Predicting Performances in Business processes using Deep Neural Networks

Gyunam Park, Minseok Song*

Department of Industrial & Management Engineering, POSTECH (Pohang University of Science and Technology), Pohang, Republic of Korea

Abstract

Online operational support is gaining increasing interest due to the availability of real-time data and sufficient computing power, such as predictive business process monitoring. Predictive business process monitoring aims at providing timely information that enables proactive and corrective actions to improve process enactments and mitigate risks. There are a handful of research works focusing on the predictions at the instance level. However, it is more practical to predict the performance of processes at the process model level and detect potential weaknesses in the process to facilitate the proactive actions that will improve the process execution. Thus, in this paper, we propose a novel method to predict the future performances of a business process at the process model level. More in detail, we construct an annotated transition system and generate a process representation matrix from it. Based on the process representation matrix, we build performance prediction models using deep neural networks that consider both spatial and temporal dependencies present in the underlying business process. To validate the proposed method, we performed case studies on three real-life logs.

Keywords: Process mining, Process management, Online operational support, Process performance prediction, Deep neural networks

*corresponding author

1. Introduction

In today's competitive and challenging business world, it is of utmost importance to consistently improve business processes [1]. Process mining, a promising discipline that aims at extracting process-oriented knowledge from event data stored in information systems, has provided practical techniques to that end. Traditionally, process mining focus on analyzing historical data in order to fully understand the process of an organization and identify possible improvements [2]. In recent years, research in process mining has shifted the spotlight from this offline analysis to online operational support due to the availability of real-time data and sufficient computing power. Different from the offline analysis, online operational support aims at monitoring and influencing running cases. As one of the approaches in online operational support, predictive business process monitoring provides timely information that enables proactive and corrective actions to improve process enactments and mitigate risks [3]. Task/resource recommendations [4] and risk notifications [5] are examples of those actions.

Existing studies in predictive business process monitoring focus on making predictions at the process instance level (e.g., predicting the remaining time for an instance to complete the process) [6] and identifying problematic instances (e.g., delayed instances) based on them. However, it is difficult for an operation manager to manage singular process instances in the complex business process [7]. Instead, it is more practical to predict the performance of processes at the process model level (e.g., predicting the processing time and waiting time of activities in the process) and detect potential weakness (e.g., delayed activities) in the process to facilitate the proactive actions that will improve the process execution. For example, in the context of traffic congestion, we focus on not each vehicle (i.e., each instance) but roads or junctions (i.e., activities in a process model).

Suppose the temporal performance of a business process evolves over a given period due to the changes in the process context (e.g., number of cases in progress, number of resources allocated to process, etc.) [2], as shown in Fig. 1.

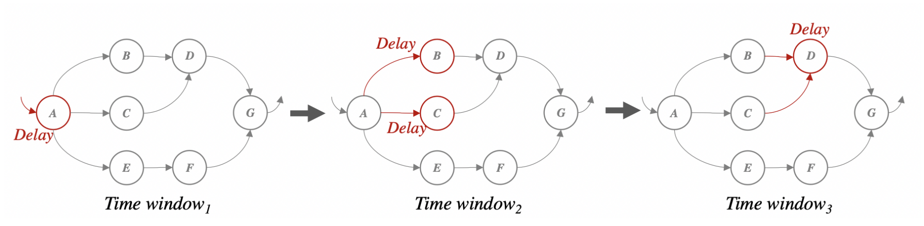


Figure 1: An example of performance evolution in a business process

During $time\,window_1$(01:00 $\sim$ 02:00), exceptionally large number of cases enter the process causing delay in *activity A*. At $time\,window_2$(02:00 $\sim$ 03:00), the limited number of resources causes the next delay in *activity B* and *activity C*. Finally, the accumulated demands for *activity D* after finishing either *activity B* or *activity C* result in the new delay on *activity D*. If it recurs regularly and is expected to be valid in the future, we call it a pattern [8]. Given this pattern, at $time\,window_2$, one can predict the problematic point in the business process (i.e., *activity D*) and take proactive actions such as assigning more resources to serve it and finding an alternative activity to bypass it.

Thus, in this paper, we aim at developing a method to predict the future performances of a business process at the process model level to enable proactive actions to improve the business process. To this end, we concentrate on the similarity between the traffic and the business process model [2]. We can think of cars in the traffic as cases in the business process model, roads as activities, and movements as events. The congestion in the traffic can be understood as bottlenecks in the business process. In the field of traffic research, the congestion prediction, that aims at predicting the future speed of roads based on historical observations, is known as an essential but challenging problem [9]. In recent years, the large volume of traffic data enables the researchers to develop novel prediction algorithms based on it. Among those data-driven methods, deep learning-based approaches, that deploy techniques for image/video processing, have achieved significant success [10, 11, 12].

Motivated by the recent breakthroughs in congestion prediction problems, we propose a novel method to predict the future performances of a business process based on deep neural networks (DNN). More in detail, we first discover a process model from an event log and annotate it with relevant information by replaying the log. Next, we generate a process representation matrix that contains information on the performances in the business process. Finally, we generate a training set from the process representation matrix and construct performance prediction models based on DNN that consider temporal evolution and spatial dependency of the process model.

The paper is organized as follows. Section 2 discusses the related work. Section 3 explains the backgrounds required to understand the proposed method. The performance prediction method is explained in Section 4 and evaluated on two real-life logs in Section 5. Section 6 discusses the usefulness and limitation of the proposed method. Finally, Section 7 concludes this paper.

2. Related work

2.1. Predictive business process monitoring

Predicting performances in business processes is concerned with the research of predictive business process monitoring, which is one of the sub-fields of process mining. It encompasses the set of methods to build predictive models aiming at providing timely information which can be used to improve the business process and mitigate possible process-related risks.

Several approaches have been proposed to predict three types of values: 1) remaining time, 2) risk probability, and 3) next event [3]. First, the remaining time prediction is concerned with the completion time of business process instances. Van der Aalst et al. [6] suggests a configurable method to construct a process model where the annotated values are used to predict the remaining time of instances. Extending on [6], Polato et al. [13] proposes a method to predict the remaining time using a set of machine learning approaches, such as Naive Bayes and Support Vector Regression (SVR). Second, the risk prediction generates various process-related risks in a business process. Pika et al. [14] propose a set of process risk indicators such as abnormal execution time and repetition of multiple events and predict them with statistical techniques. Kang et al. [15] develop the monitoring system to predict the abnormal termination of a running instance using the K-Nearest Neighbor technique. Finally, the next event prediction deals with what the next event (e.g., activity, resource, and timestamp) will become. Breuker et al. [16] develop a method to determine the next activity in a running instance using Probabilistic Finite Automaton (PFA) built upon a Petri net. Motivated by natural language processing, a couple of approaches [17, 18] apply the recurrent neural network (RNN) to predict the next event in a business process. These approaches are extended by Mehidiyev et al. [19], where the authors utilized a deep learning architecture composed of unsupervised stacked autoencoders and supervised fine-tuning with n-gram features which are leveraged by feature hashing.

Existing studies in predictive business process monitoring focus on predicting the future status of running instances. This instance-level prediction enables operational support for enhancement in productivity. For example, operation manager can take actions for instances which are expected to be delayed or prone to risks. However, this microscopic approach is not sufficient for the comprehensive management of a business process. In this regards, we need to provide the model-level predictions such as the future bottlenecks in the business process to enable proactive actions to mitigate them.

2.2. Congestion prediction

Congestion prediction problem, one of the most attractive problems in transportation management, means to predict the future speed of roads based on historical observations. Over the last few years, the availability of traffic data has been increased, and many approaches try to solve the problem in a data-driven manner [20].

Recently, deep learning algorithms with its competency in extracting features are widely applied to the congestion prediction problem. The deep-learning-based methods pay special attention to learn spatial and temporal correlations existing in a traffic network. Ma et al. [10] applies a Long-Short Term Memory neural network (LSTM NN) to capture the dynamic nature of the traffic network. A couple of approaches based on Convolutional Neural Networks (CNN) are also presented in recent years [21, 11], showing good performances. To consider network structure, Yu et al. [12] proposes a method which incorporates CNN and LSTM.

