Journal of Information and Data Management, 2025, 16:1, doi: 10.5753/jidm.2025.4241

Forecasting Business Process Remaining Time Through Deep Learning Approaches

Ronildo Oliveira da Silva [**Universidade Federal do Ceará** | ***ronildo.oliveira@alu.ufc.br***]
Regis Pires Magalhães [**Universidade Federal do Ceará** | ***regismagalhaes@ufc.br***]
Lívia Almada Cruz [**Universidade Federal do Ceará** | ***livia.almada@ufc.br***]
Criston Pereira de Souza [**Universidade Federal do Ceará** | ***criston@ufc.br***]
Davi Romero de Vasconcelos [**Universidade Federal do Ceará** | ***daviromero@ufc.br***]
José Antônio Fernandes de Macedo [**Universidade Federal do Ceará** | ***jose.macedo@dc.ufc.br***]

Universidade Federal do Ceará, Campus Quixadá, Av. José de Freitas Queiroz, 5003, Cedro, Quixadá, CE, 63902-580, Brazil.

Received: 22 March 2024 • **Published:** 22 August 2025

Abstract Business process analysis is a part of process mining, which involves predictive monitoring. It seeks to predict individual processes, such as determining the next step to execute based on past events or estimating the remaining time until process completion. Such predictions can help to prevent waits, discover process bottlenecks, and assist alert systems. This paper aims to evaluate deep learning architectures to predict the time required to complete a business process instance. We have evaluated the models using three real datasets, including two widely used public ones. The experimental results show deep learning architectures that combined dense layers with a self-attention mechanism outperformed the current state-of-the-art, demonstrating superior performance regarding the mean absolute error metric in most of the datasets analyzed.

Keywords: Business process, Remaining time prediction, Deep learning

1 Introduction

A wide range of private institutions, as well as public service organizations and departments, aim to produce better and in less time or perform tasks as efficiently as possible [Kalenkova *et al*., 2017]. Due to these factors, business process optimization is an area of research that has become popular [Reijers, 2021] and commonly deals with real process monitoring problems.

Several leaders in digital product companies believe that optimizing business processes is crucial for the impact of digital transformation. Machine learning is a core technology that supports the development of new business models, being highly effective for predictive maintenance and enhancing operational efficiency, increasing productivity, and revealing previously unattainable insights [Akhramovich *et al*., 2024]. According to [Paschek *et al*., 2017], estimates for 2020 showed that machine learning for process optimization and automation would be a very promising tool. The relationship between process management and digital transformation is quite solid, and [Stjepić *et al*., 2020] highlights a series of business segments supported by these technologies.

Additionally, the digital transformation conducted by governments has improved the quality of public services delivered to the citizens. Usually, the enormous volume of information these services manage, associated with its variety and the speed with which this data is generated, classifies it as Big Data. Processing such data to guide public policies requires innovative forms of analysis, and its potential value lies precisely in the possibilities of developing preventive and corrective measures in "real-time" and providing accurate information for planning and improving public policies. One initiative in Brazil is the Chief Scientist Program (CSP)[1], conducted by the Government of the State of Ceará. The digital transformation proposed by CSP includes the Finance Secretariat of Ceará (SEFAZ-CE)[2], which specifically has records of electronic invoices, electronic tax coupons, digital tax bookkeeping, and others ones.

In the context of SEFAZ-CE, one goal is the predictive process monitoring. Process monitoring involves leveraging advanced techniques and creating new analytical tools to characterize the intrinsic correlations between diverse data types over time and space, predicatively detect emerging or synchronized behaviors, identify causal relationships, and determine interdependent resilience measures.

Efficiently monitoring business processes requires the ability to anticipate any potential bottlenecks [Castro *et al*., 2022], cyclical dependencies, long waits, and failures [Mello *et al*., 2019, 2020]. Predicting the remaining time for process completion mitigates potential issues and optimizes decision-making systems.

This research aims to evaluate deep learning architectures to predict the time required to complete a business process instance. Based on the current purposes of improving process management, this work addresses the following Research Questions (RQ):

RQ 1 Which machine learning architectures are best suited for accurately predicting the remaining time to complete

[1] https://www.funcap.ce.gov.br/cientista-chefe-descricao-dos-programas/

[2] https://www.sefaz.ce.gov.br/

a business process?

RQ 2 What is the influence of the process stage on predictions of the remaining time to complete a business process?

RQ 3 What are the differences between the best model proposed in this work and those proposed in related works?

RQ 4 How is the model accuracy influenced by the distribution of the remaining time needed to complete the processes?

This research proposes methods for accurately predicting the time remaining to finalize a business process. We explore a range of deep learning architectures to achieve this goal. Our experimentation involves thorough evaluation using five real datasets: two widely recognized public datasets and three private datasets sourced from SEFAZ-CE. The results reveal that our approach outperforms the current state-of-the-art, demonstrating superior performance in terms of mean absolute error. This paper extends our previous work [da Silva *et al*., 2023] by conducting an experimental evaluation using three real huge datasets acquired from SEFAZ-CE, each one contains a set of processes with different variability of time. We have also added three deep learning architectures in our experiments (Section 4.2).

2 Problem definition

Definition 2.1 (Business Process)**.** A business process refers to a sequence of activities performed in a predefined order to produce a product or service.

Definition 2.2 (Business Process Instance)**.** A business process instance is an occurrence of a business process given by a sequence of activities $p = [e_1, e_2, \ldots, e_n]$, where each activity e_i is associated with a unique identifier and a timestamp of its execution. The terms activity and event interchangeably are used in this article.

Definition 2.3 (Prefix of a Business Process Instance)**.** Given a business process instance $p = [e_1, e_2, \ldots, e_n]$, a prefix of p is any subsequence $p^{'} = [e_1, e_2, \ldots, e_k]$ of size k where $1 \leq k \leq n$.

A prefix represents the sequence of activities that have been observed up to a certain point in time for a specific process instance. To ensure efficiency and accuracy, a business process should be defined with clarity, organization, and consideration for the dependencies between stages, the representation of the real model, and all possible prefixes.

Applications record the steps performed in a business process instance in an event log. An example of a process event log is presented in Table 1. The process instance is represented by a unique identifier (*Process ID*). Each row in the log corresponds to an activity that has started at a specific timestamp (*Creation time*) and is associated with an activity (*Activity*), such as "Get ticket", "Resolve request", etc. The process instance is completed when it reaches the activity *"Close"*. For example, in Table 1, the activities *"Close"*, on the last lines of process *ID 1* and *2*, respectively, specify that the processes instances *ID 1* and *ID 2* have finished.

