Deep Learning for Predictive Business Process Monitoring: Review and Benchmark

Efrén Rama-Maneiro, Juan C. Vidal, and Manuel Lama

Abstract—Predictive monitoring of business processes is concerned with the prediction of ongoing cases on a business process. Lately, the popularity of deep learning techniques has propitiated an ever-growing set of approaches focused on predictive monitoring based on these techniques. However, the high disparity of process logs and experimental setups used to evaluate these approaches makes it especially difficult to make a fair comparison. Furthermore, it also difficults the selection of the most suitable approach to solve a specific problem. In this paper, we provide both a systematic literature review of approaches that use deep learning to tackle the predictive monitoring tasks. In addition, we performed an exhaustive experimental evaluation of 10 different approaches over 12 publicly available process logs.

Index Terms—Process Mining, Business Process Monitoring, Neural Networks, Systematic Literature Review, Deep Learning

1 Introduction

Process mining is a discipline that offers techniques to discover, monitor and enhance real business processes by extracting knowledge from event logs, allowing to understand *what is really happening in a business process*, and not *what we think is going on* [1]. Process mining has three main subfields: *(i) process discovery*, where a process model is inferred from the event log, *(ii) process enhancement*, where a process model is improved with information of the event log, and *(iii) process conformance* where the process model is compared with an event log to check the degree of its conformance [2].

Predictive process monitoring is a subfield of process mining that belongs to the second category, that is concerned about forecasting *how* a running case will unfold. Most process enhancement works are concerned with a *post-mortem* analysis; that is, conformance approaches are *reactive* in the sense of detecting a violation after it has happened [3]. However, predictive monitoring approaches are *proactive* [4] in the sense of giving predictions before a violation happens, thus, improving the process performance and mitigating risks. These predictions may involve the forecasting of the next event or sequence of events of a running case, the remaining time until the end of a case or the possible outcome of a running case. For example, it could allow predicting the trajectory of a patient in a hospital since its first registration in an emergency room until the dischargement of the patient [5].

The approaches to predictive monitoring apply a wide number of techniques to perform these predictions. Some of them rely on explicit representations of the process model such as a Transition System [6], [7], a Probabilistic Finite Automaton [8], [9] or a Petri Net [10], [11]. Other approaches do not extract a process model but, instead, extract feature vectors from partial traces to train a machine learning model. These approaches use techniques such as Support Vector Machines [12], [13], [14], Clustering Analysis [15], [16], [17], Factorization Machines [18], and, more recently, Deep Learning-based techniques, the latter being the ones that have obtained the best results.

Deep learning and neural networks have gained a lot of attention in recent years because of its success in fields such as computer vision [19], [20] or natural language processing (NLP) [21], [22]. In this latter field, recurrent neural networks (RNN), a kind of neural network specialized in sequence processing, were applied with success surpassing traditional approaches. Due to the sequential nature of business processes, RNNs were a good fit to approach the predictive monitoring problem [23], [24], [25]. Nowadays, deep learning has been widely applied to the predictive monitoring of business processes and, in general, in process mining tasks such as reconstructing missing events [26], anomaly detection [27], resource allocation [28] or process discovery [29].

However, two main issues arise. First, the high number of combinations of posible architectures, ways to encode the partial traces and events, and the number of predictive tasks available may complicate the selection of a certain deep learning approach for addressing a specific problem. Second, the high disparity of datasets and experimental setups used for evaluating predictive monitoring approaches may make a fair comparison of the state-of-the-art approaches difficult.

The contribution of the paper is two-fold. We propose a categorization of the deep learning approaches for pre-

- *E. Rama-Maneiro is with the Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782, Santiago de Compostela, SPAIN. email: efren.rama.maneiro@usc.es*
- *J. C. Vidal is with the Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782, Santiago de Compostela, SPAIN. email: juan.vidal@usc.es*
- *M. Lama is with the Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782, Santiago de Compostela, SPAIN. manuel.lama@usc.es*

Manuscript received November 17, 2020 revised X

dictive monitoring, providing an in-depth analysis of the differences between them in light of the results obtained in the experimentation. Secondly, we provide an experimental evaluation of the publicly available deep learning-based predictive monitoring approaches discussed in the paper. We have made public the source code, trained models and results of the experimentation [1] in the hope that it serves as a benchmark toolbox for predictive monitoring techniques.

The rest of the paper is structured as follows: Section 2 shows some basic definitions about terms that are going to be used throughout the text. Section 3 highlights the difference between this survey and other predictive monitoring surveys. Section 4 highlights the search methodology to perform the systematic literature review. Section 5 shows the classification and taxonomy of the retrieved studies, explaining its main achievements. Section 6 shows the experimental setup, results, and discussion of the benchmarking of the available approaches. Finally, Section 7 concludes the paper highlighting future lines of work.

2 Background

2.1 Process mining

The input of process mining techniques is an *event log*, usually composed of events with at least a *case identifier*, an *activity*, and a *timestamp*, and, optionally, *case attributes*, which are values shared by all the events of the same case, and *event attributes*, which are specific of each event. Let us consider the sample log shown in TABLE 1, which is part of a real-life event log from a ticketing management process of a software company. This event log provides information about each case identifier, activity, timestamp and resource of each event (in this case, the resource is an event attribute).

TABLE 1: Excerpt of a business process log

Case ID	Activity	Timestamp	Resource
Case2118	Assign seriousness	14-01-2010 07:52:50	Resource 2
Case2118	Take in charge ticket	09-02-2010 13:01:11	Resource 21
Case2118	Resolve ticket	17-02-2010 07:44:53	Resource 21
Case2118	Closed	17-02-2020 07:44:59	Resource 21
Case2088	Assign seriousness	04-02-2010 08:37:45	Resource 2
Case2088	Take in charge ticket	04-02-2010 09:01:28	Resource 2
Case2088	Create SW anomaly	04-02-2010 09:01:35	Resource 2
Case2088	Resolve ticket	16-03-2010 13:08:40	Resource 2
Case2088	Closed	31-03-2010 11:08:53	Resource 5

Definition 1. Let A be the universe of activities, C the universe of cases, T the time domain, and $D_1, \ldots, D_m$ the universes of each of the attributes of the traces and events of the log, with $m \geq 0$. An event $e \in E$ is a tuple $(a, c, t, d_1, \ldots, d_m)$ where $a \in A$, $c \in C$, $t \in T$ and $d_i \in \{D_i \cup \epsilon\}$ with $i \in [1, m]$ and ϵ being the empty element.

Each event in a process log is *unique*, i.e. two events can not exist in the same case, with the same activity, and at the same time.

Definition 2. Let π_A, π_C, π_T and π_{D_i} be functions that map an event to an activity, a case identifier, a timestamp, and an attribute, that is, $\pi_A(e) = a$, $\pi_C(e) = c$, $\pi_T(e) = t$, and $\pi_{D_i}(e) = d_i$.

1. https://nextcloud.citius.usc.es/index.php/s/drMbTeGNKTE9axJ

Then, events are unique, that is, $\forall e_i, e_j \in E : e_i \neq e_j \implies \pi_A(e_i) \neq \pi_A(e_j) \vee \pi_C(e_i) \neq \pi_C(e_j) \vee \pi_T(e_i) \neq \pi_T(e_j)$.

For example, the second event of TABLE 1 is e_2, with $\pi_A(e_2)$ = "Take in charge ticket", $\pi_C(e_2)$ = "Case2118", $\pi_T(e_2)$ = "09-02-2010 13:01:11" and $\pi_{D_1}(e_2)$ = "Resource 21".

The sequence of events with the same case identifier is called a *trace*. In this sequence, each event has a timestamp equal or greater than its predecessor.

Definition 3. Let S be the universe of traces, a trace $\sigma \in S$ is a non-empty sequence of events $\sigma = \langle e_1, \ldots, e_n \rangle$ which holds that $\forall e_i, e_j \in \sigma; i, j \in [1, n] : j > i \wedge \pi_C(e_i) = \pi_C(e_j) \wedge \pi_T(e_j) \geq \pi_T(e_i)$ where $|\sigma| = n$.

For example the first trace of TABLE 1 consist of 4 events of the case "Case2118".

Furthermore, an *event log* can be defined as a set of traces.

Definition 4. An event log is a set of traces, $L = \{\sigma_1, \ldots, \sigma_l\}$ such as $L = \{\sigma_i | \sigma_i \in S \wedge i \in [1, l]\}$ where $|L| = l$.

Following with the example of TABLE 1, the log shown in TABLE 1 is composed of two traces, related to the cases "Case2118" and "Case2088".

Predictive monitoring approaches often partition each trace in sets of prefixes and suffixes. Prefixes and suffixes can be defined as follows:

Definition 5. Let σ be a trace such as $\sigma = \langle e_1, \ldots, e_n \rangle$ and $k \in [1, n]$ be any positive integer. The event prefix of length k, hd^k, and its event suffix, tl^k, can be defined as follows: $hd^k(\sigma) = \langle e_1, \ldots, e_k \rangle$ and $tl^k(\sigma) = \langle e_{k+1}, \ldots, e_n \rangle$. The activity prefix and suffix can be defined as the application of π_A to the whole event prefix and suffix, being $\pi_A(hd^k(\sigma)) = \langle \pi_A(e_1), \ldots, \pi_A(e_k) \rangle$ and $\pi_A(tl^k(\sigma)) = \langle \pi_A(e_{k+1}), \ldots, \pi_A(e_n) \rangle$ respectively.

For example, the activity prefix and suffix of length $k = 3$ of the trace with case identifier "Case2088" in TABLE 1 $\pi_A(hd^3(\sigma))$ = ⟨ "Assign seriousness", "Take in charge ticket", "Create SW anomaly"⟩ and $\pi_A(tk^3(\sigma))$ = ⟨ "Resolve ticket", "Closed" ⟩.

Each trace of the log may have assigned a certain outcome.

Definition 6. The *outcome* of a running case is a domain-dependent label that conveys information about the full case of interest. Let O be the universe of outcomes, where $o \in O$. Then π_O is a function that maps an event prefix to an outcome such as $\pi_O(\sigma) = \pi_O(hd^k(\sigma)) = o$.

Examples of possible outcomes are whether a case will be reopened in the future or whether a patient will be readmitted in a hospital.

2.2 Predictive monitoring

Given a certain event prefix of a running case, predictive monitoring is concerned with forecasting how different aspects of the next event or sequence of events will unfold until the end of the case. Formally, let $hd^k(\sigma)$ be a event prefix such as $hd^k(\sigma) = \langle e_1, \ldots, e_k \rangle$, e' be a predicted event by a function Ω, and let $\oplus$ be the concatenation operator

between two sequences, then, depending on the predictive task at hand, we can define the following functions Ω:

Definition 7. The next activity prediction problem can be defined as $\Omega_A(hd^k(\sigma)) = \pi_A(e'_{k+1})$.

Definition 8. The next attribute, d_i, prediction problem can be defined as $\Omega_{D_i}(hd^k(\sigma)) = \pi_{D_i}(e'_{k+1})$.

Definition 9. The next timestamp prediction problem can be defined as $\Omega_T(hd^k(\sigma)) = \pi_T(e'_{k+1}) - \pi_T(e_k)$.

Definition 10. The outcome of an event prefix can be predicted as $\Omega_O(hd^k(\sigma)) = \pi_O(hd^k(\sigma)) = o$.

Note that to predict the outcome of a certain event prefix, we do not really need information about the next event of the given event prefix.

The activity suffix prediction problem can be defined in two different ways: as an application of the Ω_A function recursively over the predicted activities until the end of the case ("[EOC]") is reached (Definition 11) or directly predicting the activity suffix (Definition 12).

Definition 11. An activity suffix can be recursively predicted as $\Omega_{SA} = \langle \Omega_A(\sigma') = \pi_A(e'_i) \mid \sigma' = hd^k(\sigma) \oplus \langle e'_{k+1}, \ldots, e'_{i-1} \rangle \rangle$ while $\pi_A(e'_i) \neq [EOC]$.

Definition 12. An activity suffix can be directly predicted $\Omega_{SA}(hd^k(\sigma)) = \pi_A(tl^k(\sigma))$.

Functions for predicting an attribute suffix and remaining time can be defined analogously.

Definition 13. An attribute suffix can be recursively predicted as $\Omega_{SD_i} = \langle \Omega_{D_i}(\sigma') = \pi_{D_i}(e'_i) | \sigma' = hd^k(\sigma) \oplus \langle e'_{k+1}, \ldots, e'_{i-1} \rangle \rangle$ while $\pi_A(e'_i) \neq [EOC]$.

Definition 14. An attribute suffix can be directly predicted as $\Omega_{SD_i}(hd^k(\sigma)) = \pi_{D_i}(tl^k(\sigma))$.

Definition 15. Let θ be the sequence of predicted next timestamps such as $\theta = \langle \Omega_T(\sigma') = \pi_T(e'_i) - \pi_T(e'_{i-1}) \mid \sigma' = hd^k(\sigma) \oplus \langle e'_{k+1}, \ldots, e'_{i-1} \rangle \rangle$ while $\pi_A(e'_i) \neq [EOC]$, then the remaining time can be calculated as $\Omega_{RT}(hd^k(\sigma)) = \sum_{i=k}^{n} \theta_i$.