Motivated by the recent breakthroughs in the congestion prediction problem, We develop a novel method to predict the future performances of a business process given historical observations. To this end, We extend the deep-learning-based models suggested in this literature.

3. Backgrounds

This section describes the background necessary to understand our approach. We first explain preliminary concepts related to process mining that we will use throughout the paper. Second, we elaborate transition system we use to discover a process model from the event log. Finally, we provide basic concepts of three different deep neural networks.

3.1. Preliminaries

Process mining techniques can extract useful information from event logs. An event log is detailed information about the activities that have been executed in a single process [2]. In this sub-section, we give formal definition of *event*, *trace* and *event log*. First, an event is a record of the execution of an activity. It contains information such as activity, resource, timestamp.

Definition 1 (Event). *Let $\mathcal{E}$ be the event universe. Events are characterized by various attributes (e.g., activity, originator, timestamp). Let $\mathcal{AN}$ be a set of attribute names. For any event $e \in \mathcal{E}$ and any attribute name $an \in \mathcal{AN}$, $\pi_{an}(e)$ is the value of an for event e. If e does not contain the attribute an, $\pi_{an}(e) = \perp$.*

Each event is associated with the process instance (e.g., customer, patient, or student) which has its trace. Trace is a sequence of events. For example, a patient might undergo a sequence of events, e.g., *Blood test*, *MRI*, and *Treatment*. An event log is a collection of process instances. The formal definition is as follows:

Definition 2 (Trace, Event log). *Let $\mathcal{E}^*$ be the set of all finite sequences over $\mathcal{E}$. A trace, $\sigma \in \mathcal{E}^*$, is a finite sequence of events. Each event in a trace appears only once and time is non-decreasing. For $\sigma = \langle e_1, e_2, ..., e_n \rangle$, $hd^k(\sigma)$ consists of first k elements, i.e., $hd^k(\sigma) = \langle e_1, e_2..., e_k \rangle$. On the other hand, $tl^k(\sigma)$ consists of the last k elements, i.e., $tl^k(\sigma) = \langle e_{n-k+1}, ..., e_n \rangle$. Let $\mathcal{C}$ be the set of traces. An event log $\mathcal{L}$ is a collection of traces, i.e., $\mathcal{L} = \{\sigma_c | c \in \mathcal{C}\}$.*

3.2. Transition system

A transition system is one of the most efficient approaches to model the behaviors in an event log [2]. A transition system is composed of states, event labels, and transitions that describe how the system moves from one state to another. States represent the status of the system, and transitions enable the system to move from a particular state to another state. Event labels indicate particular events which trigger the transitions. Transition systems have one or more initial and final states. From the transition system, one can reason about the behavior of a process. Starting from the initial states and finishing at the final states, any path in the graph corresponds to a possible execution sequence in the business process.

State representation function and event representation function are two core functions to construct the transition system. The formal definition of them is as follows:

Definition 3 (State & Event Representation Function). *A state representation function $l^s \in \mathcal{C} \rightarrow \mathbb{R}^s$ produces a representation of a (partial) trace σ, where $\mathcal{C}$ is the set of traces and $\mathbb{R}^s$ is the set of state representations (e.g., sequences, sets, multiset). An event representation function $l^e \in E \rightarrow \mathbb{R}^e$ produces a representation of an event e, where $\mathbb{R}^e$ is the set of event representations (e.g., $\pi_A(e), \pi_\tau(e)$.)*

A transition system can be produced based on l^s and l^e. With the state representation function l^s, the prefixes in the log are transformed to the states, creating a state space S (i.e., possible states of the process). Afterwards, the transition T is computed by replaying the process instances and connecting states $s \in S$, while using the event representation function l^e to generate event label E of the transition. Formally, it is defined as follows:

Definition 4 (Transition System). *A transition system TS is defined as a triplet (S, E, T) such that $S = \{l^s(hd^k(\sigma)) | \sigma \in L \wedge 0 \leq k \leq |\sigma|\}$, $E = \{l^e(\sigma(k)) | \sigma \in L \wedge 1 \leq k \leq |\sigma|\}$, and $T = \{l^s(hd^k(\sigma)), l^e(\sigma(k+1)), l^s(hd^{k+1}(\sigma)) | \sigma \in L \wedge 0 \leq k \leq |\sigma|\}$. $S^{start} = \{l^s(\langle\rangle)\}$ is the set of initial states and $S^{end} = \{l^s(\sigma) | \sigma \in L\}$ is the set of final states.*

We can produce various forms of transition systems based on different abstractions: *representation*, *horizon*, and *filter* [6]. First, *representation* determines whether to remove the order and frequency from the trace or not. Second, *horizon* means how many events are considered from the prefix to derive the states. Third, *filter* decides which events to consider when calculating the state. For a more detailed explanation, see van der Aalst et al. [6].

3.3. Deep neural networks

Deep neural networks (DNN) have been successfully applied to various domains [22]. In the following, we will explain the architectures of DNN we use in this paper.

3.3.1. Convolutional neural network

Convolutional Neural Network (CNN) has an exceptional ability to understand the spatial structure of an input (e.g., image). Having this competency, CNN has achieved remarkable improvements in visual tasks such as image and video recognition [23].

CNN is composed of model input, feature extraction layers, fully connected layer, and model output. First, let model input be:

$$x = [x_1, x_2, , x_n], \text{ where } x_i \in \mathbb{R}^n, \forall i \in [1, n]. \tag{1}$$

The combination of convolutional and pooling layers extract the features from the input. More in detail, the convolutional layer captures the local features present in the model input by a set of learnable filters called convolutional kernels $w \in \mathbb{R}^{m \times m}$. The kernel convolves every possible window of the model input, resulting in the feature map:

$$c \in \mathbb{R}^{(n+m-1)\times(n+m-1)} \text{ such that } c_{i,j} = f(\sum_k \sum_l w_{k,l} x_{i+k-1,j+l-1} + b) \tag{2}$$

,where $b \in R$ is the bias and f is a non-linear activation function. The pooling layer addresses the most important features by pooling (e.g., max pooling) over every feature map. The resulting feature map p is computed by:

$$p = [pool(c)] \tag{3}$$

Finally, the pooled feature maps from different kernels is concatenated and flattened:

$$p^{flatten} = flatten([p]) \tag{4}$$

It is then passed to the fully connected layer and transformed into model output:

$$\hat{y} = w_f \cdot p^{flatten} + b_f \tag{5}$$

where w_f is the weight of the layer and b_f is the bias.

3.3.2. Long-short-term memory neural network

Recurrent Neural Network (RNN) has a competency to learn temporal dynamics. Fig. 2 shows the architecture of RNN where the hidden states are generated in recurrent manner to maintain information over time. Let g be an activation function (e.g., sigmoid or hyperbolic tangent), x_t be the input, h_t be the hidden state, and o_t be the output at time t. Following is the recurrent equation to produce outputs from input sequences.

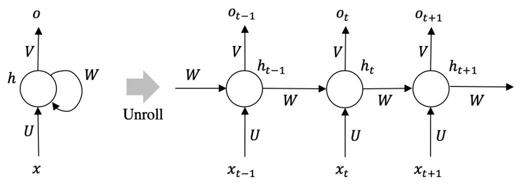


Figure 2: An architecture of Recurrent Neural Networks

$$h_t = g(Ux_t + Wh_{t-1} + b_h),\ o_t = g(Vh_t + b_o) \tag{6}$$

Long-Short-Term Memory Neural Network (LSTM NN) is a special kind of RNN which solves the gradient vanishing problem the conventional RNN models have. LSTM NN is composed of LSTM units which determine when to forget previous hidden states and when to update hidden states from new information. A typical architecture of the LSTM unit consists of a cell state and three gates, i.e., forget gate, input gate, and output gate. The cell state(i.e., C_t) allows the data to pass through the neuron without losing much information, while the gates regulate the flow of information inside the LSTM unit. Each gate is composed of a sigmoid function, σ, and a pointwise multiplication operation, $\otimes$. The forget gate(i.e., f_t) decides whether to discard information from the cell state. The input gate(i.e., i_t) allows adding new information to the cell state. The output gate(i.e., o_t) determines what the model will generate as an output. Let X, H, Y be the input time series, the hidden state of memory cells, and the output time series, respectively. The hidden state of memory cells is calculated in the following formulas:

$$\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), & i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= tanh(W_c[h_{t-1}, x_t] + b_c), & C_t &= \sigma(f_t C_{t-1} + i_t \tilde{C}_t) \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), & h_t &= o_t tanh(C_t)
\end{aligned} \tag{7}$$

where W, b indicates the weights and biases which are learned during the training phase.

