A Direct Data Aware LSTM Neural Network Architecture for Complete Remaining Trace and Runtime Prediction

Björn Rafn Gunnarsson, Seppe vanden Broucke, and Jochen De Weerdt, *Member, IEEE*

***Abstract*—Developing LSTM neural networks that can accurately predict the future trajectory of ongoing cases and their remaining runtime is an active area of research in predictive process monitoring. In this work a novel complete remaining trace prediction (CRTP) LSTM is proposed. This model is trained to directly predict the complete remaining trace and runtime of cases in contrast to single event prediction as is considered in previously published research on this topic. This makes the CRTP-LSTM robust in terms of utilizing all available attributes of previously observed events for prediction, consequently it can be considered natively data aware. In an extensive experimental assessment the authors show that CRTP-LSTMs consistently outperform other considered approaches for both remaining trace and runtime prediction. Furthermore, the authors show that including all available information contained in previously observed events has a positive impact on the performance of the CRTP-LSTM model. This indicates that valuable information can be extracted from attributes of events in order to make more accurate trace and runtime predictions. This opens up interesting avenues for future research including the incorporation of inter-case features into a modeling setup when predicting the remaining trace and runtime of cases.**

***Index Terms*—Process mining, predictive process monitoring, remaining time prediction, remaining trace prediction, long short-term memory networks.**

I. Introduction

A BUSINESS process is a collection of connected activities, decisions and events which are carried out manually or automatically in order to deliver a product or a service to a customer. Aligned with innovations, e.g., in communication and computing, these processes have become more complex and increasingly reliant on information systems. Consequently, these systems often store detailed information on the process carried out by an organisation. Event logs obtained from these information systems therefore provide a large reservoir of knowledge about the business process in question, such as information on what business activities were carried out, at what time and by whom [1], [2], [3]. Over the past decades, the field of process mining has emerged, encompassing the research area concerned with knowledge discovery from event logs. The overall goal of process mining techniques is to extract insights and information from event logs in order to improve the underlying process. Process mining techniques can be broadly classified into offline techniques, providing an ex-post view of already completed cases, and online techniques, which provide real-time information on the performance of ongoing cases. Predictive process monitoring can be viewed as a subfield of the latter, which concerns itself with developing techniques that continuously provide users with predictions about the future of an ongoing case [1], [4], [5], [6], [7]. These techniques can for example be used to provide predictions on the outcome of an ongoing case (e.g., [8] and [9]), its future trajectory (e.g., [10], [11] and [12]) and its completion time (e.g [13] and [14]).

Inspired by the recent success of recurrent neural networks (RNNs) in the field of natural language processing, research on predictive process monitoring has increasingly focused on the application of these networks, where an event is treated as analogous to a word in natural language processing and an event trace as analogous to a sentence [15]. Recently, a number of papers have been published where RNNs with a Long-Short-Term Memory (LSTM) architecture are utilized in order to simultaneously predict the full remaining trace of a case and its remaining runtime (i.e. [10] and [16]). Developing models that can accurately predict the future trajectory and remaining runtime of a case, has a number of business applications e.g., for operational support, planning and resource allocation. Fundamentally, the same modeling approach is used in previously published research on this topic where an LSTM is trained to predict the attributes of the next event for a running case, e.g., the next activity label and its timestamp. This approach therefore relies on developing an LSTM which is trained to predict a single event. In order to obtain a prediction for the full remaining trace and the remaining runtime, this modeling approach relies on continuously feeding the obtained predictions again into the model in order to obtain predictions for the attributes of events

Manuscript received 28 September 2021; revised 5 September 2022; accepted 1 February 2023. Date of publication 16 February 2023; date of current version 8 August 2023. This research was funded by the KU Leuven research chair 'Smart Airport Operational Analytics' sponsored by Brussel Airport Company NV. Recommended for acceptance by E. Damiani. *(Corresponding author: Bjorn Rafn Gunnarsson.)*

Björn Rafn Gunnarsson and Jochen De Weerdt are with the Research Center for Information Systems Engineering (LIRIS), KU Leuven, 3000 Leuven, Belgium (e-mail: bjornrafn.gunnarsson@kuleuven.be; jochen.deweerdt@kuleuven.be).

Seppe vanden Broucke is with the Research Center for Information Systems Engineering (LIRIS), KU Leuven, 3000 Leuven, Belgium, and also with the Department of Business Informatics and Operations Management, UGent, 9000, Ghent, Belgium (e-mail: seppe.vandenbroucke@kuleuven.be).

Digital Object Identifier 10.1109/TSC.2023.3245726

further into the future. This process of continuous feedback has been denoted as “hallucination” in previously published research in predictive process monitoring [10], [16], [17].

In this work we propose a novel approach for remaining trace and runtime prediction. The suggested approach builds upon previously published research on this topic by utilizing LSTM neural networks. However, in contrast to previous works, we devise a model architecture that allows for training models that can directly predict the complete sequence of remaining events of a case, as well as its remaining runtime. Therefore, we denote our technique as complete remaining trace prediction (CRTP), in contrast to single event prediction (SEP) techniques. A pivotal advantage of CRTP is that it only relies on previously observed events for prediction, even for events that will follow much further into the future of a case, without needing to rely on hallucination, as SEP techniques are required to do. This makes the CRTP model architecture more robust in terms of utilizing all available features relating to previously observed events for prediction, hence it can be considered natively data aware. In an extensive experimental assessment using six real-life datasets, we show that CRTP consistently outperforms other considered modeling approaches for both remaining trace and runtime prediction. Additionally, we demonstrate that our model architecture not only performs better when additional data attributes (i.e., case and event features) are included, but also when only time-based features are provided.

The remainder of this paper is structured as follows. An overview of the corresponding literature will be provided in Section II. Section III introduces relevant background information, i.e., relating to event logs and LSTMs. Furthermore, the SEP-LSTM modeling approach used in previously published research on this topic is discussed in this section. A detailed discussion of our proposed CRTP-LSTM model will be presented in Section IV. The experimental setup and results are presented in Section V, followed by a thorough discussion in Section VI. Finally, the last section of this paper provides concluding remarks and opportunities for further work.

II. Related Work

Developing accurate predictive process monitoring techniques has received increased attention within academia in the past decade. Consequently, a number of approaches have been developed for predicting the remaining trace and runtime of a case. These approaches can be broadly grouped based on whether they directly use a representation of a process model in order to make their prediction or not. The majority of research focused on process model dependent predictive process monitoring rely on mining an annotated transition system from event logs [7], [18]. This research builds upon the pioneering work presented in [13] where the authors illustrate how an annotated transition system can be developed and used to predict the remaining runtime of a case. A number of papers have been published aiming to expand on this work. In [19] the authors construct a stochastic Petri net which they utilize in order to predict the remaining runtime of an ongoing case in order to avoid missing deadlines. However, most relevant is the work published by [20], where the authors construct a transition system and annotate its edges with transition probabilities, which allows the model to predict the full remaining trace and runtime for a running case.

A central issue when developing process model dependent predictive process monitoring techniques relates to identifying a suitable abstraction for traces in the log. Therefore, another group of research works in predictive process monitoring does not rely on a representation of a process model. Rather, these approaches rely on machine learning to construct predictive models using features extracted from event logs [21]. In 2008, [22] suggested an approach where a non-parametric regression technique is used in order to predict the remaining runtime of a given case. Since then, a number of papers have been published expanding on this research by making use of different machine learning algorithms, such as clustering techniques, individual regressors and ensemble regressors (for a detailed overview of these works see e.g., [18]). However, to the best of our knowledge these techniques have not been used for remaining trace prediction.

More recently, research in this field has increasingly shifted its focus towards the applicability of deep learning algorithms for different predictive process monitoring tasks. In particular, LSTMs have been found to be logical candidate models for a number of these tasks due to the sequential nature of event logs. A number of papers have been published in which LSTMs have shown promising results for predicting the full remaining trace and runtime of an ongoing case, outperforming a number of previously proposed predictive process monitoring techniques such as the once proposed by [13], [22] and [20]. Broadly, this group of papers can be classified based on their prediction task. Where two groups of papers solely focus on either remaining trace or runtime prediction, while the third one focuses on both prediction tasks jointly.

