IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 8, 2021, accepted March 31, 2021, date of publication April 8, 2021, date of current version April 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3071789

Process-Aware Enterprise Social Network Prediction and Experiment Using LSTM Neural Network Models

DINH-LAM PHAM[1,2], HYUN AHN[1], KYOUNG-SOOK KIM[1], AND KWANGHOON PIO KIM[1], (Member, IEEE)

[1]Data and Process Engineering Research Laboratory, Division of Computer Science and Engineering, Contents Convergence Software Research Institute, Kyonggi University, Suwon 16227, South Korea
[2]Information Technology Institute, Vietnam National University, Hanoi, Hanoi 100000, Vietnam

Corresponding author: Kwanghoon Pio Kim (kwang@kgu.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2020R1A6A1A03040583.

ABSTRACT Process mining that exploits system event logs provides significant information regarding operating events in an organization. By discovering process models and analyzing social network metrics created throughout the operation of the information system, we can better understand the roles of performers and characteristics of activities, and more easily predict what will occur in the next operation of a system. By using accurate and valuable predicted information, we can create effective environments, provide suitable materials to perform activities better, and facilitate more efficient operations. In this study, we apply the long short-term memory, a variant of the recurrent neural network, to predict the enterprise social networks that are formed through information regarding a business system's operation. More precisely, we apply the multivariate multi-step long short-term memory model to predict not only the next activity and next performer, but also all the variants of a process-aware enterprise social network based on the next performer predictions using a probability threshold. Furthermore, we conduct an experimental evaluation on the real-life event logs and compare our results with some related researches. The results indicate that our approach creates a useful model to predict an enterprise social network and provides metrics to improve the operation of an information system based on the predicted information.

INDEX TERMS Process mining, long short-term memory neural network, process-aware enterprise social network, next event prediction.

I. INTRODUCTION

Process mining involves the extraction of information from system event logs recorded while running an information system. The main objectives of process mining are to capture what, when, and how business processes are executed [1], [2]. By reviewing these processes, we collect meaningful knowledge from event logs, e.g., hidden and duplicate tasks, loops, execution time, performers involved, and other valuable information.

Enterprises need the appropriate resources to operate successful businesses in their manner. Each different business has its characteristics in terms of resource needs. However, they all share the common feature that requires human resources to operate the system. Depending on the scope of the enterprise, the proportion of resources is allocated differently, but human resources are indispensable. The more effectively exploiting and using human resources, the more we save time and reduce enterprise businesses' costs. Moreover, it will bring higher profits, help the organization re-invest funds, and expand business operations, thereby encouraging businesses to grow more.

From the discovered information using process mining techniques, we also obtain knowledge regarding the handover of work [3] or the work transference network [4], which form a process-aware enterprise social network (ESN). Through this network, we can analyze metrics such as degree centrality, betweenness centrality, and closeness centrality to understand how different performers are involved in each activity. Therefore, the better we understand the ESN,

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

the more effective we operate our system. An interesting question is how can we predict the shape of future ESN. Using this knowledge will help us to improve the preparation for resource allocation for upcoming activities. Besides, we can save costs and improve production efficiency by preparing the appropriate resources before they are needed.

Different techniques have been proposed in several studies to predict what a system will execute in the future and the corresponding facilities and resources required (e.g., use of the Markov model with a probabilistic technique [5], [6], sequential pattern clustering and mining [7]–[10], or a transition system or stochastic Petri net simulation [11]–[13]). Regarding prediction techniques, the long-short term memory (LSTM) model [14] generally outperforms others in the prediction of the next sequence of process instances, because the LSTM network can store lengthy input sequences to predict the next sequence. That is, the current prediction made by the LSTM network can be influenced by an input sequence of activities that was input to the network many time steps ago. Over the years, different solutions have been proposed in several studies for applying the LSTM model to a variety of predictions in business process monitoring. Some of these approaches involve next activity and sequence prediction [15]–[20], and remaining time prediction [17], [19]–[21].

In this paper, we propose an approach to predict the ESN using a multivariate multi-step LSTM model, a model using multi observations at multi-step of the input data set for training and prediction. The basic idea is based on the multi-step LSTM time series forecasting model [22]. In particular, we apply the multi-step LSTM model to the category prediction problem and determine how to select parameters that yield an accurate model for predicting the next event sequence. Furthermore, we use a probability threshold to predict the next event and the numerous variants of the ESN that will be generated. We also compare our approach with some related approaches and conduct experiments using real-life event logs. These study results help us to understand the roles of performers in the system and provide effective resource allocation for the next run-time of the system.

The remainder of this paper is organized as follows: In the next section, we present some related works. Subsequently, we discuss the basic concepts related to this study and explain our approach, some experiments, and evaluations. Finally, we provide some conclusions from this study and discuss future work.

II. RELATED WORK

Evermann *et al.* [15] conducted one of the early studies by applying the LSTM model to predict process behavior at runtime. They described an initial application of deep learning to predict the next process event as an applied LSTM architecture used to predict the next event in a running process. In their early work using the deep learning approach, they achieved a certain success and evaluated the effectiveness of this approach with respect to the significant amount of work required to overcome the prediction problem in order to achieve an accurate model. However, in their research, they have not yet shown how to predict a complete process behavior and how to build a specific LSTM model.

In [17], Tax *et al.* investigated the LSTM model to build consistently accurate models for a wide range of predictive process monitoring tasks, i.e., next activity and timestamp, the suffix of a case, and remaining cycle time. They demonstrated how to apply the LSTM model to predict the next event, its timestamp and the next task, and the remaining time. Their experiment using the real-life data sets proved that the LSTM architecture outperformed existing baseline architectures and tailor-made approaches to this problem at that time.

Our approach to predicting the next event is similar to Tax's method but different in that, Tax and his co-worker only use a single step input for predictions and only predict the next step. Meanwhile, in our approach, we use multi-step and observe both activities and performers for prediction.

In [18], Lin *et al.* proposed a recurrent neural network - based predictive model, named MM-Pred, to encode multiple attributes for predicting the next event and its attributes. In the model, they introduced a component modulator to provide customized weights and representations of the event and its attributes. The name MM-Pred arises from the combination of multi-task prediction and the modulator component. The basic idea behind their solution was to use three main components in the architecture (i.e., encoders E, modulators M, and decoders D). The encoders are used to encode past events and their attributes. The modulators are used to learn the significance of attributes and to encode all information, whereas the decoders take the modulated representations as inputs and provide a decoded output. Their experimental results indicated that the model outperformed the baseline and state-of-the-art models in terms of predicting the next event and its attributes.

The similarity between our approach and the method of Lin is that we both use encoder and decoder modules. However, we conduct to predict performers and ESN meanwhile Lin *et al.* predict the next event and the suffix of events.

In [19], Camargo *et al.* introduced an approach involving the training of an LSTM architecture to predict sequences of next events, timestamps, and their associated resource pools. They addressed how to train accurate models of business process behaviors using deep learning techniques. In their study, they proposed pre-processing (scaling and n-gram encoding) and post-processing (selection of next event) methods and architectures to build and use generative models. Their approach combined the work by Tax *et al.* [17] and Evermann *et al.* [15] by not only using the embedded dimension, but also supporting categorical and numerical attributes. In particular, they used three different architectures (i.e., specialized, shared categorical, and fully shared), which were created by combining activity prefixes, role prefixes, and real-time prefixes. The results indicated that longer n-grams yield higher accuracy, log-normalization is a suitable scaling

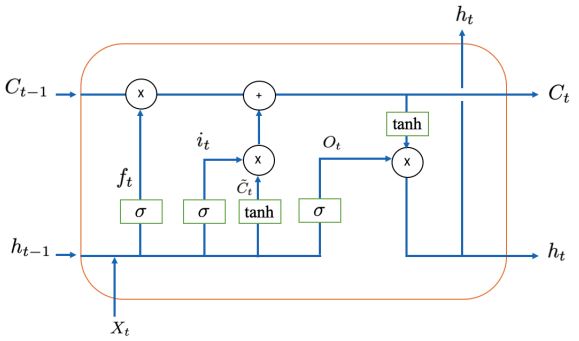


FIGURE 1. The diagram of an LSTM building block. Reproduced from [32].

method for system logs with high variability, and that randomly selecting the next event using the probabilities leads to higher accuracy.

In [21], Navarin *et al.* proposed an LSTM network to predict the remaining time in a trace of business process instances. The architecture was inspired by approaches in [17] and [23] with some modification to data attributes such that these could be input to the LSTM network.

From the discussion above, we can observe that different solutions and approaches have been proposed in several studies to improve the LSTM training model for the next event prediction. In this paper, not only predicting the next event information, we address the problem of the next phase after obtaining the prediction value. More precisely, we address the prediction of the next event sequence and the numerous variants of an ESN created from a trained multivariate multi-step LSTM model.

To the best of our knowledge, the approach in this paper is the first one deals with predict ESN from workflow execution logs.

III. BASIC CONCEPTS

A. RECURRENT NEURAL NETWORK AND LONG SHORT-TERM MEMORY NETWORK

An recurrent neural network (RNN), introduced in [24], is a type of neural network that focuses on the processing of sequences. RNNs can learn (or memorize) a sequence over time by using a special connection from output to input [27]. The term "recurrent" is used because an RNN performs the same function over time while learning sequences and using the output of the current step as the input for the next step. Backpropagation through time [28], [29] is the most common algorithm used for training RNNs. However, the problem of vanishing gradients arises while training with RNNs [30] and creates problems such as slow convergence, high complexity, and instability [31]. To overcome these limitations, we use the long short-term memory network.

The LSTM network [14] is a type of RNN that can overcome the problem of vanishing and exploding gradients while training the network [33]. Figure 1 shows a diagram of an LSTM building block. As shown in Figure 1, an LSTM block contains:

- *Input Parameters:* Input vector X_t, output of previous block h_{t-1}, and memory from previous block C_{t-1}.
- *Output Parameters:* Memory from current block C_t, output of current block h_t.

And to calculate the output of the current block, we combine the *sigmoid* and *tanh* function with the vector operations element-wise multiplication and element-wise summation by using the following equations:

$$f_t = \sigma(W_f.[h_{t-1}, X_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i.[h_{t-1}, X_t] + b_i) \quad (2)$$

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, X_t] + b_C) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

$$O_t = \sigma(W_O.[h_{t-1}, X_t] + b_O) \quad (5)$$

$$h_t = O_t * tanh(C_t) \quad (6)$$

where: $\sigma(x) = \frac{1}{1+e^{-x}}$ and $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

In these equations, (W_f, W_i, W_C, W_O) are weights and (b_f, b_i, b_C, b_O) are biases, which are learned during the training phase of the network.

To calculate the output of a block, the first step is to decide which information to remove from the previous block. This decision is made using the sigmoid function given by equation (1) to create a "forget gate". It will take an input h_{t-1}, X_t and produce a number in the range [0, 1] as the output. Output "1" indicates that it holds all previous information, whereas "0" indicates that all information will be discarded. The next step is to decide which new information will be saved in the current block state. This consists of two parts. The first is to calculate i_t, called the "input gate" to determine which values will be updated using equation (2). The second is to create a new value $\tilde{C}_t$ to add to the current state using equation (3). The two values are then combined to calculate the output memory of the current block C_t using equation (4). Finally, the output state is calculated using equations (5) and (6).

B. MULTIVARIATE MULTI-STEP LSTM MODEL

The multivariate multi-step LSTM model is used to predict multi-output values from multivariate and multi-input steps [22]. More precisely, at each time step of the input sequence, there are more than one observation and these observations will be used as the input value to predict various output values.

For example, consider two observations of an input sequence (i.e., *in_seq1, in_seq2*) and one observation of an output sequence (summation of *in_seq1* and *in_seq2*) [22]:
$in\_seq1 = [10, 20, 30, 40, 50, 60, 70, 80, 90]$
$in\_seq2 = [15, 25, 35, 45, 55, 65, 75, 85, 95]$
$out\_seq = [25, 45, 65, 85, 105, 125, 145, 165, 185]$
For the case (input-step = 3), (output-step = 1), which we later abbreviate as (3,1), the model is as shown in Figure 2a. For the case (4, 2), the model is as shown in Figure 2b.

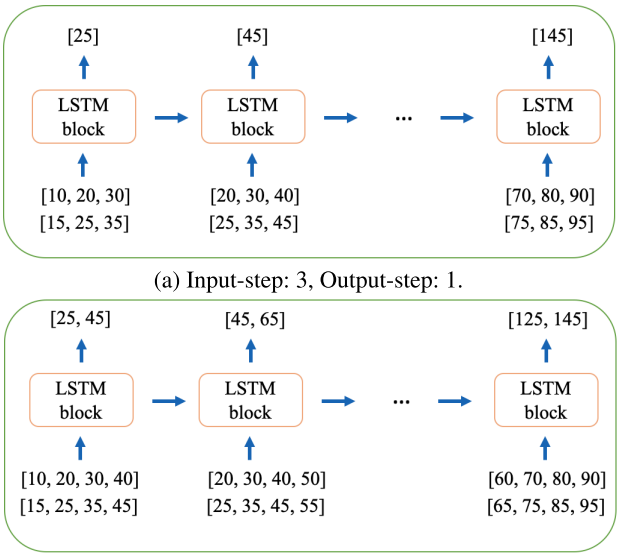


FIGURE 2. A multivariate multi-step LSTM model. Input observation: 2, output observation: 1.