Definition 16. The remaining time can be directly calculated as $\Omega_{RT}(hd^k(\sigma)) = \pi_T(e'_n) - \pi_T(e_k)$ where $\pi_T(e'_n)$ denotes the predicted timestamp for the last event.

Note that, in this paper, each of the functions Ω will be always represented by a deep neural network.

3 Related work

Several authors have addressed the problem of reviewing the current state of the art in predictive monitoring. In [30], the authors review a set of studies and classify them in process-aware methods and non-process aware methods, depending on whether they require a process model as an input or not, and also whether they treat predictive monitoring as a regression problem or as a classification problem. In [31], the authors perform a systematic literature review with the aim of helping companies to select their best suited predictive monitoring framework according to multiple dimensions such as the predictions made by the approach, the domain where it is applied, the algorithm developed, and the input used. In [32], the authors perform a study of three deep learning approaches for predictive monitoring, taking into account different dimensions, such as the encoding scheme or the prediction target.

Regarding to benchmarking studies, in [33], the authors compare the performance of 20 different supervised learning classification techniques over 6 different process logs to predict the next event in a business process. They conclude that the best machine learning classifier, in terms of accuracy, is the credal decision tree.

In [34], a comparison of the following families of techniques for the next activity prediction problem is made: *(i)* recurrent neural networks, *(ii)* markov models, *(iii)* grammar induction techniques, which learn a set of production rules that describe a language (in this case, a process log), *(iv)* process discovery-based techniques, and *(v)* automata based prediction techniques.

In [35], the authors focus their study on the prediction of the remaining time in a business process, comparing several bucketing, encoding, and supervised learning techniques in terms of the Mean Absolute Error of the predictions. They also compare the supervised learning techniques with 3 different process-aware methods. Their conclusion is that LSTMs outperform other approaches for remaining time prediction. In [36], the authors provide a systematic literature review of outcome-oriented predictive monitoring and compare the performance of four different machine learning techniques (random forest, logistic regression, support vector machines and extreme gradient boosting) with multiple encodings to predict the outcome of a business process. In [37], the authors perform a similar analysis for outcome prediction but comparing another set of machine learning techniques (random forest, support vector machines, deep feedforward networks, and LSTMs).

Only [32] has similar objectives as the review conducted in this paper. Still, it is limited to the first three deep learning approaches in predictive monitoring, and they do not perform any benchmarking of the approaches. Moreover, [32] uses the results published in the original papers and, therefore, does not perform a fair comparison between the different approaches. Other authors include Deep Learning approaches when comparing their solution with the state-of-the-art in predictive monitoring. However, they limit their comparison to evaluating LSTM [34], [35], [37], and GRU [34] alongside other machine learning algorithms. No other review study benchmarks the original deep learning implementations of the state-of-the-art approaches under controlled conditions to provide a fair comparison between them.

4 Search methodology

We aim to answer the following questions:

- RQ1 What methods for predictive monitoring are based on deep learning?
- RQ2 How could these methods be classified?
- RQ3 Which datasets are used to evaluate these methods?
- RQ4 How do these methods compare as far as prediction performance is concerned?

RQ1 is the main research question, which aims to identify existing approaches to predictive monitoring using deep

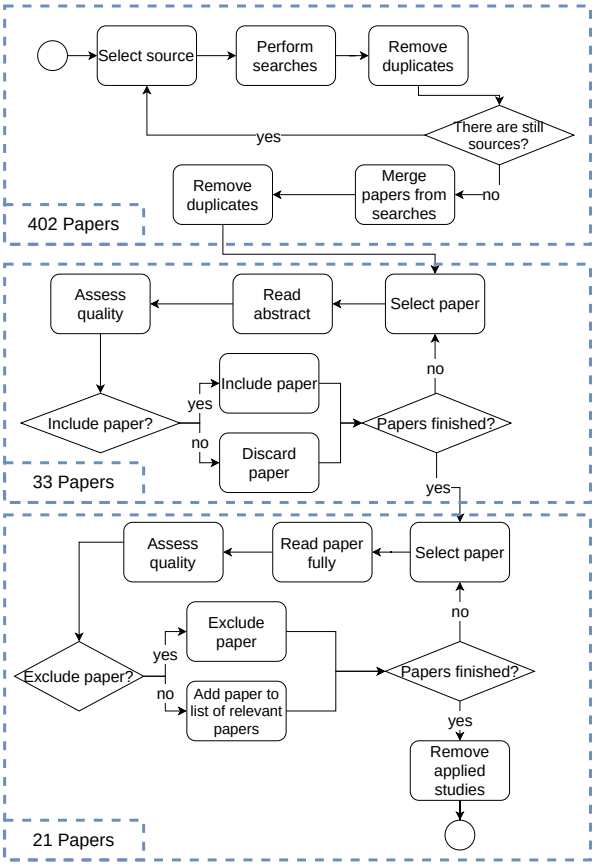

Fig. 1: Three-stage retrieval approach used to select the most relevant concerned studies.

learning. RQ2 aims to define a taxonomy to classify the approaches in groups based on the type of neural network used, the encoding of the traces, and the input data. RQ3 aims to identify how these methods have been evaluated and with which data. RQ4 investigates how these methods perform in terms of the next activity, activity suffix, next timestamp, and remaining time prediction.

4.1 Study retrieval

To retrieve the relevant studies, we performed a three-stage retrieval approach. In the first stage, we retrieved every study using the information from the title and its full text. In the second stage, we filtered the studies by assessing only the abstract of the paper. In the final stage, these papers are read and non-releveant papers are *excluded* them from the final list. This last step was necessary since the paper abstract was sometimes quite imprecise. We highlight the full approach in Fig. 1.

To look for the relevant literature, we used the following scientific databases: ScienceDirect, DBLP, Google Scholar, Scopus, ACM Digital Library, Springer, IEEE Xplore Digital Library, and arXiv. In each of these databases, we applied the following queries:

QUERY1 *("deep learning" OR "deep model" OR "deep neural") AND "predictive business process monitoring"*

QUERY2 *("deep learning" OR "deep model" OR "deep neural") AND "process prediction" AND "business process"*

QUERY3 *("deep learning" OR "deep model" OR "deep neural") AND "process mining" AND "activity prediction"*

QUERY4 *("deep learning" OR "deep model" OR "deep neural") AND "process mining" AND "time prediction"*

QUERY5 *("deep learning" OR "deep model" OR "deep neural") AND "process mining" AND "attribute prediction"*

These queries, applied over the titles, and full text of the articles, have been run in February of 2020. The initial search returned **140** results for the first query, **72** results for the second one, **66** results for the third one, **119** results for the fourth one, and **5** results for the fifth one, totalling **402** results.

The first filter ruled out irrelevant articles after analysing the title and abstract with the following inclusion criterion:

INCL1 The study is written in English.
INCL2 The study is concerned with predictive business process monitoring.
INCL3 The study uses a deep learning approach.
INCL4 The study takes an event log as an input.

The first inclusion criterion, INCL1, rules out papers that are not written in English or that do not have their full text accessible. The second inclusion criterion, INCL2, identifies papers that are not concerned with a predictive monitoring approach. INCL3 rules out classical predictive monitoring proposals and survey papers that do not use a deep learning approach. Finally, INCL4 rules out papers that do not take an event log as an input, which would be the case for papers that are focused on problems such as time series prediction.

After this inspection, a total of **33** articles are selected after applying these inclusion criteria. Apart from the differences of the queries, there are two major differences with other surveys in predictive monitoring: *(i)* there is no minimum number of citations. The reason for this criteria is that, since the use of deep learning in predictive monitoring is posterior to other machine learning techniques, the number of years covered is smaller; and *(ii)* no snowballing is applied since the queries are broad enough to capture most papers concerned with this review. Furthermore, we empirically observed that most papers tend to consistently and exhaustively cite previous work, which is also reflected in the results since the queries are applied to the full text of the papers.

These **33** articles were closely read and classified as relevant using the following exclusion criteria:

EXCL1 The study is concerned with a predictive technique and not with the use of its results for solving a different problem.
EXCL2 The study must propose a predictive monitoring technique.
EXCL3 The study must evaluate the predictive monitoring technique.

Exclusion criterion EXCL1 filters papers that do not propose a predictive monitoring technique but use the results of such techniques to perform another task such as

resource assignment or anomaly detection. EXCL2 filters papers whose paper abstract led us to erroneously think it was a predictive monitoring approach. Finally, EXCL3 excludes papers that are a research in progress.

After this inspection, a total of **26** papers were deemed as relevant. Furthermore, the papers in TABLE 2 were discarded since they where applications of a primary proposal to a specific domain (IoT, Aviation, etc.). The selected **21** original papers which are the object of our study[2] are categorized and classified in TABLE 3 according to multiple dimensions such as the neural network type, its encoding type or the prediction targets.

TABLE 2: Relation of primary works and their counterpart applications to a specific domain.

Primary study	Applied study	Domain
Evermann et al. [23]	Evermann et al. [38]	Early version
Evermann et al. [23]	Evermann et al. [39]	Reimplementation
Evermann et al. [23]	Gunnarsson et al. [40]	Airports
Evermann et al. [23]	Tello-Leal et al. [41]	IoT
Tax et al. [25]	Tax et al. [42]	Smart Homes

5 Analysis and classification

Predictive monitoring approaches that use deep learning can be classified based on the following dimensions (TABLE 3):

- *Input data*: data from the business process logs used to train the predictive model.
- *Predictions*: the elements of the events that the neural network is trained to forecast.
- *Neural Network Type*: type of neural network used such as feedforward, autoencoder, convolutional, recurrent or transformer.
- *Sequence encoding*: how event prefixes are converted into learnable tensors by the neural net.
- *Event encoding*: how each individual categorical and continuous variable is encoded in a feature vector.

5.1 Input data

The selection of inputs used in the neural net is one of the most important decisions to make when designing a predictive monitoring approach based on deep learning. Often, the more data is fed to the neural network, the better its predictive performance will be. However, this is not always true since the data in the event log can be noisy or even missing. If some attribute is missing it can be imputed or marked as an "unknown" attribute for that event. In general, all the approaches surveyed, except [53], do not use a process model as an input. Note that, every approach uses the sequence of activities of the event log since it is one of the most important sources of information of the event log.

As far as the attributes of the log are concerned, it is possible that not every attribute adds significant information to the predictive problem [43], [44], [48], [49], [51], [52], [53], [54], [56], [58] . [51] trains a predictive model where an alignment weight vector learns the importance of each attribute. [52] clusters the attributes together using the x-means algorithm. The belonging to a cluster is added as an additional feature to the feature vector of each event. Furthermore, resources in a process log are often noisy since some resources could potentially appear only once in the whole event log, even though they adhere to an organizational scheme. To solve this problem, [50] groups resources depending on their activity execution profiles. Note that, as shown in TABLE 3 this is the only approach that distinguishes the resources from other attributes of the event log. As far as the approaches that use time features [24], [25], [46], [50], [53], [54], [55], [57], they face the problem of a high variability in the time between the events so these time features may complicate the training phase.

5.2 Predictions

Regarding the prediction targets, there are multiple possibilities:

- *Activity* [23], [25], [43], [45], [46], [48], [49], [50], [51], [52], [53], [55], [57], [60]: the next activity of a running case.
- *Activity suffix* [24], [25], [46], [50], [51]: the sequence of activities given a running case.
- *Next timestamp* [25], [46], [50]: the difference between the next timestamp of an event and the timestamp of the current event.
- *Remaining time* [25], [44], [46], [50], [54]: the difference between the timestamp of the last case of a trace and the current timestamp of an event.
- *Outcome* [47], [56], [58], [59]: refers to the outcome of a given running case.
- *Attributes* [50], [51]: other event-level attributes present in the log. We include here the roles from [50] since they are derived from the resources of the log, which are, in turn, attributes of the log.
- *Attribute suffix* [50], [51]: the approaches that predict the activity suffix and use other attributes of the log as inputs must predict also the sequence of attributes since the inputs for these attributes in the predicted events would be missing otherwise.

The most common prediction problem is the prediction of the next activity given a partial event prefix. When this is the case, other prediction tasks are taken up as auxiliary tasks that may help improve the prediction performance. The only exception is the remaining time prediction problem, since it can be approached as a direct predition, as shown in definition 16.

As far as the outcome prediction problem is concerned, most studies assume that the log is fully labeled and, thus, the problem is treated in a supervised manner. The exception to this is [58], where they do not make such assumptions and, instead, consider the log as a partially labeled dataset. Thus, the problem is treated in a semi-supervised manner which has two stages: *(i)* an LSTM is trained on the full event log to predict the next activity, and *(ii)* another LSTM is initialized with the parameters learned by the previous model and it is trained over the log that is labeled to predict the outcome of a running case.