In [11] the authors develop a SEP-LSTM which is utilized for remaining trace prediction. In this approach, a model is trained which takes as input a sequence of events containing information on activity labels, life cycle information and resource information. This sequence of events is propagated into an embedding layer before being fed into a stack of LSTM layers. The model architecture is then trained to predict the attributes of the next event for a case. In order to obtain a prediction for the full remaining trace of a case, the model relies on a hallucination approach as described above. Mainly three papers have been published expanding on this work. In [17], a similar LSTM architecture is used. However, the authors expanded on previous work on using LSTMs for remaining trace prediction by incorporating knowledge on process executions into their modeling approach. In 2019, [23] published their work on using SEP-LSTMs for remaining trace prediction. This approach uses a number of categorical attributes of an event. These attributes are fed into a more elaborate LSTM architecture including separate encoders and decoders for each of the categorical attributes. This architecture also consists of a modulator which learns the relevance of each attribute when predicting the attributes of the next event. More recently, [24], used a similar LSTM architecture as proposed by [11] but expand on the architecture by adding an attention layer.

Mainly one paper has been published where an LSTM is constructed in order to predict the remaining runtime of a case. In [25], the authors proposed an LSTM architecture for remaining runtime prediction. In this modeling approach, the model takes as input a sequence of events, containing information on the activity labels and additional categorical and real-value features. The LSTM is then trained to predict the remaining runtime of a case.

Another group of papers has been published proposing models which jointly predict the remaining trace and runtime of an ongoing case. In [10] the authors proposed an SEP-LSTM architecture which can be utilized in order to jointly predict the remaining trace and runtime of a case. This approach takes as input a sequence of events containing information on activity labels and time related information. Each event is transformed into a feature vector containing the one-hot encoded activity label and the values for the different time features before being propagated into a stack of LSTM layers. The LSTM architecture is then trained to jointly predict the next activity label and timestamp. In [16] the authors combine the benefits of [10] and [11] by utilizing embedding layers when constructing an SEP-LSTM architecture for remaining trace and runtime prediction. Jointly learning LSTMs for multiple prediction tasks has been shown to improve on the performance of the model on individual prediction tasks. In their paper, [10], concluded that this is also the case when using LSTMs for remaining trace and runtime prediction. As described before, previously published research on developing LSTMs for remaining trace and runtime prediction make use of essentially the same modeling approach, where a SEP-LSTM network is trained to jointly predict the attributes of the next event such as its activity label and timestamp. In order to obtain a prediction for the full remaining trace and runtime, this approach relies on a hallucination approach, i.e., predicted events are fed back into the model to obtain predictions for events that follow further in the future. In this work, we aim to expand on the current state of the art by (i) proposing a novel modeling approach where a CRTP-LSTM network is trained to directly predict the complete remaining trace and runtime of a case, (ii) proposing an architecture that can natively incorporate all available information contained in the previously observed events for making predictions, in contrast to SEP-LSTM models, (iii) evaluating the different approaches over a number of real-life event logs obtained from a variety of business sectors therefore providing a holistic view on both the relative and the general performance of the considered modeling approaches in addition to (iv) illustrating the value of adequately incorporating available information contained in previously observed events when predicting the remaining trace and runtime of cases.

III. Background

A. Event Logs

The field of process mining comprises a set of techniques developed in order to analyse event logs. An event log usually consists of a collection of timestamped event records, where each event contains information on the execution of a specific work item of a given process. Most process mining techniques require that each event in the log contains at least three attributes: a case identifier, indicating for what case the event was recorded, an activity label, indicating what task the event refers to, and a timestamp, indicating when the event was recorded. Additionally, event logs often contain other information such as event attributes, which are specific to each event (e.g., resource) and case attributes, which are shared by all events of the same case instance (e.g., requested loan amount in a loan application process) [1]. Based on the above and in alignment with [26] we define the following:

An event is a tuple $e = (c, t, a, d_1, \ldots, d_m)$ where c is the case identifier, t is the timestamp, a is an activity label and $d_1, \ldots, d_m$ a number of additional attributes with $m \geq 0$. A labeling function maps an event to the value of one of its attributes. Let $Case : e \rightarrow c$, $Activity : e \rightarrow a$, $Time : e \rightarrow t$ and $Attribute_m : e \rightarrow d_m$ be functions that return the case identifier, activity label, timestamp and the attribute(s) for a given event. Given these attributes, events can be grouped into traces which provide a perfectly ordered sequence of events for each case instance (process execution) where each event refers to a task and all tasks in the process are included in the event log. A trace is a finite non-empty sequence of events $\sigma = \langle e_1, \ldots, e_n \rangle$, where n is the length of the trace and $e_i \in \sigma$ is the event at position i in trace σ. It holds that $\forall e_i, e_j \in \sigma, i < j : Time(e_i) < Time(e_j) \wedge Case(e_i) = Case(e_j)$, i.e., the events in the trace are ordered w.r.t. time and all events in the trace share the same case identifier. A distinction can be made between two types of attributes of events, namely case and event attributes. Case attributes relate to a characteristic of the case to which the trace of events belongs to. Therefore, the value of a case attribute is constant throughout the processing of a case, i.e., $\forall e_i, e_j \in \sigma, i < j : Attribute_m(e_i) = Attribute_m(e_j)$. Event attributes, on the other hand, are event specific. Consequently, their value can change throughout the processing of a given case.

Predictive process monitoring techniques provide users with predictions about the future of an ongoing case. Accordingly, these techniques rely on functions that partition the traces into a set of prefixes and suffixes. Given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$ and two positive integer $t < n$ and $w < n$, we define $Prefix(\sigma, t, w) = \langle e_{t-w+1}, \ldots, e_t \rangle$ and $Suffix(\sigma, t, w) = \langle e_{t+1}, \ldots, e_{t+w} \rangle$. By applying the $Activity$ function to the prefix and suffix one can obtain an activity prefix and suffix, i.e., $Activity(Prefix(\sigma, t, w)) = \langle a_{t-w+1}, \ldots, a_t \rangle$ and $Activity(Suffix(\sigma, t, w)) = \langle a_{t+1}, \ldots, a_{t+w} \rangle$.

The focus of this work is on developing predictive process monitoring methods that jointly predict the remaining trace and runtime of an ongoing case. Therefore, in addition to relying on functions which partition traces into prefixes and suffixes, we also need to define a function that computes the remaining runtime of cases. Given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$ and a positive integer $t < n$, we define $Remtime(\sigma, t) = r_t = Time(e_n) - Time(e_t)$ which provides the remaining runtime at time step t. By applying this function to the suffix at time step $t - 1$ one obtains a sequence containing the current and future remaining runtimes i.e., $Remtime(Suffix(\sigma, t-1, w)) = \langle r_t, \ldots, r_{t+w} \rangle$.

B. Long Short-Term Memory Networks

Long short-term memory networks (LSTMs) are a form of recurrent neural networks (RNNs). Conventional RNNs are augmented with edges that span adjacent time steps. At time t the network takes as input the current input x_t and the previous hidden state h_{t-1}. This information is used to compute a new hidden state: $h_t = f(Ux_t + Wh_{t-1} + b_h)$, which contains information from all previous time steps up to t. Here U, W are weight vectors and b_h a bias term. The weight vectors and bias terms are parameters learned during the training phase. Compared to conventional RNNs, LSTMs are better able to learn long term dependencies when processing sequential data. This is achieved by the introduction of the so-called memory cell as a unit of computation. The memory cell of an LSTM model can be described using the following formulas:

$$f_t = sigmoid(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = sigmoid(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C_t} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3}$$

$$C_t = f_t * C_{t-1} + i_i * \tilde{C_t} \tag{4}$$

$$o_t = sigmoid(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * tanh(C_t), \tag{6}$$

where the W-variables are weight parameters and the b-variables are biases, which are learned during the training process. The forget gate (f_t), input gate (i_t) and output gate (o_t) carry out different functions within the cell. However, as can be seen in the above equations, the same computation is carried out in these gates yielding values between 0 and 1. In (3), new candidate values ($\tilde{C_t}$) for the cell state are computed. These candidate values are combined with the cell state from the previous time step in (4). Here, the forget gate determines how important the previous cell state is given the current input (x_t) and previous hidden state (h_{t-1}). Similarly, the input gate determines how important the obtained candidate cell state is. Lastly, the output gate determines what values of the cell state are used to form the new hidden state (h_t). The LSTM cell propagates two states between time steps, namely the cell state and the hidden state. Based on the above equations, one can see that conceptually these states seem to serve rather distinct tasks where the cell state holds an aggregate memory of previously observed time steps, while the hidden state provides a depiction of the most recent time step [11], [27]. Standard LSTM layers are one directional, propagating over an input sequence in a stepwise manner starting from the first time step. However, in some applications it is beneficial to additionally consider "future" values of an input sequence in order to make a prediction. Consequently, bidirectional LSTM layers [28] have been developed. A bidirectional LSTM layer consists of two one directional LSTM layers, one which moves over the input sequence starting from the first time step and another which moves in the other direction starting from the most recent time step. Consequently, all information contained in an input sequence, e.g., a prefix of events, can be incorporated during prediction. All LSTM architectures considered in this work were constructed using bidirectional LSTM layers.