IV. METHODS

A. PREDICTION OF NEXT ACTIVITY AND NEXT PERFORMER

As mentioned previously, we can apply the multivariate multi-step LSTM model to predict the next event because of the sequence characteristics of traces in an event log. Every trace or temporal work case [34] discovered in the event log is formed as shown in Figure 3. In a single trace, an event is followed by another event and each event contains information regarding its activity, performer, and event execution time. Therefore, we can consider the trace sequence as a multivariate multi-step problem. At a specific step, the activity and performer (or other resources) are observed variables. The next step is the result of finishing the current step.

We consider:

- ω is the event log recorded from a process-aware information system. n is the total number of independent activities in ω. m is the total number of individual performers in ω. t is the total number of traces. ($n \geq 1, m \geq 1, t \geq 1$).
- $A = \{A_i, i \in \overline{1,n}\}$ is the set of activities in the event log ω.
- $P = \{P_j, j \in \overline{1,m}\}$ is the set of performers in the event log ω. A_i was performed by P_j when the system was operated.
- $T = [T_k, k \in \overline{1,t}]$ is the list of traces.
  $T_k = [\varepsilon_l, l \in \overline{1, length(T_k)}, \varepsilon = (A_i \times P_j)]$ where $length(T_k)$ is the total number of events ε in trace T_k.

Figure 4 shows the proposed LSTM architecture. Before putting the input data to the architecture, we split ω into ω_{tr} and ω_{te} as the training and testing data sets, respectively. The procedure training the model to predict the next activity and next performer in a trace T_k from event log ω is as follows.

- *Step 1:* Choose the type of observed variable: θ_A (activity only), θ_P (performer only), or θ_{AP} (both activity and performer). Using θ_A we can only predict the next activity. Using θ_P, we can only predict the next performer. Using θ_{AP}, we can predict the next both activity and performer. Because the different system has different characteristics, depending on the training accuracy value on the event log, we can select the appropriateness observations.
- *Step 2:* Choose the number of input-steps (η_i), output-steps (η_o), where ($\eta_i, \eta_o \geq 1$, $\eta_i + \eta_o \leq length(T_k)$). $length(T_k)$ is the total events in trace T_k
- *Step 3:* Encode the observed variables using a one-hot encoding vector and then reshape it to the size of the total number of individual/independent observed variables to create a batch $\Delta_{k\_\eta\_io}$.
  For example (example 1), in an event log, there are 7 activities (named as START, A, B, C, D, E, END). The 'START' and 'END' are the 'virtual' values we added to the beginning and the ending of a trace. By adding these virtual values, we easily realize the beginning and the ending of a trace. And it's also useful if we want to visualize the traces in graph format.
  A discovered trace T_k in the log: (START $\rightarrow$ A $\rightarrow$ C $\rightarrow$ B $\rightarrow$ E $\rightarrow$ END). If we choose $\eta_i = 2, \eta_o = 1$ and observe at the activity (θ_A), then we have these observations:
  (START, A) $\rightarrow$ C
  (A, C) $\rightarrow$ B
  (C, B) $\rightarrow$ E
  (B, E) $\rightarrow$ END
  The one-hot encoding vector of A: [0 1 0 0 0 0 0]
  The one-hot encoding vector of B: [0 0 1 0 0 0 0]
  The one-hot encoding vector of C: [0 0 0 1 0 0 0]
  The encode value of (A, C) $\rightarrow$ B is:
  ([0 1 0 0 0 0 0], [0 0 0 1 0 0 0], [0 0 1 0 0 0 0])
  Reshape to get batch $\Delta_{k\_\eta\_io}$:
  ([0 1 0 0 0 0 0]
  [0 0 0 1 0 0 0]
  [0 0 1 0 0 0 0])
- *Step 4:* Transfer the batch $\Delta_{k\_\eta\_io}$ to the LSTM network.
- *Step 5:* Training the network. Iterate step 3) and step 4) over the training data set ω_{tr} to improve the weights and biases of the network.
- *Step 6:* Reshaping predictive values and then decoding to get predictive activities and performers.

After training the LSTM network, using ω_{te} to evaluate the model's accuracy and record the accuracy values for each type of observed variables to select the best parameters ($\theta_A, \theta_P, \theta_{AP}$) with ($\eta_i, \eta_o$).

B. PREDICTION OF VARIANTS OF A PROCESS-AWARE ENTERPRISE SOCIAL NETWORK

In this section, we present the solution and algorithm to predict different variants of a process-aware enterprise social network with the trained LSTM networks.

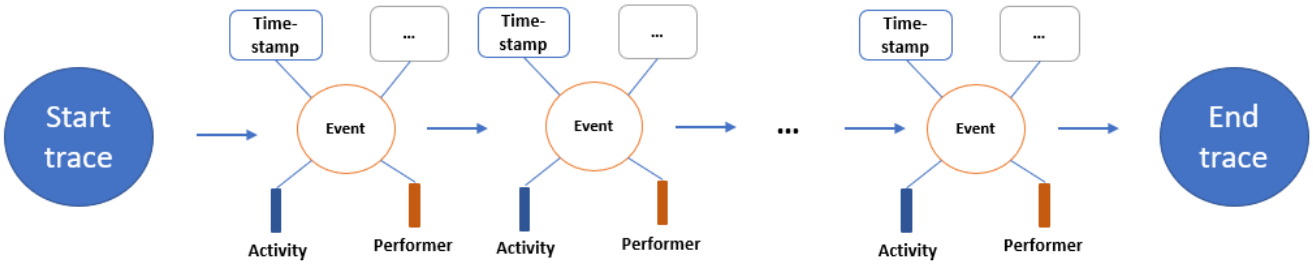


FIGURE 3. Event structure of a trace in an event log.

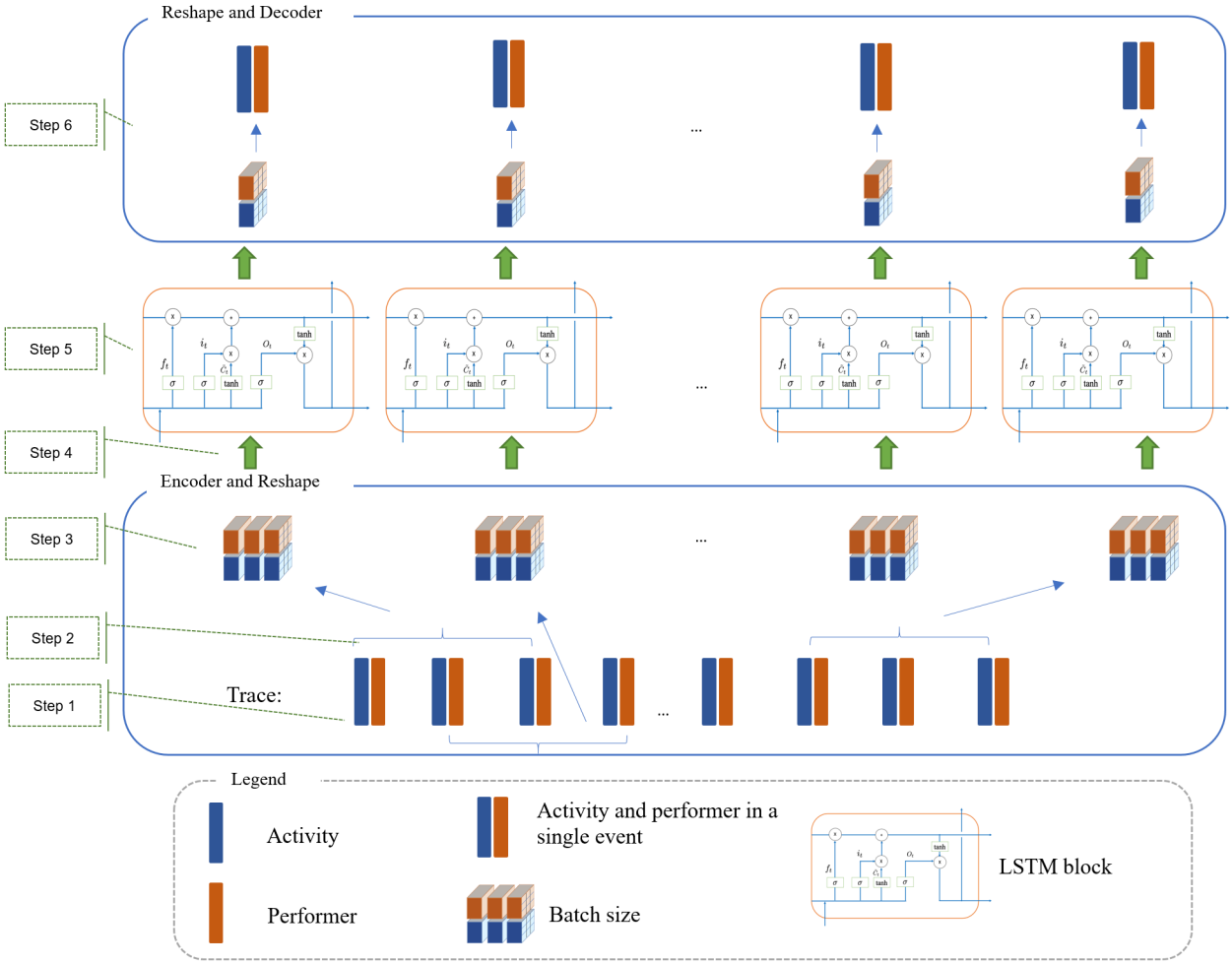


FIGURE 4. The proposed LSTM architecture for predicting next activity and next performer.

Process-Aware Enterprise Social Network Concept: Process-aware ESN is not a social network created from Internet social media such as Facebook or Twitter. It is a work transference network [4] or handover of work network [3] discovered from a process-aware system event log, which focuses on the relationships between performer and performer and can be used for social analysis (e.g., degree centrality, closeness centrality, etc.). In other words, it describes the relationships formed between two or more people who perform activities related to business processes within an organization.

Definition of Process-Aware ESN: *Let ω is the event log recorded from a process-aware information system.* $A = \{A_i, i \in \overline{1,n}\}$ *is the set of activities in* ω. $P = \{P_j, j \in \overline{1,m}\}$ *is the set of performers in* ω. $T = [T_k, k \in \overline{1,t}]$ *is the list of traces in* ω. $TRS = \{TR^{a}_{A_i,A_k} | (i,k) \in \overline{1,n}\}$ *is the set of transactions from activity* A_i *to activity* A_k, *and "a" indicates how many times the transaction occurred. The process-aware*

enterprise social network $ESN = \{RE^{b}_{P_j,P_l} | (j,l) \in \overline{1,m}\}$ *is the set of relationships between performer* P_j *and performer* P_l, *"b" indicates how many times the relationship occurred, where* P_j *performs* A_i *and* P_l *performs* A_k.

For example, we have a list of traces as following:

- $(A, P_1) \rightarrow (C, P_2) \rightarrow (B, P_3) \rightarrow (E, P_4) \rightarrow (D, P_3)$
- $(A, P_2) \rightarrow (B, P_1) \rightarrow (C, P_2) \rightarrow (D, P_3) \rightarrow (E, P_4)$
- $(A, P_2) \rightarrow (C, P_1) \rightarrow (C, P_2) \rightarrow (D, P_3) \rightarrow (E, P_4)$
- $(B, P_2) \rightarrow (A, P_3) \rightarrow (C, P_2) \rightarrow (D, P_1) \rightarrow (E, P_4)$
- $(B, P_1) \rightarrow (C, P_2) \rightarrow (C, P_1) \rightarrow (C, P_3) \rightarrow (D, P_4)$
- $(A, P_1) \rightarrow (B, P_2) \rightarrow (D, P_3) \rightarrow (C, P_4) \rightarrow (D, P_3)$

(A, P_1) means that activity A was performed by performer P_1. Then we have the ESN with the following relationships:

- $RE^{5}_{P_1,P_2}, RE^{1}_{P_1,P_3}, RE^{1}_{P_1,P_4}$,
- $RE^{4}_{P_2,P_1}, RE^{5}_{P_2,P_3}$,
- $RE^{1}_{P_3,P_2}, RE^{5}_{P_3,P_4}, RE^{2}_{P_4,P_3}$

To construct the predictive ESN, we predict the performer associated with an activity in the predictive traces by using the different parameters of $(\theta_A, \theta_P, \theta_{AP})$ with (η_i, η_o). Let $\Theta_{\theta_A}, \Theta_{\theta_P}, \Theta_{\theta_{AP}}$ be the trained LSTM networks, where the types of observed variables are θ_A, θ_P, and θ_{AP}, respectively. The predictive ESN will be created by combining these models in different ways:

- *Method 1:* Using the trained network Θ_{θ_A} to predict the next activity. Repeat the prediction until reaching the "END" state or the number of events in the predictive trace reaches a fixed value to get the predicted process $\widehat{T}_A$. Then using $\Theta_{\theta_{AP}}$ with $\widehat{T}_A$ as input to predict the associated performers in activities in $\widehat{T}_A$ to get the predicted process $\widehat{T}_{AP}$.
- *Method 2:* Using the trained network $\Theta_{\theta_{AP}}$ to predict both the next activity and performer. Repeat the prediction until reaching the "END" state or the number of events in the predictive trace reaches a fixed value to get the predicted process $\widehat{T}_{AP}$.
- *Method 3:* Using the trained network Θ_{θ_P} to predict the next performer. Repeat the prediction until reaching the "END" state or the number of event in the predictive trace reaches a fixed value to get the predicted process $\widehat{T}_P$.

