Automatic Conversion of Event Data to Event Logs Using CNN and Event Density Embedding

SUNGHYUN SIM1, (Member, IEEE), RISKA ASRIANA SUTRISNOWATI2, SEOKRAE WON2, SANGHWA LEE2, AND HYERIM BAE3, (Member, IEEE)

1Institute of Intelligent Logistics Big Data, Pusan National University, Geum-Jeong Gu, Busan 609-753, South Korea
2IOChord Inc., Haeundae-gu, Busan 48059, South Korea
3Major in Industrial Data Science & Engineering, Department of Industrial Engineering, Pusan National University, Geum-Jeong Gu, Busan 609-753, South Korea

Corresponding author: Hyerim Bae (hrbae@pusan.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by Korean Government (MSIT) under Grant 2020R1A2C110229411, and in part by NRF through the Ministry of Education under Grant NRF-2020R1A 2C1102294.

Abstract

In process mining, converting event data to event logs is related to the quality of analysis results. In general, to convert event data into event logs, it is necessary to identify process entities, such as the case identifier, activity label, activity originator, and activity timestamp, from the data fields in the event data, as well as other optional attributes. Up to now, the event log conversion process has been attempted by relying on an expert’s intuition or an analyst’s experience. However, the conversion is a challenging procedure without sufficient prior knowledge of process mining. To automate the conversion process, an event log–converting algorithm based on the convolutional neural network (CNN) was developed with a new embedding method called Event Density Embedding (EDE). To verify the performance of the proposed embedding method and the automatic event log conversion framework, a comparative experiment was performed using nine pieces of real-world event data. The experiments show that our method is 5-20% higher conversion accuracy than the other methods. It is expected that business experts will be able to easily apply the method to process mining technology by utilizing system-derived event data.

Index Terms

Process mining, automatic event log conversion, event data engineering, event density embedding.

I. INTRODUCTION

With the growing interest in hyperautomation and Robotic Process Automation (RPA), data-driven process analysis methodologies are becoming more critical [1], [2]. Process mining is a method of analyzing a process based on executed historical event data, which can be a basis for RPA and hyper-automation [3], [4]. To start process mining, acquisition of the event logs is essential. Event logs used for process mining must comply with a standard format such as eXtensible Event Stream (XES) [5]. For this reason, process mining requires a procedure for converting system-extracted log data into event logs that conform to XES, which can be achieved by mapping the columns between the two sets of data [6]. Converting event logs is a crucial step in gaining valuable processing insights but is considered a complicated task. For this reason, the procedure for preparing event logs for process mining can take 80% of the entire analysis process [7].

Furthermore, the conversion of log data to an event log should be accurate, and accuracy is highly correlated with the quality of event logs used in process mining [8]. The issue of quality in the event log was first discussed in the “Process Mining Manifesto” [9], which classified the quality of event logs into five levels. It was emphasized that with good definitions of case identifier, activity label, organizer label, and timestamp, the highest quality of process discovery can be achieved [8], [9]. Afterward, the notion of an unlabeled event log, which means the column for the case identifier is unknown, was introduced in [10]. Some researchers [11]–[13] have tackled the problem of how to handle an unlabeled event log when case identifiers are not specified. However, in the real world, there are many cases where other event log elements, such as activity label, originator, and so on, are not specified either. These problems have not yet been properly resolved. As indicated in Figure 1,
Therefore, a new method of assigning columns in log data to process mining entities in an event log (including case identifier, activity label, originator label, and attribute label) is needed [11]. The assignment can be considered a conversion process. That is, after assigning columns of log data on the left-hand side of Figure 1 to process entities, an event log can be obtained as process mining input, as seen on the right-hand side of the figure. In the figure, the XES format of process mining input is shown. Container No in the event data of Figure 1 is converted to case:concept:name in the XES format of the event log, and Job Name is converted to concept:name. This procedure repeats until columns in the event data are mapped to all of the process mining entities, which are case identifier, activity label, originator label, attribute label (called process mining entities), and timestamp [9], [10]. In this case, process mining entities are labels, which can be used in supervised learning. Mapping process mining entities to unlabeled columns of event data can be achieved through supervised learning, which can improve the quality of process mining results.

Our study proposes a method that can automatically create the event log by detecting all process mining entities. Thus, this problem is formulated as a classification problem, and a convolutional neural network (CNN)-based model is used to solve this problem. One factor that significantly affects conversion accuracy in automatically converting event logs using a CNN is the embedding method [14], [15]. Because event data include numerical and categorical variables, an embedding process is required to use them as input to the CNN.

The problems faced can be summarized as follows. First, the problem of converting event data into event logs is important, because it has a significant impact on the quality of analysis; however, existing studies have dealt only with some, not all, of the process mining entities [11]–[13]. Second, among the solutions for automatic mapping of process mining entities, an appropriate embedding method is needed to use a CNN-based approach that ensures good performance even for categorical-type variables [16], [17]. This paper makes the following contributions:

- It provides a new automatic conversion method based on the CNN
- It proposes a new embedding method called Event Density Embedding (EDE), by which the neural network can learn the event data’s structural information
- It introduces a new framework in which all event data can be automatically converted to an event log in the XES format.

The remainder of this paper is organized as follows. Section II covers the background and related work. Section III describes the proposed methodology. Section IV verifies the proposed method through a case study based on various executions of event logs. Section V analyzes robustness, while Section VI summarizes the proposed automatic event log conversion method and the present research.

II. BACKGROUND AND RELATED WORK

This section introduces the research background on process mining, event logs, and convolutional neural networks. It also discusses the related work on event log embedding issues.
A. PROCESS MINING AND EVENT LOGS

Process mining extracts meaningful knowledge from an event log generated from an information system’s execution [18]. An event log is a set of structured or semi-structured data extracted from software information systems or database systems [19]. Figure 2 provides an example of a process execution log compiled from a container-handling process. An event, \( e \), has several attributes, including the case identifier, the activity, the resource, and the timestamp, as shown in Figure 2. The first three events form a trace, \( \sigma \), of three events: \( \sigma = < e_{11}, e_{12}, e_{13} > \) (\( e_{11}, e_{12}, \) and \( e_{13} \) correspond to C01 and are generated by executing activities DQ, DM, and DY, respectively).