3.3.3. Long-term recurrent convolution network

Different from image recognition tasks, video processing requires a model to deal with variable-length input sequences, and generate variable length outputs as well. In this regard, Donahue et al. [24] proposed a novel neural network architecture called Long-term Recurrent Convolutional Networks (LRCNs). LRCN combines a deep hierarchical visual feature extractor (e.g., CNN) with a recurrent model (e.g., RNN) to recognize and learn temporal dynamics for tasks involving sequential data (e.g., video recognition).

An LRCN is composed of model input, feature extractor, sequence learning layer, and model output. Let x_t be a model input (i.e., an image or a frame from a video). The feature extractor $f_V(\cdot)$ with parameters V produce a fixed-length feature vector as follows:

$$fv_t = f_V(x_t) \tag{8}$$

A CNN is deployed for this purpose. The feature vector fv_t is then passed into a recurrent sequence learning layer. The recurrent model is LSTM NN with parameters U, W and V to learn temporal dependencies. Let h_t be a hidden state, o_t be an output. In the recurrent model, the feature vector fv_t and a previous time step hidden state h_{t-1} are used to produce output o_t as follows:

$$h_t = g(Ux_t + Wh_{t-1} + b_h),\ o_t = g(Vh_t + b_o) \tag{9}$$

Finally, a distribution $P(y_t)$ at time t is produced through a linear prediction layer:

$$\hat{y_t} = W_o o_t + b_o \tag{10}$$

where W_o and b_o are learned parameters.

4. Method

This section proposes a method to predict the future performances of a business process based on deep neural networks. A general overview is presented first, and then we explain each step in more detail.

4.1. Overview

Our method consists of three steps: 1) *annotated process model construction*, 2) *process representation matrix generation*, and 3) *prediction model construction.* Fig. 3 describes the overview of this method. As the first step, we produce a process model from an event log and annotate the derived model with measurements by replaying the log. In the process mining discipline, many process discovery algorithms have been proposed to produce better process models with different modeling notations. In this paper, we adopt a state transition system because of its ability to derive diverse forms of features (i.e., not only control-flow but also organizational/data perspective) for deep learning techniques by applying various abstraction techniques (see Section 3.2). For example, a state in a transition system can be an activity, a resource, or the combination of an activity and a resource, etc. It enables us to use different forms of process models according to the objectives.

In the second step, we generate a process representation matrix that contains information on the temporal performances in the business process, from the annotated process model. In this paper, we suggest two forms of the process representation matrix that efficiently represent the spatial dependence and temporal evolution of the business process.

Finally, we generate a training set from the process representation matrix and train the deep-learning-based prediction models to forecast the future performances of the business process based on the historical records of the performances. In this work, we utilize three deep learning architectures (i.e., *CNN*, *LSTM*, and *LRCN*) to efficiently learn the spatial and temporal dependencies, which are embodied in the process representation matrix.

4.2. Annotated process model construction

The initial step of the proposed method is to produce a transition system that describes the behaviors seen in an event log. In the rest of the paper, we assume a transition system with a state representation function $l^s(\sigma) = \langle \pi_A(\sigma(|\sigma|)) \rangle$, which represents the partial trace by the sequence of events with horizon of 1,

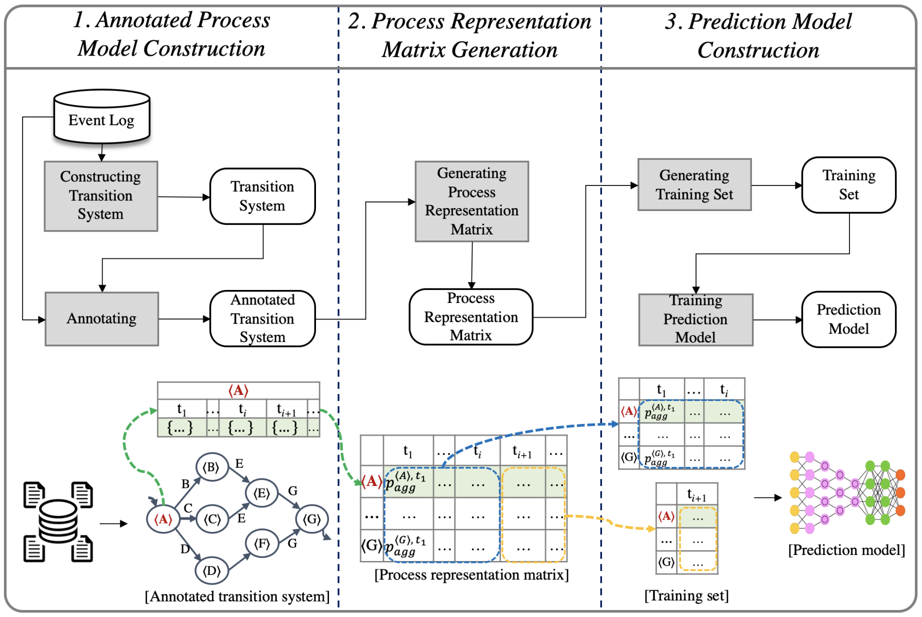

1. Annotated Process Model Construction
2. Process Representation Matrix Generation
3. Prediction Model Construction
Event Log
Constructing Transition System
Transition System
Annotating
Annotated Transition System
Generating Process Representation Matrix
Process Representation Matrix
Generating Training Set
Training Set
Training Prediction Model
Prediction Model
[Annotated transition system]
[Process representation matrix]
[Training set]
[Prediction model]

Figure 3: Overview of the proposed method

and an event representation function $l^e(e) = \pi_A(e)$, which labels the transition with activities, in order to efficiently deliver the idea behind this research.

Next, we annotate the states and transitions of the transition system with measurements by replaying the event log to it. The following definition formalize how to calculate the measurements from the two partial traces σ_1 and σ_2 such that σ_1 and σ_2 is the prefix and the postfix of a trace σ. Each measurement is annotated to the corresponding state or transition in a transition system. Although numerous measurements are possible, we use *waiting time*, *processing time*, and *sojourn time* in this paper.

Definition 5 (Measurements). *Let σ_1 and σ_2 be the prefix trace and postfix trace of given trace σ. A measurement function l^m is a function that generates tuples of measurement and its relevant timestamp (e.g., (4,00:10)). Formally, $l^m \in C \times C \rightarrow \mathbb{R}^+ \times \mathcal{T}$. Let $min_{ST/CT}(\sigma) = min\{\pi_{ST/CT}(e) | e \in \sigma\}$ and $max_{ST/CT}(\sigma) = max\{\pi_{ST/CT}(e) | e \in \sigma\}$, where ST and CT stands for the start time and the complete time. $l^m_{waiting} = (min_{ST}(\sigma_2) - max_{CT}(\sigma_1), max_{CT}(\sigma_1))$ if*

$\sigma_1 \neq \langle\rangle$ and $\sigma_2 \neq \langle\rangle$; 0 otherwise. $l^m_{processing} = (min_{CT}(\sigma_2) - min_{ST}(\sigma_2), min_{ST}(\sigma_2))$ if $\sigma_2 \neq \langle\rangle$; 0 otherwise. $l^m_{sojourn} = (min_{CT}(\sigma_2) - max_{CT}(\sigma_1), max_{CT}(\sigma_1))$ if $\sigma_1 \neq \langle\rangle$ and $\sigma_2 \neq \langle\rangle$; 0 otherwise.