In the above methods, after getting predicted traces $\widehat{T}_{AP}$, we will have the list of predicted future processes with the association of performers and hence receiving the predictive ESN.

The prediction in LSTM selects a possible next value from a list of possibilities. In the naive approach, we often choose the highest probability to achieve the best accuracy. However, the selection of the best value usually creates a loop such that we cannot reach the end state of a trace instance. Camargo *et al.* [19] solved this problem by selecting a random probability for the next possible case. This is a better solution, but sometimes can be worse if a very low probability is randomly chosen.

Our approach to this problem is to use a threshold φ and recursive function *PredictWithThreshold*. By using the threshold, a predicted value with a higher probability than φ will be selected. So that selecting a different threshold with a different number of input-step and output-step (η_i, η_o) will create a different variant of ESN. With each ESN variant, we obtain different views of the relationship between performer and performer while performing an activity in the system in the future. So a question was raised as how to choose a threshold value on each data set? We take into consideration this concern on some real-life data sets in the next section.

The pseudocode in Algorithm 1 outlines our approach for predicting the ESN by using **Method 1** described above. The algorithms for **method 2** and **method 3** are similar to algorithm 1 (by replacing the observed variables and trained models), so we do not describe it here. In Algorithm 1, the block from line number 4 to line number 10 is to get the predictive process $\widehat{T}_A$ using the trained network $\Theta_{\theta A}$. The block from line number 11 to line number 16 is to predict the associated performers in activities in $\widehat{T}_A$. After that, all predictive $\widehat{T}_A$ will be combined to create predictive future processes $\widehat{\omega_p}$ (line 17). Finally, the predictive ESN will be discovered by using the function GettingESN (line 18). Aside from the two blocks mentioned above, there are 4 supporting functions in the algorithm, i.e. Encoder, Decoder, PredictWithThreshold, and GettingESN function.

- **Encoder** function: This function is responsible for encoding the information of activities, performers to one-hot encoding vector format.
- **Decoder** function: This function is responsible for decoding the one-hot encoding vector to the original format (string) of activities and performers.
- **PredictWithThreshold** function: This function repeats predicting the next activity (with threshold φ) in the current input trace. This is a recursive function, it stops only when the predicted value is 'END' or the predicted number of events reaches a given α value. The value of α is determined based on the ω_{tr} training data set.
  For example (example 2). We have an initial input trace $T_x = (\text{START} \rightarrow \text{A})$; $(\eta_i, \eta_o) = (2,1)$; $\varphi = 0.15$; $\alpha = 7$; trained network $\Theta_{\theta A}$.
  Using the *PredictWithThreshold* function, we have the list of potential values $\widehat{potentialList}$ = [START(0.003), A(0.001), B(0.352), C(0.465), E(0.048), END(0.131)].
  Because $\varphi = 0.15 \Rightarrow$ [B, C] will be selected $\Rightarrow$ We have two new input traces (START $\rightarrow$ A $\rightarrow$ B) and (START $\rightarrow$ A $\rightarrow$ C). Then two ($\eta_i = 2$) last activities of these traces will be used for predicting the next potential activity (i.e., (A$\rightarrow$ B) and (A$\rightarrow$ C)).
  This procedure will be repeated until the probability of activity 'END' in the potential list is greater than or equal to φ (0.15) or the number of events in the input trace is equal to α (7)
- **GettingESN** function: This function discovers the relationships between performers and performers from the predictive future process $\widehat{\omega_p}$ for getting the predictive ESN.

Algorithm 1 Predicting Process-Aware Enterprise Social Network

Input: List of input trace L_T from event log ω_{te}, number of step (η_i, η_o), trained LSTM Θ_{θ_A}, $\Theta_{\theta_{AP}}$, threshold φ, maximum number of event in trace α.

Output: Process-aware enterprise social network $\widehat{ESN}$

$\widehat{\kappa} = \emptyset$, $\widehat{\omega_p} = \emptyset$, $\widehat{ESN} = \emptyset$
$dict\_encode\theta_A = \Theta_{\theta_A}$.GetEncodeDict(), $dict\_decode\theta_A = \Theta_{\theta_A}$.GetDecodeDict()
$dict\_encode\theta_{AP} = \Theta_{\theta_{AP}}$.GetEncodeDict(), $dict\_decode\theta_{AP} = \Theta_{\theta_{AP}}$.GetDecodeDict()
for ($T \in L_T$) **do**
    $T_A = T$.GetInputActivities(η_i) ▷ Get initial input activity for prediction
    T_A^e = Encoder(T_A, $dict\_encode\theta_A$) ▷ Encode T_A to one-hot encoding vector using $dict\_encode\theta_A$
    $T_A^e = T_A^e$.reshape(η_i, $dict\_encode\theta_A$.size())
    while ($\widehat{T}_A^e$ = PredictWithThreshold($\Theta_{\theta A}$, T_A^e, η_i, φ, α) ! = None) **do** ▷ Get predictive value with threshold φ
        $\widehat{T}_A$ = Decoder($\widehat{T}_A^e$, $dict\_decode\theta_A$) ▷ Decode predictive value from one-hot encoding vector
        $\widehat{\kappa}$.append($\widehat{T}_A$)
for ($\widehat{T} \in \widehat{\kappa}$) **do**
    $\widehat{T}_A = \widehat{T}$.GetAllActivities()
    $\widehat{T}_A^e$ = Encoder($\widehat{T}_A$, $dict\_encode\theta_{AP}$)
    $\widehat{T}_A^e = \widehat{T}_A^e$.reshape($\eta_i$, $dict\_encode\theta_{AP}$.size())
    $\widehat{T}_{AP}^e = \Theta_{\theta_{AP}}$.predict($\widehat{T}_A^e$)
    $\widehat{T}_{AP}$ = Decoder($\widehat{T}_{AP}^e$, $dict\_decode\theta_{AP}$)
    $\widehat{\omega_p}.append(\widehat{T}_{AP})$
$\widehat{ESN}$ = GettingESN($\widehat{\omega_p}$)
return $\widehat{ESN}$

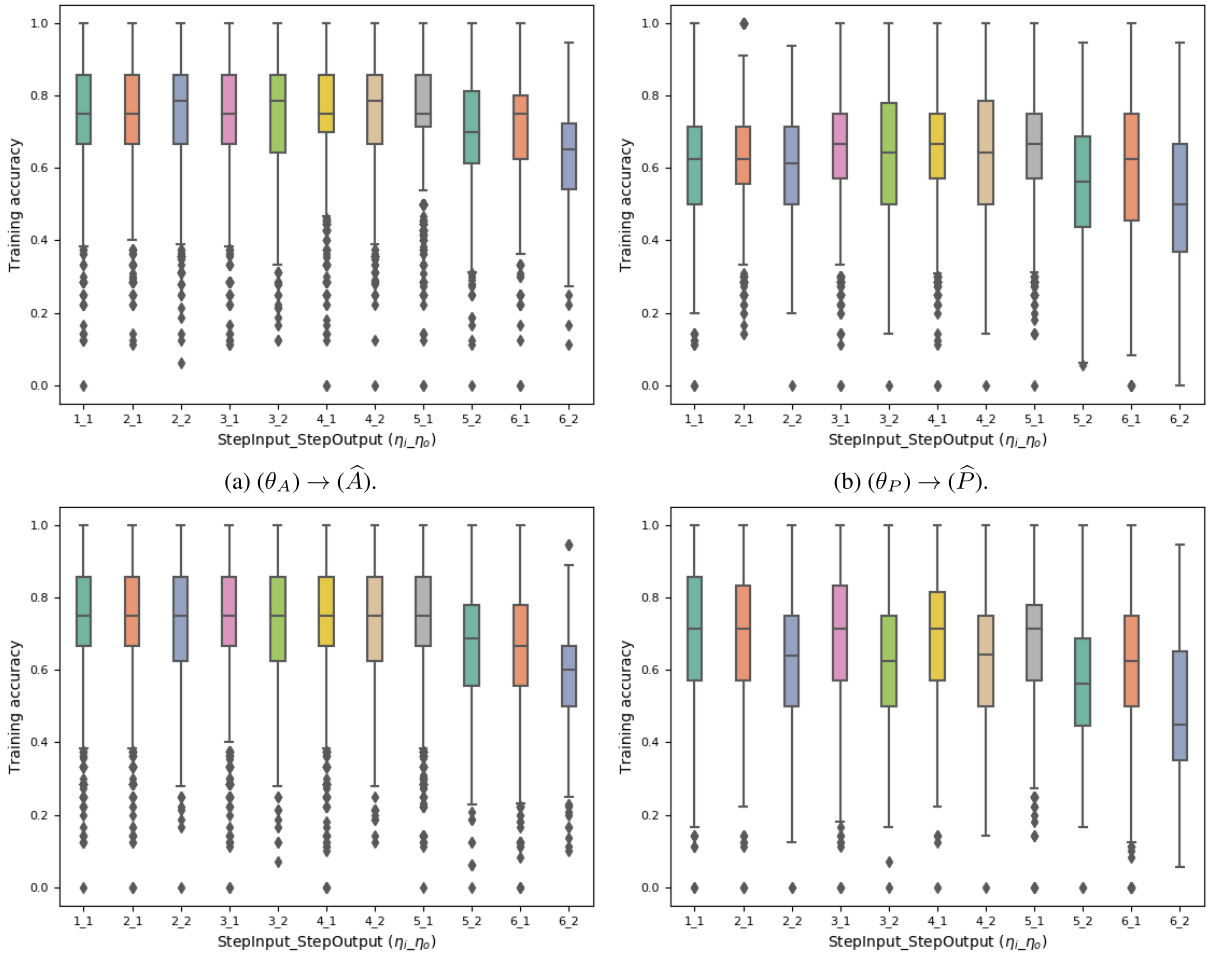
function ENCODER(T_x, $dict\_encode$)
    $T^e = \emptyset$
    for ($element \in T_x$) **do**
        $element^e = dict\_encode.GetValue(element)$ ▷ Get encode value of the key $element$ in dict
        $T^e.append(element^e)$
    return T^e
function DECODER(T_x^e, $dict\_decode$)
    $T_x = \emptyset$
    for ($element^e \in T_x^e$) **do**
        $element = dict\_decode.GetKey(element^e)$ ▷ Get key of the encode value $element^e$ in dict
        $T_x.append(element)$
    return T_x
function PREDICTWITHTHRESHOLD(Θ, T_x^e, η_i, φ, α)
    if T_x^e reached 'END' or length of $T_x^e > \alpha$ **then**
        return T_x^e
    $currentInput^e = T_x^e[-\eta_i :]$ ▷ Get the last η_i elements of T_x^e
    $\widehat{potentialList}^e = \Theta$.predict() ▷ Get list of potential values with its probabilities
    for ($element^e \in \widehat{potentialList}^e$) **do**
        if ($element^e$.GetProbability $\geq \varphi$) **then**
            $T_x' = currentInput^e$.append($element^e$)
            PredictWithThreshold(Θ, T_x', η_i, φ, α)
    return None
function GETTINGESN($\widehat{\omega_p}$)
    $\widehat{ESN} = \emptyset$
    for ($T \in \widehat{\omega_p}$) **do**
        for (i = 0; i < length(T) − 1; i++) **do**
            currentPerformer = $T_i.GetPerformer()$, nextPerformer = $T_{i+1}.GetPerformer()$
            $\widehat{ESN}.append(currentPerformer \rightarrow nextPerformer)$ ▷ Add performer relationship
    return $\widehat{ESN}$

TABLE 1. The content of discovered traces in the event log ω in example 3.