B. CONVOLUTIONAL NEURAL NETWORK

A CNN is a representative model of deep learning, and is widely used for pattern recognition and image classification. The first CNN was a feed-forward neural network called LeNet [20] and consisted of a convolutional layer, a pooling layer, and a fully connected layer. Since then, research has been actively conducted to improve LeNet, for which purpose, structures such as AlexNet [21], GoogLeNet [22], VGNet [23], and ResNet [24] have been proposed.

AlexNet, which was an improvement on the CNN, has been recognized as the first work to popularize the CNN in the field of deep learning [21]. The network structure is similar to LeNet. However, instead of alternating convolutional layers and pooling layers, AlexNet has all convolutional layers stacked together. Furthermore, this network is much larger and more profound than LeNet. Szegedy et al.’s research proposed a new architecture called GoogLeNet that applies the concept of Inception. The Inception module performs \( 1 \times 1, 3 \times 3, \) and \( 5 \times 5 \) convolutional operations to efficiently extract features, and then performs \( 1 \times 1 \) convolutional operations to combine the features extracted. This method has the advantage of reducing the number of parameters, even as the depth of the neural network increases. As a result, GoogLeNet is a particular incarnation that has 22 layers of the Inception module but fewer parameters than AlexNet.

Research by He, Zhang, Ren, and Sun [24] presented a residual learning framework that learns a residual function on the received input instead of a function where a layer is not referenced, and proposed a new CNN architecture called ResNet. That work proved beneficial for training deeper networks, since residual networks are more comfortable with optimizing and gaining accuracy. This network’s main drawback is that it is expensive to evaluate, owing to the vast number of parameters. But the number of parameters can be reduced, to an extent, by removing the first fully connected layer (most of the parameters are at this layer) without any effect on performance.

Another advantage of a CNN is the ability to perform feature extraction quite well from complex data dimensions. For this reason, some studies ([25], [26]) have been conducted to represent complex relationships (e.g., the cardinality between variables; information between different locations) that are
difficult to express in tabular data, and to then extract features using the CNN. Similarly, as in the present case, whereas event logs have a designated schema, in some domains, it is possible to compose it from a combination of multiple fields (or columns) in the event data. Hence, the CNN was deemed the appropriate solution to solve the problem at hand.

**C. EVENT LOG EMBEDDING METHOD**

Various embedding methods have been developed to apply machine learning techniques to process mining. In the initial stage of embedding for process mining, one-hot encoding is widely used. Polato et al. [27] proposed Naive Bayes (NB) and Support Vector Regression (SVR) methods to estimate an ongoing event’s end time by using one-hot encoding. One-hot encoding has been used in other studies in process mining. Hinkka et al. [28] suggested an algorithm using a Recurrent Neural Network (RNN) and one-hot encoding for process instance classification. One-hot encoding has the advantage of being easy to apply, but there is also a disadvantage in that when the data size increases, the curse of dimensionality occurs and learning becomes difficult [29]. Despite these disadvantages, the one-hot encoding method is routinely used in the process mining field [30]–[33].

To overcome one-hot encoding’s weakness, an event log representation method using Word2Vec and Doc2Vec was proposed by de Koninck, vanden Broucke, and De Weerdt [34]. That research developed vectorized embedding (i.e., Act2Vec, Trace2Vec, Log2Vec, and Model2Vec), which can represent the vectors of an activity, a trace, an event log, and a process model, respectively. They conducted comparative experiments using the K-means algorithm, and in their study, the four proposed embedding methods showed improved performance relative to other embedding methods. In studies applying deep learning to process mining domains, the vectorized embedding method of de Koninck, Vanden Broucke, and De Weerdt [34] has been widely used [35]–[38]. Ni [35] performed a study that automatically recommended the medical procedure best suited to a patient’s condition. In that study, the activity recorded in the execution event log was vectorized using Act2Vec. This study developed a good process recommendation system using Long Short-Term Memory (LSTM) and the Hierarchical Hidden Markov Model (HHMM). Another meaningful application is the conformance checking method based on activity and trace embedding for real-time process monitoring, which was developed by Peeperkorn, Vanden Broucke, and De Weerdt [34]. Their study demonstrated significantly better performance than other experiments that had randomly generated and detected noise based on virtual event logs.

Another approach to representing event logs is entity embedding, which is a technique for converting categorical data into vectors by learning the similarity between categorical variables. Entity embedding was proposed by Guo and Berkhahn [39]. This method was first used in the field of process mining by Wahid, Adi, Bae, and Choi [40]. Compared with other event log representation techniques in process mining, their research presented a new technique that has not yet been popularized. They made predictions of an event’s remaining processing time using a Deep Neural Network (DNN). Through a comparison with one-hot encoding, that research proved that their method’s prediction performance was excellent. However, due to the fact that they did not provide a comparison with the vectorized embedding method, it cannot be known which embedding method is best in terms of classification performance. The entity embedding method has recently been used in methodologies for predicting business processes as well as process remembering time [40]–[43]. However, rather than embedding the entire event log, only individual process mining entities can be used, which imposes a limitation. The research related to the event log embedding method used to apply the deep learning model in the process mining field is summarized in Table 1.

**III. THE PROPOSED METHOD**

This section describes how to automatically convert event data into event logs based on a convolutional neural network. First, the Event Density Embedding (EDE) method, which allows the network to learn the event log structure, is introduced. After that, the structure of the CNN for automatic conversion from event data to an event log is explained.

**A. EVENT DENSITY EMBEDDING**

As stated in the previous section, the existing approaches mainly consist of embedding techniques to express the relationship between activities in traces or the relationship between traces in the event log. To use a deep learning technique in the automatic conversion to process mining entities, the new embedding method, namely Event Density Embedding (EDE), herein is proposed.