The measurements are annotated to the corresponding states and transitions in the transition system to generate the annotated transition system. Let the prefix of σ be $hd^k(\sigma)$ and the postfix be $tl^{|\sigma|-k}(\sigma)$. A measurement $l^m(hd^k(\sigma), tl^{|\sigma|-k}(\sigma))$ is annotated to the state $l^s(hd^{k+1}(\sigma))$, or the transition $(l^s(hd^k(\sigma)), l^e(\sigma(k+1)), l^s(hd^{k+1}(\sigma)))$. The formal definition of the annotated transition system is as follows:

Definition 6 (Annotated transition system). *Let L be an event log and $TS = (S, E, T)$ a transition system based on a state representation function l^s and event representation function l^e. Given a particular measurement function l^m, we define a state annotation function $A_s \in S \rightarrow B(M)$ such that $A_s(s) = \sum_{\sigma \in L} \sum_{1 \leq k \leq |\sigma|-1, s=l^s(hd^{k+1}(\sigma)) \in S} [l^m(hd^k(\sigma), tl^{|\sigma|-k}(\sigma))]$. In other words, $A_s(s)$ is a multi-set composed of measurements corresponding to the state s. In addition, we define a transition annotation $A_t \in T \rightarrow B(M)$ such that $A_t(t) = \sum_{\sigma \in L} \sum_{1 \leq k \leq |\sigma|-1, t=(l^s(hd^k(\sigma)), l^e(\sigma(k+1)), l^s(hd^{k+1}(\sigma))) \in T} [l^m(hd^k(\sigma), tl^{|\sigma|-k}(\sigma))]$. In other words, $A_t(t)$ is a multi-set composed of measurements corresponding to the transition t. An annotated transition system is the tuple (S, E, T, A_s, A_t)*

4.3. Process representation generation

In this step, we build a process representation matrix from an annotated transition system. To this end, we first define temporal aggregation functions that will be used to calculate a temporary performance measure from measurements in an annotated transition system. Using the temporal aggregation function, we derive two different types of process representation matrix that will be used to produce a training set (i.e., predictor and response variables) for prediction models in the next step.

In order to evaluate the evolution of the performances in the underlying business process, we need the time window upon which we calculate the temporal performance. If we want to measure the hourly evolution of performances in the business process, we need the time windows of an hour, e.g., [Oct. 5 00:00, Oct. 5 01:00], [Oct. 5 01:00, Oct. 5 02:00], and so on. The definition of the time window is as follows:

Definition 7 (Time window). *Given a time point t, a time period p, the [t, $t + p$] will form a single block. Given a time stride s, the next block slides with s, i.e., [t+s, t+p+s] will form the next block. Suppose T is the whole range. $TW_{p,s}$ is a set of all blocks formed by p and s within T.*

After producing a set of time windows, we need to calculate the temporal performance measures of the states and transitions associated with each time window. To this end, we use temporal aggregation function (e.g., average).

Definition 8 (Temporal aggregation function and performance measure). *Let $tw \in TW$ be a time window and M be a multi-set of measurements. A temporal aggregation function, $agg_{tw}(M)$ is a function that generates a numerical value by aggregating measurements M at tw. Formally, $agg_{tw} \in M \to \mathbb{R}$ such that M is multi-set of measurements. $p_{agg}^{M,tw}$ is the temporal performance measure resulting from the temporal aggregation function agg_{tw}.*

This paper proposes two forms of process representation matrix to embed the evolution of performance in business processes efficiently. First, the two-dimensional process presentation matrix expresses the evolutionary performances of a business process in the two-dimensional matrix. Using the process model entities (i.e., state and transition) and the time windows as the first and second dimensions, we efficiently represent the spatial dependence and the temporal evolution.

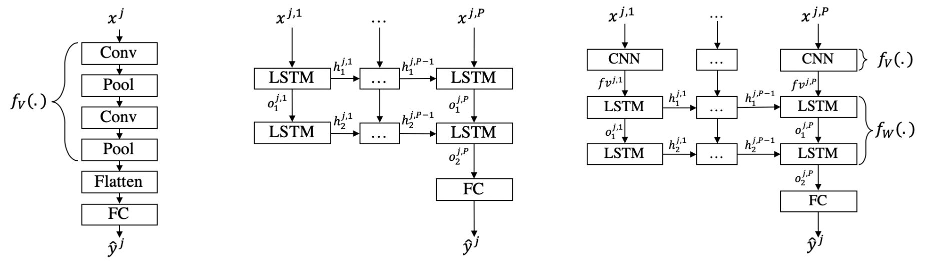
Definition 9 (two-dimensional process representation matrix). *Let $tw \in TW$ be a time window and $o \in \{state, transition\}$ be the state and transition in the annotated transition system. Then, a matrix $R^{2dim,o} \in \mathbb{R}^{|O| \times |TW|}$ is called 2-dimensional representation matrix with $r_{i,j}$ indicating the performance* *measure of o_i at tw_j, i.e., $agg_{tw_j}(A_o(o_i))$, where $A_o \in \{A_s, A_t\}$ is the state or transition annotation function.*

Second, a three-dimensional process representation matrix efficiently embodies the spatial dependency in the first two dimensions and the temporal evolution in the third dimension. The first two dimensions represent the network topology (i.e., directly-follows relations) and indicate the outgoing state (i.e., from-state) and the incoming state (i.e., to-state). The third dimension is the time window.

Definition 10 (three-dimensional process representation matrix). *Let $tw \in TW$ be a time window and $s \in S$ be the state in the annotated transition system. Let $tr(s_1, s_2)$ be a function to produce the transition connecting s_1 and s_2 in the annotated transition system. Then, a matrix $R^{3dim} \in \mathbb{R}^{|S|\times|S|\times|TW|}$ is called three-dimensional process representation matrix with $r_{i,j,k}$ representing the performance measure of transition $t = tr(s_i, s_j) \in T$ at tw, i.e., $agg_{tw}(A_t(t))$ such that A_t is the transition annotation in the annotated transition system.*

4.4. Prediction model construction

In this step, we aim at learning a performance prediction function which returns future performance measures of the states and transitions in the business process, given the historical performance measures of them. In order to consider both spatial and temporal dependencies underlying the business process, we deploy three deep neural networks that show the competency in learning spatiotemporal correlations in the traffic network, i.e., CNN [11], LSTM [10], and LRCN [12]. In the following, we will explain each of three prediction models, i.e., *CNN-based model*, *LSTM-based model*, and *LRCN-based model.* Fig. 4 shows the architecture of each prediction model. Although the proposed models have different input and model architecture, the procedure to construct them is composed of the same two phases, i.e., 1) *training set generation* and 2) *model learning.*



(a) CNN-based model **(b) LSTM-based model** **(c) LRCN-based model**

Figure 4: Architectures of prediction models

4.4.1. CNN-based prediction model

A CNN has the competency to extract essential features from the input image (See Section 3.3), $img \in \mathbb{R}^{height \times width \times channels}$ where channels commonly represent the RGB values. In order to exploit its competency, we generate image-like input, $x \in \mathbb{R}^{object \times tw \times channel}$ from two-dimensional process representation matrix $R^{2dim,o}$ by setting the number of channels as one. Let P be the length of input time windows. The model input can be written as:

$$x'^j = [r_j, r_{j+1}, , r_{j+P-1}], j \in [1, |TW| - P], \tag{11}$$

where r_j is a column vector representing performance measures at tw_j, i.e., $R^{2dim,o}_{\cdot,j}$. By adding a channel of 1, we transform $x'^j \in \mathbb{R}^{|O| \times |TW|}$ to an image-like $x^j \in \mathbb{R}^{|O| \times |TW| \times 1}$. The model output can be written as:

$$y^j = [r_{j+P}], j \in [1, |TW| - P] \tag{12}$$

where r_{j+P} is a column vector representing performance measures at tw_{j+P}, $R^{2dim,o}_{\cdot,j+P}$.