When the objective is to predict the activity suffix from a partial trace, the next event must be sampled from the output probability distribution of the last neural network's prediction layer. A simple choice would be selecting the

2. The results of the study retrieval and classification are available in https://bit.ly/2UgKySf

Author, Year	Reference	Network type	Sequence encoding	Event encoding	Input data	Prediction
Evermann et al., 2017	[23]	LSTM	CONT	EMB	ACT	ACT
Francescomarino et al., 2017	[24]	LSTM	PRFX	OH	ACT, TF, LTL	SFX
Mehdiyev et al., 2017	[43]	AE + DFNN	NGRAM	-	ACT, ATTR	ACT
Tax et al., 2017	[25]	LSTM	PRFX	OH	ACT, TF	ACT, NT, SFX, RT
Navarin et al., 2017	[44]	LSTM	PRFX	OH	ACT, ATTR	RT
Al-Jebrni et al., 2018	[45]	CNN	CONT	EMB	ACT	ACT
Khan et al., 2018	[46]	DNC	PRFX	OH	ACT, TF	ACT, NT, RT, SFX
Metzger et al., 2018	[47]	LSTM	PRFX	OH	ACT	OUT
Schönig et al. 2018	[48]	LSTM	CONT	OH	ACT, ATTR	ACT, RES
Mehdiyev et al., 2018	[49]	AE + LR	NGRAM	-	ACT, ATTR	ACT
Camargo et al., 2019	[50]	LSTM	PRFX	P-EMB	ACT, R, TF	ACT, ROLE, NT, RT, SFX, RLSFX
Lin et al., 2019	[51]	LSTM	CONT	EMB	ACT, ATTR	ACT, ATTR, SFX, ATTRSFX
Hinkka et al., 2019	[52]	GRU	PRFX	OH	ACT, ATTR, C-ATTR	ACT [CP]
Theis et al., 2019	[53]	DFNN	TSS	FB	ACT, PM, TF, ATTR	ACT
Wahid et al., 2019	[54]	DFNN	SE	EMB	ACT, TF, ATTR	RT
Pasquadibisceglie et al., 2019	[55]	CNN	PRFX	FB	ACT, TF	ACT
Wang et al., 2019	[56]	B-LSTM	PRFX	OH	ACT, ATTR	OUT
Mauro et al., 2019	[57]	CNN	PRFX	EMB	ACT, TF	ACT
Folino et al., 2019	[58]	LSTM	PRFX	OH, EMB	ACT, ATTR	OUT
Hinkka et al., 2019	[59]	GRU/LSTM	CONT	OH	ACT	OUT
Philipp et al., 2020	[60]	TRANS	CONT	EMB	ACT	ACT

TABLE 3: Collected studies for Predictive Monitoring that use Deep Learning. **Sequence encoding**: Continuous (CONT), Prefixes padded (PRFX), N-gram (NGRAM), Single Event (SE), Timed state sample (TSS). **Input data**: Activity (ACT), Time features (TF), Linear Temporal Logic (LTL), Attributes (ATTR), Clustered Attributes (C-ATTR), Process model (PM), Role (R). **Event encoding**: Embedding (EMB), One-hot encoding (OH), Pretrained Embedding (P-EMB), Frequency based (FB). **Prediction**: Activity (ACT), Activity Suffix (SFX), Next timestamp (NT), Remaining time (RT), Outcome (OUT), Resource (RES), Role (ROLE), Activity prediction in certain checkpoints (ACT [CP]), Role suffix (RLSFX), Attribute suffix (ATTRSFX). **Network Type**: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Convolutional Neural Network (CNN), Autoencoder (AE), Deep Feedforward Network (DFNN), Logistic Regression (LR), Bidirectional LSTM (B-LSTM), Transformer (TRANS)

event with the highest probability in the vector [25]. However, this solution often makes the predictions to be very repetitive on the same top probability activity when the process is complex enough. Even more, it could happen that the end of the trace is never predicted with this method, which forces to add a maximum trace length constraint to avoid the prediction to be infinite. Another solution is to perform a *beam search* that explores a limited space of solutions by retaining the b traces that have the highest composed probability of events. This is the approach used by [24] in which the generation procedure stops when a complete candidate trace conformant with a series of mined LTL rules is found. They also decrease the probability of subsequent predictions of the same event. In [50], another approach is taken, in which the next event is sampled randomly following the probability distribution of the neural network. Note that there exist more sampling methods, such as *top-k sampling* [61] or *nucleus sampling* [62].

When predicting an activity suffix, every feature used as input, such as attributes or time features, must be predicted in subsequent steps. These predictions are often approached using the same neural network to predict multiple features simultaneously. Predicting multiple outputs at once in a neural network is also called *multitask learning* [63]. It has been shown that predicting multiple tasks at once helps to enhance the generalization of the neural network in the sense that it acts as a way of implicit *data augmentation*. This is especially relevant in predictive monitoring, where the event logs are often scarce of data. Each prediction task has attached its own loss, and the set of losses must be combined so that they can be minimized. In predictive monitoring, the most usual form to combine the losses is depicted in equation 1, where L_t is the total combined loss, $|T|$ is the total number of tasks, and L_i is the loss of an individual task.

$$L_t = \sum_{i=1}^{|T|} L_i \tag{1}$$

This form of combining the task losses poses the problem that different tasks could have different magnitudes, and one task could dominate others just for a bigger value, which explains why sometimes multitask learning is avoided in predictive monitoring when the prediction target is only the next activity and not the activity suffix.

5.3 Neural network types

5.3.1 Feedforward networks

Feedforward networks [64] are the most basic models for deep learning. In this type of neural network, the information flows through it without any recurrence. This lack of recurrence makes them well suited for tabular data [54] but not in process mining, where event logs have dependencies between the activities that are not exploited by this type of network. Thus, feedforward networks are not often used in predictive monitoring. However, this type of neural network is commonly used in combination with other methods, such as autoencoders [43], [49] or after extracting features from the process model [53].

5.3.2 Autoencoders

The predictive monitoring approaches presented in [43] and [49] use autoencoders as their type of neural network. This kind of neural network learns how to reconstruct its own input. An autoencoder has two main parts: the encoder, which learns to map its input x into a hidden representation h, and the decoder, which learns to map a given hidden representation h back into the original input x'. The loss function is configured to penalize x' from being different

from x. Formally, we would train an encoder e and a decoder d such as:

$$d(e(x)) = x' \tag{2}$$

The autoencoders usually trained are *undercomplete*, which means that the dimension of the hidden representation is less than the input dimension. This forces the autoencoder to discern the most useful features of the input in its hidden representation.

Autoencoders may benefit from stacking multiple layers in the encoder and the decoder in terms of representational power and computational complexity reduction. Each layer reduces further the dimensionality of its input. The most common way of training this type of autoencoder is by greedily feeding the learned hidden features as the input of subsequent autoencoders. As shown in TABLE 3, this approach is followed by [43], [49] in which their inputs to the first autoencoder are the hashed n-grams for each event prefix used to train the network. These approaches use the two-step procedure shown in Fig. 2 to train an autoencoder. First, in Fig. 2a a stacked undercomplete autoencoder is trained to represent the most useful features by reducing the dimensionality of its input. This autoencoder is composed of an encoder, which maps the original input to a smaller hidden representation, and a decoder, which maps the hidden representation back to the original input. Then, in Fig.2b the already trained encoder is used to map the input to a hidden representation. Then, a feedforward network (Ω) is attached using the hidden representation of the encoder as an input. In the final layer, a softmax classification is applied to predict, in this case, the next activity.

In the context of predictive monitoring, an autoencoder approach may be useful when the number of attributes of the log is very high since it could help to reduce the dimensionality of the input data by selecting the most important features for each event. However, the main disadvantage of this approach resides in that they disregard longer dependencies between events of the partial trace.

The autoencoder architecture presented here could be improved by adding a sparsity penalty to the loss function [65] or by corrupting the inputs with noise [66], but these improvements have not yet been explored in predictive monitoring.

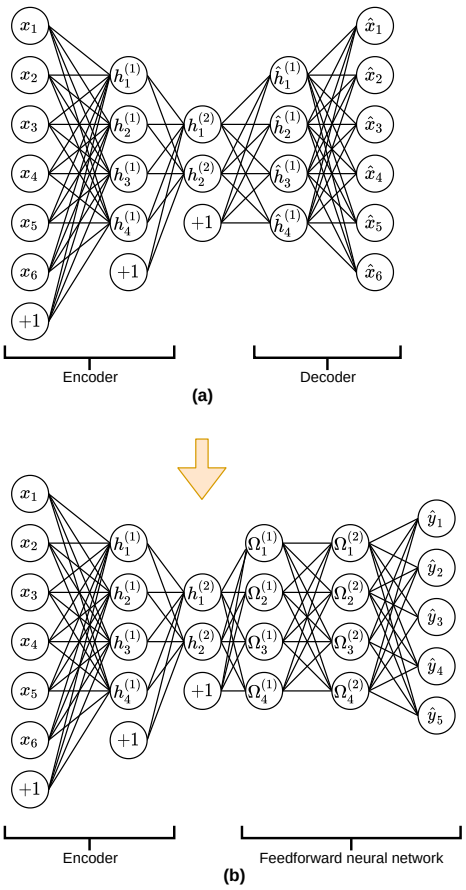

Fig. 2: Representation of the training of an autoencoder.

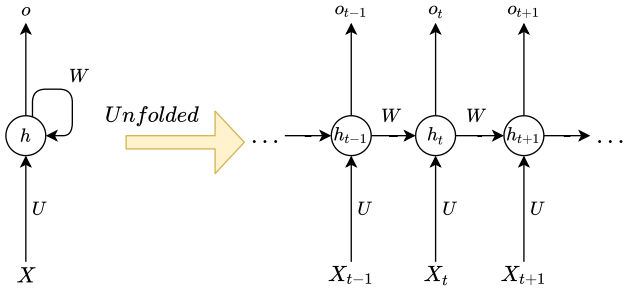

Fig. 3: Graphical representation of a vanilla recurrent neural network. In the left part, the general model is presented. In the right part, the computational graph is unfolded three timesteps.

5.3.3 Recurrent Neural Network

Many deep predictive monitoring approaches [23], [24], [25], [38], [44], [46], [47], [48], [50], [51], [52], [56], [58], [59], [59], [60] are based on Recurrent Neural Networks (RNN) [64]. RNNs are neural networks specialized in processing sequential data, that is, they operate with a sequence of vectors $x_1, \ldots, x_\tau$, where τ is the sequence length. The ability of processing sequence-like data makes this type of neural networks very useful to predictive monitoring.

Simple Recurrent Neural Networks. In their simplest form, a recurrent neural network is graphically represented in Fig. 3. It can be formally defined as follows:

$$\begin{aligned} h_t &= \tanh(b + Wh_{t-1} + Ux_t) \\ o_t &= c + Vh_t \end{aligned} \tag{3}$$

where:

- h_t denotes the hidden state of the recurrent neural network. This hidden state acts as an summary of the past sequence inputs up to the timestep t.
- x_t refers to the input vector in the timestep t.
- b and c are bias vectors and W, U and V are weight matrices. The parameters of these matrices are updated with an algorithm called *backpropagation through time* (BPTT) [67], which allows applying the backpropagation algorithm to RNNs.
- o_t is the output of the recurrent neural network in the timestep t.

This implementation of the RNN poses an important problem: the gradients propagated using BPTT either vanish or explode when trying to learn long dependencies. There exists multiple alternative models that have been proposed

to alleviate this problem: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Memory Augmented Networks (MANN).

LSTM and GRU. As shown in TABLE 3, LSTMs [23], [24], [25], [38], [44], [47], [48], [50], [51], [58], [59] and GRUs [52], [59] have been widely applied in predictive monitoring and are two of the most popular architectures in this field. LSTMs [68] and GRUs [69] create paths through time that allow the gradients to flow deeper in the sequence than in a vanilla RNN. Thus, instead of using the previous state directly, h_{t-1}, LSTM, and GRUs use a memory cell C_t that has an internal recurrence and the usual recurrence of vanilla RNN.

In the case of LSTMs, this internal recurrence is controlled by three different gates, f_t, o_t, and i_t, which control the flow of information inside the cell. f_t is called the "forget gate", which filters what information is thrown away from the cell state; i_t is the "input gate", which controls what information is going to be updated; and o_t is the "output gate", which decides what information is exposed from the cell. The definition of the formulas that define an LSTM is as follows:

$$f_t = \sigma(b_f + U_f x_t + W_f h_{t-1}) \tag{4}$$

$$i_t = \sigma(b_i + U_i x_t + W_i h_{t-1}) \tag{5}$$

$$o_t = \sigma(b_o + U_o x_t + W_o h_{t-1}) \tag{6}$$

$$\tilde{C_t} = tanh(b_C + U_C x_t + W_C h_{t-1}) \tag{7}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C_t} \tag{8}$$

$$h_t = o_t \circ tanh(C_t) \tag{9}$$

In the previous equations, x_t represents the input to the LSTM in the timestep t, b is a bias vector; U and W are trainable weight matrices; h_{t-1} represents the previous hidden state; $\tilde{C_t}$ is the calculation of the cell state for the current timestep; and, finally C_t is the combination of the past information of the cell with the current information of the cell. The $\circ$ operation denotes the Hadamard product between two matrices.