Table 1. Structure of an event log.

Row	Process ID	Activity	Creation time
1	1	Set priority	2012-10-09 14:50:17
2	1	Get the ticket	2012-10-09 14:51:01
3	1	Get the ticket	2012-10-12 15:02:56
4	1	Resolve request	2012-10-25 11:54:26
5	1	Close	2012-11-09 12:54:39
6	2	Set priority	2012-04-03 08:55:38
7	2	Get the ticket	2012-04-03 08:55:53
8	2	Resolve request	2012-04-05 09:15:52
9	2	Close	2012-05-19 09:00:28

Definition 2.4 (Remaining Time Prediction)**.** Given a set of completed business process instances *BP*, and a business process instance p, the problem is to learn a function able to estimate the time required for p to reach its finish (*"Close"* activity) based on the log of the previous executed activities.

To exemplify, consider the process instance whose *process ID* is 1 (Table 1). Suppose the last activity registered was *"Set priority"*. The model must predict the elapsed time between "2012-10-09 14:50:17", the creation time for *"Set priority"* and "2012-11-09 12:54:39", the creation time for *"Close"*.

3 Related Work

Some works in the literature propose solutions to the remaining time prediction problem or similar problems in business processes.

The work [Tax *et al*., 2017] investigates an approach based on *LSTM* to build predictive models related to process monitoring. The proposed model consists of two *LSTM* layers with normalization layers (Batch Normalization) in between. It has two outputs - one for predicting the time remaining for completion and the other for predicting the next activity. The output layer aimed at predicting the next activity refers to a classification problem for the study of predictive process monitoring. Each process instance type was represented as a 2-byte character, thus reducing the amount of memory allocated in the solution. The *Helpdesk 17* dataset was used for model training and evaluation.

The model developed by [Navarin *et al*., 2017] explores information such as start time, end time, and day of the week on which the activity occurred, in addition to representing each activity with *One-hot Encoding*, to produce a prediction of the remaining time for completion of running process instances. Similar to [Tax *et al*., 2017] work, [Navarin *et al*., 2017] model also uses *LSTM* as its core architecture and the *Helpdesk17* dataset. The model architecture proposed in that work is made up of *l* layers composed of an *LSTM* layer, *n*, as well as the number of neurons for each defined layer and the *NAdam* optimizer [3] followed by an output layer, where *l* and *n* are network construction parameters.

[Venkateswaran *et al*., 2021] predicted the remaining time to complete a business process based on a deep learning model composed of two stacked *LSTM* layers and a dense output layer. Process models in real-world settings may go through changes over time. For instance, processes may become longer or shorter, have fewer or more steps, or have different priorities. Additionally, logs utilized to train the

[3] `https://keras.io/api/optimizers/Nadam/`

models may be linked with various versions of modified processes. The proposed solution addresses these issues by employing invariant attributes that have a strong correlation with the predicted value. In this way, the objective is to alleviate distortions in the model when the distribution of data changes, ensuring more accurate predictions over time. Attributes are represented as vectors of embeddings, which performs well when considering the order of the data. The solution was evaluated with the *Helpdesk 17* and *BPI 12W* dataset, a subset of *BPI 12* limited to event logs that contain manually executed events. [Park and Song, 2020] developed a method to predict the future performance of business processes to support proactive actions for process improvement. The proposed model is hybrid and based on *CNN* and *LSTM*, called *Long-term Recurrent Convolutional Networks* (*LRCN*), which performs a combination of extracting attributes from matrices in temporal sequences. Attributes are represented as matrices that contain information about business process performance. The datasets used are mainly from *BPI* and *Helpdesk 17* competitions. [Venugopal *et al*., 2021] presented a solution to the problem of predicting the next instant of time based on graph neural networks (*GNN*). Similar to the [Tax *et al*., 2017] proposal, this work also contributes to experiments predicting the type of upcoming activity and the next instant of time. The matrix representation of the graph is given by vertices as states (e) (Activities) and transition functions (t) from that state to another under a time window, as edges (values $A_{e,t}$ of a matrix A) of the graph. Internally, the hidden layers deal with matrix operations involving vectors representing the number of distinct activities multiplied by the adjacency matrices representing the business model. The presented neural network is simple, with two dense layers and a *ReLU* activation layer between them. A linear activation function with the *Adam* optimizer is also used. The solution was evaluated with the *Helpdesk 17* and the *BPI 12W* datasets, as well as elaborated in the [Park and Song, 2020] work. The Table 2 organizes the characteristics of each related work and the proposed work.

4 Data and Methods

This section presents the datasets used in this work and the pre-processing and transformations performed on them. It also deals with the proposed deep learning architectures.

4.1 Datasets

This work uses two open datasets widely known in the literature, *BPI 12*[4] and *Helpdesk 17*[5], and three proprietary datasets from the Treasury Office of the State of Ceará (SEFAZ-CE, Brazil). These datasets are described below.

BPI 12: The *Business Process Intelligence 2012 (BPI 12)* dataset [van Dongen, 2012] was originally made available at the International Conference on Business Process Management (BPM) in 2012. The dataset is a log of events from a Dutch financial institution between January 10th, 2011, and March 4th, 2012. The log concerns personal loan application processes and contains $156{,}424$ events in $13{,}087$ business process instances. Each instance has $11{,}95$ steps on average.

Helpdesk 17: The *Helpdesk 17* dataset [Polato, 2017] comprises event logs derived from a ticket management system specifically designed to support the helpdesk of an Italian software company. It provides records of business processes from January 13th, 2010, to January 3rd, 2014. The log contains $4{,}580$ process instances with $21{,}340$ events. Each process instance has $4{,}45$ steps on average.