TraceID	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7
1	START,START	A,p1	B,p2	C,p3	D,p4	E,p5	END,END
2	START,START	A,p2	C,p2	B,p4	E,p5	END,END	
3	START,START	C,p2	D,p2	E,p5	END,END		
4	START,START	A,p1	C,p1	D,p3	E,p6	END,END	
5	START,START	B,p2	C,p1	D,p3	E,p6	END,END	
6	START,START	A,p1	D,p2	C,p3	D,p5	E,p4	END,END
7	START,START	A,p1	C,p2	C,p3	D,p4	E,p5	END,END
8	START,START	A,p1	B,p2	B,p3	C,p4	E,p5	END,END
9	START,START	B,p2	B,p2	E,p5	END,END		
10	START,START	C,p2	C,p2	D,p3	E,p5	END,END	

TABLE 2. The brief information of input data sets.

Data set	Traces	Events	Event per trace			Activity	Performer
			Min	Max	Average		
Helpdesk	4,580	21,348	4	17	4.66	14	22
BPI 2012	13,087	262,200	5	177	20.04	24	68
BPI 2015 - *Municipality 1*	1,199	52,217	2	101	43.5	398	23
BPI 2015 - *Municipality 2*	832	44,354	1	132	53.3	410	11
BPI 2017	31,509	1,202,267	12	182	38.1	26	149



(a) $(\theta_A) \rightarrow (\widehat{A})$.

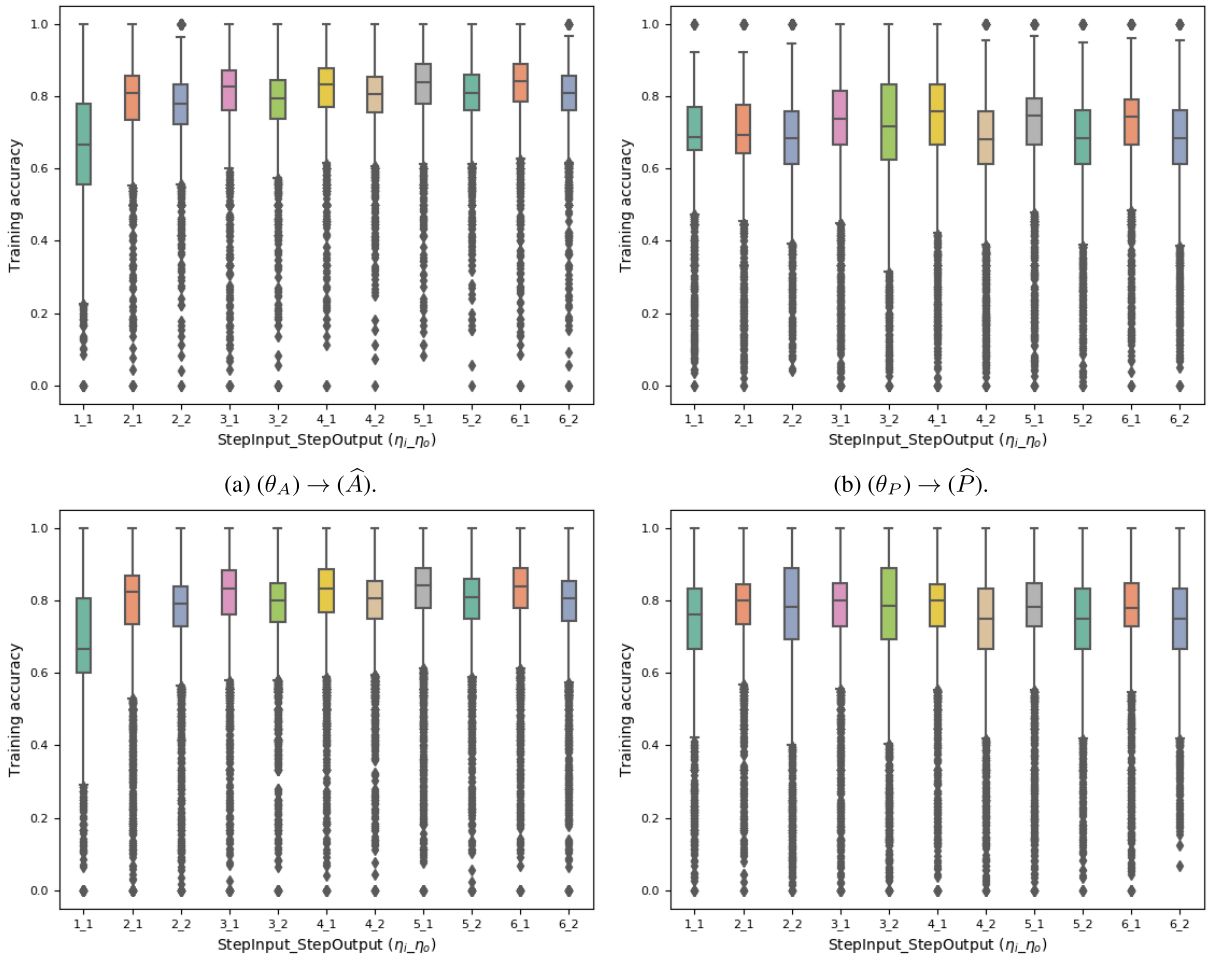
(b) $(\theta_P) \rightarrow (\widehat{P})$.

(c) $(\theta_{AP}) \rightarrow (\widehat{A})$.

(d) $(\theta_{AP}) \rightarrow (\widehat{P})$.

FIGURE 5. Training accuracy on the ***Helpdesk*** data set.

Let's run Algorithm 1 with a simple example **(example 3)**. In this example, there are 7 activities (i.e., START, A, B, C, D, E, END), and there are 8 performers (i.e., START, p1, p2, p3, p4, p5, p6, END) in the event log ω. The traces in ω are discovered as shown in Table 1. In the table, the structure (A,p1) means that activity A was performed by performer p1. As getting data from the Table, the maximum number of events in a trace is $\alpha = 7$. Assume that we trained LSTM networks with $(\eta_i, \eta_o) = (2,1)$ on the first 7 traces in Table 1 by using **Method 1** and hence getting the trained LSTM $\Theta_{\theta A}$



(a) $(\theta_A) \rightarrow (\widehat{A})$.

(b) $(\theta_P) \rightarrow (\widehat{P})$.

(c) $(\theta_{AP}) \rightarrow (\widehat{A})$.

(d) $(\theta_{AP}) \rightarrow (\widehat{P})$.

FIGURE 6. **Training accuracy on the *BPI 2012* data set.**

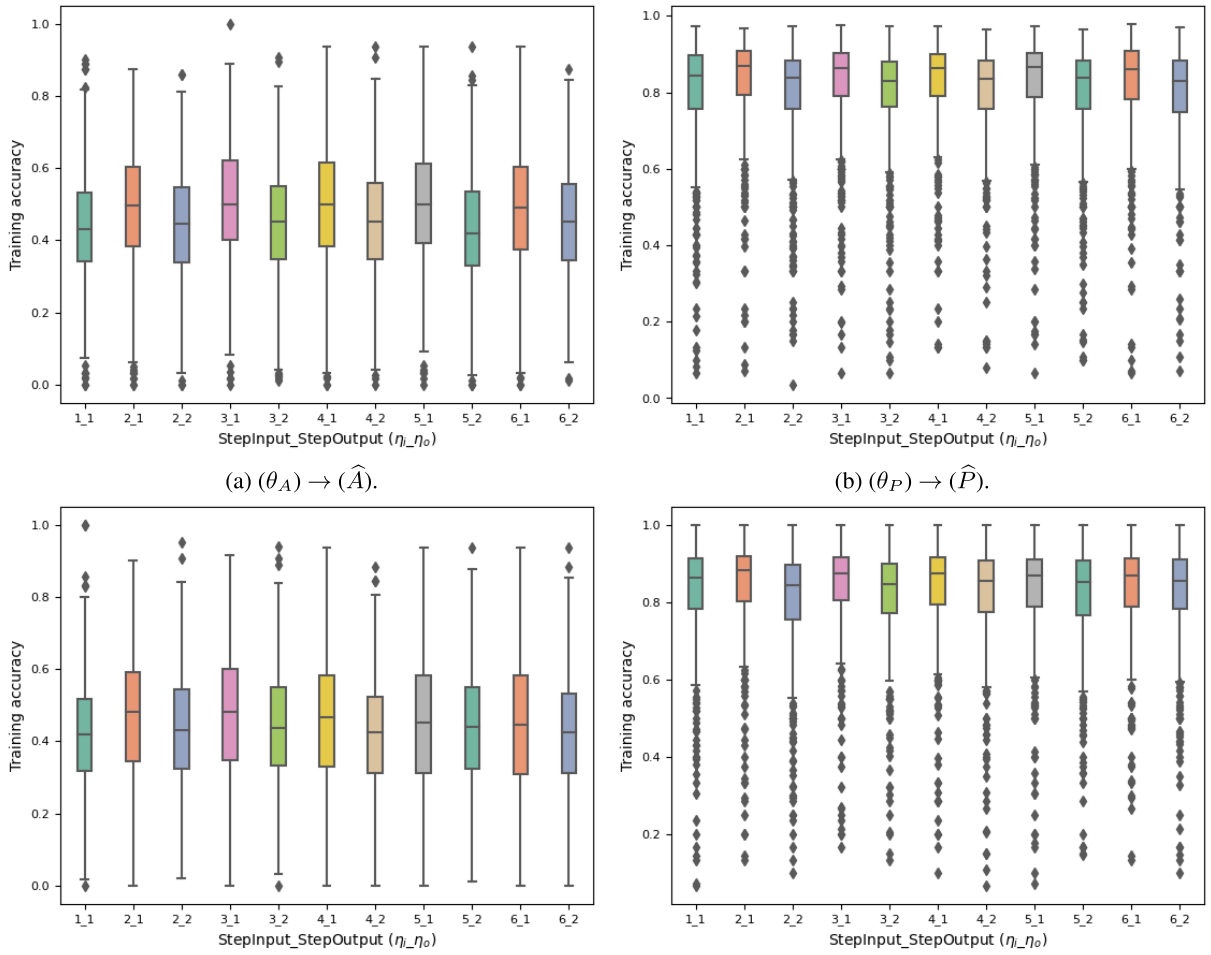
and $\Theta_{\theta_{AP}}$. The dictionaries of one-hot vectors of $\Theta_{\theta A}$ and $\Theta_{\theta_{AP}}$ are as follows.

- Dictionary of $\Theta_{\theta A}$:
  START $\leftrightarrow$ [1 0 0 0 0 0 0]
  A $\leftrightarrow$ [0 1 0 0 0 0 0]
  B $\leftrightarrow$ [0 0 1 0 0 0 0]
  C $\leftrightarrow$ [0 0 0 1 0 0 0]
  D $\leftrightarrow$ [0 0 0 0 1 0 0]
  E $\leftrightarrow$ [0 0 0 0 0 1 0]
  END $\leftrightarrow$ [0 0 0 0 0 0 1]
- Dictionary of $\Theta_{\theta_{AP}}$:
  START $\leftrightarrow$ [1 0 0 0 0 0 0 0 0 0 0 0 0 0]
  A $\leftrightarrow$ [0 1 0 0 0 0 0 0 0 0 0 0 0 0]
  B $\leftrightarrow$ [0 0 1 0 0 0 0 0 0 0 0 0 0 0]
  C $\leftrightarrow$ [0 0 0 1 0 0 0 0 0 0 0 0 0 0]
  D $\leftrightarrow$ [0 0 0 0 1 0 0 0 0 0 0 0 0 0]
  E $\leftrightarrow$ [0 0 0 0 0 1 0 0 0 0 0 0 0 0]
  p1 $\leftrightarrow$ [0 0 0 0 0 0 1 0 0 0 0 0 0 0]
  p2 $\leftrightarrow$ [0 0 0 0 0 0 0 1 0 0 0 0 0 0]
  p3 $\leftrightarrow$ [0 0 0 0 0 0 0 0 1 0 0 0 0 0]
  p4 $\leftrightarrow$ [0 0 0 0 0 0 0 0 0 1 0 0 0 0]
  p5 $\leftrightarrow$ [0 0 0 0 0 0 0 0 0 0 1 0 0 0]
  p6 $\leftrightarrow$ [0 0 0 0 0 0 0 0 0 0 0 1 0 0]
  END $\leftrightarrow$ [0 0 0 0 0 0 0 0 0 0 0 0 1 0]
  END $\leftrightarrow$ [0 0 0 0 0 0 0 0 0 0 0 0 0 1]

Now, using the first ($\eta_i = 2$) events on trace ID 8, 9 and 10, we have a list of input data for predicting the ESN:

- TraceID 8: [(START,START) $\rightarrow$ (A,p1)] $\rightarrow$?
- TraceID 9: [(START,START) $\rightarrow$ (B,p2)] $\rightarrow$?
- TraceID 10: [(START,START) $\rightarrow$ (C,p2)] $\rightarrow$?

Choose $\varphi = 0.15$, the block from line number 4 to line number 10 in the algorithm will produce the predictive process $\widehat{T_A}$ as follows (for convenient reading, we do not convert



(a) $(\theta_A) \rightarrow (\widehat{A})$. (b) $(\theta_P) \rightarrow (\widehat{P})$.