Before explaining EDE, the entity relationships that exist in event logs should be elaborated. The entities used in process mining are, as noted earlier, case identifier, activity label, originator label, attribute label, and timestamp. A single process instance (the case identifier) has a relationship with multiple activities since more than one activity is performed in the execution of the process instance. Each activity has the originator that executed it (who) and the time information when the activity was executed (when). As a result, process mining entities have hierarchical structures, and thus, cardinality information can be generated. This structural information can eventually determine which entities are case identifiers, activity labels, originator labels, and timestamps. For example, to determine the case identifier, each event data entity must be tested to determine if it can be the case identifier. To that end, all of the other entities’ structural relationships with the pivot column need to be checked by specifying each candidate as a pivot column.

The left side of Figure 3 shows an example of event data. The figure’s right side shows the frequency tables, each showing the relationships between a pivot column and the other entities. For instance, if Container No is selected as the pivot column, each value’s frequency in other entities can be
TABLE 1. Summary of related work about event log embedding method.

| Paper | Model | Input | Embedding Method |
|-------|-------|-------|------------------|
| Polato et al. [27] | SVR | ✓ | ✓ |
| Hinkka et al. [28] | RNN | ✓ | ✓ |
| J. Moon et al. [30] | GPT-2 | ✓ ✓ ✓ | ✓ |
| F. Taymouri and M. L. Rosa [31] | AE+ GAN | ✓ | ✓ |
| M. Camargo et al. [32] | LSTM | ✓ | ✓ |
| C. Huang et al. [33] | RNN | ✓ | ✓ |
| W. Ni [35] | AE | ✓ | ✓ |
| X. Qian et al. [36] | LSTM | ✓ | ✓ |
| A. Guazzo et al. [37] | AE | ✓ | ✓ |
| A. Seeliger et al. [38] | RNN | ✓ | ✓ |
| N. A. Wahid et al. [40] | CNN | ✓ | ✓ |
| V. Pasquadibisceglie et al. [41] | LSTM | ✓ | ✓ |
| A. Belhadi et al. [42] | RL | ✓ | ✓ |
| B. Dalmas et al. [43] | CNN | ✓ | ✓ |

TABLE 2. Summary of non-linear functions.

| Function | Sigmoid | TANH | ReLU | ELU |
|----------|---------|------|------|-----|
| Equation | $f(x) = \frac{e^x}{e^x+1}$ | $f(x) = \frac{1-e^x}{1+e^x}$ | $f(x) = \max(0, x)$ | $f(x) = a(e^x - 1)\quad x < 0$ |

determined. In the figure, the distinct number in Job name for the pivot column value of C01 is 3. Similarly, frequency tables for all of the columns can be compiled. These values eventually represent the cardinality between process mining entities. And, in order to learn the event log’s structural features using these values, a new embedding method herein is proposed. The proposed EDE method proceeds as shown in Figure 4. Through the preprocessing phase and projection phase shown in Figure 4, the embedded result that can learn the event log structure is derived. In the transformation phase, steps 1 to 3 are performed, and in the projection phase, Step 4 is completed. The EDE method proceeds as follows.

- Step 1: Select one pivot column from among all columns in the event data.
- Step 2: Calculate the number of intersections between the elements of the selected pivot column and the remaining columns.
- Step 3: Repeat Step 1 and Step 2 for all columns.
- Step 4: Project the results of Step 3 sequentially onto linear and non-linear functions.

In Step 4, an embedding layer and an output layer composed of non-linear functions are used to project the transformation step’s matrix. In this case, a linear projection function is used for the embedding layer, and non-linear functions are used for the output layer. Traditionally, studies related to embedding have used methods to map learning data to vector spaces. Recently, a growing number of embedding-related studies [44], [45] use activation functions to obtain embedded results in the process of embedding, showing good performance compared to past studies.

Non-linear functions used in deep learning include functions [46] such as the sigmoid function, the hyperbolic tangent (tanh) function, the rectified linear unit (ReLU), and the exponential linear unit (ELU). Table 2 shows the equations of the non-linear functions used in the EDE method. The algorithm below shows the pseudocode of the EDE method.

B. AUTOMATICALLY CONVERTING EVENT DATA TO EVENT LOGS

This section introduces a CNN structure and training method that can automatically map event data to event logs. The training procedure is performed using the EDE method proposed in Section 3.1. Figure 4 shows the procedures for generating training data using port-logistics event data, and
training using embedding and the CNN. The deep learning-based approach to automatic mapping consists of five steps.

- **Step 1:** Select a candidate pivot column.
- **Step 2:** For the pivot column selected, randomly extract \( k \) rows and generate \( n \) sub-event data, including the \( k \) randomly extracted rows.
- **Step 3:** Calculate the frequency of the intersection between the pivot column and the remaining columns from the \( n \) sub-event data extracted.
- **Step 4:** Generate input data through the embedding layer using the results of Step 3, create a dataset, and send it to the input layer for training.
- **Step 5:** With the prepared correct label information as shown in the upper right of Figure 5, perform training using the CNN consisting of five convolutional blocks, an average pooling layer, and a classification layer.

1) **GENERATING SUB-EVENT DATA**

After event data are extracted from the system, need to be preprocessed for training. For preprocessing of training data, sub-event data was used in the present study. Those data were generated from event data through a sampling procedure. There are two methods of separating event data into sub-event data. One is separating them sequentially by listing the events in the order in which they occurred. The other is separating them by randomly extracting events. The first method poses a problem where autocorrelation may appear within the same sub-event data. Therefore, to avoid the autocorrelation problem, the second method (randomly separating the event data) was applied.

The lower part of Figure 5 illustrates the procedure to separate sub-event data for training using example event data. First, the pivot column and the label to be mapped are determined. For instance, \( Container\ No \) is matched to \( Case\ Identifier \), \( Job\ Name \) to the activity label, \( Machine\ No \) takes the originator label, and \( Machine\ Type \) gets the attribute label. Then, one pivot column is selected, and elements in the column are selected randomly at a constant rate. The event data of all the rows, including the selected elements, are extracted as sub-event data. For example, suppose there are 100 independent cases from C01 to C100 in the \( Container\ No \) column. At this time, \( p\% \) of cases are randomly selected, and \( n \) sub-event data, including the selected cases, are extracted. This procedure is performed for all pivot columns.