TABLE I
PARTITIONING OF TRACES INTO PREFIXES FOR SEP-LSTM MODEL

t	w	*Prefix*$(\sigma,\ t,\ w)$
1	3	$\langle 0,\ 0,\ e_1\rangle$
2	3	$\langle 0,\ e_1,\ e_2\rangle$
3	3	$\langle e_1,\ e_2,\ e_3\rangle$
4	3	$\langle e_2,\ e_3,\ e_4\rangle$
5	3	$\langle e_3,\ e_4,\ e_5\rangle$
$\sigma = \langle e_1,\ e_2,\ e_3,\ e_4,\ e_5,\ e_6\rangle$		

C. The Existing Approach: LSTMs for Single Event Prediction

As discussed previously, essentially the same modeling approach is used in previously published research on developing LSTMs for jointly predicting the remaining trace and runtime of an ongoing case. In this approach, traces in the event log are parsed into multiple prefixes (t) of a fixed length (w), which are zero padded where necessary. This is illustrated in Table I using a simple example trace consisting of six events.

Each event in the prefix is transformed into a feature vector containing the one-hot encoded activity label. Optionally, this feature vector can be expanded to include additional attributes such as time information (e.g., time of day). The model architecture used in this approach is shown in Fig. 1. As can be seen, a prefix of feature vectors is sequentially propagated into a stack of LSTM layers consisting of a single shared LSTM layer and two separate LSTM layers for each of the prediction tasks. The shared layer takes as input the sequence of feature vectors and outputs a sequence of hidden states. This sequence is propagated into the two separate LSTM layers. The last learned hidden states of these layers are then utilized to obtain separate predictions for the activity label of the next event and the time which will elapse until the next event will be observed, i.e., $Elapsed(e_{t+1}) = Time(e_{t+1}) - Time(e_t)$.

This model architecture provides a prediction for the activity label of a single future event and the time which will elapse until this event will be observed. In order to obtain a prediction for the full remaining trace and the runtime of a case, this approach relies on applying a hallucination approach. The hallucination approaches utilized in previously published research on this topic share similar characteristics. The predicted activity label and elapsed time is utilized to construct a new event which is appended to the original prefix, which in turn is fed into the model in order to obtain new predictions. This process is continuously repeated until a special activity label, indicating that the case has ended, is predicted to be the most likely next activity label. In this manner, one obtains a prediction for the full remaining trace consisting of a sequence of future activity labels. In order to obtain the prediction for the remaining runtime, the approach simply adds all predicted elapsed times. When predicting the next activity label, the model outputs a probability distribution over all possible activity labels. Early approaches for remaining trace and runtime prediction utilized a simple greedy search algorithm where at each step the activity label

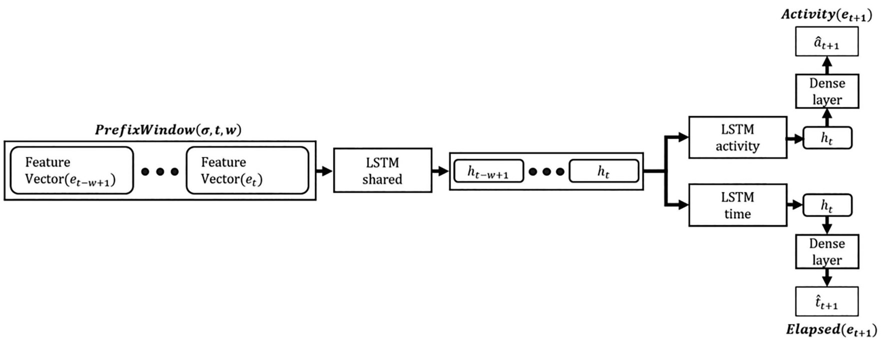

Fig. 1. SEP-LSTM: Main model components.

with the highest predicted probability is chosen as the most likely next activity. Although intuitive, more novel and better performing search algorithms have been proposed. Beam search is one such search algorithm, for which not only the activity label with the highest predicted probability is considered during prediction. Instead the k-most probable activity labels are chosen based on their predicted probability and expanded during the search [29].

As described above, this approach is constructed using a SEP-LSTM network and therefore relies on both a prefix containing previously observed events and hallucinated events in order to produce a prediction for the full remaining trace and runtime. At each step during the search, the model provides a prediction for the next activity label and the elapsed time. Therefore, this modeling approach is able to utilize both activity labels and time features during prediction and update them during the search. Case features, extracted from case attributes, are shared by all events belonging to the same case. Consequently, the values of case features are known from the beginning of a process execution. Therefore, this modeling approach can be quite naturally expanded to include information on case features. However, event features, extracted from event attributes, are event specific, therefore this modeling approach has no straightforward way of incorporating event features. One approach to incorporate these features would be to modify this modeling approach by continuously expanding the model architecture so that it can predict future values of additional event features in addition to predicting the next activity label and elapsed time. However, events often contain information on a large number of highly dimensional and rapidly changing attributes, which makes this approach infeasible in many cases. This can be clearly observed from previously published research on this topic. [11] experimented with including the resource information contained in events into the modeling setup and concluded that in general this leads to a significant performance drop for the models since resource information does not follow any clear underlying regularity, i.e., due to its dimensionality and rapidly changing nature. In an attempt to counter this, [16] proposed using a feature engineering approach where resource roles are mined from the raw resource information and included into the modeling setup. Mining roles from resource information seems sensible, however as mentioned here above, events often contain information on a large number of highly dimensional features. No general feature engineering approach has been suggested in order to preprocess event features so that they can be utilized by a SEP-LSTM model. Therefore, to the best of our knowledge, this modeling setup has no straightforward way of incorporating event features.

TABLE II
PARTITIONING OF TRACES INTO PREFIXES AND SUFFIXES FOR CRTP-LSTM MODEL

t	w	$Prefix(\sigma,\ t,\ w)$	$Activity(Suffix(\sigma,\ t, w))$	$Remtime(Suffix(\sigma, t-1, w))$
1	5	$\langle 0,\ 0,\ 0,\ 0,\ e_1 \rangle$	$\langle a_2,\ a_3,\ a_4,\ a_5,\ a_6 \rangle$	$\langle r_1,\ r_2,\ r_3,\ r_4,\ r_5 \rangle$
2	5	$\langle 0,\ 0,\ 0,\ e_1,\ e_2 \rangle$	$\langle a_3,\ a_4,\ a_5,\ a_6,\ 0 \rangle$	$\langle r_2,\ r_3,\ r_4,\ r_5,\ 0 \rangle$
3	5	$\langle 0,\ 0,\ e_1,\ e_2,\ e_3 \rangle$	$\langle a_4,\ a_5,\ a_6,\ 0,\ 0 \rangle$	$\langle r_3,\ r_4,\ r_5,\ 0,\ 0 \rangle$
4	5	$\langle 0,\ e_1,\ e_2,\ e_3,\ e_4 \rangle$	$\langle a_5,\ a_6,\ 0,\ 0\ ,0 \rangle$	$\langle r_4,\ r_5,\ 0,\ 0,\ 0 \rangle$
5	5	$\langle e_1,\ e_2,\ e_3,\ e_4,\ e_5 \rangle$	$\langle a_6,\ 0,\ 0\ ,0\ ,0 \rangle$	$\langle r_5,\ 0,\ 0,\ 0,\ 0 \rangle$
$\sigma = \langle e_1,\ e_2,\ e_3,\ e_4,\ e_5,\ e_6 \rangle$				

IV. A Data Aware LSTM Neural Network Architecture for Complete Remaining Trace and Runtime Prediction

The main motivation for this paper is to add to the existing body of literature on using LSTMs for predictive process monitoring by proposing a novel modeling approach for remaining trace and runtime prediction. In contrast to previous works, the model architecture is designed to directly predict the complete remaining trace consisting of activity labels and the remaining runtime of a case in contrast to single event prediction[1]. In this approach, traces in the event log are again parsed into multiple prefixes (t) of fixed length (w). Additionally, two suffixes consisting of the remaining activity labels and runtimes, are constructed. This is illustrated in Table II, zero padding is used in order to keep the length of all prefixes and suffixes constant.