(c) $(\theta_{AP}) \rightarrow (\widehat{A})$. (d) $(\theta_{AP}) \rightarrow (\widehat{P})$.

FIGURE 7. Training accuracy on the ***BPI 2015 - Municipality 1*** data set.

activities and performers to one-hot encoding vectors).

$$\left.\begin{array}{c}(START) \rightarrow (A) \\ \Theta_{\theta_A}\end{array}\right\} \Rightarrow \left.\begin{array}{c}(START) \rightarrow (A) \rightarrow (B) \\ (START) \rightarrow (A) \rightarrow (C) \\ \Theta_{\theta_A}\end{array}\right\}$$

$$\Rightarrow \ldots$$

$$\Rightarrow \left\{\begin{array}{c}(START) \rightarrow (A) \rightarrow (B) \rightarrow (C) \rightarrow \ldots \rightarrow (END) \\ (START) \rightarrow (A) \rightarrow (C) \rightarrow (D) \rightarrow \ldots \rightarrow (END) \\ (START) \rightarrow (A) \rightarrow (B) \rightarrow (E) \ldots \rightarrow (END) \\ \ldots \\ (START) \rightarrow (A) \rightarrow \ldots \rightarrow (END)\end{array}\right.$$

After getting the predictive process $\widehat{T_A}$, the block from line number 11 to line number 17 in the algorithm will create predictive future processes $\widehat{\omega_p}$ like this.

$$\left.\begin{array}{c}(START) \rightarrow (A) \rightarrow (B) \rightarrow (C) \rightarrow \ldots \rightarrow (END) \\ (START, START) \rightarrow (A, p1) \\ \Theta_{\theta_{AP}}\end{array}\right\}$$

$$\Rightarrow \left\{\begin{array}{l}(START, START) \rightarrow (A, p1) \rightarrow (B, p2) \\ \rightarrow (C, p3) \rightarrow \ldots \rightarrow (END, END)\end{array}\right.$$

$$\left.\begin{array}{c}(START) \rightarrow (A) \rightarrow (C) \rightarrow (D) \rightarrow \ldots \rightarrow (END) \\ (START, START) \rightarrow (A, p1) \\ \Theta_{\theta_{AP}}\end{array}\right\}$$

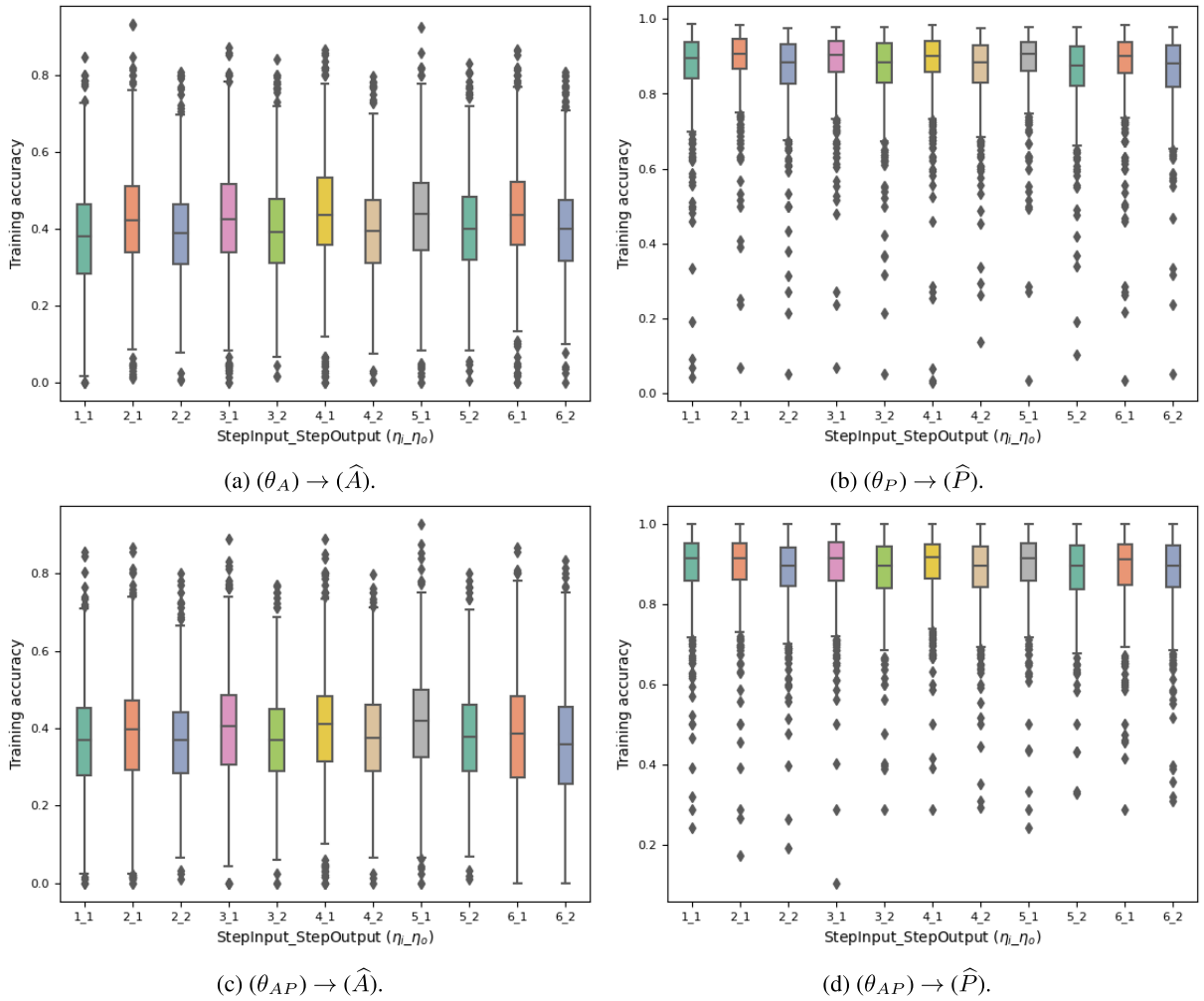
$$\Rightarrow \left\{\begin{array}{l}(START, START) \rightarrow (A, p1) \rightarrow (C, p1) \\ \rightarrow (D, p4) \rightarrow \ldots \rightarrow (END, END)\end{array}\right.$$

Lastly, the predictive ESN will be discovered by using the function *GETTINGESN* with $\widehat{\omega_p}$ as the input parameter.

V. EXPERIMENTAL RESULTS

We conduct the experiment on 5 real-life data sets. Three data sets were used in the studies in [15], [17]–[19] and two data sets were provided in BPI challenge 2015 [37]. The main purpose of the experiment in this section is as follows:

- Evaluate the efficiency of the proposed architecture and algorithm in predicting the next activity and performer.
- Compare our approach's results with other published studies.
- Address the question: How to choose the suitable models and parameters for different event logs.



(a) $(\theta_A) \rightarrow (\widehat{A})$.

(b) $(\theta_P) \rightarrow (\widehat{P})$.

(c) $(\theta_{AP}) \rightarrow (\widehat{A})$.

(d) $(\theta_{AP}) \rightarrow (\widehat{P})$.

FIGURE 8. Training accuracy on the *BPI 2015 - Municipality 2* data set.

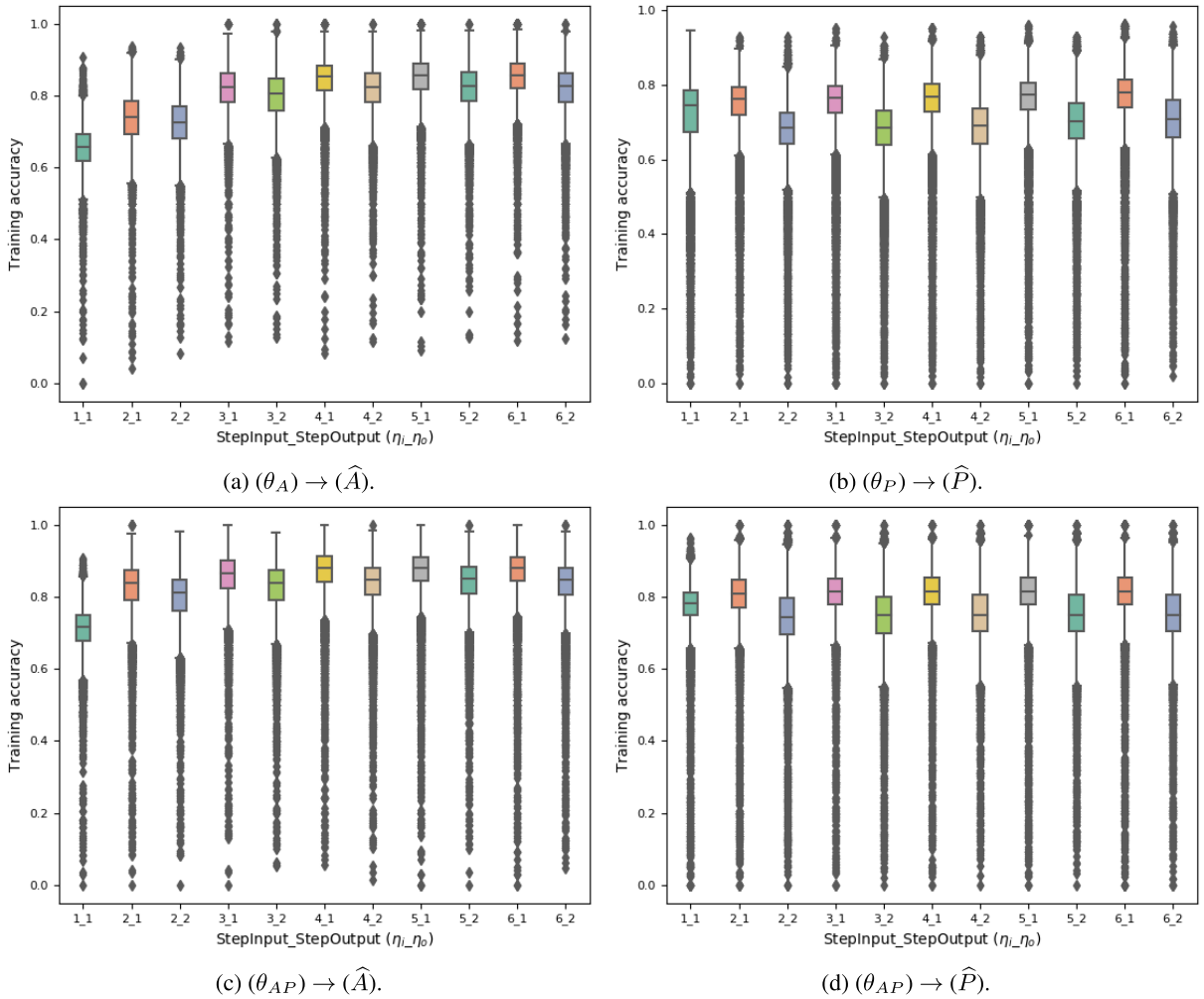
- Evaluate the relationship between the threshold value and predictive results.
- Visualize the predicted ESN.

We also use the same chronological order of the data sets for training and evaluating as in [15], [17]–[19], i.e., the first two-thirds of the data sets for training and the remainder for evaluating. The brief information of the data sets is shown in Table 2 and as follows:

- *Helpdesk [35]:* This data set contains information from a ticketing management process belonging to an Italian software company.
- *BPI 2012 [36]:* This data set is an application process for a personal loan or overdraft within a global financing organization taken from a Dutch financial institute.
- *BPI 2015 [37]:* The business process intelligence challenge 2015 provided 5 real-life data sets in five Dutch municipalities. Each data set described all building permit applications over about four years in a different municipality in Dutch. In this study, we select two data sets for conducting our experiment (The **Municipality 1** [38] and **Municipality 2** [39]).
- *BPI 2017 [40]:* This data set pertains to a loan application process used by a Dutch financial institute and contains all applications filed through an online system.

For each trace in the data sets, we observe both the activities and performers and select $\eta_i \in [1, 6]$ and $\eta_o \in [1, 2]$. The reason we choose the number of the input-step $\eta_i \in [1, 6]$ and the number of the output-step $\eta_o \in [1, 2]$ is that, in the process-aware information systems (which generated event logs), activities are performed based on the planned workflow models, in that, the decision of operating activities in the current step are usually based on the roles and results of some recent activities. To predict a full trace, we use the current prediction value as input data for the next prediction step and repeat this action until reaching the 'END' state.

To build and train the LSTM models, we use Keras API (version 2.3.1) [41]. The training accuracy is being measured by using the integrated accuracy function in Keras (this function calculates the accuracy value by dividing the total by



(a) $(\theta_A) \rightarrow (\widehat{A})$. (b) $(\theta_P) \rightarrow (\widehat{P})$.