2) **DEEP LEARNING-BASED MODEL DEFINITION FOR AUTOMATIC EVENT LOG CREATION**

The model-training step executes the embedding process on the generated sub-event data, and trains using the CNN model. In the embedding procedure, each sub-event is handled in a format that can be learned by performing the transformation and projection procedures in Figure 4. The embedding process transforms the data into a three-dimensional array at \( 225 \times 225 \times 3 \), and the transformed 3-D array is fed back into the CNN input layer. A CNN network composed of five convolutional blocks and one classification layer is constructed to secure good training performance.
FIGURE 4. EDE method’s procedure using example event data.

The convolutional blocks consist of a convolutional layer, batch normalization, an activation function, dropouts, and max pooling. The convolutional layer contains filters for learning parameters. In general, the filter size should be smaller than the input data’s height and weight, and various sizes of filter, such as $2 \times 2$, $3 \times 3$, and $5 \times 5$, can be used. The convolutional layer’s operation is expressed in Equation 1 [47]:

$$z_{i,j,k} = \sum_{h=0}^{f_h-1} \sum_{v=0}^{f_v-1} \sum_{k'=0}^{f_k'-1} x_{i',j',k'} \times w_{u,v,k',k}$$

$$\text{with } \begin{cases} i' = i \times s_h + h \\ j' = j \times s_v + v \end{cases}$$  \hspace{1cm} (1)$$

where $z_{i,j,k}$ represents the output of neurons located in the $i$ and $j$ columns in the $k$-feature map of the convolutional layer (the $l$-layer); $f_h$ and $f_v$ represent filters for height and width, respectively; $s_h$ and $s_v$ represent the strides of height and width, respectively; $f_{k'}$ represents the number of feature maps in the previous convolutional layer (the $l-1$ layer); $x_{i',j',k'}$ represents the output of neurons in the $i'$ row, $j'$ column, and $k'$-feature map of the previous convolutional layer (the $l - 1$ layer). Finally, $w_{u,v,k',k}$ represents the neuron weights of the $k'$-feature map, the $u$-row, and the $v$-column associated with the filter in the $k$-feature map of the $l$-layer.

Batch normalization [48] is a technique that solves the internal covariate shift problem that occurs during network learning to prevent gradient vanishing and gradient exploding. The batch normalization layer is known to be used between the convolutional layer and the activation function. In the present study, the batch normalization layer was introduced to prevent gradient vanishing and gradient exploding.

The Exponential Linear Unit (ELU) function was chosen as the activation function [46]. The ELU function contains all the ReLU function’s advantages. It solves the dying ReLU problem, where the ReLU function dies in a small part of $x < 0$. It also has the advantage of the output value being almost zero-centered. The equation for the ELU function is shown in Table 2.

For the dropout layer, Srivastava, Han, Kumar, and Singh [49] proposed a technique to reduce overfitting. The dropout layer’s main advantage is that the weights of all neurons cannot be synchronously optimized. In other words, overfitting can be avoided by preventing learning neurons from converging to the same goal.

The last classification block has a configuration similar to the convolutional block, but there is no dropout layer and uses an adaptive average pooling layer rather than a max-pooling layer. Since the number of labels to be classified can vary, there is a disadvantage in that the filter size of the pooling layer must be changed each time it is trained. In other words, the filter size automatically changes according to the output size without having to adjust the kernel size every time the label changes.
The cross-entropy function is used as the loss function. Cross-entropy loss is known to show good learning performance in multi-label classification problems [50]. The following equation gives the cross-entropy function:

\[ J = - \frac{1}{N} \sum_{i=1}^{N} y_i \times \log(\hat{y}_i) \]  

(2)

where \( N \) refers to the size of the data, and \( y \) is both the real value and the predicted value.

Table 3 shows the overall composition of the CNN model with the proposed input and output sizes, while the configuration design is similar to the AlexNet network structure, which is one of the most famous models among the existing CNN architectures. Some changes to the structure of the CNN were made, based on the structure of AlexNet and the latest CNN-related research [50], plus reliable experience. In a study by Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov [49] and in the one by Ioffe and Szegedy [48], comparisons of training performance according to various layer arrangements of the CNN model were performed. As a result, the best learning effect was found when the batch normalization layer was located between the convolutional layer and the activation function layer, and when the dropout layer was located between the activation function and the pooling layer. For this reason, the convolutional block was organized in the following order: convolutional layer, batch normalization layer, activation function, dropout layer, and pooling layer. A 3 × 3 filter was used for the convolutional layer, as well as a 2 × 2 filter for the stride size, and a 2 × 2 filter for the padding size.

### IV. DATASETS AND EXPERIMENTS

To validate the proposed approach, real-life event logs from nine different areas were used for experimentation. First, the datasets used for the experiments are described below. Then, the experiment design is introduced. Finally, the comparison results are presented.

### A. DATASETS

The experiments used nine event datasets (LD1 to LD9) from port logistics, steel manufacturing, finance, IT, and government administration, as shown in Table 3. Note that LD3 to LD9 are publicly available event data used in Business Process Intelligence Challenge 2012 (BPIC 2012) to BPIC 2018 [51]–[57]. The detailed descriptions of each set of event data can be found in Table 4.

Since the CNN algorithm used in the present study follows supervised learning, an input/output set was needed. The
input is the embedding of event data (LD1 to LD9), which is represented as images for the CNN. From the resulting images, process mining entities could be specified, which means that process mining could be initiated with this information. For LD1 and LD2, the information was acquired by domain experts, and for LD3 to LD9, the information was taken from the winning results of BPIC 2012 to BPIC 2018 [58]–[64]. This information was used to validate the accuracy of the automatic event log conversion algorithm.