Afterward, we train a CNN-based prediction model that extracts spatiotemporal features embedded in the model input, where the first and second dimensions represent the temporal and spatial information, respectively. Fig. 4-(a) shows the architecture of the proposed CNN model. The model input x^j is passed to a feature transformation $f_V(.)$ that consists of the combination of convolution and pooling layers. Through the transformation, the model learns the spatiotemporal features of the business process. The extracted features $f_V(x^j)$ from different filters are concatenated to a dense vector by flattening, and then they are passed into a fully connected layer to be transformed to the model output $\hat{y}^j$.

We train all sets of network weights using *RMSProp algorithm* [25] such that the mean absolute error (MAE) between the predicted value $\hat{y}$ and the actual value y is minimized. We use convolutional filters of size $(3, 3)$ and max poolings of size $(3, 3)$. Convoultional layers consecutively transform the number of channels into 32 and 16 with the corresponding number of convolutional filters. As regularization strategies, we use *Dropout* [26] and *Batch Normalization* [27].

4.4.2. LSTM-based prediction model

Long-Short Term Memory Neural Networks (LSTM) has the competency to learn long temporal dependency for the input sequence. The strength to learn the temporal dependency is suitable for predicting the future performances of the business process based on historical performances. Furthermore, by producing a sequence of vectors as model input where each vector represents the snapshot of performances on all locations in the business process model, we can construct the prediction model to reflect both the spatial and temporal information effectively. To this end, we first generate a training set from the two-dimensional process representation matrix $R^{2dim,o}$ using equation (11) and (12) in Section 4.4.1.

Afterward, we train an LSTM-based prediction model from the training set. Fig. 4-(b) shows the unrolled structure of the proposed LSTM-based prediction model. Note that the number of steps LSTM unrolled is P that is the length of input time windows. The model is composed of two hidden LSTM layers, where each layer contains multiple memory cells. Each layer contains LSTM cells as much as the number of column vectors in x^j. The model is trained in a recurrent manner. In other words, LSTM cell $f^1_{W_1}(.)$ with parameters W takes as input not only the element $x^{j,t}$ in the model input x^j, but also the hidden state $h_1^{j,t-1}$ generated from the previous LSTM cell. The output $o_1^{j,t}$ of the

LSTM cell in the first layer is passed to the next LSTM $f^2_{W_2}(.)$layer as an input. The output $o_2^{j,t}$ of the last LSTM cell in the second layer becomes an input to a fully connected layer $f_L(.)$, which then produces the prediction results $\hat{y}^j$.

We train all sets of network weights using *RMSProp algorithm* [25] such that MAE between the predicted value $\hat{y}$ and the actual value y is minimized. The number of hidden states in the first layer is twice as large as the input dimension, and the number of hidden states in the second layer is four times larger than the input dimension. As regularization strategies, we use *Dropout* [26] and *Batch Normalization* [27].

4.4.3. LRCN-based prediction model

It is paramount to reflect the process model topology to build an accurate prediction model. To this end, we produce a model input based on the three-dimensional process representation matrix R^{3dim}, which involves the directly-follows relations. The model input can be written as:

$$x^j = [img_j, img_{j+1}, , img_{j+P-1}], j \in [1, |TW| - P] \tag{13}$$

, where img_j is an image generated from $R^{3dim}_{\cdot,\cdot,j}$ (i.e., the performance at j_{th} time window) by setting the channel as one just as we did in Section 4.4.1.

The model output can be written as:

$$y^j = [img_{j+P}], j \in [1, |TW| - P] \tag{14}$$

, where img_{j+P} is an image generated from $R^{3dim}_{\cdot,\cdot,j+P}$ (i.e., the performance $j+P_{th}$ time window) by setting the channel as one.

We can understand the prediction task as the video recognition problem where the previous sequence of frames (i.e., images) is used to predict the future frame. An LRCN, a class of neural network architectures for video recognition, combines a visual feature extractor (i.e., CNN) and a recurrent model (i.e., LSTM) which recognize the temporal dynamics of the sequential images. Fig. 4-(c) shows the architecture of our LRCN-based prediction model. We start by passing each element $x^{j,t}$ in x^j through a CNN, f_V, with parameters V to produce a feature vector $fv^{j,t}$. The resulting feature vector $fv^{j,t} = f_V(x^{j,t})$ is then passed into a LSTM layer. The LSTM layer f_W works in recurrent manner by taking both feature vector $fv^{j,t}$ and previous hidden state $h_1^{j,t-1}$ as an input. For example, $h_1 = f_{W_1}(fv^{j,1}, h_1^{j,0}) = f_{W_1}(fv^{j,1}, 0)$, $h_2 = f_{W_1}(fv^{j,2}, h_1^{j,1}) = f_{W_1}(fv^{j,2}, f_{W_1}(fv^{j,1}, 0))$, etc. We stack another LSTM layer as we do for the *LSTM-based model.* The output of the second LSTM layer $o_2^{j,P}$ is passed through a fully connected layer, which then generate the model output $\hat{y}^j$.

Since this model is the combination of *CNN-based model* and *LSTM-based model*, we follow the same procedure as the previous two approaches to train the model.

5. Evaluation

In order to evaluate the applicability of the proposed method to predict the future performances of a business process, we conduct three case studies with real-life logs: healthcare service, BPI Challenge 2012 (BPIC'12) [1], and helpdesk [2]. When applying the proposed method, we initialized all network weights using a uniform random distribution over [0.1, 0.1]. The models were trained with a batch size of 16 with 100 epochs and a learning rate of 0.001. The learning rate decreased by a factor of 0.1 when a metric has stopped improving. The training was stopped if the loss stops decreasing for five epochs. In the case studies, we compare the proposed method with the two baseline approaches, i.e., statistical approach and search-based approach.

Statistical approach. Existing studies [13] enrich each state with a prediction model to conduct a prediction task. Based on these works, we design an approach that builds an individual prediction model for each state and transition in the transition system. Through the prediction model, we can predict the future performance of individual state and transition based on its historical

[1]`https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`
[2]`https://doi.org/10.17632/39bp3vv62t.1`

performances. The statistical algorithms we utilize for this purpose are *linear regression (LR)*, *random forest (RF)*, and *support vector machine (SVR)*.

Search-based approach. The performance of a business process is likely to be repeated with some periodicity. In this regard, one intuitive method to predict future performance in the business process is first to find the most similar past status with the current status and then suggest its next performance record as a prediction. To this end, we take a snapshot for each time window in the form of a one-dimensional vector whose elements indicate the performances of states from the transition system. Next, we calculate the distance between the historical snapshots and the current one. Afterward, we find the prior snapshot having the shortest distance and provide its next snapshot as a prediction. The distance metrics we utilize in this approach are *Euclidean Distance (Euc.)*, *Chebyshev Distance (Che.)*, and *Cosine Distance (Cos.)*.

Note that the proposed method and two baseline approaches are implemented in Python3.6.8, and the source code and supplementary materials required to reproduce the experiments can be found at the Github repository [3].

In each case study, the experiments were performed with 5-fold cross-validation. To measure and compare the prediction accuracy, we used two metrics: Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). Let $y_{i,t}$ and $\hat{y}_{i,t}$ denote the actual and predicted performance at time t at object(i.e., state or transition) i.

5.1. Case study I: healthcare service process

The real-life log used in this case study is from an emergency department in a tertiary hospital in South Korea. It contains event records of the treatment process in the emergency department, collected from January 2018 to December 2018. The log is comprised of 459,700 events by 29,871 patients who visit the

[3] `https://github.com/gyunamister/performance_prediction.git`

emergency department. Each patient goes through 15 activities on average until they leave. The average time for each patient to stay in the department is approximately 8.5 hours. The unique number of activities conducted in the process is 19. In total, 754 resources serve the activities in a shift system.

In this case study, we aim at evaluating the applicability of our proposed method in different settings. To this end, we define four purposeful tasks by discussing with domain experts, each of which represents if the task is short-term or long-term and with enough information or limited information. We consider the next hour prediction as a short-term prediction and the next three-hour prediction as a long-term prediction. Also, the available performance records for 12 times and 24 times of the prediction length are regarded as limited and enough information, respectively. Let tasks be specified with the *(time interval of output, time interval of input)*. The four tasks are as follows:

- Task 1 *with (1,24)* : 1 hour prediction using past 24 hours, i.e., short-term prediction with enough information
- Task 2 *with (1,12)*: 1 hour prediction using past 12 hours, i.e., short-term prediction with limited information
- Task 3 *with (3,72)*: 3 hour prediction using past 72 hours, i.e., long-term prediction with enough information
- Task 4 *with (3,36)*: 3 hour prediction using past 36 hours, i.e., long-term prediction with limited information

For each task, we predict the average sojourn time for states and transitions in the business process model. The experiments were performed using an entire log with 5-fold cross-validation.