SEFAZ-CE: The dataset is not publicly available, and it comprises business processes performed between April 1st, 2015 to December 9th, 2021. In total, there are $1{,}201{,}347$ different processes and more than 7 million ($7{,}719{,}966$) process events distributed across 356 different areas. From these varied data, we choose three subsets of processes, each one related to a different topic and variability of duration. This dataset was anonymized, and we did not have access to the specific content of the subjects. We estimate the mean and the standard deviation of the remaining time for each distinct subject and the global set of processes. Regarding these metrics, we point out that the lower the standard deviation, the less variability in duration the set of processes has. We also expect that more variability increases the difficulty in predicting the remaining time for the processes of a specific subject. Three subjects (S_25, S_50, and S_75) have been selected to serve as experimental material based on the following requirements:

- The processes of *S_25* dataset, with $38{,}738$ instances, represent the subject with the set of processes having the standard deviation closest to the first quartile regarding the global standard deviation (≈ 34 days). This subset involves processes between April 1st, 2020, and December 9th, 2021;
- The processes of *S_50* dataset, with $18{,}416$ instances is related to taxes, which date between October 8th, 2020, to December 9th, 2021. *S_50* represents the subject with the set of processes having the standard deviation closest to the median regarding the global standard deviation (≈ 55 days);
- The processes of the *S_75* dataset, with $25{,}685$ instances, represent the subject with the set of processes having the standard deviation closest to the third quartile regarding the global standard deviation ($\approx$ 100 days). These processes deal with the taxation regime captured between April 8th, 2015, and September 28th, 2021.

Each process instance is represented by multiple activities in the datasets. The variety and quantity of events in a process is an important factor that reflects the input dimension for each of the architectures presented in this work. Table 3 contains information about the size of the process instance with the largest number of steps found in each dataset. The *S_75* and *BPI 12* datasets have at least one process with many steps. The largest processes of *S_25* and *S_50* datasets are similar in size. The *Helpdesk 17* dataset is, in general, made up of processes with a few steps.

[4]https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

[5]https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

Table 2. Comparison of related works.

Work	Architecture type	Encoding actitivy	Dataset	Problem type
[Tax *et al*., 2017]	LSTM	Unicode	Helpdesk 17 BPI 12 W	Prediction of remaining time for completion Prediction of next activity
[Navarin *et al*., 2017]	LSTM	One Hot Encoding	Helpdesk 17 BPI 12	Prediction of remaining time to completion
[Park and Song, 2020]	LRCN	Matrix Representation	Helpdesk 17 BPI 12, BPI 13	Prediction of remaining time for completion Prediction of next activity
[Bukhsh *et al*., 2021]	Transformer	Token and Positional Embedding	Helpdesk 17	Remaining time prediction for completion Next activity prediction
[Venkateswaran *et al*., 2021]	LSTM	Categorical	Helpdesk 17 BPI 13, BPI 15, BPI 18, BPI 19	Prediction of the next instant of time Prediction of the next activity
[Venugopal *et al*., 2021]	GNN	Matrix Representation	Helpdesk 17 BPI 12 W	Prediction of the next instant of time Prediction of the type of the next activity
This work	BiLSTM Self-attention Transformer	Tokens	Helpdesk 17 BPI 12 S_25, S_50, S_75	Prediction of remaining time to completion

Table 3. Size of the largest process instance found in the datasets *BPI 12*, *Helpdesk 17*, S_25, S_50 and S_75.

Dataset	Size of largest process instance
BPI 12	55
Helpdesk 17	10
S_25	24
S_50	22
S_75	69

Finally, Table 4 organize the average remaining time for completion and its standard deviation for each of the subsets defined for the experiment. It is worth mentioning that *BPI 12* and *Helpdesk 17* are datasets with processes that involve manual and automatic tasks, while the tasks in the SEFAZ-CE dataset are manual, depending on human interaction to advance to the next task.

Data Pre-processing

The data partitioning into training, validation, and test sets was applied to respect the chronological order of the events. As a result, the training set includes events occurring before the ones in the validation set, and the validation set includes events occurring before the ones in the test set [Lakshmanan *et al*., 2020]. Details regarding the partitioning and distribution of process instances and their events in each set are organized in Table 5. To guarantee the effectiveness of our training, we used only those processes that have been completed entirely, i.e., processes in which the last step is the "Close" activity. After this filter, the *BPI 12* and *Helpdesk 17* datasets have 13,087 and 4,557 process instances, respectively. Eventually, the average number of steps per process changes, resulting in 3.65 in the *Helpdesk 17* dataset and remaining the same in *BPI 12*, as this dataset is formed only by processes that have reached completion. The other datasets, S_25, S_50, and S_75, have respectively 6,140, 2,812, and 1,400 process instances with 38,738, 18,416, and 25,685 events.

At this stage, the labels that represent activities were tokenized, in which a unique numerical coding was assigned to each type of activity. Furthermore, an indicative value is given for the stage in which the event is. A step takes on a value between $1...n$, where n is the maximum number of steps the process has (*Step*). Three attributes were created, all representing time in days derived from the *Instant of creation* of the event: the duration (*Duration*) of the event, the execution time of the process until the current event (*Elapsed Time*) and the time remaining for the process to complete (*Remaining time to complete*). The latter represents the label for this experiment's training, validation, and test sets. Below, some items regarding data partitioning are listed:

1. The composition of the training, validation, and testing sets follows the proportion 60%, 20%, and 20%, respectively;
2. Each process instance is represented by several samples (events) in the datasets;
3. The process instances are complete. There is no part of the same process in two or more sets;
4. Events whose *"Remaining time to complete"* column has a zero value are removed from the training, validation, and test sets, as there is no need to predict the remaining time in the last step. This is the case of the lines corresponding to steps 4 and 5 in the example in Table 6.

Another transformation applied consists of joining the same activity consecutively repeated in the same process instance, collapsing them into a unique activity, and adjusting timestamps to accumulate the values of the collapsed samples. This step allows us to summarize the data, minimize its complexity, and reduce the number of process steps. Furthermore, we believe that merging consecutive steps with identical activities enhances the recurrent model's ability to learn activity transitions effectively. Table 6 presents samples from the *Helpdesk 17* dataset. The *BP1 12* dataset has the same structure but different values. The sequence repetition of activities in steps (*Step*) 4 and 5 of the process instance of *ID 1* (Table 1) allows them to be collapsed and adjusted the data from the columns *Duration*, *Elapsed Time* and *Remaining time to complete* with the sum of the values of the collapsed rows. Furthermore, this sequence of activities (lines 1 to 3 of Table 6) can be represented as a sequence of prefixes in matrix form (Table 7) where the number of steps in the largest process is 6 (the *BP1 12* dataset has the same structure with *padding* equal to 55).

In the example above (Table 7), values equal to 0 are used as paddings (*paddings*), which makes it possible to standardize the number of columns in the representation of the ac-

Table 4. Average and standard deviation (in days) of the *labels* of the training, validation and test sets of *BPI 12*, *Helpdesk 17*, S_25, S_50 and S_75.