### B. EXPERIMENTAL METHOD

The proposed algorithm was evaluated using the real-life event logs introduced in the previous section. The purpose was to determine whether the newly proposed EDE method can solve the automatic event data conversion problem. Experimentation was also undertaken to see if the proposed method can provide superior performance to that of a state-of-the-art method used in process mining. Two comparative experiments were performed. EXP I checked the conversion accuracy among the different types of non-linear functions in the EDE method, and EXP II compared the conversion accuracy and learning performance between an existing embedding method and the EDE method.

In this study, two performance indices were used. The first was conversion accuracy, which indicates whether the trained model can accurately convert each column in the event data to the event log’s required entities (e.g., case identifier, activity label, activity originator, and timestamp). The second was the learning performance index, which is based on the value of the cross-entropy loss measured during training.

EXP I determined which non-linear function is most suitable from among the several non-linear functions used in the EDE method. EXP I had three steps.

- **Step 1:** From all event data, 1,000 sub-event data were created using the data separation method.
- **Step 2:** After applying four different non-linear functions (sigmoid, tanh, ReLU, ELU) in the EDE method to the generated sub-event data, a learning process of over 100 epochs was conducted using a CNN of the same structure.
- **Step 3:** As measures of learning performance, the test data’s conversion accuracy (classification accuracy) and the deviation between the average accuracy and the learning performance index were calculated.

### TABLE 4. Descriptions of experimental event datasets.

| Event Data | No. Case | No. Event | Data Descriptions | Process Mining Entities |
|------------|----------|-----------|-------------------|-------------------------|
| LD1        | 13,954   | 55,004    | Logistics process of a Korean port | caseid: Container identifier, activity label: Job name, originator label: Machine number, attribute label: Remaining columns of event data |
| LD2        | 77,405   | 1,048,575 | Steel manufacturing processes of a Korean company | caseid: Production identifier, activity label: Job name, originator label: Machine Name, attribute label: Remaining columns of event data |
| LD3 [51]   | 13,087   | 13,087    | Bank lending process in a Netherlands company | caseid: CaseID, activity label: Activity, originator label: Resource, attribute label: Remaining columns of event data |
| LD4 [52]   | 7,554    | 65,533    | Incident management process within Volvo | caseid: CaseID, activity label: Activity, originator label: Resource, attribute label: Remaining columns of event data |
| LD5 [53]   | 46,616   | 644,738   | ITIL process for the Rabobank Group | caseid: Incident_ID, activity label: IncidentActivity_Type, originator label: Assignment_Group, attribute label: Remaining columns of event data |
| LD6 [54]   | 1,199    | 52,217    | Building permit processes in five Dutch municipalities | caseid: case, activity label: action_code, originator label: resource, attribute label: Remaining columns of event data |
| LD7 [55]   | 26,279   | 1,070,215 | Handling process for employee insurance within a Dutch administration | caseid: session_id, activity label: page_name, originator label:user_id, attribute label: Remaining columns of event data |
| LD8 [56]   | 31,509   | 1,202,267 | Financial service processes in a Dutch bank | caseid: caseID, activity label: taskID, originator label: originator, attribute label: Remaining columns of event data |
| LD9 [57]   | 17,617   | 1,048,575 | Payment application and execution processes for German farmers | caseid: docid_uid, activity label: doctype, originator label: resource, attribute label: Remaining columns of event data |
classification accuracy were applied. All of these performance measures were obtained from 30 iterations.

EXP II checked whether the proposed embedding method is suitable for learning the structure of event data, compared to the existing embedding method. The experiment was performed as follows.
- Step 1: From all event data, 1,000 sub-event data were created using the data separation method.
- Step 2: Five embedding methods (one-hot encoding, entity embedding, Act2Vec, Trace2Vec, and EDE) were applied to the generated sub-event data, which were then input into the CNN and learned for 100 epochs in the same environment.
- Step 3: As measures of learning performance, the test data’s conversion accuracy (classification accuracy) and the deviation between the average accuracy and the classification accuracy were applied. All of these performance measures were obtained from 30 iterations.

In order to measure conversion accuracy, test data was inputted into the trained model to map each column of event data to each process mining entity. Then, a confusion matrix was derived by comparing it with the actual process mining entities. Using the confusion matrix thus derived, the conversion accuracy was calculated by following equation [65]:

$$\sum_{i=1}^{k} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

where $k$ is the number of process mining entries; $TP_i$ represents a case where the predicted value of the model and the actual value are both true; $TN_i$ indicates the converse case, where the predicted and actual values of the model are both false, and $FP_i$ ($FN_i$) represents a case where the model does not get the correct answer, the actual value is false (true), but (and) the model’s predicted value is true.

Both experiments used 70% of the data for training and 30% for testing. For example, in LD1, 38,502 rows of events were used as training data and the remaining 16,501 rows of events were used as test data. The model structure employed in the experiment used the structure shown in Table 3. For model parameter optimization, the Adam optimizer [66], known to show the most stable learning performance among the previously developed optimization tools, was utilized. Model training was performed by applying an exponential learning rate schedule [67] technique known to be relatively stable. Finally, when performing the proposed EDE, 30 samples are randomly extracted, after which the experiment was performed (Section III.B. Step-2). The experiments were conducted using a standalone AMD Ryzen 7 2700 eight-core (3.2 GHz) computer with 64 GB of memory running Windows 10. The model training was done using the NVIDIA GeForce RTX 2070 with 8 GB of dedicated GPU memory using CUDA v.10.2 and PyTorch 1.2.0.

### C. EXPERIMENTAL RESULTS

In this section, the results of EXP I and EXP II are summarized and discussed. First, EXP I compared the performance of the EDE method based on the non-linear function used.