GRUs are similar to LSTMs with the main difference that they do not have an output gate. Formally, GRUs can be formally defined as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{10}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{11}$$

$$\tilde{h_t} = tanh(W_h x_t + U_h(r_t \circ h_{t-1}) + b_z) \tag{12}$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h_t} \tag{13}$$

In the previous equations, z refers to the "update gate", which controls the amount of information that flows from the past to the future; r is called the "reset gate", which filters how much information from the past is forgotten; $\tilde{h}$ represents the calculation of the current memory; and h_t corresponds to the final calculation of the memory of the cell, which can be interpreted as how much information is retained from the past and how much information is updated.

In practice, there is almost no difference in the performance of GRUs against LSTMs, but the former has the advantage of a faster training [70], [71].

Bidirectional Recurrent Neural Network. [56] uses a bi-directional LSTM with an attention mechanism to predict the outcome of a running case. Bi-directional recurrent neural networks, such as [56], consist on applying one RNN forward and another RNN backwards, concatenating their hidden states of each timestep. Moreover, the attention mechanism, originally devised in [72] and [73], allows learning an alignment vector to weight the importante of each timestep in the prediction.

Memory Augmented Neural Network. In the predictive monitoring approach of [46], an architecture that belongs to the family of the Memory Augmented Neural Networks (MANN) is proposed. The family of MANN architectures may be useful in predictive monitoring for learning longer dependencies when the traces of the log are very long or when cycles of the same event may make the LSTM and GRU to "forget" events in the beginning of the trace. However, these architectures are expensive to train and often very sensitive to the hyperparameters used. MANNs use an external memory unit to enhance the learning of longer term dependencies in sequences. The controller is often a Feedforward or a RNN that reads the inputs and, with the help of data from the memory, produces the corresponding outputs.

The oldest MANN architecture is the Neural Turing Machines [74] (NTM). Instead of depending on a single cell for having information from the past, the NTMs use an addressing mechanism to access to this external memory cells. This addressing is based on an attention mechanism that provides a weight vector, w, which highlights the region of the memory more relevant for reading or writing at each timestep. This addressing mechanism allows the neural network to both interact with contiguous regions of memory and jump random addresses. One possible implementation of the memory addressing would use as keys the internal state of the controller LSTM in a certain timestep [75]. This kind of neural network is able to learn longer-term dependencies than its LSTM counterpart. However, the NTMs suffer from training issues (slow convergence, NaNs in gradients, etc.). The Differentiable Neural Computer [76] (DNC) further improves the memory management of the NTM by allowing freeing allocated blocks, keeping track of the writes in memory and avoiding overlapping between memory blocks.

In [46], a variation of the DNC is proposed. In this architecture, the controller is separated into two controllers, the encoder controller and the decoder controller, where both controllers are LSTMs. The encoder reads the input event prefix reading and writing the contents of the memory when necessary. Then, the decoder is initialized with the last state of the LSTM encoder, and the suffix is then predicted.

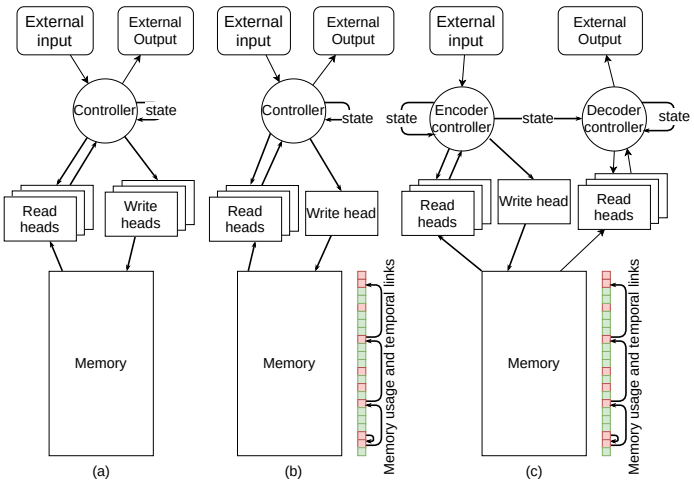

Fig. 4: Comparison between the Neural Turing Machine [74] (Fig. 4a) the Differentiable Neural Computer [76] (Fig. 4b) and the variation of the DNC proposed by Khan et al. [46] (Fig. 4).

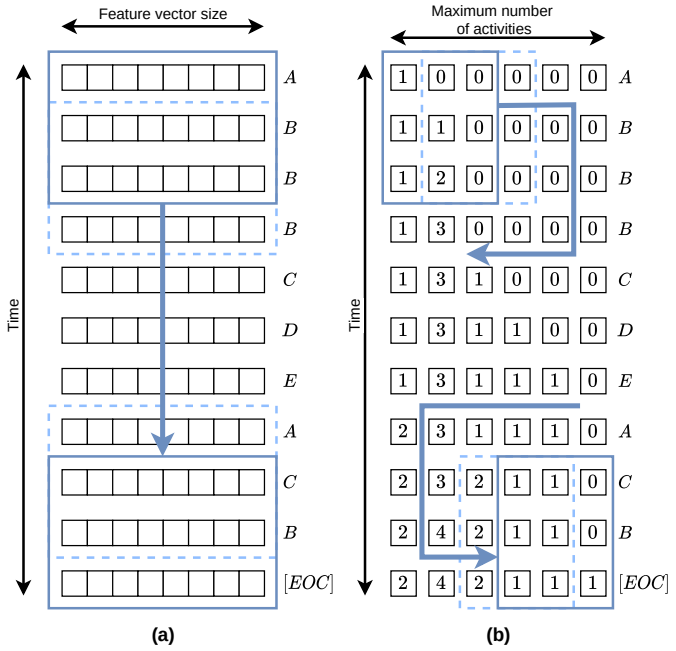

Fig. 5: Differences between 1D convolutions [45], [57] (Fig. 5a) and 2D convolutions [55] (Fig. 5b) for processing the trace $ABBBCDEACB[EOC]$ assuming a total of 6 different activities, including the end of case, on the whole log. The blue square represents the filter and the blue arrow represents the direction of movement over the input matrix. For 1D convolutions, the filter is slided over the time dimension using the full width of the feature vector corresponding to the activities. For 2D convolutions, the filter is slided from right to left and from top to bottom using a smaller kernel size.

One notable aspect of this architecture is that the decoder is prevented from writing into memory, so this architecture is write-protected. Fig. 4 shows a comparison between the NTM from [74], the DNC implementation from [76], and the DNC implementation from [46].

Transformers. The predictive monitoring approach of [60] uses a different architecture named the *transformer*. These type of architecture, originally proposed in [21], substitutes the recurrence entirely by attention mechanisms. They rely on performing an attention operation over different parts of the sequence simultaneously (multi-head attention). In [60] , instead of training a encoder-decoder like in the original proposal of the architecture, they only use the decoder part of the Transformer. Even though the transformer allows a faster training and inference due to the usage of only attention modules, it is still unclear how would the transformer deal with multiple heterogeneous input data, that is, when the inputs to the transformer model are both categorical and continuous data, such as the resources or time-related measures from the events.

5.3.4 Convolutional neural network

Convolutional Neural Networks were applied in the approaches of [45], [57] and [55], as shown in TABLE 3. This type of neural network is specialized in processing grid-like data. Most CNN applied to sequence prediction problems process the data as if it were an one-dimensional (1D) grid [45], [57]. In contrast, some CNNs reengineer their preprocessing of the sequences to adapt them to a two-dimensional (2D) grid [55]. The two main operations performed by this type of neural network are the *convolution* and the *pooling* operations.

The convolution operation takes two different arguments: *(i)* the input to the convolution operation, and *(ii)* the *kernel* (also called filter), which is a matrix of learnable parameters much smaller than the input data. The output of the convolution operation is called the *feature map*. The convolution operation slides the kernel through the input grid accross the input grid's width and height. More formally, the usual convolution operation for 2D data is defined as follows [64]:

$$S(i,j) = (I*K)(i,j) = \sum_{m}^{M}\sum_{n}^{N} I(i+m, j+n)K(m,n) \quad (14)$$

Where $S(i,j)$ is the element (i,j) of the feature map, $*$ is the convolution operation, M is the total height of the kernel, N is the total width of the kernel, I is the input to the convolution operation, and $K(m,n)$ refers to the element (m,n) of the kernel. In the case of 1D data, the convolution operation is applied over the tensor full width, which is the feature vector size, and with a certain height (which is the time axis of the tensor, as shown in Fig. 5a.

Fig. 5 highlights the differences between a 1D convolution and a 2D convolution. Note that for a 2D convolution the time dimension has to be added as an additional channel of the input, while for the 1D convolution, it would just enlarge the feature vector of each event. Thus, for 2D convolutions, including information about the attributes of the event log requires adding additional channels, which might pose a problem when the number of different attributes is mismatched from the number of different activities (the width of the matrix would not be the same).

After a convolution layer, the most usual next layer is the pooling layer, which applies a statistical summary of its output, often the maximum or average of certain portions of the input grid. This kind of transformation has the advantage of learning invariant features to the position inside the trace (local translation invariance). Furthermore, applying a

pooling operation can further reduce the dimensionality of the problem, thus speeding up the training procedure.

The most usual basic architecture for a CNN is a series of building blocks. Each block has a convolutional layer followed by a non-linear activation function, and a pooling layer. The works [45], [55] are based on this neural architecture. [45] follows the 1D convolutional approach, using embeddings for the log categorical variables, whereas [55] uses the 2D convolutional approach, using a frequency-based encoding for continuous and discrete variables, as shown in Fig. 5b. In both studies, the non-linear activation function used between layers is the ReLU function, which allows a faster training and alleviates the vanishing gradient problem [77].

There are two ways to increase the expressivity of a neural network: increasing its depth or its width. When increasing the depth the vanishing gradient problem arises: the gradient updates in latter layers of the network are too small and impede the network to learn properly [19]. When increasing the width, the number of parameters, and therefore, its computational complexity, grows very rapidly. This latter problem has been tackled in [78], where multiple modules of the neural perform both a convolutional operation with different kernel sizes and a pooling operation simultaneously. This approach is used by [57] but with two main differences: they use 1D convolutions, and they do not perform a 1×1 convolution before applying a convolution with a bigger size.

Comparing CNNs against RNNs for predictive monitoring, the former may have the advantage of a faster training and inference, specially for events logs with longer traces. However, RNNs may capture longer depencencies between the events of the trace since the hidden state for a given event depends on every event before it, whereas on a CNN it only depends on the k most recent events, where k is the size of the kernel.

5.4 Sequence encoding

In deep learning architectures, traces must be encoded in tensors of fixed size. However, there is a big variability in the length of every trace in a business process, so this step poses an important challenge. Furthermore, this step also conditions how the training targets are fed to the neural network. We have identified the following encoding formats:

- *Continuous* [23], [45], [51], [59], [60]: this encoding technique is inspired by the training of neural language models [79]. Here, the log is viewed as a text, each trace as a sentence of that text, and each activity as a word of a sentence. In this type of encoding, only a window W of events is considered, and each window can include events from different traces. In case a window is incomplete, it could either be discarded or padded with zeroes. For example, let L be the set of traces $\{[A, B, C], [C, D]\}$. Then, with a window size $W = 3$ we would have the set of windows $\{[A, B, C], [EOC, C, D], [EOC, 0, 0]\}$ as an input. In this case, the training targets fed to the neural network are, in each timestep, the same set of windows shifted one position to the left, i.e, $\{[B, C, EOC], [C, D, EOC], [0, 0, 0]\}$.
- *Prefixes padded* [24], [25], [44], [46], [47], [50], [52], [55], [56], [57], [58]: in this type of encoding, every possible set of event prefixes hd^k where $k \in (1, ..., n)$ for each trace is considered. There are two different approaches to apply this encoding. The first one considers only the W most recent events (as in [50]). The second one considers all the events (as in [25]). In both approaches, the event prefixes must be padded with zeroes in case they are shorter than the specified vector length. In the second case, the vector length is often set to the length of the longest trace of the log. The approaches that use this encoding set the training target for each event prefix to the next event that follows in the event prefix, even though the full event suffix of events could also be the training target.
- *N-gram* [43], [49]: this encoding, used by [43], represents each trace as a set of all subsequences up to length k contained in it. The total number of possible sequences for an event log of $|A|$ distinct activities can be calculated using equation 15.

$$N = \sum_{i=1}^{k} |A|^i \tag{15}$$

  Since the space of n-gram combinations is very large, the "hashing trick" [80] (also known as "feature hashing") is used to reduce this dimensionality to a fixed length vector. The hashing trick is defined as in equation 16:

$$n_i = \sum_{i:h(i)=k} \xi(i) x_i \tag{16}$$

  A hash function h is used to determine the k position in the fixed vector that has to be updated with a feature x_i. Another hash function ξ counters the effect of hash collisions by determining the sign of the update.
  In predictive monitoring, this type of encoding has only been used for autoencoders [43], [49] so, in these proposals the training targets as well as the inputs of the NN are equal.
- *Single event* [54]: in this encoding, only a single event and its attributes are considered, so the sequence of events in the trace is disregarded. The approach of [54] uses this encoding, setting the training targets as the next event to the event in question.
- *Timed state* [53]: recently proposed by [53], this encoding represents the inner state of a Petri net after replaying a partial trace in it. Each place of the Petri net is enhanced with a "decay function" that counts the time between the current timestamp and the last time a token was in a given place. The sequence encoded vector is defined as a concatenation of the following hand-crafted vectors: F_t gives the value of the decay function for each place of the Petri net; C_t counts the number of times a token has gone through a place of the Petri net; M_t counts where the tokens are in the petri net, and R_t: counts the occurrence of other attributes of the trace. The training target is the next event after the replayed activity prefix on the Petri net.