Table 1 shows mean absolute error (MAE) and mean absolute percentage error (MAPE) of predicting average sojourn time for states and transitions. Our deep-learning approach generally works better than both the statistical approach and the search-based approach both in the short-term (i.e., Task 1 and Task 2) and the long-term (i.e., Task 3 and Task 4) prediction. The bold font

Table 1: MAE (hours) and MAPE of predicting average sojourn time for states and transitions in healthcare service process using 5-fold cross validation (ST: statistical approach, SB: search-based approach, DL: deep-learning approach)

		Task 1				**Task 2**			
		State		**Transition**		**State**		**Transition**	
		MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
ST	*LR*	0.42 ± 0.01	0.15 ± 0.01	0.15 ± 0.01	0.06 ± 0.01	0.41 ± 0.01	0.14 ± 0.01	0.15 ± 0.01	0.06 ± 0.01
	RF	0.42 ± 0.01	0.14 ± 0.01	0.15 ± 0.01	0.06 ± 0.01	0.41 ± 0.02	0.13 ± 0.01	0.15 ± 0.01	0.06 ± 0.01
	SVR	0.36 ± 0.02	0.14 ± 0.01	0.24 ± 0.01	0.05 ± 0.01	0.35 ± 0.01	0.13 ± 0.01	0.24 ± 0.01	0.06 ± 0.01
SB	*Euc.*	0.42 ± 0.02	0.14 ± 0.01	0.13 ± 0.01	0.07 ± 0.01	0.42 ± 0.02	0.14 ± 0.01	0.13 ± 0.01	0.06 ± 0.01
	Che.	0.41 ± 0.01	0.15 ± 0.01	0.14 ± 0.01	0.08 ± 0.01	0.41 ± 0.01	0.15 ± 0.01	0.14 ± 0.01	0.07 ± 0.01
	Cos.	0.40 ± 0.05	0.14 ± 0.05	0.11 ± 0.01	0.07 ± 0.05	0.40 ± 0.05	0.14 ± 0.05	0.11 ± 0.01	0.07 ± 0.05
DL	*CNN*	0.30 ± 0.02	0.13 ± 0.01	0.10 ± 0.01	0.05 ± 0.03	$\mathbf{0.25 \pm 0.01}$	$\mathbf{0.11 \pm 0.02}$	0.09 ± 0.01	0.05 ± 0.01
	LSTM	0.33 ± 0.11	0.16 ± 0.02	0.11 ± 0.02	0.05 ± 0.01	0.40 ± 0.05	0.21 ± 0.05	0.13 ± 0.05	0.08 ± 0.02
	LRCN	$\mathbf{0.26 \pm 0.02}$	$\mathbf{0.13 \pm 0.01}$	$\mathbf{0.08 \pm 0.01}$	$\mathbf{0.04 \pm 0.01}$	0.26 ± 0.01	0.12 ± 0.01	$\mathbf{0.08 \pm 0.01}$	$\mathbf{0.05 \pm 0.01}$

		Task 3				**Task 4**			
		State		**Transition**		**State**		**Transition**	
		MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
ST	*LR*	0.51 ± 0.01	0.16 ± 0.01	0.34 ± 0.01	0.12 ± 0.01	0.52 ± 0.02	0.16 ± 0.01	0.34 ± 0.01	0.18 ± 0.12
	RF	0.50 ± 0.01	0.15 ± 0.01	0.32 ± 0.01	0.11 ± 0.01	0.52 ± 0.02	0.15 ± 0.01	0.32 ± 0.01	0.11 ± 0.01
	SVR	0.46 ± 0.02	0.15 ± 0.01	0.37 ± 0.01	0.10 ± 0.01	0.47 ± 0.02	0.15 ± 0.01	0.37 ± 0.01	0.10 ± 0.01
SB	*Euc.*	0.42 ± 0.02	0.17 ± 0.01	0.45 ± 0.02	0.11 ± 0.01	0.42 ± 0.02	0.16 ± 0.01	0.45 ± 0.02	0.11 ± 0.01
	Che.	0.41 ± 0.01	0.17 ± 0.01	0.46 ± 0.01	0.11 ± 0.01	0.41 ± 0.01	0.17 ± 0.01	0.46 ± 0.01	0.12 ± 0.01
	Cos.	0.40 ± 0.05	0.18 ± 0.05	0.38 ± 0.01	0.13 ± 0.04	0.40 ± 0.05	0.17 ± 0.05	0.38 ± 0.01	0.17 ± 0.04
DL	*CNN*	0.32 ± 0.05	0.15 ± 0.01	0.32 ± 0.02	0.10 ± 0.01	$\mathbf{0.28 \pm 0.02}$	$\mathbf{0.12 \pm 0.02}$	0.32 ± 0.01	0.10 ± 0.01
	LSTM	0.31 ± 0.06	0.16 ± 0.02	0.34 ± 0.03	0.11 ± 0.01	0.29 ± 0.02	0.13 ± 0.01	0.33 ± 0.03	0.11 ± 0.01
	LRCN	$\mathbf{0.29 \pm 0.01}$	$\mathbf{0.13 \pm 0.01}$	$\mathbf{0.16 \pm 0.01}$	$\mathbf{0.08 \pm 0.01}$	0.29 ± 0.02	0.13 ± 0.01	$\mathbf{0.17 \pm 0.01}$	$\mathbf{0.09 \pm 0.01}$

indicates the best result in a specific task. Our proposed CNN-based model and LRCN-based model show the most accurate predictions in all tasks. In particular, LRCN-based model demonstrates the stable results for all tasks. It attributes to its ability to learn network topology present in the three-dimensional process representation matrix. However, our deep learning approach does not outperform in all tasks. In the short-term prediction (i.e., Task 1 and Task2), the LSTM-based model fails to properly learn the spatiotemporal dependencies underlying the business process when enough information is not given, showing inferior results than baseline approaches. Also, in long-term transition prediction tasks (i.e., transition prediction in Task 3 and Task 4), the CNN-based model and LSTM-based model show inferior results compared to the statistical approach.

5.2. Case study II: BPIC'12

In this case study, we use a real-life log of the application procedure for a personal loan or overdraft at a global financing organization over the six months from October 2011 to March 2012. Approximately 262,200 events regarding 13,087 cases are recorded for the period. This log contains three types of process: one that refers to the states of the application, one that refers to the states of an offer, and one that tracks the states of work items that occur during the approval process. Since we are only interested in the events performed manually, we only investigate the third type.

Using this log, we aim at predicting the average processing time for states and transitions in the next 6 hours, given the historical performance of the last 144 hours. The experiment was performed using 5-fold cross-validation. Table 2 shows the mean absolute error (MAE) and mean absolute percentage error (MAPE) of predicting average processing time for states and transitions. Our proposed deep-learning-based models perform well both in state and transition predictions. In particular, the CNN-based and the LRCN-based prediction models outperform the two baseline approaches both in the state and transition predictions. It verifies that the competency of CNN in extracting critical features is crucial to produce accurate predictions. However, the LSTM-based model shows inferior prediction accuracy in terms of MAPE, compared to LR and RF, failing to learn the temporal dynamics.

5.3. Case study III: Helpdesk

The third case study log concerns a ticketing management system designed for the help desk of an Italian software company. The process starts with the insertion of a new ticket into the ticketing management system. The ticket is managed by resources, and the process ends when the problem is resolved. In this case study, we use 8,988 events by 2,542 cases from January 2011 to June 2012.