Figures 6, 7, and 8 show the average training losses recorded when training 100 times per iteration. From the results in Figure 6 to Figure 8, each embedding method’s effect on the convergence of automatic mapping model training can be seen. It is intuitively apparent that the
FIGURE 7. Average curves for training loss after 30 iterations on LD2 to LD6.
proposed embedding method obtained the fastest training loss convergence on all of the data used in the experiments. The Trace2Vec embedding method reached convergence faster than Act2Vec, with the entity embedding method and Trace2Vec being similar, and converging slightly faster. Finally, it was found that, on average, the training losses for models using the one-hot encoding method were the slowest to converge.

In Table 5, the linear unit functions (ReLU, ELU) showed comparatively good results. The ReLU series function showed a better result because the ReLU series function returns zero or a near-zero value for a negative area, and returns the value itself for a positive area. This feature has the advantage of distinguishing the object of classification more clearly [68]. In our experiments as well, the ReLU series function showed a similar effect.

In Table 5, the linear unit functions (ReLU, ELU) showed comparatively good results. The ReLU series function showed a better result because the ReLU series function returns zero or a near-zero value for a negative area, and returns the value itself for a positive area. This feature has the advantage of distinguishing the object of classification more clearly [68]. In our experiments as well, the ReLU series function showed a similar effect.
Table 6 shows the time performance for each embedding method. To compare the time performance, the time spent on training for 1 epoch was measured, and this process was repeated a total of 30 times. In the results, the EDE method had the fastest learning speed for five out of the total of nine data. One-hot encoding took the longest time to train, and Trace2Vec took the second-longest time to train.

Table 7 summarizes the mean and standard deviation for conversion accuracy in the test data after 30 training iterations over 100 epochs. In the results, EDE showed the best mapping accuracy. Compared to the existing embedding method, the mapping accuracy by EDE showed higher value of at least 2% to 3%, and up to as much as 15% to 20%. Based on the experiment results in Table 5, EDE showed the best accuracy, compared to the other embedding methods. Thus, when using deep learning to solve the event log–mapping problem, the proposed embedding method is suitable, and at the same time, stable.

V. ANALYSIS OF ROBUSTNESS OF PROPOSED METHOD

In the previous section, it was shown that EDE outperforms the existing embedding methods. Now, as detailed in this section, it will be confirmed that the proposed embedding model works robustly, even when it uses various event data domains to train and test the model. Specifically, eight out of the nine event data domains were merged and then used as training event data, saving the remaining dataset for validation. For example, a model for automatic event log transformation from the proposed method was trained by using LD1 to LD8, and verification was performed using LD9. This process was repeated with LD1 to LD9 and was carried out over 30 iterations to verify whether the trained model can perform well, even with event data not involved in training. In this experiment, the ELU function, which generally showed excellent results in EXP I, was used.

Table 8 lists the results, and shows that except for two experiments (using LD6 and LD7 as testing data), most of
the experiments showed more than 70% conversion accuracy, and some experiments showed more than 85% accuracy. To find the cause of this result, the process model's structural complexity for each event dataset was analyzed. Table 9 summarizes the results from analyzing the complexity of the process model. In fact, LD6 and LD7 (with the lowest conversion accuracy in the above experiments) had higher complexity in the process model than other event data.

To confirm this provisional fact, the process model's complexity was lessened by randomly reducing the number of AND and OR gates in LD6 and LD7, and an additional experiment (EXP III) was conducted by reiterating the experiment conducted above. EXP III was carried out in the following steps:

- Step 1: Sampling was performed randomly from LD6 and LD7. Extracting sub-event data made the total number of gates 50% to 90% (in 10% intervals). At this time, the event data extracted through sampling were denoted LD6∗ and LD7∗.
- Step 2: The experiment was repeated 30 times by following the same procedure as above. Then, the accuracy and standard deviation from the testing data were obtained.

Table 10 presents the results from EXP III and shows that accuracy increases when decreasing the ratio extracted from LD6 and LD7. This experiment revealed that the number of complex gates included in the event data correlates with the results. For example, by reducing LD6 by 50%, a test accuracy of 74.79% was obtained, which was about a 6% improvement (the accuracy in EXP II for LD6 was 66.82%). This effect was similarly shown in the other complex event log, LD7. These results mean that if the proposed method does not have a large total number of gates, the trained model can work robustly, even on event data with different structures. While complex processes tend to be less accurate when included, the accuracy reduction was only 6% to 7% with the EDE method.

### VI. CONCLUSION AND DISCUSSION

This paper introduces a deep learning-based conversion framework that automatically converts event data to event logs. An appropriate embedding method for converting event data into an event log is needed to transform input data so that a network can learn the event log structural information. A new Event Density Embedding (EDE) method was proposed herein. Unlike existing embedding methods used in process mining, the proposed method depends on how much the process entities are interrelated in terms of occurrence frequency. From the event data, the pivot column was selected, in relation to which a matrix with the remaining columns' frequency was derived. Then, the data was projected onto a linear function and a non-linear function. For training of the conversion-to-process-mining input, a CNN-based structure that can guarantee excellent accuracy was proposed herein. The neural network structure was customized sufficiently to achieve high performance from the automatic conversion of event data into an event log.

In the experiments, a non-linear function that can maximize conversion accuracy was found. The experiments were performed using actual event data collected from 9 different domains for comparison with existing embedding methods. The results showed that the proposed embedding method outperforms the existing methods. The results can be summarized in terms of learning time and accuracy as follows. 1) Training time performance: the proposed method improved learning speed by at least 10% relative to the other methods in more than half of the experimental data. In particular, the learning speed was improved by at least 40% relative to one-hot encoding as affected by the size of the variable, and relative to trace2vec as affected by the length of the trace. 2) Test accuracy: in all of the test event data, the proposed embedding method showed a 5-20% higher conversion accuracy than the other methods. While the existing embedding methods considered only specific elements of the process mining entity, the proposed embedding method showed high accuracy by considering the relationship between process mining entities.