Average Time Remaining to completion (± Standard deviation)					
Dataset	**BPI 12**	**Helpdesk 17**	**S_25**	**S_50**	**S_75**
Training	11.0929 (± 13.0282)	36.1666 (± 11.1193)	23.8269 (± 42.3115)	48.6381 (± 70.2296)	57.1359 (± 109.5282)
Validation	11.2576 (± 11.2576)	37.7909 (± 9.9398)	16.2409 (± 23.2113)	19.7326 (± 25.0358)	71.3337 (± 89.6194)
Test	8.4947 (± 8.5743)	29.1493 (± 12.5011)	17.0729 (± 24.9786)	20.3092 (± 25.3953)	71.2313 (± 89.6764)

Table 5. Number of process instances (Pro. Ins.) and events related to the partitioning of datasets.

Dataset		**Train**	**Validation**	**Test**	**Total**
BPI 12	ProT. Ins.	7,852	2,618	2,617	13,087
	Events	85,963	26,865	30,509	143,337
Helpdesk 17	Pro. Ins.	2,745	896	916	4,557
	Events	10,254	3,417	3,004	16,675
S_25	Pro. Ins.	3,684	1,228	1,228	6,140
	Events	23,180	7,845	7,713	38,738
S_50	Pro. Ins.	1,687	562	563	2,812
	Events	10,784	3,744	3,888	18,416
S_75	Pro. Ins.	840	280	280	1,400
	Events	14,495	5,688	5,502	25,685

tivities of the instances of processes. This resulting matrix replaces the *Activity* column (Table 6).

4.2 Proposed architectures

The architectures proposed (Figure 1) are structured in the following way. Embedding layers codify the activities into a vector with reduced dimension. A *BiLSTM* layer supports the temporal and sequential data given by the business process. Attention layers identify the importance of certain stages of the process, similar way to the work [Wang *et al*., 2019]. The different architectures were obtained by combining one or more of these layers. The SA_BiLSTM model (Figure 1-*A*) is a sequential model where the input passes through an embedding layer and is then processed by a self-attention layer (Self-Attention). Then, the output of this layer is concatenated to the input temporal data. The result is a new entry for each *BiLSTM* layer concatenated with the temporal data layer. To evaluate the architecture, the models were obtained from variations of the SA_BiLSTM model. The variations are mainly based on removing some layers or changes in the order in which the attention and *BiLSTM* layers are arranged. Furthermore, experiments are carried out with Transformer Encoding, a generic model for ingesting sequences and manipulating representations through an encoder and decoder. All models described in Figure 1 receive as input the sequences of pairs formed by the prefixes from the previous process steps (*Prefix*) and the temporal data (*Duration*, *Elapsed Time* and *Step*), as shown in Table 6. 'NAdam' optimizer was used for all proposed architectures and the network parameters were: *learning_rate=0.001*; *kernel_regularizer=l2(0.01)*; *loss='mae'*; *dropout=0.1*; *activation='relu'*; *activation='linear'* (output). The following models were evaluated:

A) SA_BiLSTM. The input corresponding to *Prefix* is sent to an embedding layer, which converts the data representing the activities into vectors. These vectors are sent to the self-attention layer, which handles different parts of the initial input sequence. Then, the attention layer is connected to a *BiLSTM* layer, mainly used to process data sequences in both directions and capture long-term dependencies. This layer can provide additional context and result in more complete learning from the data. The *Duration*, *Elapsed Time*, and *Step* correspond to the temporal data of each stage in the process execution. This, in turn, is concatenated to the *BiLSTM* layer, which is then linked to a dense layer. The dense layer finally outputs the predicted value.

B) SA_DENSE. This architecture does not include a *BiLSTM* layer. Its organization analyzes the behavior of the neural network without interactions between layers in the backward and forward directions but maintaining the influence of the self-attention layer, which, in turn, compares all elements (process steps) of the input *Prefix* and modifies the corresponding sequence positions of the output. Then, the result of the self-attention layer is concatenated with temporal inputs to pass by the dense layer.

C) DENSE. Following the idea of reducing the complexity of the model, this architecture is one of the simplest presented in this section. In addition to the absence of *BiLSTM*, the attention layer is also not used here. This model resembles classic machine learning models.

D) BiLSTM_SA. This version is a variation of architecture A, where the attention layer comes after the BiLSTM layer. The proposal here is to perform the attention layer processing after BiLSTM learning.

E) BiLSTM. Similarly to architecture C, the attention layer is removed, leaving this architecture in a slightly simpler RNN configuration, as presented in [Tax *et al*., 2017] and [Navarin *et al*., 2017] however, with an architecture that works in both directions and with the concatenation of temporal data with those related to activities.

F) TRANSFORMER_ENCODER. The second-simplest architecture of this experiment, concerning layers, is configured like the C architecture and uses a transformation and encoding layer followed by the output instead of an embedding layer.

G) LSTM Tax *et al.* [2017]. The LSTM model proposed in the work has a few layers, which simultaneously solve regression and classification problems.

H) LSTM Navarin *et al.* [2017]. Also, using an LSTM with few layers solves regression problems, such as predicting the remaining time for completion of a process instance.

I) TKN_TRANSFORMER. [Bukhsh *et al*., 2021] The Tokenizer Transformer uses tokenization, prefixes, and transformers to solve the same problem presented in this work.

4.2.1 Assessment methodology

The evaluation is based on MAE (*Mean Absolute Error*), calculated by the sum of absolute errors divided by the sample size (n), where y_i is the observed value and $\hat{y}_i$ is the predicted

Table 6. Dataset samples *Helpdesk 17*.

Row	ID	Activity	Step	Duration	Elapsed Time	Remaining time to complete
1	1	1	1	0.0000	0.0000	31.0087
2	1	12	2	16.0084	0.0000	15.0003
3	1	9	3	0.0001	15.0002	15.0002
4	1	2	4	15.0002	15.0003	0.0000
5	1	2	5	0.0000	31.0087	0.0000
6	2	1	1	0.0000	0.0000	30.9822
7	2	12	2	5.8750	25.1053	25.1072
8	2	9	3	0.0019	25.1072	25.1053
9	2	2	4	25.1053	30.9822	0.0000

Table 7. Prefixes in the sample dataset *Helpdesk 17* without the last instance data.