Table 2: MAE (hours) and MAPE of predicting average processing time for states and transitions in BPIC'12 using 5-fold cross validation

		State		**Transition**	
		MAE	MAPE	MAE	MAPE
Statistical Approach	*LR*	1.03 ± 0.18	1.20 ± 0.38	0.45 ± 0.06	0.75 ± 0.02
	RF	0.86 ± 0.22	0.70 ± 0.01	0.39 ± 0.06	0.65 ± 0.01
	SVR	0.53 ± 0.09	0.59 ± 0.01	0.30 ± 0.03	0.63 ± 0.01
Search-based approach	*Euc.*	0.69 ± 0.12	0.77 ± 0.05	0.30 ± 0.07	0.79 ± 0.04
	Che.	0.67 ± 0.12	0.78 ± 0.05	0.29 ± 0.06	0.79 ± 0.05
	Cos.	0.45 ± 0.10	0.93 ± 0.15	0.20 ± 0.05	0.93 ± 0.14
Deep Learning approach	*CNN*	0.44 ± 0.10	0.55 ± 0.02	0.19 ± 0.03	0.65 ± 0.10
	LSTM	0.45 ± 0.10	0.87 ± 0.45	0.21 ± 0.03	0.90 ± 0.18
	LRCN	$\mathbf{0.44 \pm 0.09}$	$\mathbf{0.51 \pm 0.04}$	$\mathbf{0.19 \pm 0.03}$	$\mathbf{0.61 \pm 0.03}$

We predict the average sojourn time for states and transitions in the next five days using the historical performances of the last 30 days. As in the previous case study, the experiments were performed using 5-fold cross-validation. Table 3 shows the mean absolute error (MAE) and mean absolute percentage error (MAPE) of predicting average sojourn time for states and transitions. Our proposed deep-learning-based models perform well both in the state prediction and transition prediction in terms of MAE and MAPE. For state prediction, the LSTM-based model and the CNN-based model achieve outstanding performances in terms of MAE and MAPE, respectively. On the other hand, the search-based approach using the Euclidean metric outperforms other approaches in transition predictions.

5.4. Further experiments

In the case studies, we use a transition system with a state representation function $l^s(\sigma) = \langle \pi_A(\sigma(|\sigma|) \rangle$ that represents the partial trace by the sequence of events with the prefix length (i.e., horizon) of 1 and an event representation function $l^e(e) = \pi_A(e)$. We performed the experiments to investigate the effect of the different horizons to the prediction accuracy and computation time. As

Table 3: MAE (days) and MAPE of predicting average sojourn time for states and transitions in Helpdesk using 5-fold cross validation

		State		**Transition**	
		MAE	MAPE	MAE	MAPE
Statistical Approach	*LR*	2.62 ± 0.16	0.64 ± 0.02	1.63 ± 0.05	0.85 ± 0.01
	RF	2.64 ± 0.17	0.63 ± 0.03	1.62 ± 0.06	0.84 ± 0.01
	SVR	2.24 ± 0.17	0.68 ± 0.02	1.31 ± 0.05	0.88 ± 0.01
Search-based approach	*Euc.*	2.58 ± 0.32	0.63 ± 0.02	1.00 ± 0.06	0.59 ± 0.02
	Che.	2.68 ± 0.27	0.63 ± 0.03	0.99 ± 0.06	0.60 ± 0.02
	Cos.	2.49 ± 0.15	0.63 ± 0.01	1.03 ± 0.06	0.58 ± 0.03
Deep Learning approach	*CNN*	2.00 ± 0.21	0.83 ± 0.16	0.79 ± 0.05	0.63 ± 0.03
	LSTM	1.85 ± 0.18	0.75 ± 0.11	$\mathbf{0.72 \pm 0.05}$	$\mathbf{0.58 \pm 0.01}$
	LRCN	$\mathbf{1.71 \pm 0.11}$	$\mathbf{0.61 \pm 0.02}$	0.75 ± 0.05	0.58 ± 0.05

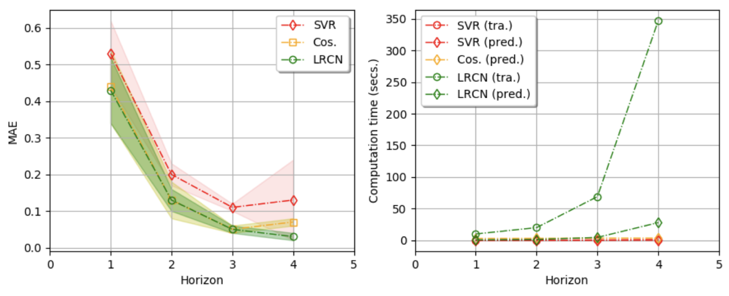
the size of the horizon increases, the transition system has more states and transitions. It enables prediction models to reflect more features when training.

The detailed experiment results are found in the Github repository [4]. The experiment results show that the *Case Study II* using the BPIC'12 event log is affected by the increase of horizon, while there is almost no effect in the first and third case studies. Fig. 5-(a) shows the effects of the different horizons in the *Case Study II*. Note that the most accurate models of each approach (i.e., *SVR*, *Cos.*, and *LRCN*) are selected. The *LRCN* model shows the most improved prediction accuracy as the horizon expands. On the other hand, the extended feature exposes the prediction models to be vulnerable to overfitting. Besides, as depicted in Fig. 5-(b), it needs more computation time for training and prediction when the horizon increases.

6. Discussion

The three case studies on three real-life logs suggest that our proposed methods based on deep neural networks outperform the two baseline approaches by

[4]`https://github.com/gyunamister/performance_prediction.git`



(a) Effect of horizon to MAE

(b) Effect of horizon to computation time

Figure 5: Effects of varying horizon to the prediction accuracy and computation time: experiments on the same setting as in *Case Study II*

successfully learning the temporal evolution and the spatial dependency underlying the business process. The first case study also shows that our proposed method applies to any prediction tasks regardless of the prediction length (i.e., short-term or long-term) and the availability of data (i.e., enough or limited information). Among the deep learning approach, the LRCN-based model shows stable performances on all prediction tasks by learning network topology (i.e., directly-follows relations) that is provided by the three-dimensional process representation matrix.

Existing works on predictive business process monitoring aims at providing timely information that enables proactive and corrective actions to improve the business process. However, these techniques focus on making predictions, not at the process model level, but at the instance level, failing to provide useful knowledge of future problematic points in the business process. In this regard, the proposed method bridges the gap between the process performance mining and the predictive business process monitoring to provide practical information which will facilitate proactive actions to improve the business process.

The proposed approach has some limitations. Firstly, this work does not assess what extent managers can rely on prediction results to make operational management decisions. There are two possible directions to deal with this limitation. The first one is to calculate the required level of prediction accuracy by conducting sensitivity analysis concerning the effect of prediction accuracy in managerial decisions. For example, Park and Song provide the required level of prediction accuracy to execute the prediction-based resource allocation in business processes [28]. In the experiment on a real-life business process, the sensitivity analysis demonstrates that the suggested resource allocation technique is applicable if the prediction accuracy is above 60 percent. The other way to deal with this limitation is to develop prediction models to quantify the prediction uncertainty, such as Bayesian Neural Networks (BNNs), which is a promising direction for future works.

Second, the proposed approach assumes that past behaviors in a business process can be used to predict future behaviors. However, this assumption is invalid if there are some changes in the business environment. The changes affect business processes and lead to concept drift in prediction models. Concept drift means that the relation between the feature and the target variable changes due to the external factors that causes a decrease of the prediction accuracy over time. In this case, the predictive models should be adapted in an online manner, i.e., the models should be updated if the prediction accuracy is below a certain threshold.

Thirdly, we only utilize historical performances as the only input for training the prediction models. In other words, we predict the average waiting time for transitions based on the historical records of average waiting time in the business process. However, these one-to-one matches between the model input and the model output ignore other possible inuencing factors such as other related performance measures, the context of cases, and the availability of resources.

Fourth, our intention to make the method as general as possible leads to several parameters for the practitioners to determine before implementing the proposed method. First of all, constructing a transition system from an event log requires to specify the abstractions to represent the business process better. Second, one needs to define the time window $TW_{p,s}$ by setting the period p and the stride s. Since both of them are numerical values, the possible options for the time window are infinite. Besides, the length of the model input P should be presented before training the model.