As before, each event in the prefix is transformed into a feature vector. As shown in Fig. 2, this prefix of feature vectors is sequentially propagated into a LSTM stack which consists of a single shared LSTM layer and a separate LSTM layer for each prediction task. As mentioned here above, all LSTM based architectures considered in this work are constructed using bidirectional LSTM layers. Consequently, these layers are

1.Code illustrating the training mechanism and structure of the proposed model can be found at https://github.com/bjornragu/CRTP-LSTM

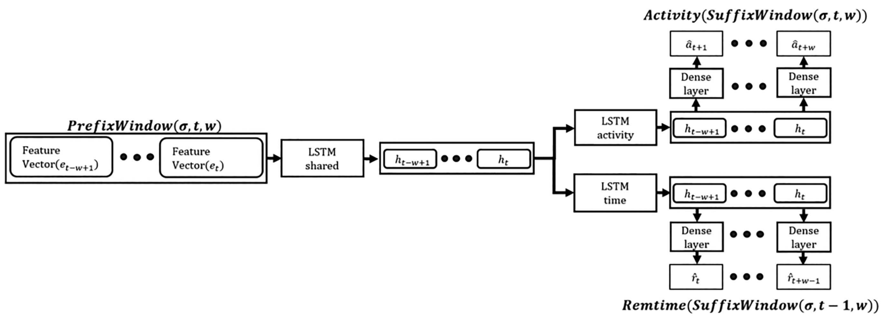

Fig. 2. CRTP-LSTM: Main model components.

able to use all information contained in their input sequence. The shared LSTM shown in the figure for example consists of two one directional LSTM layers where one layer sequentially propagates from left to right, starting from the first feature vector of the prefix, and the other moves from right to left, starting from the most recent feature vector. As an example, one can look at the prefix given at the fifth time step in Table II. The shared LSTM would sequentially propagate over this prefix, where a single one directional LSTM starts by processing the feature vector extracted from the first event of the prefix (i.e., e_1) and another one directional LSTM starts by processing the feature vector extracted from the last event of the prefix (i.e., e_5). The shared LSTM returns a sequence of hidden states, containing a concatenation of the hidden states produced by each of the one directional layers of this bidirectional LSTM layer. These hidden states are then sequentially propagated into a separate bidirectional LSTM layer for each prediction task. The two separate bidirectional LSTM layers process this input sequence of hidden states in the same manner as the shared LSTM, learning a relevant representation at each time step using the full input sequence of hidden states. The model is trained to directly predict the complete remaining trace consisting of a sequence of activity labels and a sequence of remaining runtimes. Therefore, the two separate LSTMs, used for activity and time prediction, both return a sequence of hidden states which are then used to predict sequences of activity labels and remaining runtimes. Conceptually, the motivation for this modeling choice is that, given sufficient training, the cell state will learn to accurately propagate relevant information between time steps and that the sequence of hidden states returned by the two separate LSTMs will come to hold information on the future events to be predicted (i.e., their activity labels and remaining runtimes).

During prediction, the model outputs a sequence of probability distributions for future activities and a sequence of remaining runtime predictions. In order to obtain a prediction for the full remaining trace and runtime, the most probable activity label at each future time step is greedily chosen along with the runtime prediction until an activity label indicating that the case has ended is chosen as the most likely activity label. It should be noted that the elapsed times between predicted events can be easily obtained using this approach by computing the difference between the predicted remaining runtimes, e.g., $\hat{r}_{t+1} - \hat{r}_t$. Consequently, the approach can predict remaining traces that contain both information on activity labels of future events as well as their timestamps.

Sentences in a natural language and traces consisting of process events share a number of interesting characteristics, e.g., natural language is constrained by grammatical rules while the structure of traces is determined by a underlying process logic [11]. However, there are also some stark differences. Events contain a number of case and event features contrary to words in a natural language. When predicting a sequence of future activity labels and the remaining runtimes, the values of these features, contained in previously observed events, can provide valuable information, which can be utilized during prediction in order to improve the performance of the predictive model. The CRTP-LSTM network proposed here is trained to directly predict the full remaining trace and a sequence of remaining runtimes for an ongoing case. One benefit of this is that the model only relies on previously observed events during prediction. Consequently, this modeling approach can naturally utilize all relevant information contained in previously observed events when predicting the remaining trace and the runtime of a case.

V. Experimental Evaluation

A. Event Logs

In order to evaluate the considered modeling approaches, six real-life event logs were used. Four of the event logs have been made available for the BPI Challange (BPIC). That is, the BPIC20,[2] BPIC19,[3] BPIC17[4] and BPIC12.[5] The BPIC12-Sub event log is a filtered version of the BPIC12 event log where events having the same activity label as the previously occurring event are removed. Lastly, the BAG event log is an event log obtained from a luggage system at an international airport, namely Brussels Airport. No cases were removed from the majority of the event logs. However, although a majority of cases in the BPIC19 event log started their process execution in 2018 ($\approx 97\%$) a few cases had events dating long before that (e.g., 1948). Therefore, a number of outlying cases were removed based on in what year they were processed. Additionally, cases which only consisted of a single event were removed from this event log.

Information on the characteristics of the different event logs considered is provided in Table III. It details the business sector

2.10.4121/uuid:3f422315-ed9d-4882-891f-e180b5b4feb5
3.10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1
4.10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
5.10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

TABLE III
CONSIDERED EVENT LOGS AND THEIR CHARACTERISTICS

Event log	Sector	Cases	Variants	Avg. TL	Coverage	Structure	Affinity	Case feat.	Event feat.	Variability (EF)
BPIC20	University	10366	98	5.43	0.82	0.87	0.83	1	2	0.84
BPIC19	Retail	248455	11909	6.38	0.38	0.72	0.62	10	3	0.95
BPIC17	Finance	31509	4047	17.83	0.21	0.77	0.61	3	7	0.35
BPIC12	Finance	13087	4366	20.04	0.43	0.79	0.36	1	1	0.38
BPIC12-Sub	Finance	13087	1309	11.95	0.49	0.81	0.46	1	1	0.41
BAG	Operations	432357	7826	5.29	0.35	0.83	0.46	14	2	0.54

from where the event log originates (Sector), the number of cases (Cases), and the number of variants of sequences of activity labels (Variants). Furthermore, we report on the average trace length (Avg. TL) and the number of available case features (Case feat.) and event features (Event feat.). Additionally, in order to provide an initial view of the variability in event features we report on the average variability of the most variable event feature in each event log (Variability (EF)). In order to obtain this measure, we count the number of value changes observed for an event feature in a trace and normalized this value by the trace length. This computation was carried out for all traces and event features in the considered event logs. We then computed the average variability of all event features available in the event logs. Finally, we report on the average variability of the event feature with the highest average variability for each event log. In order to provide an initial view of the complexity of the considered event logs, we also report on the fraction of cases that share the three most frequently observed traces of activity labels (Coverage) and the average similarity between all traces of activity labels (Affinity). Similarity is here measured using Levenshtein similarity. Levenshtein distance counts the minimum number of insertions, deletions and replacements needed in order to make two traces equal [30]. The obtained distance is normalized and subtracted from 1 in order to obtain a normalized Levenshtein similarity. Lastly, we report on the structure of the traces of activity labels in the different event logs (Structure) as defined by [31]. The structure of an event log is measured by computing the inverse relative amount of direct following relation compared to the maximal amount of direct following relations possible. It can therefore be seen as a measure of the amount of observed behavior compared to the amount of theoretically possible behavior. As can be seen from the table, the considered event logs are both varied in terms of size (i.e., number of cases, trace lengths and available features) and complexity (i.e., number of variants, variability of event features, coverage, affinity and their structuredness). Observe that time independent data splits were applied when training and evaluating the considered models. Hereto, cases in the event logs were first order based on their timestamps. For a majority of the event logs, the last occurring quarter of cases was chosen as a test set and the remaining cases were used to train the model. Cases in the training set that were still ongoing when the first case in the test set was being executed were removed, resulting in a time independent training and test set. Lastly, a quarter of the time independent training set was randomly sampled and used as a validation set. This procedure was carried out for all but one event log, namely the BPIC19 event log. A continuous decreasing trend in execution times was observed for this event log which seems to be a consequence of how the event log was extracted. When a time independent data split was carried out where the last occurring quarter of cases were chosen as a test set, as described here above, this resulted in a bias where execution times in the training set were in general much higher than in the test set. In order to counter this, a time independent data split was carried out where the last occurring half of cases was chosen as the test set and the remaining cases were used to train the models. Cases in the training set that were still ongoing when the first case in the test set was being executed were again removed. In order to obtain a validation set, a quarter of the resulting time independent training set was randomly sampled.