Row	Prefix and Padding					Temporal Features			Label
	1	2	3	4	5	Duration	Elapsed Time	Step	Remaining Time to Complete
1	**1**	0	0	0	0	0.0000	0.0000	1	31.0087
2	**1**	**12**	0	0	0	16.0084	0.0000	2	15.0003
3	**1**	**12**	**9**	0	0	0.0001	15.0002	3	15.0003
4	**1**	**12**	**9**	**2**	0	15.0002	15.0003	4	0.0000

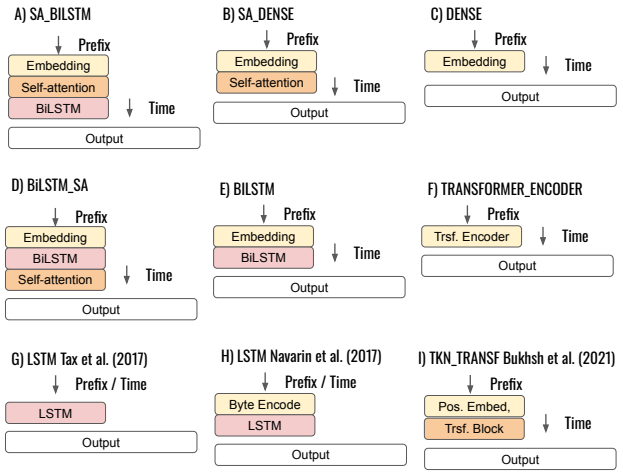

Figure 1. Deep neural network architectures used.

value:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \quad (1)$$

It is also possible to consider evaluation by process stage, that is, how well the learning model can predict using only activities at a certain stage as a test set.

5 Experiments and Results

This section discusses the results of experiments with the proposed predictive learning models. The following strategies were used as baseline solutions: Dummy Regressor, which returns the average value of the expected values in the training set without observing the input values; Regression with Random Forest, Linear Regression, Regression with XGBoost [Chen and Guestrin, 2016] and LightGBM [Ke *et al*., 2017]. The proposed solution and its variations were also compared to state-of-the-art deep learning solutions for predicting the remaining time of processes, two solutions based on LSTM [Navarin *et al*., 2017; Tax *et al*., 2017], and one solution based on Transformers [Bukhsh *et al*., 2021]. The metrics presented for these works are obtained from running new experiments using partitioning and filtering that are different from those used in the original works. Therefore, to enable a fairer comparison between all approaches, we aimed for the same filtering and partitioning. Table 8 presents the value of MAE metric and the respective 95% confidence interval for all datasets.

5.1 Discussion of results

The following list discusses the features and results of each architecture presented in this work.

5.1.1 SA_BiLSTM

By comparing this scenario with BiLSTM_SA, we can observe that the order in which the attention layer is configured may not impact the result. For the *BPI 12* dataset, its MAE is 1.44 lower than the best baseline (LightGBM). For the *Helpdesk 17* dataset, the MAE is 0.29 greater than the best baseline (LightGBM) and the third best among the proposed architectures, indicating this model may not be too good for smaller processes. *SA_BiLSTM* also reached the best result for *S_25* dataset, which has longer sequences than *Helpdesk 17*. The *S_50* has an average of events per process similar to the *S_25* dataset and configure the second-best result with this architecture. The *SA_BiLSTM* architecture is still more effective for processes that involve more steps. We attribute these results to using a recurrent layer supporting larger sequences. Finally, the SA_BiLSTM architecture using the *S_75* dataset did not produce very promising results. *S_75*, in addition to having the highest average number of events per process, the standard deviations that make up the training, validation, and test sets are also very high (Table 4), which may difficult the prediction.

5.1.2 DENSE

The *DENSE* architecture is the simplest one, based only on dense layers. The results indicate this architecture is the worst among the deep learning models presented in this work. The absence of attention or recurrent layers may have compromised achieving a promising result. This result is expected, given the nature of sequential data and the impor-

Table 8. Model performance using the MAE metric. * Confidence interval below and without intersection. ** It was not possible to carry out the complete experiment. The best results are in bold text.

Model	BPI 12	Helpdesk 17	S_25	S_50	S_75
	MAE (95% C.I.)*	**MAE (95% C.I.)***	**MAE (95% C.I.)***	**MAE (95% C.I.)***	**MAE (95% C.I.)***
Dummy Regressor	7.64 (7.58 - 7.69)	12.06 (11.77 - 12.32)	19.06 (18.65 - 19.47)	29.49 (29.00 - 29.98)	61.81 (60.02 - 63.59)
Linear Regression	6.03 (5.98 - 6.08)	12.02 (11.48 - 12.67)	11.89 (11.57 - 12.21)	20.49 (19.92 - 21.05)	50.88 (49.17 - 52.59)
Random Forest	4.89 (4.82 - 4.96)	**5.52 (5.34 - 5.69)**	10.37 (10.02 - 10.72)	17.30 (16.57 - 18.03)	**44.89 (43.13 - 46.65)**
XGBoost	5.02 (4.95 - 5.09)	5.73 (5.55 - 5.90)	10.59 (10.23 - 10.95)	18.12 (17.38 - 18.87)	**44.85 (43.09 - 46.62)**
LightGBM	4.84 (4.78 - 4.90)	**5.41 (5.26 - 5.57)**	11.32 (10.96 - 11.68)	17.97 (17.33 - 18.61)	48.88 (47.15 - 50.61)
LSTM Tax et al. (2017)	N.A.**	6.17 (5.50 - 6.33)	N.A.**	N.A.**	N.A.**
LSTM Navarin et al. (2017)	7.25 (7.17 - 7.34)	8.46 (8.30 - 8.62)	N.A.***	N.A.***	N.A.***
TKN_TRANSF Bukhsh et al. (2021)	4.91 (4.88 - 4.94)	5.69 (5.53 - 5.86)	9.45 (9.14 - 9.75)	**10.92 (10.50 - 11.35)**	47.30 (45.56 - 49.05)
SA_BiLSTM	**3.80 (3.74 - 3.87)**	5.70 (5.50 - 5.89)	**8.89 (8.53 - 9.25)**	12.83 (12.23 - 13.42)	52.08 (50.11 - 54.04)
DENSE	4.25 (4.18 - 4.31)	7.58 (7.42 - 7.75)	10.06 (9.71 - 10.42)	14.53 (13.94 - 15.13)	49.88 (47.92 - 51.85)
BiLSTM_SA	**3.72 (3.65 - 3.79)**	5.89 (5.70 - 6.07)	**8.91 (8.55 - 9.26)**	12.67 (12.07 - 13.26)	51.58 (49.62 - 53.53)
BiLSTM	**3.72 (3.65 - 3.79)**	**5.58 (5.39 - 5.77)**	**8.48 (8.12 - 8.84)**	12.85 (12.23 - 13.47)	**46.15 (44.32 - 47.97)**
SA_DENSE	**3.71 (3.65 - 3.78)**	**5.22 (5.04 - 5.40)**	**8.98 (8.60 - 9.35)**	13.76 (13.11 - 14.40)	**47.45 (45.51 - 49.38)**
TRANSF_ENCODER	4.24 (4.18 - 4.31)	6.64 (6.48 - 6.80)	9.91 (9.56 - 10.27)	14.49 (13.92 - 15.07)	51.17 (49.18 - 53.15)

tance of specific stages of process execution, which are not prioritized in this experiment.