Although event data have different types of model structures, the proposed model can be applied and can obtain good conversion accuracy. In addition, it is expected that by embedding the entire process mining entity, the proposed method can be used as an event data embedding method that

### TABLE 10. Number of XOR gates and AND gates in LD1 to LD9.

| Sample Ratio | LD1 | LD2 | LD3 | LD4 | LD5 | LD6∗ | LD7∗ | LD8 | LD9 | Test Accuracy |
|--------------|-----|-----|-----|-----|-----|------|------|-----|-----|----------------|
| 90%          | Tr  | Tr  | Tr  | Tr  | Tr  | Ts   | Tr   | Tr  | Tr  | 67.33 (1.12)   |
| 80%          | Tr  | Tr  | Tr  | Tr  | Tr  | Ts   | Tr   | Tr  | Tr  | 65.27 (1.33)   |
| 70%          | Tr  | Tr  | Tr  | Tr  | Tr  | Ts   | Tr   | Tr  | Tr  | 67.90 (2.09)   |
| 60%          | Tr  | Tr  | Tr  | Tr  | Tr  | Ts   | Tr   | Tr  | Tr  | 65.78 (2.50)   |
| 50%          | Tr  | Tr  | Tr  | Tr  | Tr  | Ts   | Tr   | Tr  | Tr  | 69.49 (2.61)   |
|              |     |     |     |     |     |      |      |     |     | 66.51 (2.81)   |
|              | 71.65 (3.09) | 68.20 (3.20) | 74.79 (3.38) | 71.86 (3.66) |

Tr: Training, Ts: Testing

Avg. % (Std. Dev. %)

VOLUME 10, 2022
can provide higher dimensions of information as input data when utilizing deep learning in process mining.

Besides the research contributions and benefits explained above, the present research has some limitations, which are twofold. 1) In this work, it was assumed that labels are given in order to transform event data into event logs. However, mappable labels can vary, depending on the analytical perspective, be it workflow, control flow, or resources. For this reason, research is needed on how to flexibly auto-map process model complexity. For these reasons, further research is needed.

REFERENCES

[1] A. Jiménez-Ramírez, “Humans, processes and robots: A journey to hyper-automation,” in Proc. Int. Conf. Busi. Process Manage., 2021, pp. 3–6.

[2] J. G. Enriquez, A. Jimenez-Ramirez, F. J. Dominguez-Mayo, and J. A. Garcia-Garcia, “Robotic process automation: A scientific and industrial systematic mapping study,” IEEE Access, vol. 8, pp. 39113–39129, 2020.

[3] B. Vajgel, P. P. Correa, T. T. De Sousa, R. V. E. Quille, J. A. R. Bedoya, G. M. D. Almeida, L. V. L. Filgueiras, V. R. S. Demuner, and D. Mollica, “Development of intelligent robotic process automation: A utility case study in Brazil,” IEEE Access, vol. 9, pp. 71222–71235, 2021.

[4] V. Leno, A. Polvyanya, M. Dumas, M. L. Rosa, and F. M. Maggi, “Robotic process mining: Vision and challenges,” Bus. Inf. Syst. Eng., vol. 63, no. 3, pp. 301–314, Jun. 2021.

[5] IEEE Standard for Extensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams, Standard 1849-2016, 2016, pp. 1–50.

[6] R. Andrews, C. G. van Dun, M. T. Wynn, W. Kratsch, M. Roglinger, and A. H. ter Hofste, “Quality-informed semi-automated event log generation for process mining,” Decis. Support Syst., vol. 132, May 2020, Art. no. 113265.

[7] E. G. L. de Murillas, W. M. van der Aalst, and H. A. Reijers, “Process mining on databases: Unearthing historical data from redo logs,” in Proc. Int. Conf. Busi. Process Manage., 2016, pp. 367–385.

[8] S. Suriadi, R. Andrews, A. H. M. ter Hofste, and M. T. Wynn, “Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs,” Inf. Syst., vol. 64, pp. 132–150, Mar. 2017.

[9] W. V. D. Aalst, A. Adriansyah, A. K. A. D. Medeiros, F. Arcieri, T. Baier, T. Bliwick, J. C. Bose, P. V. D. Brand, R. Brandjian, and J. Buiks, “Process mining manifesto,” in Proc. Int. Conf. Busi. Process Manage., 2011, pp. 169–194.

[10] D. Bayomie, I. M. A. Helal, A. Awad, E. Ezat, and A. ElBastawissi, “Deducing case IDs for unlabeled event logs,” in Proc. Int. Conf. Busi. Process Manage., 2016, pp. 242–254.

[11] A. A. Andaloussi, A. Burattin, and B. Weber, “Toward an automated labeling of event log attributes,” in Proc. Enter. Busi. Process Inf. Syst. Modeling, 2018, pp. 82–96.

[12] E. G. L. de Murillas, H. A. Reijers, and W. M. P. van der Aalst, “Case notion discovery and recommendation: Automated event log building on databases,” Knowl. Inf. Syst., vol. 62, no. 7, pp. 2539–2575, Jul. 2020.

[13] B. Ramos-Gutiérrez, L. Parody, and M. T. Gómez-López, “Towards the detection of promising processes by analysing the relational data,” in Proc. Int. Conf. Theor. Prac. Digi Lib, 2020, pp. 283–295.

[14] H. A. Almuzaini and A. M. Azmi, “Impact of stemming and word embedding on deep learning-based Arabic text categorization,” IEEE Access, vol. 8, pp. 127913–127928, 2020.

[15] M. Xu, Z. Wang, H. Zhang, D. Pantazis, H. Wang, and Q. Li, “A new graph Gaussian embedding method for analyzing the effects of cognitive training,” PLOS Comput. Biol., vol. 16, no. 9, Sep. 2020, Art. no. e1008186.

[16] M. Cho, J. Ha, C. Park, and S. Park, “Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition,” J. Biomed. Informat., vol. 103, Mar. 2020, Art. no. 103381.
[42] A. Belhadi, Y. Djenouri, V. G. Diaz, E. H. Houssein, and J. C. Lin, “Hybrid intelligent framework for automated medical learning,” Expert Syst., early access, May 7, 2021, doi: 10.1111/exsy.12737.