B. Configuration of Models

Configuration of SEP-LSTM Models. The SEP-LSTM based approach presented in previously published research on jointly predicting the remaining trace and runtime of a case was first proposed by [10]. As described above, in this approach a feature vector is constructed that contains the one-hot encoded activity labels and a number of time related features. In total four time related features are extracted indicating the time of week, time of day, duration of case so far and elapsed time since last observed activity. As described above, this feature vector is propagated into a stack of LSTM layers. In order to obtain a prediction for the next activity label, a softmax activation function is used. However, a linear activation function is used to obtained a prediction for the elapsed time until the next activity will be observed. Based on literature recommendations and explorations, a number of alterations to this modeling setup were carried out. First, rather than considering continuous attributes indicating the time of the week and time of the day, categorical attributes indicating the day of the week and the hour of the day were extracted. When propagating categorical attributes into the stack of LSTMs, embedding layers were used so that the model can learn more interesting associations between attributes. Second, in [10] a linear activation function is applied when predicting the time which will elapse until the next activity is observed. However, elapsed times cannot be negative therefore a linear function does not seem a sensible choice. Therefore, a variant of Relu namely parametric ReLU (PReLU) (see e.g., [32]) was considered instead. Also, in [10], the LSTM stack is constructed using unidirectional LSTM layers which read their input sequence from left to right. Contrary to this approach, LSTM stacks in this work are constructed using bidirectional LSTM layers, which are allowed to read the input

TABLE IV
AN OVERVIEW OF ALL CONSIDERED MODELS

Baseline	Time aware	Data aware: CF	Data aware: CF + EF
TS($w = 1, k = 1$)	SEP-LSTM ($k = 1$)	SEP-LSTM ($k = 1$)	SEP-LSTM ($k = 1$)
TS($w = 1, k = 3$)	SEP-LSTM ($k = 3$)	SEP-LSTM ($k = 3$)	SEP-LSTM ($k = 3$)
TS($w = 2, k = 1$)	CRTP-LSTM	CRTP-LSTM	CRTP-LSTM
TS($w = 2, k = 3$)	-	-	-

sequence from both left to right and from right to left. Finally, the approach used in [10] relies on a simple and intuitive search procedure in order to obtain a prediction for the full remaining trace and remaining runtime where at each step the activity label with the highest predicted probability is chosen. However, as discussed above, more novel search procedures have been proposed in the literature including beam search. Therefore, two search procedures are considered here, i.e., arg-max search (as utilized in [10]) and beam search were the beam width was set to 3 (i.e., $k = 3$).

Configuration of CRTP-LSTM Models. In order to evaluate the relative performance of CRTP-LSTM models compared to SEP-LSTM models the configurations of the two approaches were in general kept the same. Again, embedding layers are utilized when propagating categorical attributes to the stack of LSTM layers in the CRTP-LSTM model. However, as described in the last section, the stack of LSTM layers is trained to predict the complete remaining trace consisting of activity labels and a sequence of remaining runtimes. When predicting activity labels and remaining runtimes softmax and PReLU activation functions were again considered. Lastly, bidirectional LSTM layers were again preferred over unidirectional LSTM layers when training CRTP-LSTM models.

Considered Models. An overview of the considered models is given in Table IV. The time aware models take as input time related information extracted from the events in addition to information on the activity labels of previously observed events as discussed above. Data aware case feature models (Data aware: CF) take as input all available case features in addition to information utilized by time aware models. Additionally, data aware case and event feature models (Data aware: CF + EF) use all available event features and therefore utilize all available information contained in events, namely case and event features in addition to time features and activity labels. Lastly, transition systems (TS) are included in our comparisons as baseline models in order to provide a relative view of the performance of the more complex LSTM based approaches. In order to construct a transition system, traces in the event log are again parsed into multiple prefixes (t) of a fixed size (w) as illustrated in Table I. Each prefix is then mapped to a state, which here is a sequence of previously observed activity labels. During training the states are annotated with measurements, i.e., the remaining runtime of a case and the next observed activity label. During prediction, observed prefixes are again mapped to states. When predicting the remaining runtime of a case, the average remaining runtime observed during training for a given state is predicted. In order to predict the next activity label proportions for all possible next activity labels are computed corresponding to their frequency in the training set for each state. Like before, both arg-max and beam search are considered in order to predict the full sequence of activity labels and a sequence of remaining runtimes (for a more detailed discussion on transition systems see e.g., [13], [20]). As discussed in a previous section, the SEP-LSTM model provides a prediction for the time that will elapse until the next event will be observed in a addition to a prediction for the activity label of that event. Therefore, this approach can naturally use time related features since they can be updated during the search. Case features are case specific. Consequently, this modeling approach can also be expanded to include case features. Event features, on the other hand, are event specific. One way to incorporate event features into this modeling approach would be to continuously expand the model architecture so it can provide predictions for future values of considered event features. However as discussed before, events can contain information for a large number relevant event features. Furthermore, these features are often highly dimensional and rapidly changing, which makes them hard to predict. After experimentation, it was found that keeping event features constant at their last observed value resulted in better prediction for the remaining trace of activity labels and runtimes. Therefore, this approach was preferred. As previously discussed, the CRTP-LSTM model relies solely on information from previously observed events during prediction, instead of observed and hallucinated events. Therefore, this approach is designed so that it can make use of information from a large number of case and event features.

Optimizer, Feature Scaling and Window Size. As in [10], the Adam optimiser with Nesterov momentum was used in order to train the LSTM based models. Embedding layers were used when propagating categorical case and event features in the same manner as with categorical time features discussed above. Continuous features and elapsed times were scaled by dividing their value with their average value as in [10]. Remaining runtimes were scaled by dividing their value with the remaining runtime value at a given percentile, e.g., the 95th percentile. For three of the event logs, namely BPIC12, BPIC12-Sub and BPIC17, a logarithmic transformation was carried out for the remaining runtimes before scaling. In order to set the length of the prefix for the SEP-LSTM based approach, four values were selected with equal step size from the interval $[0.5 \times median, 1.5 \times median]$, we then report on the average performance over the considered prefix lengths. When training models with the CRTP-LSTM model, the window size was set equal to the number of events in the longest observed trace in the training set. The intuition for this being that the model will in general be able to directly provide a complete prediction for traces of future process executions. Lastly, dropout regularization was considered where two versions of each LSTM model were constructed, one where the dropout rate was set to 20% and another where no dropout was implemented, the model with the lower validation loss was then included in the final comparison.

C. Results

The obtained results from a comparison of the considered modeling approaches based on six real-life event logs is provided in this section. The models are evaluated for remaining trace

TABLE V
PREDICTION RESULTS FOR TRACE PREDICTION EVALUATED USING LEVENSHTEIN SIMILARITY. WE FIRST COMPUTE THE AVERAGE LEVENSHTEIN SIMILARITY OVER ALL PREFIXES FOR A EACH CASE AND REPORT ON THE AVERAGE OVER ALL CASES IN THE TEST SET. THE BEST PERFORMING MODEL IN EACH GROUP OF MODELS IS GIVEN IN BOLD. AN UNDERSCORE IS USED TO INDICATE THE OVERALL BEST PERFORMING MODEL