(c) $(\theta_{AP}) \rightarrow (\widehat{A})$. (d) $(\theta_{AP}) \rightarrow (\widehat{P})$.

FIGURE 9. Training accuracy on the *BPI 2017* data set.

count). To validate the prediction values, we use the same methodology as the function in Keras. That is counting the total correct predicted value and then dividing the total by count. For example, if the total activities need to predict is 85 and the number of exactly predicted activity is 56, the accuracy value is $\frac{56}{85} \approx 0.658$.

A. PREDICTIVE NEXT ACTIVITY AND NEXT PERFORMER

Figures 5, 6, 7, 8, and 9 show the comparisons in training accuracy among the different combinations of observations for activities, performers, and (η_i, η_o). In these figures:

- $(\theta_A) \rightarrow (\widehat{A})$: observing activity to predict activity.
- $(\theta_P) \rightarrow (\widehat{P})$: observing performer to predict performer.
- $(\theta_{AP}) \rightarrow (\widehat{A})$: observing both activity and performer to predict activity.
- $(\theta_{AP}) \rightarrow (\widehat{P})$: observing both activity and performer to predict performer.

Based on these figures (5, 6, 7, 8, and 9), some comments with respect to training LSTM networks with the data sets can be made:

- **Helpdesk** data set: predicting activities yields similar accuracy for training values with (1,1), (2,1), (2,2), and (3,1). Meanwhile, predicting performers yields different training accuracies, and the best training accuracy is achieved when observing activities and performers with (1,1).
- **BPI 2012** data set: predicting activities with (1,1) yields the lowest training accuracy and the best results with (4,1), (5,1), and (6,1). Predicting performers yields the highest training accuracy when observing both activities and performers with (2,2) and (3,2).
- **BPI 2015 - Municipality 1** data set: Predicting activities yields the lower accuracy comparing to the other data sets. The training accuracy values are concentrated in the range [0.3 - 0.6] and have median values of approximately 0.4. Meanwhile, the training accuracy values for predicting performers yield the best values at (2,1), and (6,1) with median values of approximately 0.85.
- **BPI 2015 - Municipality 2** data set: Like the Municipality 1 data set, the operation of predicting activities

TABLE 3. The validation values with different trained LSTM networks.

Helpdesk					BPI 2012				
$\eta_i\_\eta_o$	$\theta_A \to \widehat{A}$	$\theta_{AP} \to \widehat{A}$	$\theta_P \to \widehat{P}$	$\theta_{AP} \to \widehat{P}$	$\eta_i\_\eta_o$	$\theta_A \to \widehat{A}$	$\theta_{AP} \to \widehat{A}$	$\theta_P \to \widehat{P}$	$\theta_{AP} \to \widehat{P}$
1_1	0.819	0.763	0.258	0.523	1_1	0.534	0.597	**0.694**	0.674
2_1	0.691	0.859	0.455	0.616	2_1	0.717	0.779	0.686	**0.823**
2_2	0.613	0.755	0.081	0.263	2_2	0.407	0.465	0.428	0.509
3_1	0.836	0.873	0.472	0.056	3_1	0.720	0.739	0.612	0.817
3_2	0.114	0.664	0.208	0.380	3_2	0.461	0.452	0.267	0.357
4_1	0.869	0.931	0.641	0.802	4_1	**0.863**	**0.873**	0.465	0.799
4_2	0.868	0.819	0.121	0.164	4_2	0.612	0.639	0.364	0.487
5_1	0.928	**0.969**	**0.875**	**0.912**	5_1	0.805	0.811	0.668	0.775
5_2	0.568	0.502	0.185	0.420	5_2	0.635	0.644	0.403	0.573
6_1	**0.960**	0.953	0.856	0.903	6_1	0.856	0.871	0.662	0.799
6_2	0.663	0.271	0.064	0.360	6_2	0.694	0.736	0.351	0.493
BPI 2015 - Municipality 1					**BPI 2015 - Municipality 2**				
$\eta_i\_\eta_o$	$\theta_A \to \widehat{A}$	$\theta_{AP} \to \widehat{A}$	$\theta_P \to \widehat{P}$	$\theta_{AP} \to \widehat{P}$	$\eta_i\_\eta_o$	$\theta_A \to \widehat{A}$	$\theta_{AP} \to \widehat{A}$	$\theta_P \to \widehat{P}$	$\theta_{AP} \to \widehat{P}$
1_1	**0.108**	**0.171**	0.680	0.690	1_1	0.237	0.195	0.785	0.753
2_1	0.099	0.086	0.855	**0.853**	2_1	0.236	0.270	**0.787**	0.800
2_2	0.060	0.065	0.804	0.669	2_2	0.117	0.099	0.724	0.609
3_1	0.091	0.145	0.810	0.782	3_1	**0.275**	**0.278**	0.685	0.780
3_2	0.075	0.052	0.701	0.701	3_2	0.115	0.099	0.432	0.641
4_1	0.088	0.101	**0.868**	0.837	4_1	0.257	0.261	0.705	0.826
4_2	0.037	0.031	0.789	0.715	4_2	0.114	0.101	0.437	0.705
5_1	0.091	0.131	0.855	0.759	5_1	0.255	0.266	0.582	**0.828**
5_2	0.047	0.043	0.789	0.716	5_2	0.109	0.093	0.578	0.619
6_1	0.079	0.108	0.759	0.794	6_1	0.235	0.239	0.668	0.624
6_2	0.027	0.026	0.802	0.610	6_2	0.095	0.085	0.636	0.555
BPI 2017									
$\eta_i\_\eta_o$	$\theta_A \to \widehat{A}$	$\theta_{AP} \to \widehat{A}$	$\theta_P \to \widehat{P}$	$\theta_{AP} \to \widehat{P}$					
1_1	0.658	0.705	0.548	0.610					
2_1	0.751	0.821	**0.658**	0.668					
2_2	0.590	0.674	0.368	0.491					
3_1	0.832	0.868	0.645	**0.673**					
3_2	0.686	0.718	0.156	0.492					
4_1	0.861	**0.881**	0.612	0.664					
4_2	0.723	0.715	0.234	0.488					
5_1	0.853	0.857	0.630	0.661					
5_2	0.546	0.728	0.367	0.480					
6_1	**0.867**	0.871	0.596	0.656					
6_2	0.720	0.724	0.333	0.478					

TABLE 4. The comparison of different approaches.

Data set	Tax et al. [17]	Camargo et al. [19]	Lin et al. [18]	Our approach
Helpdesk	0.712	0.789	0.916	**0.960**
BPI 2012	0.760	0.786	**0.974**	0.863
BPI 2017			**0.974**	0.867

TABLE 5. The predicted ESN with $(\eta_i, \eta_o) = (1, 1)$ combined with different thresholds φ in viewing performer relationship.

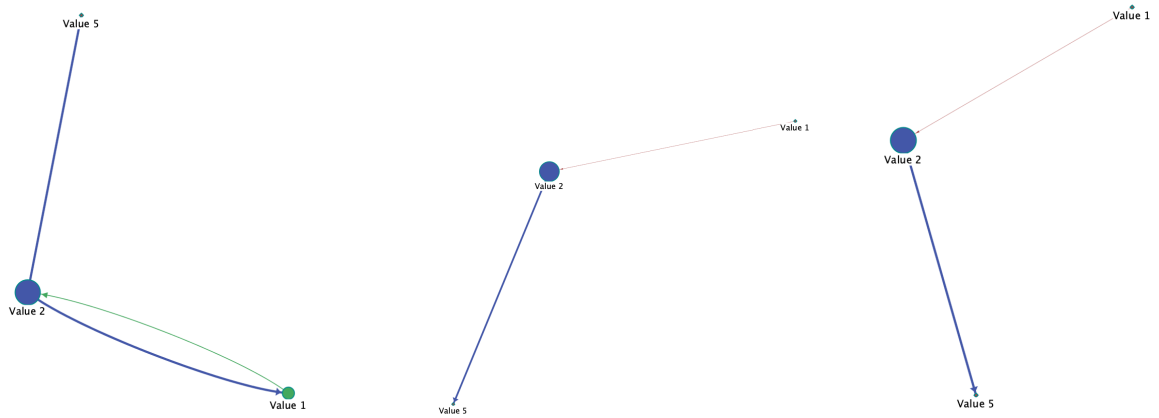
$\varphi = 0.05$			$\varphi = 0.10$			$\varphi = 0.15$		
From performer	To performer	Count	From performer	To performer	Count	From performer	To performer	Count
Value 1	Value 2	1698	Value 1	Value 2	1	Value 1	Value 2	1
Value 2	Value 1	1248	Value 2	Value 5	3	Value 2	Value 5	7
Value 2	Value 5	18650						

yields low accuracy. Meanwhile, predicting performers achieve the best values with median values of approximately 0.9 at (1,1), and (3,1).

- **BPI 2017** data set: predicting activities achieves the best training with (4,1), (5,1), and (6,1), and observing using both activities and performers produces better results than just observing activities. Predicting performers yields better training accuracy results than the previous two data sets.

Table 3 presents the validation results on the last one-third of the data sets. As shown in the table, the observations that achieve the best values on each data set are as follows.

- **Helpdesk** data set: observing both activity and performer with a 5-step input to predict a 1-step output of the activity or performer.
- **BPI 2012** data set: observing both activity and performer with a 4-step input to predict a 1-step output of the activity or observing both activity and performer with a 2-step input to predict a 1-step output of the performer.
- **BPI 2015 - Municipality 1** data set: observing performer with a 4-step input to predict a 1-step output of the performer or observing both activity and performer with a 2-step input to predict a 1-step output of the performer.

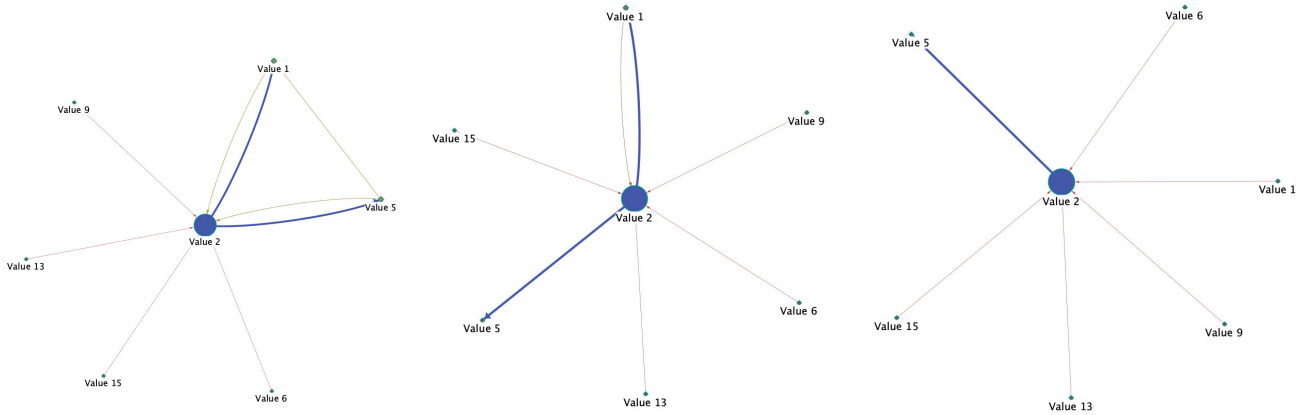


(a) $\varphi = 0.05$ (b) $\varphi = 0.10$ (c) $\varphi = 0.15$

FIGURE 10. Predicted ESN with $(\eta_I, \eta_O) = (1, 1)$.

TABLE 6. The predicted ESN with $(\eta_I, \eta_O) = (2, 1)$ combined with different thresholds φ in viewing performer relationship.

$\varphi = 0.05$			$\varphi = 0.10$			$\varphi = 0.15$		
From performer	**To performer**	**Count**	**From performer**	**To performer**	**Count**	**From performer**	**To performer**	**Count**
Value 5	Value 2	79	Value 9	Value 2	3	Value 1	Value 2	12
Value 5	Value 1	110	Value 6	Value 2	3	Value 9	Value 2	12
Value 2	Value 1	2432	Value 15	Value 2	164	Value 15	Value 2	33
Value 6	Value 2	7184	Value 13	Value 2	166	Value 13	Value 2	34
Value 1	Value 2	7770	Value 2	Value 5	343	Value 6	Value 2	35
Value 15	Value 2	9235	Value 2	Value 1	580	Value 2	Value 5	126
Value 9	Value 2	9780	Value 1	Value 2	587			
Value 13	Value 2	10601						
Value 2	Value 5	40179						



(a) $\varphi = 0.05$ (b) $\varphi = 0.10$ (c) $\varphi = 0.15$

FIGURE 11. Predicted ESN with $(\eta_I, \eta_O) = (2, 1)$.

