Remaining cycle time prediction with Graph Neural Networks for Predictive Process Monitoring

Le Toan Duong, Louise Travé-Massuyès, Audine Subias, Christophe Merle

Remaining cycle time prediction with Graph Neural Networks for Predictive Process Monitoring

Le Toan Duong*
LAAS-CNRS, Université Fédérale de Toulouse, CNRS, INSA
ANITI, Université Fédérale Toulouse Midi Pyrénées
Vitesco Technologies France SAS
Toulouse, France
ltduong@laas.fr

Louise Travé-Massuyès
Audine Subias
LAAS-CNRS, Université Fédérale de Toulouse, CNRS, INSA
ANITI, Université Fédérale Toulouse Midi Pyrénées
Toulouse, France

Christophe Merle
ANITI, Université Fédérale Toulouse Midi Pyrénées
Vitesco Technologies France SAS
Toulouse, France

ABSTRACT

Predictive process monitoring is at the intersection of machine learning and process mining. This subfield of process mining leverages historical data generated from process executions to make predictions about the ongoing process. One of the predictive process monitoring tasks with high interest is predicting the remaining cycle time of process instances. Recently, deep neural networks, particularly long short-term memory, have attracted much attention due to their ability to learn relevant features automatically and predict with high accuracy. While these models require data defined in the Euclidean space, graph neural networks have the advantage of handling data that can be represented as graphs. This paper proposes the use of graph neural network models to predict the remaining cycle time, which has not yet been studied in the literature. The proposed models are evaluated on real-life event logs and compared to a state-of-the-art long short-term memory model. The results show that graph neural network models can improve prediction accuracy, particularly for complex processes.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Machine learning approaches** → **Neural networks**.

KEYWORDS

Predictive process monitoring, Remaining cycle time prediction, Machine learning, Graph neural networks

ACM Reference Format:
Le Toan Duong, Louise Travé-Massuyès, Audine Subias, and Christophe Merle. 2023. Remaining cycle time prediction with Graph Neural Networks for Predictive Process Monitoring. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Predictive Process Monitoring (PPM) is a subfield of process mining that refers to a family of techniques that try to predict the outcome or the future properties of an ongoing process case. For this purpose, historical data on process executions stored in event logs are used as input. Then, depending on the specific needs, a predictive model is built and trained to predict the relevant information of the current process. Several PPM problems have been addressed and solved. Among them, Rama-Maneiro et al. [23] have formalized the most common ones. The first one is the prediction of attributes, i.e., activity and timestamp related to the next event. The second one is predicting the sequence of events and their attributes until the end of the case. The third problem consists in predicting the outcome of an event prefix [12, 30]. For example, in an order-to-cash process, the outcome of a case may be the level of customer satisfaction or the risk of delay. In a manufacturing process, it may be the quality of the final product, i.e., whether the product will be returned or not by the customer. Another PPM problem that has emerged in recent years, which is the focus of our study, is the prediction of the remaining cycle time (RCT).

Especially given an event prefix of a running case, the RCT prediction model seeks to forecast the time until the completion of the case. The results can be used to prevent process instances with long cycle times and support process managers in making resource allocation decisions or adjusting process behavior. Several studies have been conducted on this problem [16, 20, 22, 35]. There are two main approaches to predicting the RCT of a running process. The first consists of recursively predicting the next activity and its timestamp until the end of a process. In this case, the RCT is derived by simply summing the time between all intermediate activities. The second approach is to directly predict the RCT from the prefix. This approach learns from historical data a function Ω from the prefix space to $\mathbb{R}$. A systematic review of the state-of-the-art models is presented in Section 2. Whereas for the recursive approach, the model developed for predicting the next event is reused to predict the RCT, the direct approach must build its own model. However, the disadvantage of the recursive method is that many intermediate prediction steps have to be performed to obtain the final result. Consequently, the error can accumulate through these steps and become significant for the final outcome. This problem is apparent

when the model has to handle long processes with many events and especially processes with a lot of duplicated activities [26]. Some experimental results that confirm this intuition are presented in Section 4. In this paper, we focus mainly on the development of models for the direct approach. The paper presents the first use of Graph Neural Networks (GNN) for RCT prediction. The motivation of the study is that the precedence relationships between activities in a process can be represented as graphs. Hence, using GNN could help leverage this information in prediction. Two GNN models have been evaluated and compared with the Long Short-Term Memory (LSTM) model in public datasets and a real manufacturing process. The results show the GNN model's effectiveness in predicting the remaining cycle time.