The most used encodings are *Continuous* and *Prefixes padded*, since they are versatile enough to be used with both CNNs and RNNs. The *single event* encoding is

used less since it disregards the dependencies between the events of the log. The *timed state* encoding has the advantage of using the model as an input and, potentially, to capture dependencies between the activities that are not present by examining the literal ordering of the events in the log. However, this encoding is not directly compatible with more expressive models such as RNNs or CNNs. In the comparison between *continuous* and *prefix padded* encodings, the former may benefit of a faster training at the expense of not capturing longer dependencies than in the latter encoding.

5.5 Event encoding

Attribute variables can be either categorical variables or continuous variables. On the one hand, continuous variables must be normalized before feeding to the neural network. There are multiple techniques such as log-normalization [50], min-max normalization [50], z-score normalization [81] or tanh-estimators [82]. On the other hand, each categorical variable must be encoded in fixed feature vectors that uniquely represents them. There are various strategies used in the literature for that:

- *One-hot* [24], [25], [44], [46], [47], [48], [52], [52], [56], [58]: categorizing the variable with an integer is not enough since this categorization assumes that the higher the value of the variable is, the more important it is. To avoid that problem, the feature is represented in a binary vector where its size corresponds to the number of possible distinct values for that variable, and its position in the vector is a one if the category corresponds with the variable.
- *Embedding* [23], [45], [51], [54], [57], [58], [60]: the embedding encoding creates a matrix $W \in \mathbb{R}^{n \times f}$ where each row corresponds to each of the categories of the variable, and columns correspond to the feature dimension. The parameters of this matrix can be either established randomly or be learned with stochastic gradient descent, so the learned embeddings are optimal for the prediction task at hand.
- *Frequency-based* [53], [55]: this type of encoding [83] indicates how many times the activity i has happened until the current event of the trace. This encoding is useful when temporal information must be added to the encoding of the activities. Note that in the case of [53] the frequency does not represent directly activities but the number of times a token has gone through a place.
- *Pretrained embedding* [50]: instead of directly training the embedding vectors with stochastic gradient descent, the embeddings can be pretrained for another task that gives additional information. For example, in [50], embeddings are pretrained as a combination of the learned embeddings of roles and activities.

While the *pretrained embedding* and *embedding* approaches have the main purpose of learning a set of embeddings to represent the categorical variables, the former may benefit of allowing a better convergence of the neural network since they are trained for a task that is different from the next activity prediction task. Thus, the embeddings learned with pretrained embeddings may provide more information, which eases the convergence of the neural network. Moreover, the *one-hot* encoding is a good solution when the number of distinct possible variables is low since it does not use additional parameters, otherwise, the size of the vector could dramatically increase the memory usage of the neural network.

6 Benchmark

6.1 Experimental setup

6.1.1 Datasets

We performed the experiments using 12 real-life event logs from a variety of domains. These event logs were extracted from the *4TU Center for Research Data*[3] and are also available in the repository of our comparison tool. TABLE 4 shows some relevant statistics from these logs, namely, the number of cases, the number of different activities, the number of events, the average and maximum case length, the maximum and average event duration in days, the average and maximum case duration in days and the number of different variants. Most logs have a high event time variability (difference betweeen average event duration and maximum event duration), and a high trace length variability. The log "Nasa" shows 0 in the time related measures since the time variability in this log is low.

6.1.2 Data split

All approaches have been evaluated with the same logs' split and in the same conditions. This is specially important so as to obtain comparable results between every tested approach. We aim to simulate the situation in which the knowledge of the past is used to train a predictive model, and then the model is used to predict the future. For that, the traces of the event logs are ordered by its first event timestamp and then split out in train-validation-test sets, with a trace distribution 64%-16%-20%. This split procedure is used since a cross-validation could dramatically increase the training time for some approaches.

Furthermore, many approaches add an "end of case" token at the end of every trace of the log. This modification has two main advantages. Firstly, state of the process is reduced. Secondly, it gives a clear stop condition for activity suffix generation. To unify the procedure and to make the metrics of next activity prediction comparable, we augmented the log with an end of case activity at the end of each trace for every approach tested.

6.1.3 Metrics

Depending on the predictive task, we use the following metrics for evaluating the performance of the approaches:

- *Next activity prediction*: since the next activity prediction task is a classic classification problem, we use the *accuracy* metric. The accuracy measures the proportion of correct classifications in relation to the number of predictions done. Other measures such as the Matthews Correlation Coefficient [84] or the weighted F1 score are reported by the approaches tested but we found that the

3. https://data.4tu.nl/repository/collection:event_logs_real

Event log	Num. cases	Num. activities	Num. events	Avg. case length	Max. case length	Avg. event duration	Max. event duration	Avg. case duration	Max. case duration	Variants
Helpdesk	4580	14	21348	4.66	15	11.16	59.92	40.86	59.99	226
BPI 2012	13087	36	262200	20.04	175	0.45	102.85	8.62	137.22	4366
BPI 2012 Complete	13087	23	164506	12.57	96	0.74	30.92	8.61	91.46	4336
BPI 2012 W	9658	19	170107	17.61	156	0.7	102.85	11.69	137.22	2621
BPI 2012 W Complete	9658	6	72413	7.5	74	1.75	30.92	11.4	91.04	2263
BPI 2012 O	5015	7	31244	6.23	30	3.28	69.93	17.18	89.55	168
BPI 2012 A	13087	10	60849	4.65	8	2.21	89.55	8.08	91.46	17
BPI 2013 closed problems	1487	7	6660	4.48	35	51.42	2254.84	178.88	2254.85	327
BPI 2013 incidents	7554	13	65533	8.68	123	1.57	722.25	12.08	771.35	2278
Sepsis	1049	16	15214	14.48	185	2.11	417.26	28.48	422.32	845
Env. permit	1434	27	8577	5.98	25	1.09	268.97	5.41	275.84	116
Nasa	2566	94	73638	28.7	50	0.0	0.0	0.0	0.0	2513

TABLE 4: Statistics of the event logs used for benchmarking. Time related measures are shown in days.

results were aligned with the accuracy measure and do not give additional information. Therefore, these results are not reported in this paper.

- *Activity suffix prediction*: when predicting an activity suffix in the context of predictive monitoring, it is important to take into accout that the activities in the process may occur in parallel [25]. Thus, instead of the metrics used for the next activity prediction task, we use the *Damerau-Levenshtein distance* metric. This metric measures the edit distance between two given strings without penalizing too harshly transpositions of tokens, which, in the context of predictive monitoring, could mean a pair of parallel activities. These two strings represent the predicted activity suffix for a given event prefix and its ground truth activity suffix. The Damerau-Levenshtein metric measures the number of insertions, deletions, substitutions, and transpositions needed to transform one string into another. This value is then normalized by the lengths of the two strings, obtaining a value of similarity between 0 and 1.
  In relation with the activity suffix prediction, we also report results of the *Brier score* measure. The Brier score is similar to the accuracy measure, but it also gives a sense of how well the predictions are calibrated with respect to the ground truth. This is specially useful if we are interested in sampling from the probability vector, which is the case of the activity suffix prediction task. The Brier score is calculated as the square of the differences between the predicted probabilities for a given item and the ground truth. More formally, in a multiclass setting, the Brier score is defined by equation 17:

$$BS = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{R}(f_{ti} - o_{ti})^2 \in [0, 2] \tag{17}$$

- *Next timestamp prediction and remaining time prediction*: since the time prediction problem is a regression task, the metric selected for measuring the performance in the next timestamp prediction tasks and remaining time prediction tasks is the *Mean Absolute Error* (MAE). This metric is defined in equation 18 and has the advantage of not overpenalizing the variability in the observations [85], which is the case in time prediction in predictive process monitoring, where the time between two events in a trace can be potentially large [25], [35].

$$MAE = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{N} \tag{18}$$

6.1.4 Approaches and experimental setup

We performed the benchmark with 10 different approaches from the state of the art [23], [25], [44], [46], [50], [52], [53], [55], [57]. We left out of the evaluation approaches that predict outcome [47], [59] since it would require a preprocessing that would be too specific for some datasets. We used the source code of each approach whenever it was publicly available. Otherwise, we contacted the authors of every surveyed approach of which we could not find the publicly available code, or had problems to reproduce the experiments [24], [46]. We modified the approaches to load the splits, substitute their split procedures (if present), allow a proper command-line usage, calculate more metrics than originally intended by the authors or update the used libraries to more recent versions. The original implementations locations are specified in TABLE 5. For the approach of [53] we show the location of the code used in the original paper and its reimplementation from the same authors.

Approach	URL(s)
Khan et al. [46]	https://github.com/thaihungle/MAED/tree/deep-process
Evermann et al. [23]	https://joerg.evermann.ca/docs/rnn_process_data_scripts_results.tar.gz
Navarin et al. [44]	https://github.com/nickgentoo/DALSTM_PM
Tax et al. [25]	https://github.com/verenich/ProcessSequencePrediction
Theis et al. [53]	https://github.com/Julian-Theis/DREAM-NAP and https://github.com/Julian-Theis/PyDREAM
Mauro et al. [57]	https://github.com/nicoladimauro/nnpm
Pasquadibisceglie et al. [55]	https://github.com/vinspdb/ImagePPMiner
Camargo et al. [50]	https://github.com/AdaptiveBProcess/GenerativeLSTM/
Hinkka et al. [52]	https://github.com/mhinkka/articles/tree/master/Exploiting%20Event%20Log%20Event%20Attributes%20in%20RNN%20Based%20Prediction
Francescomarino et al. [24]	https://github.com/yesanton/Process-Sequence-Prediction-with-A-priori priori-knowledge

TABLE 5: Original code repositories from the studies evaluated.

In particular, we made the following adaptations:

- **Pasquadibisceglie et al. [55]**. We adapted the code to support more use-cases than in the original paper.

Moreover, we optimized the memory usage by changing the datatype of the encoding from 64-bit to 16-bit floating point, and by avoiding loading the entire dataset in RAM to support bigger logs than originally intended.

- **Mauro et al. [57]**. No major changes were made.
- **Tax et al. [25]**. We refactored the code and updated the python version used from 2.7 to 3.6.
- **Evermann et al. [23]**. There are two code versions of the paper provided by the authors: the original one, implemented in Tensorflow 0.12, and an updated version implemented in TensorFlow 1.0. The first one has the disadvantage of requiring the log in a text format and the difficulty of porting it to a newer version of Tensorflow. The second one is tightly coupled with a graphical interface, making it unsuitable to use in the command line. Since the first implementation is based on the Tensorflow RNN tutorial (as stated in the comments of the code), we reimplemented the approach using a newer version of the RNN tutorial[4], which uses the same training procedure as the implementation of Evermann. We used the same hyperparameters as in the original approach.
- **Khan et al. [46]**. The code provided originally by the authors does not contain the preprocessing and encoding stage for any log since it loads the already preprocessed datasets. As far as we know, the code also does not support the prediction of activity suffixes and remaining time, even though in the original paper these results are reported. Therefore, the results reported in this paper are only concerned with the next activity and timestamp. We used the same preprocessing as in Tax et al. [25].
- **Theis et al. [53]**. Running the code for each log is done in two stages. In the first stage, a process model is mined using the Split Miner [86] by performing a grid search using the same search space, as reported in the original paper [53]. The best model is selected as the mined model with the highest fitness. In the second stage, for generating the training sets and training the neural network, a reimplementation made by the authors of the paper is used[5], instead of the original implementation of the paper[6]. The reason for this change is that the original implementation uses a cross-validation procedure that is difficult to circumvent. There is also a ProM plugin[7] which also generates cross-validation folds, but it was not used since the plugin is already compiled and is very hard to modify. Furthermore, we reported the results for two variants of the implementation: one that uses resources "(w/ resources)" and other that does not use them "(w/o resources)".
- **Navarin et al. [44]**. No major changes were made. Since the original paper claims that the neural network is not very sensitive to hyperparameters, we trained every dataset using a two-layer with 150 hidden units per layer LSTM. Note that for the BPI 2013 Incidents dataset, due to the high number of attributes used, the memory usage escalates quickly, achieving an usage of 160GB of RAM.
- **Camargo et al. [50]**. We reported results for two different methods of sampling the next activity when predicting an activity suffix of activities: always selecting the highest probability activity "(argmax)" and randomly sampling the next activity following the probability distribution of the neural network output "(random)". We performed a random search hyperparameter optimization procedure following the search space provided in the code and training 20 models per dataset. The architecture used is fixed to the "Shared categorical" architecture reported in the paper [50]. In this architecture, a shared layer integrates the information for the activity and role prefixes, whereas the time prediction is taken separated from the shared layer.
- **Hinkka et al. [52]**. Apart from changing the data split procedure and implementing an adapter to convert logs from CSV to JSON, no major changes were made.
- **Francescomarino et al. [24]**. The execution of this approach involves two main steps: mining the LTL rules from each dataset, which are a series of temporal restrictions that the activities of the log shall follow, and predicting the activity suffix using those rules. We restrict the experimentation to only a "Strong" rule set since it seems to have better results reported in the original paper. To mine the rules we rely on the Rule Mining Tool[8], and, to avoid leaking information from the testing set, we use the whole validation set with a "minimum constraint support" of 10% and vacuity detection enabled. The templates used for mining are "existence" ($\Diamond A$) and "response" ($\Box(A \rightarrow \Diamond B)$). After mining the rules, we apply the LTL Checker to verify how many rules are satisfied in the validation event log. However, the count reported by the tool counts the number of events that the rule fullfils and not the number of traces in which is fulfilled. Furthermore, a same rule can fulfill a trace multiple times. Thus, instead of taking the rules that are satisfied in more of the 50% of the validation set, we take the rules that have a number of fulfillments more than 50% of the activations ordered by the number of fulfillments up to a maximum of three different rules.