5.1.3 BiLSTM_SA

This architecture is a variation of the *SA_BiLSTM* architecture and has reached similar results. For the *BPI 12*, S_25, and S_75 datasets, the MAE obtained has a confidence interval overlapping the best results presented in Table 8. In such cases,*SA_BiLSTM* was better than almost all baselines and architectures in the related works. This architecture has reached the best result among deep learning models for the *S_50* dataset. Once again, the use of the BiLSTM layer was shown to help the models to be more assertive in predicting the remaining time. Concerning *S_75* and *Helpdesk 17* datasets, this architecture produces a *MAE* close to the median compared to other architectures. With larger sequences in the log of events and larger distributions of activities, the average absolute error was slightly smaller concerning the baselines presented. On the other hand, it is not the best in predicting smaller sequences with a smaller distribution of activities (*Helpdesk 17*). The number of instances and the maximum size of activities per process execution may influence the combination of self-attention layers with *BiLSTM*.

5.1.4 BiLSTM

Recurrent models are widely used in solving problems involving data sequences to capture temporal dependencies, such as those used in this work and related work. The BiLSTM architecture achieved the best results in all compared datasets, except for the *S_50* dataset, being the most stable architecture to predict the remaining time in our experiments.

5.1.5 SA_DENSE

This architecture achieved the best MAE for both *BPI 12* and *Helpdesk 17* datasets, and confidence intervals comparable to the ones of *BiLSTM* for the other datasets, except to the *S_50*.

5.1.6 TRANSFORMER_ENCODER

This model is similar to the *DENSE* architecture's organization described in Figure 1. However, it uses a Transformer Encoding layer instead of an embedding layer to obtain vector representations of the prefix input. Among all the proposed architectures, the *TRANSFORMER_ENCODER* only outperformed the *DENSE*. Except for *S_75* dataset, where the dense architecture (*DENSE*) is slightly more efficient than *TRANSFORMER_ENCODER*. We hypothesize this is due to the fact that neither architecture has layers to support sequences.

5.1.7 LSTM [Tax *et al.*, 2017]

This architecture requires the process instances in train, validation, and test partitions to have a maximum size given by the maximum number of activities of all process samples. The modeling of this architecture does not allow the complete execution of the experiment when there are disparities in maximum sizes among the mentioned subsets. Consequently, it was not well-suited for the experiments proposed in this work.

5.1.8 LSTM [Navarin *et al.*, 2017]

The architecture proposed by Navarin *et al*. [2017] requires more computational resources to deal with long sequences or sets of processes with many distinct activities, as in the case of the *S_25*, *S_50*, and *S_75* datasets. For those datasets, the available computing environment could not complete the experiments. It was possible to evaluate the case that contains smaller processes and few distinct activities, such as the *BPI 12* and *Helpdesk 17* datasets. In those cases, the model does not outperform baselines such as Random Forest, XGBoost, and LightGBM.

5.1.9 TKN_TRANSF [Bukhsh *et al.*, 2021]

This architecture does not require recurrent layers. In addition, it uses transformers to tokenize activities organized by prefixes and convert temporal data to integers. It produces

the best result when the target is the *S_50* dataset. It demonstrates to be competitive according to the results for *BPI 12* and *Helpdesk 17* datasets, but it did not perform as well on the other datasets.

5.2 Discussion guided by research questions

This section aims to answer the research questions of the work based on the analyses and conclusions about the experiments carried out.

RQ1 - What are the best-suited machine learning models or architectures for accurately predicting the remaining time to complete a business process?

Predicting the remaining time to complete a business process is challenging, mainly due to the variability in executing process instances. For example, process activities may change over time regarding efficiency, name, and even existence, as some activities may be extinguished. These characteristics affect the difference between the sets since the partitioning is done following the temporal correspondence. The more diverse the set, the more difficult it will be to have uniform partitioning, which also impacts the learning model. From the results, we observed that using the self-attention mechanism may improve the result of *DENSE* architecture when applied directly to embedding vectors of activities. Furthermore, there was no improvement in the use of *BiLSTM* combined with the attention layer. Finally, in general, *BiLSTM* and *SA_DENSE* architectures were similar or more efficient than the baselines and the other proposed models. *BiLSTM* and *SA_DENSE* were also better than the solutions proposed by the works Tax *et al*. [2017], Navarin *et al*. [2017] and Bukhsh *et al*. [2021], except for the *S_50* dataset which the model of Bukhsh *et al*. [2021] presents a lower MEA. Finally, the *XGBoost* and *LightGBM* obtained the lower MAE of the *S_75* dataset, but their confidence interval have considerable overlapping with *BiLSTM* and *SA_DENSE*.

RQ2 - What is the influence of the process stage on predictions of the remaining time to complete a business process?

We want to understand if is easy to predict the remaining time for processes in the final stages. Still, it should be harder to predict the conclusion time for the processes in the initial stages. To answer the RQ2 question, the stages of the processes are varied from 1 to n to serve as input to the model, then we obtain the MAE considering the set of processes in each stage. In this experiment, we consider only the models that reached the best MAE in the previous experiment for each dataset.

Figures 2, 3, 4, 5, 6 and 7 present the *MAE* obtained for each stage of the process from the datasets *BPI 12*, *Helpdesk 17*, *S_25*, *S_50* and *S_75* , respectively.