The paper is organized as follows. Section 2 presents preliminaries and the related work. The different predictive models studied in this work are presented in Section 3. In Section 4, we present the prediction results obtained with GNN and the comparison, and Section 5 concludes the work.

2 RELATED WORK

Several researches have been conducted on the topic of PPM. These can be categorized into different classes depending on the PPM problems, the application domains, the prediction model, etc. In [15], the authors classify the approaches into two families, process-aware and non-process-aware methods. Di Francescomarin et al. [6] review existing methods for the PPM based on prediction type, input data required, tool support, the validity of the algorithm, and the family of algorithms to help industries select the method that best suits their problem. The authors in [10] focus only on deep learning approaches based on Recurrent Neural Networks (RNN), LSTM, and Stacked Autoencoder. A complete review of the state-of-the-art deep learning methods in the PPM is presented in [17, 23].

Regarding the RCT prediction problem, as in the PPM, the approaches are classified into process-aware and non-process-aware methods. Process awareness means whether or not the approach uses a process model as input to make the prediction. Most process-aware methods discover process models from event logs because the model is not always known and may be different from the real behavior. In general, a transition system [2, 20, 21, 31] or a Petri net [35] is constructed and used to predict the remaining time. Meanwhile, Verenich et al. [38] make a prediction based on the process tree obtained from historical traces. Non-process-aware approaches usually apply machine learning algorithms to learn a model from labeled training data, i.e., supervised learning [37]. Several regression models can be used such as linear regression [1], random forest [32], XGBoost [25] and neural networks [3, 8, 11, 14, 16, 26, 28, 29, 39]. In [5], the authors propose a framework that uses both process and non-process-aware methods for the RCT prediction. They combined the prediction given by a transition system-based model with the prediction from a multiple linear regression model to make the final results. The approach proposed in our paper is non-process-aware as it does not use any process model. It takes only the process state represented in the prefixes to make the prediction.

In recent years, deep learning has been widely exploited in the PPM, particularly in the RCT prediction, due to its promising results against classical methods [23]. The authors in [37] present a systematic review and a benchmark of the existing methods for RCT prediction. The results show that LSTM-based models achieve the best accuracy in terms of Mean Absolute Error (MAE) at the time the benchmark is performed. Recently more sophisticated models based on deep learning have been proposed. Wang et al. [40] present a remaining time predictor with reinforcement learning. Taymouri et al. [27] propose a Generative Adversarial Net (GAN) for the suffix and the RCT prediction. Over the last few years, GNNs have been used to solve PPM problems. Philipp et al. [18] present the first use of GNN to predict the amount of disbursement per area regarding a process of application request. They develop a model that contains two graph convolutional layers followed by one linear layer. Venugopal et al. [36] presents a comparison of Graph Convolutional Network (GCN) with the Convolutional Neural Networks (CNN) and LSTM models along with a Multi-Layer Perceptron (MLP) for the next activity and timestamp prediction. In addition, a Gated Graph Neural Network (GGNN) is used in [41]. The authors in [24] build a process model in the form of a Petri net, then feed it to a model that integrates GCNs and RNNs. Whiorrini et al. [4] develop a GNN model that performs well in the event log with a high presence of parallelism. However, all of these works focus only on the problem of predicting the next activity and timestamp. There have been no studies so far about GNN on the RCT prediction. In this paper, we present the first use of the GNN model and compare its performance with the LSTM model of [26]. We choose this latter model as a baseline because it shows the most promising results of the benchmark in [37].

According to input data, all studies process the event log with at least one case ID, activity, and timestamp. Many methods use additional event attributes, and case attributes to enrich the prediction models [21, 34]. Others use contextual information to obtain more accurate predictions [9, 25]. The goal of this study is to evaluate a new family of models, GNN, for the RCT prediction. Since then, we only use mandatory attributes, case ID, activity, and timestamp to keep the approach generalized and applicable to all event logs.