	BPIC12	BPIC12-Sub	BPIC17	BPIC19	BPIC20	BAG
Baseline						
TS($w=1, k=1$)	0.0811	0.3284	0.1642	0.6758	**0.9335**	0.2997
TS($w=1, k=3$)	0.4020	0.4426	0.5256	0.6390	0.9335	0.6172
TS($w=2, k=1$)	0.3849	0.4419	0.1649	0.7074	0.9335	**0.6723**
TS($w=2, k=3$)	**0.4803**	**0.5538**	**0.5217**	**0.7074**	0.9335	0.6493
Time aware						
SEP-LSTM ($k=1$)	0.3060	0.3619	0.6194	**0.6999**	**0.9303**	0.7215
SEP-LSTM ($k=3$)	0.4258	0.5101	**0.6315**	0.6972	0.9195	0.7260
CRTP-LSTM	**0.5436**	**0.6161**	0.6078	0.6925	0.9282	**0.7273**
Data aware: CF						
SEP-LSTM ($k=1$)	0.4792	0.5955	0.6202	0.7237	**0.9414**	0.9015
SEP-LSTM ($k=3$)	0.4913	0.6109	**0.6363**	0.7281	0.9383	**0.9049**
CRTP-LSTM	**0.5446**	**0.6341**	0.6149	**0.7485**	0.9406	0.8935
Data aware: CF + EF						
SEP-LSTM ($k=1$)	0.4510	0.5265	0.5187	0.3209	0.6644	0.9057
SEP-LSTM ($k=3$)	0.4716	0.5372	0.5313	0.3283	0.6728	**0.9074**
CRTP-LSTM	**0.5496**	**0.6439**	**0.6906**	**0.7499**	**0.9419**	0.9037

and runtime prediction for all prefixes of cases in the test set. The normalized Levenshtein similarity between the actual and predicted trace was used in order to evaluate the performance of the models when predicting the full remaining trace consisting of activity labels. It should be noted here that previously published research on using LSTMs for remaining trace prediction has in general relied on a normalized version of the Damerau-Levenshtein similarity, which adds a swapping operation to the set of operations used by the Levenshtein similarity, in order to evaluate trace similarity. This is usually motivated by the fact that business processes can include branches where two events are executed in parallel. However, this assumes that LSTMs are able to learn parallel behavior, which has not been shown to be the case. Furthermore, even if LSTMs had been shown to be able to learn parallel behavior, it would only seem sensible to allow for swapping of events for events that are truly executed in parallel, but not in general. Therefore, the more restrictive Levenshtein similarity will be used here.

The prediction results for trace prediction are provided in Table V. As can be seen from the table, the data aware case and event feature CRTP-LSTM model, which utilizes all available information contained in previously observed events, performs best on all event logs considered, with one exception namely the BAG event log where a data aware case and event feature SEP-LSTM model performs best. However, the differences in performance between data aware case and event feature models is negligible when evaluated for trace prediction on this event log. From the table we can also see that a competitive solution is obtained using a transition system for trace prediction on a number of the event logs. A good solution in both relative and absolute terms is for example obtained using a transition system on the BPIC20 event log. Additionally, we can see that applying beam search when using transition systems for remaining trace prediction in general results in a significant performance increase for these models. Also, when considering the results for the time aware models, we can see that the CRTP-LSTM model is the best performer for remaining trace prediction on three of the considered event logs, namely the BPIC12, BPIC12-Sub and the BAG event log. It should be noted here that the difference in performance between time aware models is negligible on the BAG event log. Furthermore, no meaningful performance differences are observed for time aware models on the BPIC19 and BPIC20 event log. However, a SEP-LSTM model outperforms the time aware CRTP-LSTM on the BPIC17 event log. Similarly, when the results for data aware case feature models are observed we can see that the CRTP-LSTM model is the best performing model on three out of the considered event logs, namely the BPIC12, BPIC12-Sub and BPIC19 event logs. However, the SEP-LSTM model outperforms the data aware case feature CRTP-LSTM model on the BPIC17 and also to a lesser extent on the BAG event log. No meaningful performance differences are observed between data aware case feature models on the BPIC20 event log. Lastly, when the LSTM based models are allowed to utilize all available information contained in previously observed events, the data aware case and event feature CRTP-LSTM model is the best performing model on all event logs with one exception, namely the BAG event log, as mentioned here above.

In order to evaluate the performance of the different models when predicting the remaining runtime of cases both the mean absolute error (MAE) and the root-mean-square error (RMSE) were considered. The results for remaining runtime prediction are provided in Table VI. We report on the prediction error in minutes on all event logs except for the BAG event log for which errors are reported in seconds, since process executions in that event log in general take shorter time to complete than in other event logs considered here. As can be seen from the table, the data aware case and event feature CRTP-LSTM model, which utilizes all available information contained in previously observed events, is the overall best performer for predicting the remaining runtime of cases. More specifically, it is the best performing model for remaining runtime prediction on all event logs considered here when evaluated using the MAE. When evaluated using the RMSE, the data aware case and event feature CRTP-LSTM model is the overall best performing model on all event logs with two exception, namely the BPIC12 where a

TABLE VI
PREDICTION RESULTS FOR REMAINING RUNTIME PREDICTION EVALUATED USING MEAN ABSOLUTE ERROR (MAE) AND ROOT MEAN SQUARE ERROR (RMSE). WE COMPUTE THE MAE AND RMSE OVER ALL PREFIXES FOR EACH CASE AND REPORT ON THE AVERAGE MAE AND RMSE OVER ALL CASES. THE BEST PERFORMING MODEL IN EACH GROUP OF MODELS IS GIVEN IN BOLD. AN UNDERSCORE IS USED TO INDICATE THE OVERALL BEST PERFORMING MODEL

	BPIC12		**BPIC12-Sub**		**BPIC17**		**BPIC19**		**BPIC20**		**BAG (sec.)**	
Baseline	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
TS($w = 1, k = 1$)	**8889**	**10005**	**10069**	**11214**	12788	14387	47838	**52125**	7398	8168	980	1164
TS($w = 1, k = 3$)	8889	10005	10069	11214	12788	14387	47838	52125	7398	8168	980	1164
TS($w = 2, k = 1$)	8898	10013	10231	11414	**12223**	**14325**	**47798**	52392	**7216**	**8033**	**963**	**1154**
TS($w = 2, k = 3$)	8898	10013	10231	11414	12733	14325	47798	52392	7216	8033	963	1154
Time aware												
SEP-LSTM ($k = 1$)	13249	19284	10123	11433	12153	13548	45588	51034	6391	7519	605	789
SEP-LSTM ($k = 3$)	8986	12379	6428	8237	12621	14100	45826	51279	6412	7553	585	760
CRTP-LSTM	**5248**	**6582**	**5796**	**7330**	**11667**	**13090**	**41127**	**43843**	**6356**	**7255**	**570**	**677**
Data aware - CF												
SEP-LSTM ($k = 1$)	13964	18659	5784	6990	12095	13384	**39965**	43820	6362	7459	501	675
SEP-LSTM ($k = 3$)	15846	23195	5750	$\underline{\textbf{6908}}$	12802	14207	39971	44102	6360	7459	**461**	**613**
CRTP-LSTM	**5239**	$\underline{6445}$	**5748**	7138	**11739**	**13097**	40076	**42107**	**6310**	**7220**	550	654
Data aware - CF + EF												
SEP-LSTM ($k = 1$)	10561	16808	5850	7423	15434	20464	72390	90789	13045	18168	310	408
SEP-LSTM ($k = 3$)	9956	13730	5734	7220	14991	19778	71591	89672	12944	17984	299	393
CRTP-LSTM	$\underline{\textbf{5214}}$	**6581**	$\underline{\textbf{5579}}$	**6986**	$\underline{\textbf{10766}}$	$\underline{\textbf{12750}}$	$\underline{\textbf{38929}}$	$\underline{\textbf{40624}}$	$\underline{\textbf{6128}}$	$\underline{\textbf{7050}}$	$\underline{\textbf{293}}$	$\underline{\textbf{349}}$

data aware case feature CRTP-LSTM model performs best and the BPIC12-Sub where a data aware case feature SEP-LSTM model is the best performing model. From the table we can also see that a competitive solution is obtained using a transition system for remaining runtime prediction on the BPIC12 event log. Here a simple transition system which only predicts the average remaining runtime given the last observed activity label outperforms all SEP-LSTM models based on both the MAE and RMSE. However, it should be noted that SEP-LSTM models in general outperform the considered transition systems for remaining runtime prediction. When the results for time aware models are observed one can see that the CRTP-LSTM model outperforms the SEP-LSTM model on all event logs based on both the MAE and the RMSE. When the results for the data aware case feature models are observed one can see that the CRTP-LSTM model outperforms SEP-LSTMs on four event logs, namely BPIC12, BPIC12-Sub, BPIC17 and BPIC20, when evaluated using the MAE. When evaluated using the RMSE, the CRTP-LSTM model also outperforms the SEP-LSTM models on four event logs, namely the BPIC12, BPIC17, BPIC19 and BPIC20. Lastly, when the models are allowed to utilize all available information contained in previously observed events the data aware case and event feature CRTP-LSTM model outperforms SEP-LSTM models on all event logs based on both the MAE and RMSE.