For BPI 12 (Figure 2) and S_75 (Figure 7) datasets, the behavior of the trend line observed consists of MAE decreasing up to the final stages with some minor variations, which should indicate some correlation between stage and the capability to model to predict the remaining time. It is important to note that the processes comprising these datasets are longer sequences with many stages, which also may facilitate visually delineating this trend. However, the same trend is not so well observed for the datasets with smaller processes.

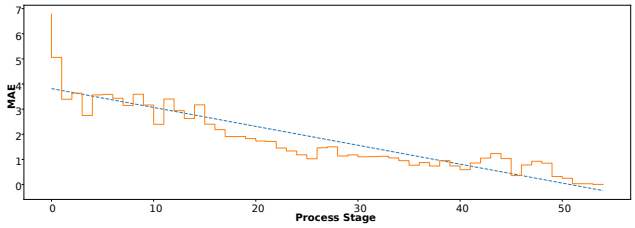

Figure 2. *BPI 12* dataset - MAE by process stage (*SA_DENSE*).

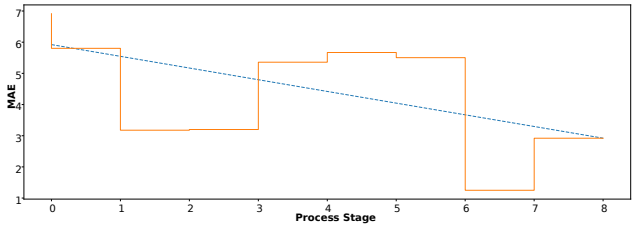

Figure 3. Helpdesk 17 dataset - MAE by process stage (SA_DENSE).

For *Helpdesk 17* dataset (Figure 3), this experiment was run with *SA_DENSE* model. The results present a relatively higher MAE value in the middle stages and slightly better MAE values in subsequent (final) stages.

For the S_25 dataset (Figure 4), which the best model was the *BiLSTM*, similarly, the trend line result per stage also decreases when the stage increases, and the MAE values decrease in the initial stages and increase in the medium-sized stages. However, the MAE has a peak in the final stage, which means it is harder to predict the remaining time when these processes are close to the end. This result may indicate more delays in the final steps of process execution.

For *S_50* dataset (Figure 5), the best results were obtained using *TKN_TRANSF* [Bukhsh *et al*., 2021] model. The trend line obtained by the model is non-decreasing. We can also observe a decrease of MAE up to the middle of stages, followed by a peak in the *MAE* value in stages 12 to 15 — an abrupt increase. In the last stages, the value of *MAE* decreases again.

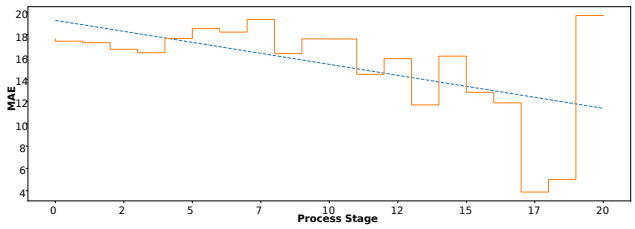

Figure 4. S_25 dataset - MAE by process stage (BiLSTM).

On the other hand, *BiLSTM_SA* architecture obtained the second-best result for the dataset *S_50*. We observed its behavior in Figure 6. Unlike the Bukhsh *et al*. [2021] model, this trend line decreases when the stage increases. However, similar to that model, there is a peak in the final stages.

Figure 7 shows the same analysis for the *S_75* dataset. XGBoost obtained the lower MAE metric value. The be-

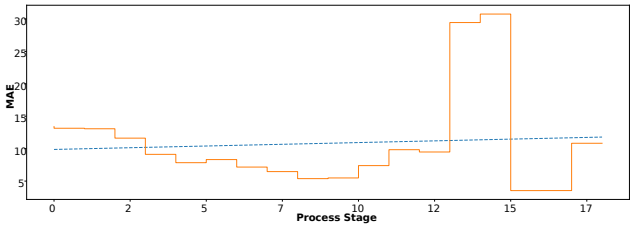

Figure 5. S_50 dataset - MAE by process stage (TKN_TRANSF [Bukhsh *et al.*, 2021]).

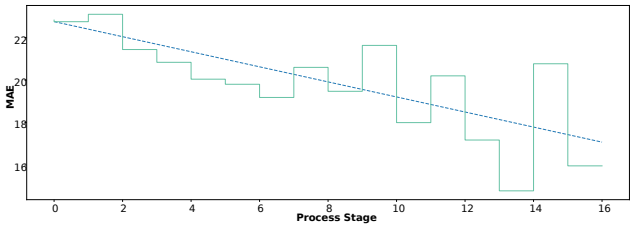

Figure 6. S_50 dataset - MAE by process stage (BiLSTM_SA).

havior of the trend line consists of decreasing up to the final stages with some minor variations. In this case, XGBoost reached the lower MAE. The behavior of the trend line observed consists of decreasing up to the final stages with some minor variations. It is important to note these processes comprise longer sequences with many stages, which may facilitate visually delineating the trend line.

Finally, in general, we observe that the prediction error increases in the first stages of the process. After a few stages, the error keeps varying and decreases in later stages. This behavior may indicate that if there is more information about the execution of the process and if it is closer to the final state, it becomes easier to predict the remaining time. This behavior may also indicate that the models can better distinguish the activities carried out toward the end of the process execution.

RQ3 - How does the best model proposed in this work compare with those proposed in related works?

The models presented in this work produce similar results. Still, using an attention layer produces results with a lower average absolute error despite being within the confidence interval presented in Table 8. The architectures proposed in this work that use *LSTM* and Attention Layers technically have the same ability to predict the remaining time to complete a business process. The models presented in related works differ in the complexity of *LSTM* [Tax *et al.*, 2017] and [Navarin *et al.*, 2017] and in the existence of a *LSTM* to support the sequences [Bukhsh *et al.*, 2021], although both

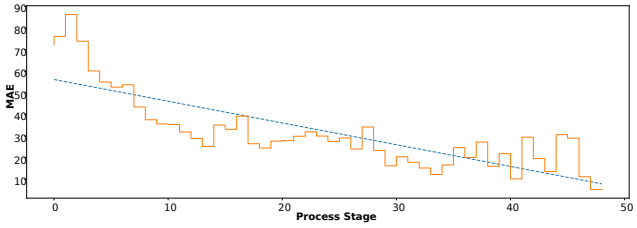

Figure 7. S_75 dataset - MAE by process stage (*XGBoost*).

works use embeddings. The best model presented in this work can produce results that outperform related work and other more traditional machine learning strategies.