Finally, since it is computationally expensive to learn deep neural networks, it took more time to implement our deep learning approach than two baseline approaches. The use of nonlinear activation functions in the neural network makes the optimization problem non-convex. Since the non-convex optimization problem contains many local optima, flat spots, and cliffs, it is challenging to find an optimal solution to this problem. To deal with this problem, the optimizer in deep neural network repeats the steps of evaluating the model and updating the model parameters to step down the error surface. This search process, which is called gradient optimization, is known to be slow.

7. Conclusion

This paper has proposed a novel method for predicting the future performances of a business process on the process model level. The proposed method incorporates the process discovery technique in process mining and the congestion prediction technique in traffic network research. It is composed of three steps where the first step discovers the business process model we aim at analyzing, the second step represents the performance in the business process model as a matrix upon which we can construct prediction models, and the third step builds prediction models to predict the future performances of the business process.

The proposed method has been validated using three real-life event logs from different domains. All three experiments show that our proposed approach successfully predicts the performances at the process model level. It suggests that our proposed method is applicable to various domains with its competency to learn spatiotemporal dependency and reflect the network topology.

Our work has important implications for both research and practice. From an academic research standpoint, the proposed method provides a novel method to predict the performance of business processes at the process model level. As such, it shifts the focus of predictive process monitoring from the instance level to the process model level. Existing works in predictive process monitoring provide predictions at the instance level. This information requires proactive actions for the singular instances, which is infeasible to managers of complex business processes. Instead, the information acquired by predictions at the process model level is more actionable in that it enables managers to identify weaknesses in the process and apply remedial actions to improve them.

Moreover, this research links the realms of traffic research and the business process management based on the analogy of traffic and business processes. This research borrows concepts from congestion prediction in traffic research to predict performance in business processes. This effort can be extended to deploy other prediction tasks such as traffic flow prediction to achieve relevant prediction results in the field of business process management. Further, other techniques developed in traffic research, such as a method for analyzing traffic violations, can be applied to solve related problems in the business process management.

This paper demonstrates the importance of reflecting spatiotemporal information when building a prediction model at the process model level. The experimental results show that the suggested models, which are designed to incorporate spatial dependence and temporal evolution, outperform two baseline approaches, which do not reflect them. Also, learning proper network topology (e.g., directly-follows relations) is another critical aspect of building reliable prediction models.

When it comes to implications for practice, the proposed method gives practitioners a ready to use tool to predict weaknesses in business processes. This information enables them to make more informed decisions for taking corrective and proactive actions to improve business processes and mitigate risks (e.g., by resource allocation and risk notification). The fast evolution of technology combined with the ever-changing needs of customers forces organizations to swiftly and frequently adapt their business processes.The ability to forecast possible problems in business processes and reacting in a proactive manner is one of the most crucial success factors for organizations. Also, while conducting our case study in the hospital, we noted an increasing requirement of executives to flexibly and proactively deal with operational issues.

As future work, we plan to extend the proposed method by incorporating other process-related performance measures and contextual information into the prediction models. We also plan to apply our proposed method to predict the performance measures from different dimensions other than time, such as quality. Second, our proposed method efficiently predicts the future performances of a business process at the process model level and identifies the weakness in the process. However, the predictions must be transformed into concrete remedial actions. In order to deal with this, future works should present a method for improving the performance in business processes by recommending proactive actions with optimization and simulation techniques. Another important direction of future work is to deploy prediction models that quantify the prediction accuracy to support business managers to make decisions on the remedial actions to improve performances and mitigate risks.

Acknowledgment

This paper was supported by Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOTIE) (N0008691, The Competency Development Program for Industry Specialist)

References

[1] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, 2nd Edition, Springer, 2018.

[2] W. M. P. van der Aalst, Process Mining: Data Science in Action, 2nd Edition, Springer, 2016.

[3] A. E. Mrquez-Chamorro, M. Resinas, A. Ruiz-Corts, Predictive monitoring of business processes: A survey, IEEE Transactions on Services Computing 11 (6) (2018) 962–977.

[4] G. Park, M. Song, Prediction-based resource allocation using lstm and minimum cost and maximum flow algorithm, in: 2019 International Conference on Process Mining (ICPM), 2019, pp. 121–128.

[5] S. A. Fahrenkrog-Petersen, N. Tax, I. Teinemaa, M. Dumas, M. de Leoni, F. M. Maggi, M. Weidlich, Fire now, fire later: Alarm-based systems for prescriptive process monitoring, CoRR abs/1905.09568 (2019). arXiv:1905.09568.

[6] W. M. P. van der Aalst, M. Schonenberg, M. Song, Time prediction based on process mining, Information Systems 36 (2) (2011) 450–475.

[7] E. D. Arnheiter, J. Maleyeff, The integration of lean management and six sigma, The TQM Magazine 17 (1) (2005) 5–18.

[8] F. Gullo, From patterns in data to knowledge discovery: What data mining can do, Physics Procedia 62 (2015) 18 – 22, 3rd International Conference Frontiers in Diagnostic Technologies, ICFDT3 2013, 25-27 November 2013, Laboratori Nazionali di Frascati, Italy.

[9] C.-H. Wu, J.-M. Ho, D. T. Lee, Travel-time prediction with support vector regression, Trans. Intell. Transport. Sys. 5 (4) (2004) 276–281.

[10] X. Ma, Z. Tao, Y. Wang, H. Yu, Y. Wang, Long short-term memory neural network for traffic speed prediction using remote microwave sensor data, Transportation Research Part C: Emerging Technologies 54 (2015) 187–197.

[11] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, Y. Wang, Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction, Sensors 17 (4) (2017) 818.

[12] H. Yu, Z. Wu, S. Wang, Y. Wang, X. Ma, Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks, Sensors 27 (2017) 1501.

[13] M. Polato, A. Sperduti, A. Burattin, M. D. Leoni, Time and activity sequence prediction of business process instances, Computing 100 (9) (2018) 1005–1031.

[14] A. Pika, W. M. P. van der Aalst, C. J. Fidge, A. H. M. ter Hofstede, M. T. Wynn, Profiling event logs to configure risk indicators for process delays, in: C. Salinesi, M. C. Norrie, Ó. Pastor (Eds.), Conference on Advanced Information Systems Engineering, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 465–481.

[15] B. Kang, D. Kim, S.-H. Kang, Real-time business process monitoring method for prediction of abnormal termination using knni-based lof prediction, Expert Systems with Applications: An International Journal 39 (2012) 6061–6068.

[16] D. Breuker, M. Matzner, P. Delfmann, J. Becker, Comprehensible predictive models for business processes, MIS Q. 40 (4) (2016) 1009–1034.

[17] J. Evermann, J.-R. Rehse, P. Fettke, Predicting process behaviour using deep learning, Decision Support Systems 100 (2017) 129–140.

[18] N. Tax, I. Verenich, M. L. Rosa, M. Dumas, Predictive business process monitoring with lstm neural networks., in: E. Dubois, K. Pohl (Eds.), CAiSE, Vol. 10253 of Lecture Notes in Computer Science, Springer, 2017, pp. 477–492.

[19] N. Mehdiyev, J. Evermann, P. Fettke, A novel business process prediction model using a deep learning method, Business & Information Systems Engineering (07 2018).

[20] J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems 12 (4) (2011) 1624–1639.

[21] J. Wang, Q. Gu, J. Wu, G. Liu, Z. Xiong, Traffic speed prediction and congestion source exploration: A deep learning method, in: IEEE 16th International Conference on Data Mining, 2016, pp. 499–508.

[22] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–44.

[23] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 160–167.

[24] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, T. Darrell, Long-term recurrent convolutional networks for visual recognition and description, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (4) (2017) 677–691.

[25] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning (2012).

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[27] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML'15, JMLR.org, 2015, pp. 448–456.

[28] G. Park, M. Song, Prediction-based resource allocation using bayesian neural networks and minimum cost and maximum flow algorithm (2019). arXiv:1910.05126.