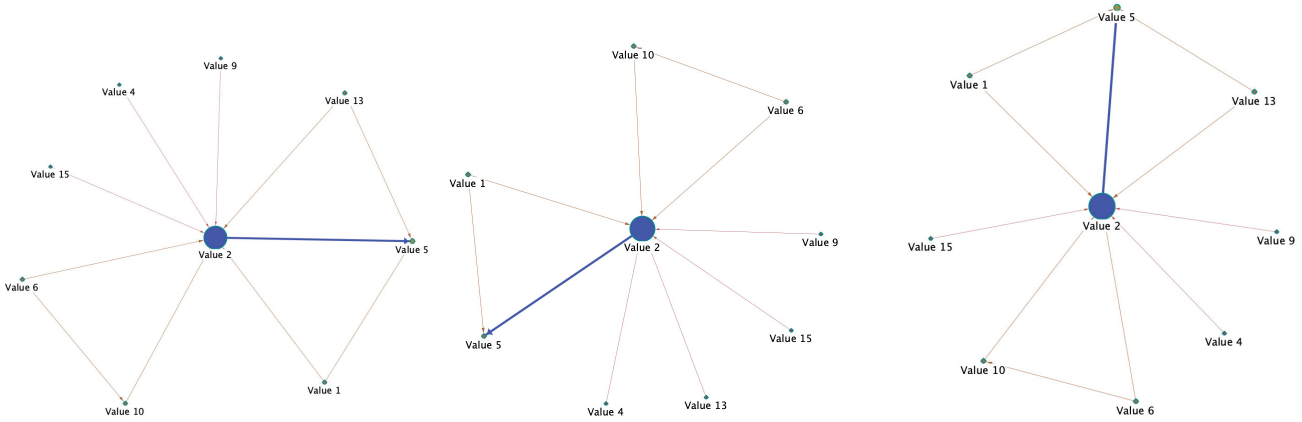
- **BPI 2015 - Municipality 2** data set: observing performer with a 2-step input to predict a 1-step output of the performer or observing both activity and performer with a 5-step input to predict a 1-step output of the performer.
- **BPI 2017** data set: observing both activity and performer with a 4-step input to predict a 1-step output of the activity or observing both activity and performer with a 3-step input to predict a 1-step output of the performer.

As we can see in the validation results, most data sets and observations get the best results with the number output-step is 1. In BPI 2015 data sets, although the number of traces used for training is small compared to other data sets, and it contains a large number of activities (see table 2). However, observing and predicting performers gave us good results equivalent to the remaining data sets.

At the same time, the best results did not happen with the highest number of input-steps. As can be seen, in workflow

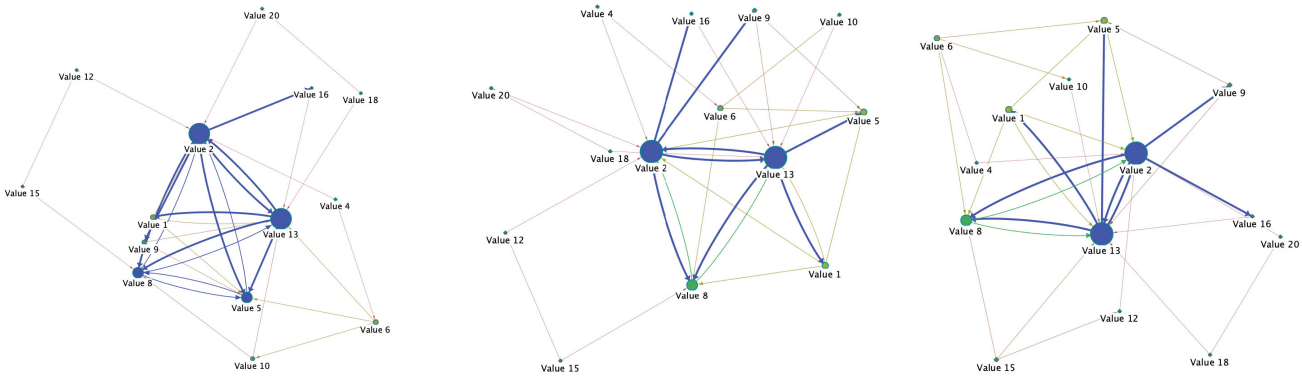
TABLE 7. The predicted ESN with $(\eta_i, \eta_o) = (3, 1)$ combined with different thresholds φ in viewing performer relationship.

$\varphi = 0.05$			$\varphi = 0.10$			$\varphi = 0.15$		
From performer	**To performer**	**Count**	**From performer**	**To performer**	**Count**	**From performer**	**To performer**	**Count**
Value 13	Value 5	1	Value 1	Value 5	2	Value 9	Value 2	1
Value 1	Value 5	2	Value 10	Value 2	12	Value 13	Value 5	1
Value 10	Value 2	12	Value 9	Value 2	12	Value 1	Value 5	1
Value 9	Value 2	12	Value 6	Value 10	12	Value 15	Value 2	2
Value 6	Value 10	12	Value 4	Value 2	12	Value 10	Value 2	12
Value 4	Value 2	12	Value 6	Value 2	12	Value 6	Value 10	12
Value 6	Value 2	12	Value 13	Value 2	24	Value 4	Value 2	12
Value 1	Value 2	59	Value 1	Value 2	47	Value 6	Value 2	12
Value 15	Value 2	195	Value 15	Value 2	83	Value 13	Value 2	24
Value 13	Value 2	481	Value 2	Value 5	235	Value 1	Value 2	36
Value 2	Value 5	1000				Value 2	Value 5	113



(a) $\varphi = 0.05$ (b) $\varphi = 0.10$ (c) $\varphi = 0.15$

FIGURE 12. Predicted ESN with $(\eta_i, \eta_o) = (3, 1)$.



(a) $\varphi = 0.05$ (b) $\varphi = 0.10$ (c) $\varphi = 0.15$

FIGURE 13. Predicted ESN with $(\eta_i, \eta_o) = (4, 1)$.

models, a group of activities and roles determines the next activity be executed. If using too many activities associated with the next activity prediction, the result will not be as good as using the right amount of appropriate activity. So that with each different workflow model, we need to determine the appropriate number of input-steps for getting the best results.

In Table 4, we compare the results for predicting the next event using our approach with the research results in [17]–[19]. As shown in the table, our approach using the multivariate multi-step LSTM model is not the best but provides a valuable solution for predicting the next event. The reason our result on the Helpdesk data set is better than the result of Lin *et al.* [18] is that: In the Helpdesk data set, the average number of events per trace is 4.66. Our training models get the best result with $(\eta_i, \eta_o) = (6,1)$. With 6-input step and 1-output step, we keep the traces with the number of events greater than 7 for training and validation (1870 traces); while

TABLE 8. The predicted ESN with $(\eta_i, \eta_o) = (4, 1)$ combined with different thresholds φ in viewing performer relationship.

$\varphi = 0.05$			$\varphi = 0.10$			$\varphi = 0.15$		
From performer	To performer	Count	From performer	To performer	Count	From performer	To performer	Count
Value 15	Value 8	1	Value 6	Value 5	1	Value 8	Value 2	1
Value 6	Value 5	1	Value 9	Value 5	1	Value 18	Value 20	1
Value 2	Value 9	1	Value 18	Value 13	1	Value 20	Value 2	1
Value 9	Value 5	1	Value 2	Value 9	1	Value 15	Value 8	1
Value 18	Value 13	1	Value 6	Value 10	3	Value 6	Value 5	1
Value 15	Value 12	2	Value 10	Value 13	3	Value 13	Value 1	1
Value 12	Value 2	2	Value 4	Value 6	4	Value 9	Value 5	1
Value 6	Value 10	2	Value 1	Value 5	4	Value 18	Value 13	1
Value 9	Value 13	2	Value 13	Value 1	5	Value 2	Value 9	1
Value 10	Value 8	3	Value 9	Value 13	5	Value 15	Value 13	2
Value 4	Value 6	4	Value 15	Value 12	6	Value 5	Value 2	2
Value 6	Value 13	4	Value 18	Value 20	6	Value 16	Value 13	2
Value 1	Value 5	5	Value 20	Value 2	6	Value 13	Value 2	2
Value 10	Value 13	6	Value 12	Value 2	6	Value 2	Value 16	2
Value 1	Value 13	9	Value 5	Value 2	8	Value 15	Value 12	3
Value 8	Value 2	20	Value 16	Value 13	8	Value 12	Value 2	3
Value 8	Value 5	20	Value 13	Value 2	8	Value 6	Value 10	3
Value 16	Value 13	21	Value 2	Value 16	8	Value 13	Value 8	3
Value 2	Value 16	21	Value 6	Value 8	9	Value 10	Value 13	3
Value 5	Value 2	21	Value 4	Value 2	10	Value 4	Value 6	4
Value 18	Value 20	23	Value 13	Value 8	13	Value 1	Value 5	4
Value 20	Value 2	23	Value 1	Value 13	14	Value 9	Value 13	5
Value 5	Value 8	40	Value 1	Value 8	15	Value 1	Value 13	6
Value 13	Value 1	41	Value 8	Value 2	17	Value 6	Value 8	8
Value 2	Value 5	41	Value 15	Value 8	17	Value 1	Value 2	9
Value 1	Value 2	68	Value 1	Value 2	19	Value 1	Value 8	9
Value 2	Value 8	155	Value 2	Value 13	30	Value 4	Value 2	9
Value 13	Value 2	205	Value 2	Value 8	46	Value 2	Value 13	13
Value 4	Value 2	268	Value 8	Value 13	111	Value 2	Value 8	15
Value 2	Value 13	496	Value 13	Value 5	160	Value 8	Value 13	41
Value 13	Value 8	565				Value 13	Value 5	71
Value 8	Value 13	762						
Value 13	Value 5	888						

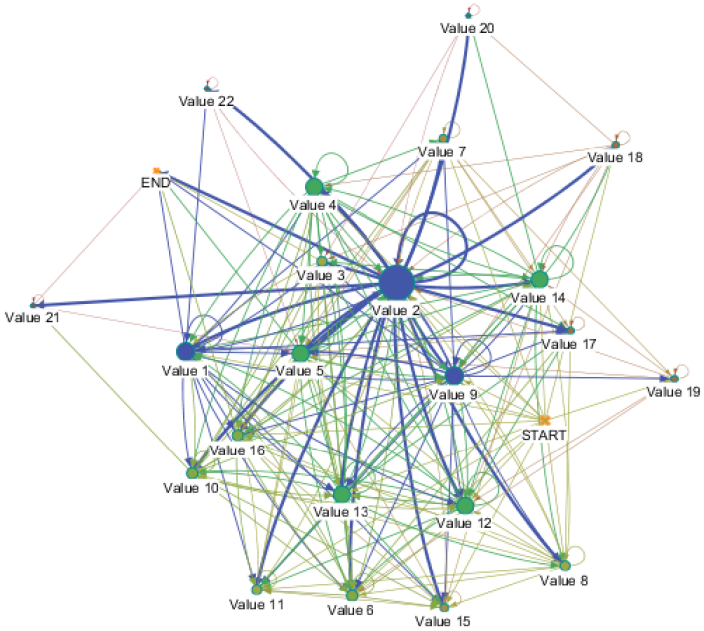


FIGURE 14. The original ESN discovered from the remaining one-third of the Helpdesk data set.

Lin *et al.* keep traces with the number of events greater than 3 (4580 traces).

B. PREDICTIVE ENTERPRISE SOCIAL NETWORK

In this subsection, we explain the predictions of different ESN variants on the Helpdesk data set by applying Algorithm 1.

The Helpdesk data set was chosen because it has a small number of performers (22 performers) and average event per trace (4.66). It is suitable for observing the changes of the enterprise social network between different thresholds; while the other data sets have a large number of performers or average event per trace. As analyzed above, the number output-step is 1 giving the best results, besides, the average number of event per trace is 4.66, so we choose parameters as follows: $(\eta_i, \eta_o) = [(1, 1), (2, 1), (3, 1), (4, 1)]$. $\alpha = 17$, since the largest number of events in a trace in the Helpdesk data set is 17.

We conduct predicting ESN with different thresholds φ: 0.05, 0.10, and 0.15 (It means that predicting ESN variant with the probability execution of the next activity is greater than 5%, 10%, and 15%, respectively). The prediction results are shown in Figures 10, 11, 12, and 13.

Figure 10 describes the predicted ESN with the number of input step and output step $(\eta_i, \eta_o) = (1, 1)$ and the relationship between performer and performer is shown in table 5. As shown in figure 10 and table 5, there are only 3 performers that were predicted involving in the future process and form the ESN. The number of work transference between performers and performers with $(\varphi = 0.10)$ and $(\varphi = 0.15)$ are similar. Meanwhile, there is a significant change with $(\varphi = 0.05)$.

Figure 11 describes the predicted ESN with the number of input step and output step $(\eta_i, \eta_o) = (2, 1)$. The relationship in the network is shown in table 6. As shown in figure 11 and table 6, the list of predicted performers participating in the future process is the same among different φ values. However, there is a huge change in the work transference from performer to performer with different φ values. Especially with $(\varphi = 0.05)$.