VI. DISCUSSION

Various novel insights with regards to the performance of the considered approaches for remaining trace and runtime prediction can be drawn from the results presented above.

Consistent Best Performance by Our CRTP-LSTM Model for Remaining Trace Prediction. The CRTP-LSTM model proposed in this paper consistently outperforms all other considered approaches when predicting a trace consisting of a sequence of activity labels. Here, the data aware case and event feature CRTP-LSTM model performs best on all event logs considered, with one exception, namely the BAG event log where a data aware case and event feature SEP-LSTM model performs best. However, it should be reiterated that the differences in performance between data aware case and event feature models are negligible when evaluated for trace prediction on this event log. A large performance gain is obtained from using the data aware case and event feature CRTP-LSTM model over the best performing SEP-LSTM model on the BPIC12-Sub, BPIC12, BPIC17 and the BPIC19 event logs. From Table III, we can see that these event logs are relatively complex in terms of trace structure compared to the BPIC20 and BAG event log. Also, these event logs have a considerably lower average trace similarity (Affinity) compared to the BPIC20 event log. This seems to indicate that the observed performance gain from using the CRTP-LSTM model becomes more evident for event logs with more complex trace structures. The data aware case and event feature CRTP-LSTM model is the overall best performing model on the BPIC20 event log. However, only a minimal performance gain is observed compared to the best performing SEP-LSTM model and even the best performing transition system on this event log. This becomes intuitive when the characteristics of this event log are observed. As can be seen in Table III, this event log is relatively simple based on all complexity measures considered. Lastly, as discussed here above no meaningful differences in performance for trace prediction are observed for data aware case and event feature models based on the BAG event log and all data aware models perform quite well in absolute terms for trace prediction on this event log. As can be seen from Table III, this event log contains a number of case features e.g., on the arrival and departing flights for bags processed at the airport. Flights at the airport follow a predetermined schedule. Therefore, when these attributes are included in the modeling setup, the data aware models are in general able to determine the routes that bags will take through the luggage system at the airport, resulting in minimal performance differences for the considered approaches.

Consistent Best Performance by Our CRTP-LSTM Model for Remaining Runtime Prediction. The CRTP-LSTM models also consistently outperform other considered models for remaining runtime prediction. More specifically, the data aware case and event feature CRTP-LSTM model is the overall best performer for predicting the remaining runtime of cases on all event logs when evaluated using MAE. When evaluated using the RMSE, the data aware case and event feature CRTP-LSTM model is the best performing model on four out of the considered event logs, namely the BPIC17, BPIC19, BPIC20 and the BAG event log. A data aware case feature CRTP-LSTM model is the overall best performing model based on the BPIC12 event log when evaluated using the RMSE. However, a data aware case feature SEP-LSTM model is the overall best performing model on the BPIC12-Sub event log when evaluated using that metric. From Table III, we can see that the BPIC12 and the BPIC12-Sub event logs only contain a single event feature. When the performance of the best performing data aware case feature model is compared to the best performing data aware case and event feature model on these event logs one can see that the inclusion of this event feature seems to have a positive impact when the models are evaluated using the MAE. However, when evaluating the models using the RMSE a performance decrease is observed when this feature is included. When results for time aware models are observed one can see that the CRTP-LSTM model outperforms all SEP-LSTM models for remaining runtime prediction on all event logs and performance measure considered. When data aware case features models are evaluated using the MAE the CRTP-LSTM model is the best performer on four of the considered event logs, namely the BPIC12, BPIC12-Sub, BPIC17 and BPIC20. Similarly, when evaluated using the RMSE the CRTP-LSTM model outperforms the SEP-LSTM models on four event logs, namely the BPIC12, BPIC17, BPIC19 and the BPIC20. Lastly, when the results for data aware case and event feature models are observed one can see that the CRTP-LSTM model outperforms all SEP-LSTM models for remaining runtime prediction on all event logs and performance measures considered here. When the results for remaining trace prediction for the BPIC20 event log were observed only minimal performance differences were identified between the best CRTP-LSTM model, the best SEP-LSTM model and the best transition system. However, when the result for remaining runtime prediction are evaluated a clear performance gain is observed for time aware LSTM models compared to the best performing transition system on this event log. When additional attributes of events, i.e., case and event features, are included in the modeling setup, a considerable performance increase is observed for the CRTP-LSTM model resulting in the overall best performing model for remaining runtime prediction on this event log. Lastly, no meaningful performance differences were observed between data aware case and event feature models when evaluated for remaining trace prediction on the BAG event log. This is also the case when the performance these models is observed for remaining runtime prediction on this event log. A considerable performance gain is observed between time aware models and the best performing transition system on this event log. Furthermore, a considerable performance gain is observed when additional attributes of events are included in the modeling setup for both SEP-LSTM models and the CRTP-LSTM model. Although the data aware case and event feature CRTP-LSTM model outperforms all other considered models for remaining runtime prediction, only minimal performance differences are observed between data aware case and event feature models on this event log.

Beneficial Impact of Incorporating Case Attributes. Case features are case specific and therefore shared by all events belonging to a case. Due to this, both SEP-LSTM and CRTP-LSTM models can be naturally expanded in order to incorporate case features. When the results for the best performing data aware case features models are compared to the best performing time aware models a general performance increase can be observed when models are allowed to utilize case features in addition to time features and activity labels in order to predict the remaining trace and runtime of cases. This performance increase is especially evident on two out of the considered event logs, namely the BAG event log and the BPIC19 event log. From Table III, we can see that these event logs contain a relatively large number of case features compared to the other event logs. Therefore, it is not surprisings that the models are able to utilize information contained in case features in these event logs in order to make improved predictions.

Beneficial Impact of Adequately Incorporating Event Attributes. As previously discussed, the SEP-LSTM model has no straightforward way of incorporating event features, which can be clearly observed from the above results. The performance of the SEP-LSTM models reduces drastically when event features are included in the modeling setup for a number of the considered event logs, namely the BPIC17, BPIC19 and BPIC20 event logs. From Table III, we can see that these are event logs that in general contain a relatively large number of rapidly changing event features. However, event features often contain valuable information for predicting the remaining trace and runtime of cases. This can also be clearly observed from the presented results, where a considerable performance gain is achieved with the inclusion of event features for models trained with the CRTP-LSTM model. This clearly illustrates the benefit of this modeling approach which is designed so that it can include information from a large number of additional attributes of events in order to make predictions.

Simple Models Work for Simple Event Logs. The proposed CRTP-LSTM models outperform the considered transition systems for both remaining trace and runtime prediction on all event logs considered here. However, it should be noted that transition systems provide a competitive solution for trace prediction on a number of the considered event logs especially when beam search is considered. One example of an event log where transition systems provide a strong performance is the BPIC20 event log where a transition system which simply predicts the most frequently observed next activity label based on the most recently observed activity label ($w = 1, k = 1$) performs remarkably well for remaining trace prediction. This illustrates the benefit of considering transition systems as simple baselines models in order to provide a relative view of the performance of the more complex and computationally expensive LSTM based models. However as discussed here above, when the characteristics of the BPIC20 event log is observed as given in Table III one can clearly see why simple transition systems perform well on this event log. From the table one can see that this event log is quite simple, i.e., a relatively low number of variants (98) is observed, a large majority of cases (82%) share the three most frequently observed variants, the event log is relatively structured (0.87) and has a high estimated average similarity between traces (83%). Therefore, it is not surprising that this simple model is able to provide a good solution for this relatively simple event log.

VII. Conclusion

Developing models that can accurately predict the future trajectory of a case and its remaining runtime has a number of business applications, e.g., for planning purposes and operational support. In this work a novel CRTP-LSTM model was proposed. This model is trained to directly predict the complete remaining trace and runtime of cases. Furthermore, this model is natively data aware and can therefore incorporate all available information contained in previously observed events in order to make predictions, in contrast to SEP-LSTM models. In order to evaluate the performance of the proposed CRTP-LSTM model, the model was compared to SEP-LSTM models on six real-life event logs. Transition systems were included in this comparison as baseline models in order to provide a relative view of the performance of the more complex LSTM based models. Additionally, both time and data aware variants of the two LSTM based models were considered in order to illustrate the benefit of adequately incorporating additional features relating to events into the modeling setup.