[43] B. Dalmas, F. Baranski, and D. Cortinovis, “Predicting process activities and timepoints with entity-embeddings neural networks,” in Proc. Int. Conf. Res. Challenges Inf. Sci., 2021, pp. 393–408.

[44] A. Q. Ohi, M. F. Mirhadi, F. B. Safir, M. A. Hamid, and M. M. Monowar, “AutoEmbedder: A semi-supervised DNN embedding system for clustering,” Knowl.-Based Syst., vol. 204, Sep. 2020, Art. no. 106190.

[45] Z. Tan, J. Chen, Q. Kang, M. Zhou, A. Abusorrah, and K. Sedaoudi, “Dynamic embedding projection-gated convolutional neural networks for text classification,” IEEE Trans. Neural Netw. Learn. Syst., early access, Jan. 8, 2021, doi: 10.1109/TNNLS.2020.3047692.

[46] Y. Wang, Y. Li, Y. Song, and X. Rong, “The influence of the activation function in a convolution neural network model of facial expression recognition,” Appl. Sci., vol. 10, no. 5, p. 1897, Mar. 2020.

[47] S. Hong and S.-K. Song, “Kick: Shift-N-overlap cascades of transposed convolutional layer for better autoencoding reconstruction on remote sensing imagery,” IEEE Access, vol. 8, pp. 107244–107259, 2020.

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

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 56, pp. 1929–1958, 2014.

[50] W. Liao, Y. Wang, Y. Yin, X. Zhang, and P. Ma, “Improved sequence classification via CNN and initialized fully connection,” Neurocomputing, vol. 382, pp. 188–195, Mar. 2020.

[51] B. Van Dongen, “BPI challenge 2012 dataset,” Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep. 1, 2012, doi: 10.4121/uuid:3926db30-f712-4394-aebc-5976070e91f1.

[52] W. Steeman, “BPI challenge 2013 dataset,” Ghent Univ., Ghent, Belgium, Tech. Rep. 1, 2013, doi: 10.4121/500573e6-acc-ce40b-957605f85e8b610cece.

[53] B. Van Dongen, “BPI challenge 2014 dataset,” Rabobank Nederland, Utrecht, The Netherlands, Tech. Rep. 1, 2014, doi: 10.4121/uuid:3cfa2260-f5c5-4f76-ab97-b70d352886d6.

[54] B. Van Dongen, “BPI challenge 2015 dataset,” Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep. 1, 2015, doi: 10.4121/uuid:a0add1da-2044-4541-a540-fd89f1e16d17.

[55] D. Marcus and B. Van Dongen, “BPI challenge 2016 dataset,” Employee Insurance Agency, Amsterdam, The Netherlands, Tech. Rep. 1, 2016, doi: 10.4121/uuid:360795c8-1dd6-4a5b-a443-185001076eab.

[56] B. Van Dongen, “BPI challenge 2017 dataset,” Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep. 1, 2017, doi: 10.4121/uuid:5f3067df-f10b-45da-b9b6-86ae4c7a310b.

[57] B. Van Dongen and B. Florian, “BPI challenge 2018 dataset,” Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep. 1, 2018, doi: 10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972.

[58] A. Adriansyah and J. M. Buijs, “Predicting process activities and timestamps with entity-embeddings neural networks,” in Proc. Int. Conf. Res. Challenges Inf. Sci., 2021, pp. 393–408.

[59] B. Dalmas, F. Baranski, and D. Cortinovis, “Predicting process activities and timepoints with entity-embeddings neural networks,” in Proc. Int. Conf. Res. Challenges Inf. Sci., 2021, pp. 393–408.

[60] I. Teinemaa, A. Leontjeva, and K.-O. Masing, “BPIC 2015: Diagnostics of human and technology interaction,” in Proc. Int. Conf. Res. Challenges Inf. Sci., 2021, pp. 393–408.

[61] B. Van Dongen, “BPI challenge 2014 dataset,” Rabobank Nederland, Utrecht, The Netherlands, Tech. Rep. 1, 2014, doi: 10.4121/uuid:3926db30-f712-4394-aebc-5976070e91f1.

[62] I. Teinemaa, A. Leontjeva, and K.-O. Masing, “BPIC 2015: Diagnostics of human and technology interaction,” in Proc. Int. Conf. Res. Challenges Inf. Sci., 2021, pp. 393–408.

[63] Z. Li and S. Arora, “An exponential learning rate schedule for deep learning,” Nov. 2019, arXiv:1910.07454.

[64] K. Ecke and J. Schmidt-Hieber, “A comparison of deep networks with ReLU activation function and linear spline-type methods,” Neural Netw., vol. 110, pp. 232–242, Feb. 2019.

SUNGHYUN SIM (Member, IEEE) received the B.S. degree in statistics and the M.S. and Ph.D. degrees in industrial engineering from Pusan National University, Republic of Korea, in 2016 and 2021, respectively. His research interests include data mining, process mining, big data analytics for operational intelligence, and process optimization based on the deep-learning method.

SEOKRANG WON received the B.S. degree in material science and engineering and the M.S. degree in management and technology from Pukyong National University, in 2002 and 2018, respectively. He joined IOChord Inc., in 2017, and is currently working as the Director of the R&D Center. He is working closely on cloud computing, big data systems, process mining, and data mining.

SANGHWA LEE received the B.S. degree in computer science from Northern Kentucky University, USA, in 1992, and the M.S. degree in computer science from the University of Cincinnati, USA, in 1995. He is one of the Founder of IOChord Inc. He is currently the CEO of the company. He is also working closely on cloud computing, big data, process mining, and data mining.

HYERIM BAE (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in industrial engineering from Seoul National University, Republic of Korea. From 2002 to 2003, he worked with Samsung Credit Card Corporation, Seoul, Republic of Korea. Since 2005, he has been a Professor with the Department of Industrial Engineering, Pusan National University, Republic of Korea. His current research interests include information system design, cloud computing, business process management systems, process mining, big data analytics for operational intelligence, analyzing huge volumes of event logs from port logistics, and shipbuilding industries using process mining techniques.

* * *