The experiments were carried out in a server equipped with an Intel Xeon Gold 5220, 192 GB RAM, and 2 Nvidia Tesla V100S 32GB. TABLE 6 reflects the event attributes used by the approaches. Trace level attributes have not been taken into consideration. To perform the specific preprocessing for each dataset, we rely on the Pm4Py library [87]. As shown in the aforementioned table, the highest count of attributes is present in the datasets "BPI 2013 Incidents" and "Nasa". Note that not every dataset has available the resource assigned to the event ("org:resource") so, if the approach uses specifically this attribute, it can not be evaluated.

6.2 Results and discussion

In this section we detail the results of the benchmarking for each of the prediction tasks evaluated: next activity pre-

4. https://www.tensorflow.org/tutorials/text/text_generation
5. https://github.com/Julian-Theis/PyDREAM
6. https://github.com/Julian-Theis/DREAM-NAP/
7. https://prominentlab.github.io/ProM-DREAM/
8. https://sep.cs.ut.ee/Main/RuM

Dataset	Attributes
BPI 2013 Incidents	org:group, resource country, organization involved, org:role, impact, product, lifecycle:transition, org:resource
BPI 2013 Closed Problems	org:group, org:resource
BPI 2012	lifecycle:transition, org:resource
BPI 2012 Complete	org:resource
BPI 2012 W	lifecycle:transition, org:resource
BPI 2012 W Complete	org:resource
BPI 2012 A	org:resource
BPI 2012 O	org:resource
Sepsis	org:group
Nasa	apploc:joinpoint, apprun:exthrowtype, apploc:etype, apprun:excatchtype, lifecycle:transition
Env. permit	org:group,org:resource
Helpdesk	org:resource

TABLE 6: Event attributes used for "data-aware" approaches.

diction, activity suffix activity prediction, next timestamp prediction and remaining time prediction.

6.2.1 Next activity prediction

TABLE 7 shows the accuracy scores obtained by the approaches in the tested datasets. Overall, the approaches obtain very close results as far as the accuracy is concerned with the notable exception of the "BPI 2013 Incidents" log, in which the differences are more prominent. In particular, the approaches of Camargo et al. [50], Tax et al. [25], Hinkka et al. [52] and Mauro et al. [57] obtain very similar results in every dataset except "BPI 2013 Incidents", and "BPI 2013 closed problems". Taking into account that Tax et al. and Mauro et al. use almost the same input data (Tax et al. uses more time features), this may indicate that LSTMs and CNN 1D are not very far away in performance. However, LSTMs may have a slight advantage in logs where the strict sequence of activities is more important than how the patterns of activities are arranged inside the trace. Moreover, in some datasets, namely "BPI 2012" and "BPI 2012 W Complete", the approach of Camargo et al. performs a bit worse than expected. This is because the "BPI 2012" dataset, and some of its subprocesses, lack information about the resources of some of the events. For those events, the resource assigned is "Unknown". Thus, the role discovery algorithm may underperform in that situation.

The approach of Hinkka et al. [52] obtains the best score in 6 of the 12 datasets. However, the difference with the second-best approach is less than 1% except in "BPI 2013 Incidents". In this log, as TABLE 6 shows, the number of attributes is high, showing that the clustering of attributes may give important information to predict the next activity.

The approach from Theis et al. [53] obtains the best score in two of the tested datasets. Even though the gap between the best result in the Nasa dataset is low (the distance between Theis et al. and Tax et al. is 1.45%), the difference is more important in the "BPI 2012 W Complete" dataset (excluding Khan et al., the distance between Theis et al. approach and Tax et al. is 9.17%). While the encoding of Theis et al. does not impose a strict ordering on the events in the trace, showing that the process model may have important information that can be used to predict the next activity in a partial trace. The other close performing approach is Khan et al, which uses a Memory Augmented Neural Network (MANN) and the same inputs as both Mauro et al. and Tax et al. The difference of performance may indicate that learning long dependencies may be benefitial in logs where the presence of loops may "confuse" LSTMs and CNN to predict always the activity that is causing the loop.

Tax et al. [25] approach obtains the best score in two of the tested datasets. While the difference in the "Sepsis" dataset is small (1.05%, w.r.t the second-best approach), it is more significant in the "BPI 2013 closed problems" (with a difference of 4.43%). It appears that the usage of extra attributes, as shown in the TABLE 6, does not help the prediction performance in this log.

Comparing Tax et al. w.r.t Mauro et al., which almost uses the same inputs (Mauro et al. uses less time features), the difference in the prediction performance in the "BPI 2013 closed problems" may indicate either some of the additional extracted time features by Tax et al. are useful (namely: time since the start of the case, week of the day, and time since midnight [25]) or that the strict ordering of the activities is important in this log, which would make LSTM more suitable than CNN.

If we compare Khan et al. against Tax et al., which uses the same preprocessing with the same time features, since the "BPI 2013 closed problems" has a low number of cases, as shown in TABLE 4, may indicate that the MANN is overfitting in this dataset since it is far more complex than the LSTM used by Tax et al.

The approach from Khan et al. [46] obtains the best score in 1 of the 12 tested datasets with a difference between the second-best approach of a 0.92%. Furthermore, this approach outperforms the Tax et al. approach in 5 of the 12 datasets. Most notably, it obtains good results in the "W" subprocesses of the "BPI 2012" log (namely "BPI 2012 W" and "BPI 2012 W Complete"). As mentioned before, this may be the consequence of learning longer dependencies than the LSTMs and the CNNs avoiding to predict always the same activities that belong to the loops in the process.

The approach from Pasquadibisceglie et al. obtains the best score in 1 of the 12 datasets. Even though the difference between the second and third best approaches is low ($<$ 1%) it may show some potential in the sense of using a frequency-based encoding approach, instead of using a one-hot encoding to represent the activities. However, this approach severly underperforms in the datasets "BPI 2013 closed problems" and "BPI 2013 incidents". Those logs, as shown in TABLE 4, have one of the highest maximum event and case durations and a low number of activities. The number of activities in the log affects the length of the vector that is going to be used to represent the time features. A low length of this vector combined with a high time variability may make the approach to underperform due to a lack of feature representation. Moreover, if the number of activities is higher or the time variability is less accused, the approach performs better, which is the case of the "Env permit" log with 27 activities.

6.2.2 Activity Suffix prediction

TABLE 8 shows the activity suffix prediction performance of five different approaches. These approaches present three different ways to select the next activity from the output probability vector of the neural network: *(i)* always selecting the highest probability activity ("argmax"), *(ii)* sampling

	Helpdesk	BPI 2012	BPI 2012 Complete	BPI 2012 W	BPI 2012 W Complete	BPI 2012 O	BPI 2012 A	BPI 2013 closed problems	BPI 2013 Incidents	Sepsis	Env permit	Nasa
Pasquadibisceglie et al.	65.84	82.59	74.55	81.59	66.14	77.51	71.47	24.35	31.10	56.71	**91.47**	87.96
Tax el al.	**75.06**	**85.20**	**79.39**	84.90	67.80	81.22	77.75	**65.57**	67.50	**65.87**	89.24	**88.15**
Camargo et al.	**76.51**	83.41	**79.22**	83.29	65.19	**85.13**	**78.92**	**60.62**	68.01	-	**91.38**	-
Hinkka el al.	**77.90**	**86.05**	**79.76**	83.52	67.24	**85.51**	**79.27**	**61.14**	**77.95**	**64.44**	89.46	87.89
Khan et al.	69.13	82.93	75.50	**86.69**	**75.91**	**84.48**	75.62	55.57	64.34	64.34	89.78	85.51
Evermann et al.	70.07	60.38	63.37	75.22	65.38	79.20	74.44	55.66	**68.15**	34.37	84.33	20.43
Mauro et al.	74.77	**84.56**	78.72	**85.11**	65.01	81.52	**78.09**	56.97	**71.09**	**64.82**	87.29	**88.50**
Theis et al. (w/o attributes)	67.80	77.64	73.10	**85.77**	**76.97**	81.52	66.23	52.31	57.65	55.72	**91.32**	**89.60**
Theis et al. (w/ attributes)	66.25	64.23	65.21	76.16	**72.52**	73.56	65.12	47.69	63.51	56.47	82.35	87.24

TABLE 7: Accuracy measures in % for the next activity prediction (higher is better). The best, second best and third best systems are highlighted in red, blue and green respectively.

the next activity following the probability distribution of the output probability vector ("random") or *(iii)* exploring the probability tree keeping the most promising candidates ("beam search"). Camargo et al. uses both the "argmax" and "random" procedures, Tax et al. uses only the "argmax" procedure, Evermann et al. uses the "random" procedure, and Francescomarino et al. use the "beam search" procedure.

Overall, Camargo et al. using the "random" procedure outperforms the other approaches in 6 of the 10 tested datasets while Evermann et al., which is also a "random" procedure, outperforms Tax et al., which is an "argmax" procedure, in the "Sepsis" and "Nasa" logs. The "random" procedure works better in unstructured logs where the number of cases per trace is high and has many cycles repeating the same activity. In the "argmax" procedure, these cycles may make the neural network to be stuck in always predicting the same cycle activity. In these cases, the neural network may not even predict the end of the case token at all. However, when the process is more structured or simple (such as in the "Nasa" or "Helpdesk" logs), the "argmax" procedure works better.

Since the "argmax" procedure seeks for the most probable activity in each moment, and the accuracy measure matches it with the most probable activity, there is a correlation between the accuracy measure and the activity suffix prediction performance when the sampling procedure is "argmax". However, this is not the case when the sampling procedure is "random" since this method takes into account the full probability vector and not only the most probable activity. For example, the accuracy measure does not explain the prediction performance of logs, such as the "BPI 2013 closed problems", where the Camargo et al. approach outperforms Evermann et al. in terms of accuracy but not in terms of activity suffix prediction. Another case where this happens is in the "BPI 2012 W Complete" log, where Evermann et al. outperforms Camargo et al. in terms of accuracy. In these cases, it is more useful to look at the Brier score measures from TABLE 9. This metric takes into account the whole probability output vector instead of only comparing the highest selected activity. Thus, this metric allows us to evaluate whether the neural network "is doubtful" when it fails to predict the next activity, i.e, allows to dictaminate whether the neural network is "confident" on its predition error (high probability on the wrong next activity) or, instead, if it assigns a lower probability to the erroneous activity. This is especially important since neural networks are often overconfident in their predictions [88]. This could happen due to using multiple event inputs, such as time and role, may help calibrate the neural network predictions and, thus, improve the activity suffix prediction performance.

When comparing the "argmax" procedures between them, Camargo et al. outperforms Tax et al. in 9 of the 10 tested datasets. These results are due to two main reasons: a higher prediction accuracy, as reported in TABLE 7, and a predefined maximum length of the predicted activity suffix that does not depend on the maximum trace length of the log.

When comparing the "random" procedures between them, the Camargo et al. approach outperforms Evermann et al. approach in 8 of the 10 tested datasets. This may be due to the use of the roles apart from the activity sequence, which would help to discover the true probability distribution of the activities in the event log, enhancing the sampling performance, and the prediction performance, as shown in the TABLE 9.

Comparing the approach of Francescomarino et al. with Tax et al, which uses the same architecture and inputs, the approach of Francescomarino et al. outperforms Tax et al. in 3 of the 12 tested datasets: the BPI 2012 W, BPI 2012 W Complete and BPI 2013 Incidents. These result may be due to the fact that the combination of the beam search with a successfull mining of the LTL rules, is beneficial in complex process logs.