RQ4 - Is the model error influenced by the distribution of the remaining time needed to complete the processes?

We evaluated three datasets, each related to a specific topic and with completion time with different deviations. Our hypothesis is that the sets of processes with larger deviations should be more challenging, since they comprise processes with less regular behaviors, compared to the public databases used in this work. Furthermore, since the partitioning criterion for the training, validation, and test sets is temporal, this irregularity of the processes may also be reflected in differences between the processes in these data sets. For example, processes that occur later may have different flows even though they belong to the same topic. This will probably impact the learning of the models too.

For the data sets evaluated, S_25 had the smallest deviations in the remaining time. For this data set, all the models evaluated had a lower MAE than the models that used S_50 and S_75 datasets. The best MAE value for S_25 was 8.48. Similarly, using the S_50 dataset, the models performed worse than using S_25, the best MAE among all proposed models being 12.67. Finally, the performance of the models for the S_75 data set was the worst among all, with the best MAE obtained being 46.15. It is also observed that the margin of error of the models was also greater for the more dispersed data sets.

6 Conclusions and Future Work

This article investigated different approaches for predicting the time remaining to complete a process. Among the approaches evaluated in this work, deep learning and the self-attention layer deal well with logs of real events, regardless of having automated and manual instances in the same dataset or business process instances with several activities or completely different natures. As future work, it is important to consider other datasets, explore logs from different competitions aimed at predictive process monitoring, verify how more recent architectures can contribute to improving the solution to this problem, compare other algorithms and models classics with the current ones, mix these architectures such as the proposals of [Ma *et al.*, 2023] and continue with the availability of a database and repository with the experiments developed to support new practices, research and reproductions.

Funding

This research was partially funded by the projects *FUNCAP-CE* (Ceará Foundation to Support Scientific and Technological Development) and *FASTEF* (Foundation to Support Technical Services, Teaching, and Research Promotion) 04772314/2020, 04772420/2020 and 04772551/2020 and was part of the Chief Scientist Project that aims to unite academia and public management to identify solutions for science, technology, and innovation that can

be implemented to improve services and, in this way, provide a better quality of life for the population of the state of Ceará (Brazil).

Authors' Contributions

RM, LC, DV, and CS contributed to the conception of this study. RS is the main contributor and writer of this manuscript and performed the experiments and validation. Also supported by RM and LC writing. JM was research support.

Competing interests

The authors declare that they have the following competing interests.

Availability of data and materials

The public datasets and code experiments performed in the current study are available on: https://github.com/RonildoSilva/PTRC_PN_DL.

References

Akhramovich, K., Serral, E., and Cetina, C. (2024). A systematic literature review on the application of process mining to industry 4.0. *Knowledge and Information Systems*, 66(5):2699–2746.

Bukhsh, Z. A., Saeed, A., and Dijkman, R. M. (2021). Processtransformer: Predictive business process monitoring with transformer network. *arXiv preprint arXiv:2104.00721*.

Castro, M. A., Souza Jr, N., Escovedo, T., Lopes, H., and Kalinowski, M. (2022). Mineração de processos aplicada à auditoria interna na marinha do brasil. In *Anais do XXXVII Simpósio Brasileiro de Bancos de Dados*, pages 241–253. SBC.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

da Silva, R. O., Magalhães, R. P., Cruz, L. A., de Souza, C. P., de Vasconcelos, D. R., and de Macêdo, J. A. F. (2023). Predição de tempo restante para conclusão de processos de negócio utilizando aprendizado profundo. In *Anais do XXXVIII Simpósio Brasileiro de Bancos de Dados*, pages 141–153. SBC.

Kalenkova, A., Ageev, A., Lomazova, I. A., and van der Aalst, W. M. (2017). E-government services: Comparing real and expected user behavior. In *International Conference on Business Process Management*, pages 484–496. Springer.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Lakshmanan, V., Robinson, S., and Munn, M. (2020). *Machine learning design patterns*. O'Reilly Media.

Ma, H., Yang, P., Wang, F., Wang, X., Yang, D., and Feng, B. (2023). Short-term heavy overload forecasting of public transformers based on combined lstm-xgboost model. *Energies*, 16(3):1507.

Mello, P., Santoro, F., and Revoredo, K. (2020). It incident solving domain experiment on business process failure prediction. *Journal of Information and Data Management*, 11(1).

Mello, P. O., Revoredo, K., and Santoro, F. (2019). Business process failure prediction: a case study. In *Anais do VII Symposium on Knowledge Discovery, Mining and Learning*, pages 89–96. SBC.

Navarin, N., Vincenzi, B., Polato, M., and Sperduti, A. (2017). Lstm networks for data-aware remaining time prediction of business process instances. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE.

Park, G. and Song, M. (2020). Predicting performances in business processes using deep neural networks. *Decision Support Systems*, 129:113191.

Paschek, D., Luminosu, C. T., and Draghici, A. (2017). Automated business process management–in times of digital transformation using machine learning or artificial intelligence. In *MATEC Web of Conferences*, volume 121, page 04007. EDP Sciences.

Polato, M. (2017). Dataset belonging to the help desk log of an italian company.

Reijers, H. A. (2021). Business process management: The evolution of a discipline. *Computers in Industry*, 126:103404.

Stjepić, A.-M., Ivančić, L., and Vugec, D. S. (2020). Mastering digital transformation through business process management: Investigating alignments, goals, orchestration, and roles. *Journal of entrepreneurship, management and innovation*, 16(1):41–74.

Tax, N., Verenich, I., Rosa, M. L., and Dumas, M. (2017). Predictive business process monitoring with lstm neural networks. In *International Conference on Advanced Information Systems Engineering*, pages 477–492. Springer.

van Dongen, B. (2012). Bpi challenge 2012.

Venkateswaran, P., Muthusamy, V., Isahagian, V., and Venkatasubramanian, N. (2021). Robust and generalizable predictive models for business processes. In *Business Process Management: 19th International Conference, BPM 2021, Rome, Italy, September 06–10, 2021, Proceedings*, pages 105–122. Springer.

Venugopal, I., Töllich, J., Fairbank, M., and Scherp, A. (2021). A comparison of deep-learning methods for analysing and predicting business processes. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Wang, J., Yu, D., Liu, C., and Sun, X. (2019). Outcome-oriented predictive process monitoring with attention-based bidirectional lstm neural networks. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 360–367. IEEE.