Mainly two conclusions can be drawn from this comparison. First, the CRTP-LSTM model consistently outperforms all other considered approaches for both remaining trace and runtime prediction. More specifically, a data aware case and event feature CRTP-LSTM model is the best performer for all event logs when predicting remaining traces consisting of activity labels, with one exception where a data aware case and event feature SEP-LSTM model performs better on the BAG event log. However, the difference in performance between data aware case and event feature models on this event log is negligible. The data aware case and event feature CRTP-LSTM model is the overall best performing model for remaining runtime prediction for all event logs when evaluated using the MAE. When evaluated using the RMSE, this model is also the best performing model on all event logs with two exception, namely the BPIC12 event log where a data aware case feature CRTP-LSTM model performs best and the BPIC12-Sub event log where a data aware case feature SEP-LSTM model is the overall best performing model. Second, the inclusion of additional attributes of events, i.e., case and event features, has a positive impact on the performance of the CRTP-LSTM model. This is a clear indication that valuable information can be extracted from the attributes of events in order to make more accurate trace and runtime predictions. Therefore, developing models that can naturally incorporate all relevant information contained in events is beneficial.

Developing LSTM based models which are designed so that they can use information from a large number of event features opens up interesting avenues for future research in predictive process monitoring. So far, research on developing LSTMs for remaining trace and runtime prediction has largely focused on the use of a limited number of intra-case features and therefore assumed that cases progress in isolation. However, ongoing cases often compete for scarce resources during their execution. Therefore, developing a more elaborate feature engineering approach where in addition to intra-case features inter-case features, providing information on other concurrently running cases, are constructed seems sensible. Research on the use of inter-case features for predicting the completion time of cases using conventional machine learning algorithms (e.g., tree based approaches) has already shown promising results (see e.g., [33] and [34]). Process simulation is another active area of research in process monitoring. Developing natively data aware methods for process simulation is another interesting avenue of future research. Our initial exploration into this indicates that a data aware case and event feature CRTP-LSTM model in general outperforms SEP-LSTM models for early prediction of both remaining traces and runtimes. Additionally, one would expect that including all information contained in previously observed events would lead to improved performance for predicting future values of event features, e.g., resource information. Lastly, future research could expand on this research by taking into account other novel deep learning algorithms such as transformer networks (see [35]), which are designed to handle sequential data and have shown promising results for natural language processing.

References

[1] M. Dumas et al., *Fundamentals of Business Process Management*, Berlin, Germany: Springer, 2013.

[2] W. Van Der Aalst, *Process Mining: Data Science in Action*. Berlin, Germany: Springer, 2016.

[3] J. De Weerdt, M. De, J. Backer Vanthienen, and B. Baesens, “A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs,” *Inf. Syst.*, vol. 37, no. 7, pp. 654–676, 2012.

[4] S. K. vanden Broucke, J. De Weerdt, J. Vanthienen, and B. Baesens, “Determining process model precision and generalization with weighted artificial negative events,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1877–1889, Aug. 2014.

[5] W. Van der Aalst and E. Damiani, “Processes meet Big Data: Connecting data science with process science,” *IEEE Trans. Services Comput.*, vol. 8, no. 6, pp. 810–819, Nov./Dec. 2015.

[6] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortés, “Predictive monitoring of business processes: A survey,” *IEEE Trans. Services Comput.*, vol. 11, no. 6, pp. 962–977, Nov./Dec. 2018.

[7] C. Di Francescomarino, C. Ghidini, F. M. Maggi, and F. Milani, “Predictive process monitoring methods: Which one suits me best?,” in *Proc. Int. Conf. Bus. Process Manage.*, Springer, 2018, pp. 462–479.

[8] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, “Outcome-oriented predictive process monitoring: Review and benchmark,” *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 2, pp. 1–57, 2019.

[9] C. Di Francescomarino, M. Dumas, F. M. Maggi, and I. Teinemaa, “Clustering-based predictive process monitoring,” *IEEE Trans. Serv. Comput.*, vol. 12, no. 6, pp. 896–909, Nov./Dec. 2019.

[10] 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.*, Springer, 2017, pp. 477–492.

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

[12] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, “A multi-view deep learning approach for predictive business process monitoring,” *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 2382–2395, Jul./Aug. 2022.

[13] W. M. Van der Aalst, M. H. Schonenberg, and M. Song, “Time prediction based on process mining,” *Inf. Syst.*, vol. 36, no. 2, pp. 450–475, 2011.

[14] B. R. Gunnarsson, S. K. vanden Broucke, and J. De Weerdt, “Predictive process monitoring in operational logistics: A case study in aviation,” in *Proc. Int. Conf. Bus. Process Manage.*, Springer, 2019, pp. 250–262.

[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.*, Springer, 2016, pp. 327–338.

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

[17] C. Di Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, and A. Yeshchenko, “An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring,” in *Proc. Int. Conf. Bus. Process Manage.*, Springer, 2017, pp. 252–268.

[18] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and I. Teinemaa, “Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 4, pp. 1–34, 2019.

[19] A. Rogge-Solti and M. Weske, “Prediction of business process durations using non-markovian stochastic petri nets,” *Inf. Syst.*, vol. 54, pp. 1–14, 2015.

[20] M. Polato, A. Sperduti, A. Burattin, and M. de Leoni, “Time and activity sequence prediction of business process instances,” *Computing*, vol. 100, no. 9, pp. 1005–1031, 2018.

[21] A. Bevacqua, M. Carnuccio, F. Folino, M. Guarascio, and L. Pontieri, “A data-adaptive trace abstraction approach to the prediction of business process performances,” in *Proc. Int. Conf. Enterprise Inf. Syst.*, 2013, pp. 56–65.

[22] B. F. van Dongen, R. A. Crooy, and W. M. van der Aalst, “Cycle time prediction: When will this case finally be finished?,” in *Proc. OTM Confederated Int. Conf. “On Move Meaningful Internet Syst.*, Springer, 2008, pp. 319–336.

[23] 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*, SIAM, 2019, pp. 118–126.

[24] A. Jalayer, M. Kahani, A. Beheshti, A. Pourmasoumi, and H. R. Motahari-Nezhad, “Attention mechanism in predictive business process monitoring,” in *Proc. IEEE 24th Int. Enterprise Distrib. Object Comput. Conf.*, 2020, pp. 181–186.

[25] 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.*, 2017, pp. 1–7.

[26] E. Rama-Maneiro, J. C. Vidal, and M. Lama, “Deep learning for predictive business process monitoring: Review and benchmark,” 2020, *arXiv:2009.13251*.

[27] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” 2015, *arXiv:1506.00019*.

[28] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[29] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, U.K.: Pearson, 2010.

[30] G. Navarro, “A guided tour to approximate string matching,” *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, 2001.

[31] C. W. G Günther, “Process mining in flexible environments,” Ph.D. dissertation, Eindhoven Univ. Technol., 2009.

[32] M. M. Lau and K. H. Lim, “Review of adaptive activation function in deep neural network,” in *Proc. IEEE-EMBS Conf. Biomed. Eng. Sci.*, 2018, pp. 686–690.

[33] A. Senderovich, C. Di Francescomarino, C. Ghidini, K. Jorbina, and F. M. Maggi, “Intra and inter-case features in predictive process monitoring: A tale of two dimensions,” in *Proc. Int. Conf. Bus. Process Manage.*, Springer, 2017, pp. 306–323.

[34] A. Senderovich, C. Di Francescomarino, and F. M. Maggi, “From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring,” *Inf. Syst.*, vol. 84, pp. 255–264, 2019.

[35] A. Vaswani et al., “Attention is all you need,” 2017, *arXiv:1706.03762*.

Björn Rafn Gunnarsson received the MSc degree of statistics and data science from KU Leuven, Belgium, in 2018. Currently, he is a PhD researcher with the Research Center for Information Systems Engineering (LIRIS) of the KU Leuven. His research interests span across the fields of predictive process monitoring, operational analytics and data mining.

Seppe vanden Broucke received the PhD degree in applied economics with KU Leuven, in 2014. Currently, he is working as an assistant professor with the Department of Business Informatics, UGent, Belgium and is a lecturer with KU Leuven, Belgium. His research interests include business data mining and analytics, machine learning, process management, process mining.

Jochen De Weerdt received the PhD degree in business economics from KU Leuven, in 2012 on the topic of Automated Business Process Discovery. He is an associate professor with the Research Centre of Information Systems Engineering (LIRIS) of the Faculty of Economics and Business, KU Leuven. Subsequently, he worked as a postdoctoral research fellow with the Information Systems School of the Queensland University of Technology (Brisbane, Australia). His research and teaching interests include business information systems, business analytics, process mining, machine learning, learning analytics, and business process management.