6.2.3 Next timestamp prediction and remaining time prediction

TABLE 10 shows the next timestamp prediction performance of 2 different approaches, and TABLE 11 shows the remaining time prediction performance of 4 different approaches.

As far as the next timestamp prediction performance is concerned, the approach of Tax et al. outperforms Khan et al. in 9 of the 11 tested datasets, even though the difference in the performance is low, except in the logs "Helpdesk" and "BPI 2013 closed problems". Note that for these datasets, the accuracy metric, as shown in TABLE 7, is significantly higher in the case of Tax et al. with respect to Khan et al. so this result is expected. However, the approach of Khan et al. obtains better results in the logs "BPI 2012 O" and "BPI 2012 A" than Tax et al. even though its accuracy performance is worse. This could happen due to the multitasking nature

	Helpdesk	BPI 2012	BPI 2012 Complete	BPI 2012 W	BPI 2012 W Complete	BPI 2012 O	BPI 2012 A	BPI 2013 closed problems	BPI 2013 Incidents	Sepsis	Env permit	Nasa
Evermann et al.	0.8069	0.1979	0.2679	0.2769	0.3315	0.5431	0.5817	0.6712	0.4786	**0.2791**	0.5717	**0.1266**
Tax et al.	0.8121	0.1213	0.1750	0.0956	0.0619	0.5651	0.4541	0.4856	0.2327	0.0977	0.8780	0.0601
Camargo et al. (argmax)	**0.8799**	0.1633	0.1828	0.1491	0.0271	**0.6872**	0.6336	**0.7257**	0.3004	-	**0.8889**	-
Camargo et al. (random)	0.8365	**0.3800**	**0.4530**	**0.3391**	**0.3332**	0.5946	**0.6421**	0.5127	**0.5512**	-	0.7556	-
Francescomarino et al.	0.1991	0.1189	0.0902	0.1013	0.2581	0.3481	0.2375	0.4341	0.2743	0.0767	0.1760	0.0530

TABLE 8: Activity suffix prediction Damerau Levenshtein distance (higher is better). The best system is highlighted in bold.

	Helpdesk	BPI 2012	BPI 2012 Complete	BPI 2012 W	BPI 2012 W Complete	BPI 2012 O	BPI 2012 A	BPI 2013 closed problems	BPI 2013 Incidents	Sepsis	Env permit	Nasa
Pasquadibisceglie el al.	0.6584	0.2276	0.3558	0.2686	0.4597	0.3143	0.3535	0.9037	0.9031	0.5905	0.1787	0.1590
Tax el al.	0.3865	**0.1960**	**0.2856**	0.2198	0.4354	0.2570	0.2722	**0.4763**	0.4183	**0.4425**	0.1829	0.1370
Camargo et al.	0.3715	0.2090	0.2867	0.2438	0.4708	**0.1822**	**0.2506**	0.5671	0.4184	-	**0.1615**	-
Hinkka el al.	**0.2729**	**0.1130**	**0.1634**	**0.1265**	**0.2282**	0.2549	0.3155	0.6422	**0.2718**	**0.3155**	0.2225	**0.0953**
Khan et al.	0.4606	0.2168	0.3349	**0.1780**	0.3413	**0.2143**	0.2950	0.6319	0.4196	0.4498	0.1772	0.1809
Evermann et al.	0.4161	0.6271	0.5763	0.4012	0.4968	0.3015	0.3273	**0.4827**	0.4836	0.7732	0.4169	0.8984
Mauro el al.	**0.3710**	0.2008	0.2946	0.2173	0.4676	0.2553	**0.2686**	0.4888	**0.3581**	0.4726	0.2282	0.1377
Theis et al. (w/o attributes)	0.5197	0.3194	0.3675	0.2096	**0.3261**	0.2627	0.4124	0.6290	0.5336	0.5311	**0.1715**	**0.1273**
Theis et al. (w/ attributes)	0.4917	0.5399	0.4843	0.3785	0.4302	0.3848	0.4305	0.6733	0.5054	0.5290	0.2577	0.1918

TABLE 9: Brier score measures for the next activity prediction (lower is better). The best, second best and third best system is highlighted in red, blue and green respectively.

	Helpdesk	BPI 2012	BPI 2012 Complete	BPI 2012 W	BPI 2012 W Complete	BPI 2012 O	BPI 2012 A	BPI 2013 closed problems	BPI 2013 Incidents	Sepsis	Env permit
Tax et al.	5.7766	0.3063	0.4887	0.4960	1.2084	1.6136	0.8387	7.7007	0.4684	0.9361	0.3032
Khan et al.	6.3551	0.3169	0.5191	0.5011	1.3254	1.4495	0.7494	8.7538	0.5450	1.0079	0.3974

TABLE 10: Next timestamp MAE in days (lower is better). Results for the "Nasa" dataset are not reported due to its low time variability.

	Helpdesk	BPI 2012	BPI 2012 Complete	BPI 2012 W	BPI 2012 W Complete	BPI 2012 O	BPI 2012 A	BPI 2013 closed problems	BPI 2013 Incidents	Sepsis	Env permit
Tax et al.	71.5013	330.6143	91.8374	387.8066	210.1586	38.0906	28.5530	192.7177	38.4138	899.9605	**1.4576**
Camargo et al. (argmax)	11.1468	30.5633	11.2842	32.0330	7.9684	19.7792	11.8968	396.3654	260.6419	-	3.7753
Camargo et al. (random)	10.5269	30.9614	9.0870	32.1346	9.2409	19.7944	11.8968	397.2954	260.7660	-	3.7753
Navarin et al.	**10.3792**	**6.1257**	**6.1840**	**6.6268**	**6.4787**	**6.7310**	**6.0542**	**15.6252**	**2.9701**	**17.8189**	1.4595
Francescomarino et al.	273.7709	449.2984	106.4971	329.5750	12.5605	52.7364	36.3395	77.5304	139.7793	759.8114	6.9079

TABLE 11: Remaining time MAE in days (lower is better). The best system is highlighted in bold. Results for the "Nasa" dataset are not reported due to its low time variability.

of the next timestamp prediction: Khan et al. is focusing on predicting time, disregarding the next activity prediction, whereas Tax et al. focuses on predicting the next activity.

As far as the remaining time prediction is concerned the approach from Navarin et al. [44] outperforms the other approaches in 10 of the 11 tested datasets. Note that the results from Tax et al. seem to be very different compared to other approaches. This misadjustment could be due to the denormalization procedure of the predictions, which is different in training than in testing.

Moreover, the differences in the performance between Navarin et al. and Camargo et al. are significant. That is due to the differences in the training procedure between the two approaches. Recalling from the Definition 15, there are two possible ways to train a network to predict the remaining cycle time: training to predict the times of each activity separately or training to predict directly the remaining time. Thus, while the approach of Camargo et al. is trained to predict the next timestamp[9], Navarin et al. approach is trained to directly predict the time until the end of the case. This difference causes the approaches that use the Definition 15, such as Camargo et al., to accumulate errors in each timestep of the prediction whereas approaches that use Definition 16, such as Navarin et al., avoid this problem.

7 Conclusions

In this paper, we presented a systematic literature review of Deep Learning approaches to predictive business process monitoring. We made an analysis and classification of

9. Unfortunately, the code provided by Camargo et al. does not report the performance for the next timestamp prediction.

these approaches, supported by an exhaustive experimental evaluation of 10 approaches on 12 different process logs, according to five different perspectives: *(i)* input data, *(ii)* predictions, *(iii)* neural network type, *(iv)* sequence encoding, and *(v)* event encoding.

Regarding to the input data, as shown by the experimentation, the usage of additional attributes available on the event log may benefit the predictive performance of the predictive task at hand. As far as the predictions is concerned, we showed how the different sampling strategies affect the predictive performance of the activity suffix prediction task, concluding that *random sampling* approaches outperform *argmax sampling* approaches when the length of the traces of the log is big and viceversa. Relating to the neural network type, LSTMs and GRUs work well along a wide array of predictive tasks. However, when tackling a new predictive monitoring problem, it could be relevant to explore other architectures such as CNNs or MANNs due to the slow training speed of LSTMs and the fact that LSTMs tend to "forget" information in business processes with many cycles or very long traces. Concerning the sequence and event encoding perspectives, even though the experimentation does not highlight differences on the performance of the multiple possibilities of encoding, there are differences on the efficiency between the encodings. For example, the *one-hot* event encoding uses more memory than the *embedding* event encoding when the number of different activities is very high or the *continuous* sequence encoding has a faster training procedure than the *prefixes padded* sequence encoding.

We expect the experiments serve as a baseline for future works in predictive monitoring. As a future line of work, this experimentation could be enhanced with non-deep learning approaches or by integrating it with a graphical interface that eases the benchmarking procedure and visualization.

8 Acknowledgments

This research was funded by the Spanish Ministry for Science, Innovation and Universities (grant TIN2017-84796-C2-1-R and TIN2015-73566-JIN) and the Galician Ministry of Education, University and Professional Training (grants ED431C 2018/29 and "accreditation 2016-2019, ED431G/08"). All grants were co-funded by the European Regional Development fund (ERDF/FEDER program). E. Rama-Maneiro is supported by the Spanish Ministry of Education, under the FPU national plan (FPU18/05687).

References

[1] W. M. P. van der Aalst, *Process Mining*. Springer Berlin Heidelberg, 2011.

[2] W. M. P. van der Aalst et al., "Process mining manifesto," in *Proceedings of the 9th International Business Process Management Workshops (BPM 2011)*, ser. Lecture Notes in Business Information Processing, vol. 99. Springer, 2011, pp. 169–194.

[3] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," *Information Systems*, vol. 54, pp. 209–234, 2015.

[4] F. M. Maggi, C. D. Francescomarino, M. Dumas, and C. Ghidini, "Predictive monitoring of business processes," in *Proceedings of the 26th International Conference on Advanced Information Systems Engineering (CAISE 2014)*, ser. Lecture Notes in Computer Science, vol. 8484. Springer, 2014, pp. 457–472.

[5] F. Mannhardt and D. Blinde, "Analyzing the trajectories of patients with sepsis using process mining," in *Proceedings of the CEUR Workshop*, ser. CEUR Workshop Proceedings, vol. 1859. CEUR-WS.org, 2017, pp. 72–80.

[6] W. van der Aalst, M. Schonenberg, and M. Song, "Time prediction based on process mining," *Information Systems*, vol. 36, no. 2, pp. 450–475, 2011.

[7] A. Senderovich, M. Weidlich, A. Gal, and A. Mandelbaum, "Queue mining for delay prediction in multi-class service processes," *Information Systems*, vol. 53, pp. 278–295, 2015.

[8] J. Becker, D. Breuker, P. Delfmann, and M. Matzner, "Designing and implementing a framework for event-based predictive modelling of business processes," in *Proceedings of the Enterprise Modelling and Information Systems Architectures (EMISA 2014)*, ser. LNI, vol. P-234. GI, 2014, pp. 71–84.

[9] D. Breuker, , M. Matzner, P. Delfmann, J. Becker, , and and, "Comprehensible predictive models for business processes," *MIS Quarterly*, vol. 40, no. 4, pp. 1009–1034, 2016.

[10] A. Rogge-Solti and M. Weske, "Prediction of business process durations using non-markovian stochastic petri nets," *Information Systems*, vol. 54, pp. 1–14, 2015.

[11] A. Senderovich, A. Shleyfman, M. Weidlich, A. Gal, and A. Mandelbaum, "Optimal model simplification for improving accuracy in process performance prediction," in *Proceedings of the 14th International Conference on Business Process Management (BPM 2016)*, ser. Lecture Notes in Computer Science, vol. 9850. Springer, 2016, pp. 418–436.

[12] B. Kang, D. Kim, and S.-H. Kang, "Periodic performance prediction for real-time business process monitoring," *Industrial Management & Data Systems*, vol. 112, no. 1, pp. 4–23, 2012.

[13] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and C. D. Francescomarino, "Minimizing overprocessing waste in business processes via predictive activity ordering," in *Proceedings of the 28th International Conference on Advanced Information Systems Engineering (CAISE 2016)*, ser. Lecture Notes in Computer Science, vol. 9694. Springer, 2016, pp. 186–202.

[14] C. Cabanillas, C. D. Ciccio, J. Mendling, and A. Baumgrass, "Predictive task monitoring for business processes," in *Proceedings of the 12th International Conference on Business Process Management (BPM 2014)*, ser. Lecture Notes in Computer Science, S. W. Sadiq, P. Soffer, and H. Völzer, Eds., vol. 8659. Springer, 2014, pp. 424–432.

[15] 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 *Proceedings of the 15th International Conference on Enterprise Information Systems (ICEIS 2013)*, INSTICC. SciTePress, 2013, pp. 56–65.

[16] F. Folino, M. Guarascio, and L. Pontieri, "A prediction framework for proactively monitoring aggregate process-performance indicators," in *Proceedings of the 2015 IEEE 19th International Enterprise Distributed Object Computing Conference (EDOC 2015)*. IEEE, 2015, pp. 128–133.