Figure 12 describes the predicted ESN with the number of input step and output step $(\eta_i, \eta_o) = (3, 1)$. The relationship in the network is shown in table 7. As shown in figure 12 and table 7, with $\varphi = 0.05$ and $\varphi = 0.010$, the number of predicted work transferences between performers and performers were significant reduced comparing with $(\eta_i, \eta_o) = (2, 1)$.

Figure 13 describes the predicted ESN with the number of input step and output step is $(\eta_i, \eta_o) = (4, 1)$. The relationship in the network is shown in table 8. As shown in figure 13 and table 8, more performers are predicted to participate in the future process than the previous parameters. The number of work transference is similar between different φ. With $\varphi = 0.05$, some performer pairs are predicted having a large amount of work transference ([Value 2, Value 13], [Value 8, Value 13], [Value 13, Value 5]), comparing with the rest.

As can be seen, choosing the different number of input step (η_i) gives very different results. Different threshold φ on the same number of input step parameter gives the similarly predicted ESN results. However, there are significant differences in the number of work transference from performer to performer in the predicted network. The smaller the threshold φ value, the more work transference is predicted and vice versa.

So how to choose a good threshold value? Based on our opinion and this experimental result, it depends on the characteristics of the business process model created the data set as well as the experimental of comparing predicted results with actual results. Figure 14 illustrates the original ESN of the remaining one-third of the Helpdesk data set. So that, in this experimental, selecting $\varphi = 0.05$ and $(\eta_i, \eta_o) = (4, 1)$ is the best choice. Although selecting this parameter is good, there is still a significant difference between the predicted ESN and the original ESN. However, the key performers and work relationships are predicted well.

VI. CONCLUSION

Prediction of the next event information and enterprise social network is important for understanding and planning an organization's resources. There are several different approaches to accomplish this; however, using LSTM is one of the best solutions currently. In this study, dealing with the problem of the next stage after obtaining the predicted event information in the business process, we:

- Summarized some approaches to predict next event information based on business process predictive monitoring;
- Introduced our approach and algorithm using the multivariate multi-step LSTM model to predict not only the next event information, but also various enterprise social network variants. As far as we know, the approach in this paper is the first study addressing the prediction of ESN from the workflow execution log.
- Evaluated our approach with some real-life data sets and compared the results with some related research.

In real-life companies and enterprises, understanding ESN is critical for their operation development. Because human is the main factor in the organization, getting knowledge from ESN means understanding more about the working relationship between people in the organization. The different thresholds will give different variants of the generated ESN based on operating the business process. These variants provide a multi-dimensional perspective to the operators of the organization and hence give the appropriate arrangement for the organization's improvement and allocation of resources.

In this approach, our study still has some limitations such as evaluating the predicted ESN to compare with the original ESN, comparing the degree centrality metrics between two networks. In future work, we will try to deal with these concerns as well as how to allocate human resources in enterprises based on the predicted ESN to improve business processes in enterprises.

REPRODUCIBILITY

The source code, trained networks, and related resources to produce the experiments in this paper can be downloaded at https://github.com/DinhLamPham/PredictingESN.git

REFERENCES

[1] W. M. P. van der Aalst and A. J. M. M. Weijters, "Process mining: A research agenda," *Comput. Ind.*, vol. 53, no. 3, pp. 231–244, Apr. 2004.

[2] D. A. W. Van, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, and J. Buijs, "Process mining manifesto," in *Proc. Int. Conf. Bus. Process Manage.* Berlin, Germany: Springer, 2011, pp. 169–194.

[3] W. M. P. van der Aalst, H. A. Reijers, and M. Song, "Discovering social networks from event logs," *Comput. Supported Cooperat. Work*, vol. 14, no. 6, pp. 549–593, Dec. 2005.

[4] H. Ahn and K. P. Kim, "Formal approach for discovering work transference networks from workflow logs," *Inf. Sci.*, vol. 515, pp. 1–25, Apr. 2020.

[5] G. T. Lakshmanan, D. Shamsi, Y. N. Doganata, M. Unuvar, and R. Khalaf, "A Markov prediction model for data-driven semi-structured business processes," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 97–126, Jan. 2015.

[6] S. Pandey, S. Nepal, and S. Chen, "A test-bed for the evaluation of business process prediction techniques," in *Proc. 7th Int. Conf. Collaborative Comput., Netw., Appl. Worksharing*, 2011, pp. 382–391.

[7] M. Ceci, P. F. Lanotte, F. Fumarola, D. P. Cavallo, and D. Malerba, "Completion time and next activity prediction of processes using sequential pattern mining," in *Proc. Int. Conf. Discovery Sci.* Cham, Switzerland: Springer, 2014, pp. 49–61.

[8] A. Bevacqua, M. Carnuccio, F. Folino, M. Guarascio, and L. Pontieri, "A data-driven prediction framework for analyzing and monitoring business process performances," in *Proc. Int. Conf. Enterprise Inf. Syst.* Cham, Switzerland: Springer, 2013, pp. 100–117.

[9] A. Bolt and M. Sepúlveda, "Process remaining time prediction using query catalogs," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2013, pp. 54–65.

[10] B. Kang, D. Kim, and S.-H. Kang, "Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 6061–6068, Apr. 2012.

[11] W. M. P. van der Aalst, M. H. Schonenberg, and M. Song, "Time prediction based on process mining," *Inf. Syst.*, vol. 36, no. 2, pp. 450–475, Apr. 2011.

[12] A. Rogge-Solti and M. Weske, "Prediction of remaining service execution time using stochastic Petri nets with arbitrary firing delays," in *Proc. Int. Conf. Service-Oriented Comput.* Berlin, Germany: Springer, 2013, pp. 389–403.

[13] A. Rogge-Solti and M. Weske, "Prediction of business process durations using non-Markovian stochastic Petri nets," *Inf. Syst.*, vol. 54, pp. 1–14, Dec. 2015.

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

[15] J. Evermann, J. R. Rehse, and P. Fettke, "A deep learning approach for predicting process behaviour at runtime," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2016, pp. 327–338.

[16] J. Evermann, J.-R. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," *Decis. Support Syst.*, vol. 100, pp. 129–140, Aug. 2017.

[17] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.* Cham, Switzerland: Springer, 2017, pp. 477–492.

[18] L. Lin, L. Wen, and J. Wang, "MM-Pred: A deep predictive model for multi-attribute event sequence," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2019, pp. 118–126.

[19] M. Camargo, M. Dumas, and O. González-Rojas, "Learning accurate LSTM models of business processes," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2019, pp. 286–302.

[20] G. Park and M. Song, "Prediction-based resource allocation using LSTM and minimum cost and maximum flow algorithm," in *Proc. Int. Conf. Process Mining (ICPM)*, Jun. 2019, pp. 121–128.

[21] N. Navarin, B. Vincenzi, M. Polato, and A. Sperduti, "LSTM networks for data-aware remaining time prediction of business process instances," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–7.

[22] J. Brownlee, "Deep learning for time series forecasting: Predict the future with MLPs, CNNs and LSTMs in Python," *Mach. Learn. Mastery Blog*, pp. 135–150, 2018. [Online]. Available: https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/

[23] M. Polato, A. Sperduti, A. Burattin, and M. de Leoni, "Data-aware remaining time prediction of business process instances," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 816–823.

[24] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.

[25] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[26] M. I. Jordan, "Serial order: A parallel distributed processing approach," *Adv. Psychol.*, vol. 121, pp. 471–495, Jan. 1997.

[27] M. Levy, "Machine learning at the edge," Tech. Rep., 2019, pp. 549–601. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128094488000151

[28] R. Sutton, P. Werbos, N. Gupta, E. Rosenfeld, J. Mccool, and L. Jolla, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, USA, 1974.

[29] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[30] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001. [Online]. Available: https://ml.jku.at/publications/older/ch7.pdf

[31] A. Y. Alanis and E. N. Sanchez, *Discrete-Time Neural Observers: Analysis and Applications*. U.K.: Academic, 2017, pp. 99–101.

[32] C. Olah. (2015). *Understanding LSTM Networks*. Accessed: Apr. 27, 2020. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[33] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 473–479.

[34] H. Ahn, D.-L. Pham, and K. P. Kim, "An experimental analytics on discovering work transference networks from workflow enactment event logs," *Appl. Sci.*, vol. 9, no. 11, p. 2368, Jun. 2019.

[35] I. Verenich. (2016). *Helpdesk Data Set. Mendeley Data, V1*. Accessed: Mar. 8, 2020. [Online]. Available: http://dx.doi.org/10.17632/39bp3vv62t.1

[36] *4TU, Research Data: BPI 2012 Data Set*. Accessed:Aug. 3, 2020. [Online]. Available: http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

[37] *4TU, Research Data: BPI 2015 Data Set*. Accessed: Feb. 2, 2021. [Online]. Available: https://www.win.tue.nl/bpi/doku.php?id=2015:challenge

[38] *4TU, Research Data: BPI 2015 Data Set—Municipality 1*. Accessed: Feb. 2, 2021. [Online]. Available: http://dx.doi.org/10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17

[39] *4TU, Research Data: BPI 2015 Data Set—Municipality 2*. Accessed: Feb. 2, 2021. [Online]. Available: http://dx.doi.org/10.4121/uuid:63a8435a-077d-4ece-97cd-2c76d394d99c

[40] *4TU, Research Data: BPI 2017 Data Set*. Accessed: Aug. 3, 2020. [Online]. Available: https://www.win.tue.nl/bpi/doku.php?id=2017:challenge

[41] *Open-Source Library: Keras Deep Learning API*. Accessed: Aug. 3, 2020. [Online]. Available: https://keras.io/

DINH-LAM PHAM received the B.S. and M.S. degrees in computer science from Thai Nguyen University, Vietnam, in 2008 and 2010, respectively, and the Ph.D. degree in computer science from Kyonggi University, South Korea, in 2021. He is currently a Research Professor with the Contents Convergence Software Research Institute and a Research Member with the Data and Process Engineering Research Laboratory, Kyonggi University. His research interests include process mining, deep learning-based process predicting, workflow systems, workflow-supported social and affiliation networks discovery and analysis, and the process-aware Internet of Things architecture and services.

HYUN AHN received the B.S. and M.S. degrees and the Ph.D. degree in computer science from Kyonggi University, South Korea, in 2011, 2013, and 2017, respectively. He is currently an Assistant Professor with the Department of Software Convergence, Hanshin University, South Korea. He is also a Research Professor with the Division of Computer Science and Engineering, Kyonggi University, and the Contents Convergence Software Research Institute, Kyonggi University. His research interests include process-aware enterprise information systems, business process intelligence, process mining, predictive process monitoring, and deep learning and knowledge engineering using active contents big data.

KYOUNG-SOOK KIM received the B.S. degree in computer science from Kyonggi University, South Korea, in 1984, and the M.S. and Ph.D. degrees in computer engineering from Kyunghee University, in 2001 and 2019, respectively. She worked as a Researcher and a Developer with FAST Systems, Inc., and OSI Corporation, South Korea. She is currently an Adjunct Professor with the Department of Computer Science and Engineering, Kyonggi University. Her research interests include large-scale database systems and applications, enterprise information systems, real-time databases, distributed databases, temporal databases, federate databases, object-oriented database management systems, and the Internet of Things collaboration architectures and services.

KWANGHOON PIO KIM (Member, IEEE) received the B.S. degree in computer science from Kyonggi University, South Korea, in 1984, the M.S. degree in computer science from Chungang University, in 1986, and the M.S. and Ph.D. degrees from the Computer Science Department, University of Colorado Boulder, in 1994 and 1998, respectively. He is currently a Full Professor with the Division of Computer Science and Engineering, and the Founder and a Supervisor of the Data and Process Engineering Research Laboratory, Kyonggi University. At Kyonggi, he has been in charge of the Dean of the Computerization and Informatics Institute and the Director of the Contents Convergence Software Research Institute that is newly designated as the National Science and Engineering Research Institute, where he is leading and fulfilling a multi-million dollars research project during 2020–2029 fully funded by the Korea National Research Foundation of the Education Ministry of South Korea. He worked as a Researcher and a Developer with Aztek Engineering, American Educational Products Inc., IBM, USA, and the Electronics and Telecommunications Research Institute (ETRI), South Korea. His research interests include groupware, workflow and business process management systems, BPM, CSCW, collaboration theory, grid/P2P distributed systems, process warehousing and mining, predictive process monitoring, workflow-supported social networks discovery and analysis, process-aware information systems, data intensive workflows, and process-aware Internet of Things, predictive process modeling, process deep-learning, and active contents big data engineering and applications supporting crime prevention and prediction. He is also the Vice-Chair of the BPM Korea Forum. He has also been in charge of a Country-Chair (South Korea) and the ERC Vice-Chair of the Workflow Management Coalition. He has also been on the editorial board of the journal of *KSII Transactions on Internet and Information Systems*, and the committee member of the several conferences and workshops.