[17] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and C. D. Francescomarino, "Complex symbolic sequence clustering and multiple classifiers for predictive process monitoring," in *Proceedings of the 13th International Conference on Business Process Management Workshops (BPM 2015)*, ser. Lecture Notes in Business Information Processing, vol. 256. Springer, 2015, pp. 218–229.

[18] W. L. J. Lee, D. Parra, J. Munoz-Gama, and M. Sepúlveda, "Predicting process behavior meets factorization machines," *Expert Systems with Applications*, vol. 112, pp. 87–98, 2018.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. IEEE, 2016, pp. 770–778.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, Y. Bengio and Y. LeCun, Eds., 2015.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need,"

in *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS 2017)*, 2017, pp. 5998–6008.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS 2014)*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 3104–3112.

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

[24] C. D. 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 *Proceedings of the 15th International Conference on Business Process Management (BPM 2017)*, ser. Lecture Notes in Computer Science, vol. 10445. Springer, 2017, pp. 252–268.

[25] N. Tax, I. Verenich, M. L. Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proceedings of the 29th International Conference on Advanced Information Systems Engineering (CAISE 2017)*, ser. Lecture Notes in Computer Science, vol. 10253. Springer, 2017, pp. 477–492.

[26] H. T. C. Nguyen, S. Lee, J. Kim, J. Ko, and M. Comuzzi, "Autoencoders for improving quality of process event logs," *Expert Systems with Applications*, vol. 131, pp. 132–147, 2019.

[27] T. Nolle, A. Seeliger, and M. Mühlhäuser, "Binet: Multivariate business process anomaly detection using deep learning," in *Proceedings of the 16th International Conference on Business Process Management (BPM 2018)*, ser. Lecture Notes in Computer Science, vol. 11080. Springer, 2018, pp. 271–287.

[28] G. Park and M. Song, "Prediction-based resource allocation using LSTM and minimum cost and maximum flow algorithm," in *In Proceedings of the 2019 International Conference on Process Mining (ICPM 2019)*. IEEE, 2019, pp. 121–128.

[29] T. Shunin, N. Zubkova, and S. Shershakov, "Neural approach to the discovery problem in process mining," in *Proceedings of the 7th International Conference on Analysis of Images, Social Networks and Texts (AIST 2018)*, ser. Lecture Notes in Computer Science, vol. 11179. Springer, 2018, pp. 261–273.

[30] A. E. Marquez-Chamorro, M. Resinas, and A. Ruiz-Cortes, "Predictive monitoring of business processes: A survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 962–977, 2018.

[31] C. D. Francescomarino, C. Ghidini, F. M. Maggi, and F. Milani, "Predictive process monitoring methods: Which one suits me best?" in *Proceedings of the 16th International Conference on Business Process Management (BPM 2018)*, ser. Lecture Notes in Computer Science, vol. 11080. Springer, 2018, pp. 462–479.

[32] N. Harane and S. Rathi, "Comprehensive survey on deep learning approaches in predictive business process monitoring," in *Studies in Computational Intelligence*. Springer International Publishing, 2020, pp. 115–128.

[33] B. A. Tama and M. Comuzzi, "An empirical comparison of classification techniques for next event prediction using business process event logs," *Expert Systems with Applications*, vol. 129, pp. 233–245, 2019.

[34] N. Tax, I. Teinemaa, and S. J. van Zelst, "An interdisciplinary comparison of sequence modeling methods for next-element prediction," *Software and Systems Modeling*, 2020.

[35] 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 Transactions on Intelligent Systems and Technology*, vol. 10, no. 4, pp. 1–34, 2019.

[36] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 2, pp. 1–57, 2019.

[37] W. Kratsch, J. Manderscheid, M. Röglinger, and J. Seyfried, "Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction," *Business & Information Systems Engineering*, 2020.

[38] J. Evermann, J. Rehse, and P. Fettke, "A deep learning approach for predicting process behaviour at runtime," in *Proceedings of the 14th International Business Process Management Workshops (BPM 2016)*, ser. Lecture Notes in Business Information Processing, vol. 281, 2016, pp. 327–338.

[39] ——, "XES tensorflow - process prediction using the tensorflow deep-learning framework," in *Proceedings of the 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017)*, ser. CEUR Workshop Proceedings, vol. 1848. CEUR-WS.org, 2017, pp. 41–48.

[40] B. R. Gunnarsson, S. K. L. M. vanden Broucke, and J. D. Weerdt, "Predictive process monitoring in operational logistics: A case study in aviation," in *Proceedings of the 17th International Conference on Business Process Management Workshops (BPM 2019)*, ser. Lecture Notes in Business Information Processing, vol. 362. Springer, 2019, pp. 250–262.

[41] E. Tello-Leal, J. Roa, M. Rubiolo, and U. M. Ramirez-Alcocer, "Predicting activities in business processes with LSTM recurrent neural networks," in *Proceedings of the 2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU-K 2018)*. IEEE, 2018, pp. 1–7.

[42] N. Tax, "Human activity prediction in smart home environments with LSTM neural networks," in *Proceedings of the 14th International Conference on Intelligent Environments (IE 2018)*. IEEE, 2018, pp. 40–47.

[43] N. Mehdiyev, J. Evermann, and P. Fettke, "A multi-stage deep learning approach for business process event prediction," in *Proceedings of the 2017 IEEE 19th Conference on Business Informatics (CBI 2017)*. IEEE, 2017, pp. 119–128.

[44] N. Navarin, B. Vincenzi, M. Polato, and A. Sperduti, "LSTM networks for data-aware remaining time prediction of business process instances," in *Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI 2017)*. IEEE, 2017, pp. 1–7.

[45] A. Al-Jebrni, H. Cai, and L. Jiang, "Predicting the next process event using convolutional neural networks," in *Proceedings of the 2018 IEEE International Conference on Progress in Informatics and Computing (PIC 2018)*. IEEE, 2018, pp. 332–338.

[46] A. Khan, H. Le, K. Do, T. Tran, A. Ghose, H. Dam, and R. Sindhgatta, "Memory-augmented neural networks for predictive process analytics."

[47] A. Metzger and A. Neubauer, "Considering non-sequential control flows for process prediction with recurrent neural networks," in *Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2018)*. IEEE, 2018, pp. 268–272.

[48] S. Schönig, R. Jasinski, L. Ackermann, and S. Jablonski, "Deep learning process prediction with discrete and continuous data features," in *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*. SCITEPRESS - Science and Technology Publications, 2018, pp. 314–319.

[49] N. Mehdiyev, J. Evermann, and P. Fettke, "A novel business process prediction model using a deep learning method," *Business & Information Systems Engineering*, 2018.

[50] M. Camargo, M. Dumas, and O. G. Rojas, "Learning accurate LSTM models of business processes," in *Proceedings of the 17th International Conference on Business Process Management (BPM 2019)*, ser. Lecture Notes in Computer Science, vol. 11675. Springer, 2019, pp. 286–302.

[51] L. Lin, L. Wen, and J. Wang, "Mm-pred: A deep predictive model for multi-attribute event sequence," in *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM 2019)*. SIAM, 2019, pp. 118–126.

[52] M. Hinkka, T. Lehto, and K. Heljanko, "Exploiting event log event attributes in RNN based prediction," in *Proceedings of the 9th International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA 2019)*, ser. Lecture Notes in Business Information Processing, vol. 379. Springer, 2019, pp. 67–85.

[53] J. Theis and H. Darabi, "Decay replay mining to predict next process events," *IEEE Access*, vol. 7, pp. 119 787–119 803, 2019.

[54] N. A. Wahid, T. N. Adi, H. Bae, and Y. Choi, "Predictive business process monitoring – remaining time prediction using deep neural network with entity embedding," *Procedia Computer Science*, vol. 161, pp. 1080–1088, 2019.

[55] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Using convolutional neural networks for predictive process analytics," in *Proceedings of the 2019 International Conference on Process Mining (ICPM 2019)*. IEEE, 2019, pp. 129–136.

[56] J. Wang, D. Yu, C. Liu, and X. Sun, "Outcome-oriented predictive process monitoring with attention-based bidirectional LSTM neural networks," in *Proceedings of the 2019 IEEE International Conference on Web Services (ICWS 2019)*. IEEE, 2019, pp. 360–367.

[57] N. D. Mauro, A. Appice, and T. M. A. Basile, "Activity prediction of business process instances with inception CNN models," in *Proceedings of the 18th International Conference of the Italian Association*

for Artificial Intelligence (AIIA 2019), ser. Lecture Notes in Computer Science, vol. 11946. Springer, 2019, pp. 348–361.
[58] F. Folino, G. Folino, M. Guarascio, and L. Pontieri, "Learning effective neural nets for outcome prediction from partially labelled log data," in *Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI 2019)*. IEEE, 2019, pp. 1396–1400.
[59] M. Hinkka, T. Lehto, K. Heljanko, and A. Jung, "Classifying process instances using recurrent neural networks," in *Proceedings of the 16th International Conference on Business Process Management Workshops (BPM 2018)*, ser. Lecture Notes in Business Information Processing, vol. 342. Springer, 2018, pp. 313–324.
[60] P. Philipp, R. Jacob, S. Robert, and J. Beyerer, "Predictive analysis of business processes using neural networks with attention mechanism," in *Proceedings of the 2nd International Conference on Artificial Intelligence in Information and Communication (ICAIIC 2020)*. IEEE, 2020, pp. 225–230.
[61] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*. Association for Computational Linguistics, 2018, pp. 889–898.
[62] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net, 2020.
[63] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.
[64] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[65] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS '06)*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, pp. 1137—1144.
[66] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 3563–3593, 2014.
[67] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," *Complex Systems*, vol. 3, 1989.
[68] F. Gers, "Learning to forget: continual prediction with LSTM," in *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN '99)*. IEEE, 1999, pp. 850–855.
[69] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.
[70] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv e-prints*, 2014.
[71] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative Study of CNN and RNN for Natural Language Processing," *arXiv e-prints*, 2017.
[72] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*. Association for Computational Linguistics, 2015, pp. 1412–1421.
[73] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015.
[74] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," *arXiv e-prints*, 2014.
[75] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot Learning with Memory-Augmented Neural Networks," *arXiv e-prints*, 2016.
[76] A. G. et al., "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.
[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[78] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*. IEEE, 2015, pp. 1–9.
[79] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, T. Kobayashi, K. Hirose, and S. Nakamura, Eds. ISCA, 2010, pp. 1045–1048.
[80] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009)*. ACM Press, 2009, pp. 1113–1120.
[81] "Score normalization," in *Encyclopedia of Biometrics*. Springer US, 2009, pp. 1134–1135.
[82] W. J. Scheirer, A. Rocha, R. J. Micheals, and T. E. Boult, "Robust fusion: Extreme value theory for recognition score normalization," in *Proceedings of the 11th European Conference on Computer Vision (ECCV 2010)*, ser. Lecture Notes in Computer Science, vol. 6313. Springer, 2010, pp. 481–495.
[83] A. Leontjeva, R. Conforti, C. D. Francescomarino, M. Dumas, and F. M. Maggi, "Complex symbolic sequence encodings for predictive monitoring of business processes," in *Proceedings of the 13th International Conference on Business Process Management (BPM 2015)*, ser. Lecture Notes in Computer Science, vol. 9253. Springer, 2015, pp. 297–313.
[84] B. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.
[85] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, pp. 79–82, 2005.
[86] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, and A. Polyvyanyy, "Split miner: automated discovery of accurate and simple business process models from event logs," *Knowledge and Information Systems*, vol. 59, no. 2, pp. 251–284, 2018.
[87] A. Berti, S. J. van Zelst, and W. M. P. van der Aalst, "Process mining for python (pm4py): Bridging the gap between process- and data science," *CoRR*, vol. abs/1905.06169, 2019.
[88] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 1321–1330.

9 Biographies

EFRÉN RAMA-MANEIRO received the B.Eng. degree in computer engineering and the M.Sc. degree in Big Data from the University of Santiago de Compostela, Spain in 2018 and 2019 respectively. He is a Researcher and currently working toward the Ph.D. degree at the Centro Singular de Investigación en Tecnoloxías Intelixentes, University of Santiago de Compostela. His research interests include process mining and deep learning.

JUAN C. VIDAL received the B.Eng. degree in computer science from the University of La Coruña, La Coruña, Spain, in 2000, and the Ph.D. degree in artificial intelligence from the University of Santiago de Compostela (USC), Santiago de Compostela, Spain, in 2010, where he was an Assistant Professor with the Department of Electronics and Computer Science, from 2010 to 2017. He is currently an Associate Researcher with the Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), USC. His research interests include process mining, fuzzy logic, machine learning, and linguistic summarization.

MANUEL LAMA received the Ph.D. degree in physics from the University of Santiago de Compostela, in 2000, where he is currently an Associate Professor of artificial intelligence. He has collaborated on more than 30 projects and research contracts financed by public calls, participating as a principal investigator in 20 of them. These activities were implemented in areas, such as process discovery, predictive monitoring, and management of dynamic processes. As a result of this research, he has published over 150 scientific articles with review process in conference and national and international journals.