3 METHODOLOGY

3.1 Problem statement

This section provides necessary concepts and notations for the formalization of the studied problem. To illustrate the above concepts, Table 1 presents an example of event log. Each row in the table corresponds to an event.

Let $\mathcal{E}$ be the event universe and AN be the set of attribute names. For any event $e \in \mathcal{E}$ and attribute name $n \in AN$, $\#_n(e)$ is the value of attribute n for event e.

Definition 3.1 (Event). An event e is represented by a tuple of attributes $(\#_c(e), \#_a(e), \#_t(e), \#_{d_1}(e), ..., \#_{d_j}(e), ..., \#_{d_m}(e))$, where c is the case ID, a is the activity, t is the timestamp and d_j is the j^{th} additional attribute of the event.

The additional attributes may be the resource (e.g., person in charge of the task) or the associated cost, etc. Let $\mathcal{E}^*$ be the set of all finite subsets of $\mathcal{E}$.

Table 1: Example of an event log

Event ID	Case ID	Activity	Timestamp
e_1	1	a	2022-01-30 08:20:07
e_2	1	b	2022-02-08 08:58:46
e_3	1	b	2022-02-08 09:59:05
e_4	1	c	2022-02-11 17:27:35
e_5	1	d	2022-02-15 09:45:20
e_6	2	a	2022-01-30 08:38:54
e_7	2	b	2022-02-07 09:30:07
e_8	2	c	2022-02-10 15:12:25
e_9	2	a	2022-02-10 17:15:08
e_{10}	2	d	2022-02-12 16:31:20

Definition 3.2 (Trace). A trace is a finite, non-empty sequence of events $\sigma \in \mathcal{E}^* \setminus \{\emptyset\}$, $\sigma = \langle e_1, ..., e_{|\sigma|} \rangle$ such that for $1 \leq i < j \leq |\sigma|$: $\#_c(e_i) = \#_c(e_j)$.

Definition 3.3 (Event log). An event log L is a set of complete traces, i.e., traces representing the execution from the beginning to the end of a case.

Definition 3.4 (Prefix). A prefix is the head of a trace σ with a length $0 < k < |\sigma|$. It is denoted as $hd^k(\sigma) = \langle e_1, ..., e_k \rangle$.

For example of the trace $\sigma = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ associated with case ID number 1 in Table 1, $hd^2(\sigma) = \langle e_1, e_2 \rangle$ and $hd^3(\sigma) = \langle e_1, e_2, e_3 \rangle$.

The RCT prediction takes as input the prefix of a running case and outputs time remaining until the end of that case.

A RCT prediction model is a function Ω such that $\Omega(hd^k(\sigma)) = \#_t(e_{|\sigma|}) - \#_t(e_k)$, with $k \in 1..(|\sigma| - 1)$.

3.2 Prefix encoding

One of the challenges when working with event log data is that it requires a lot of feature engineering, especially when using machine learning algorithms that take input as numerical tensors, e.g., vectors or matrices. However, a trace representing a process instance's behavior is a sequence of events. Each event contains numerical data such as cost and duration and text data such as activity and resource. Hence, events must be encoded into numerical vectors. The prefix encoding method may affect the performance of the prediction task. Figure 1 shows the methods used in this study, described in the following paragraphs.

3.2.1 Event encoding. We use the event encoding technique presented in [26]. Especially each event is represented by a vector which is a concatenation of its attributes. In this study, we use only two mandatory attributes in all event logs to keep the work as generalized as possible: *activity* and *timestamp*. Categorical attributes such as *activity* are converted to numeric by one hot encoding method due to its simplicity. This is also known as dummy coding. The method creates a dummy variable for each level of a categorical variable which is equal to 1 if the level is present and 0 if it is absent. According to the *timestamp*, we create four time-based features for each event which are the time from the previous event in the same case, the time from the start of its case, the time within the day, and the day within the week.

3.2.2 Prefix encoding. In the task of prediction of the RCT, inputs are prefixes with variable lengths. One solution is to train multiple predictors by dividing all prefix traces into several buckets. Each bucket contains prefixes of a specific length. The disadvantage of this approach is that it requires a lot of models and training, especially when working with long traces. The other solution is to combine all the prefix traces and use a single model to learn them. We used the latter for the presented study. In this case, a prefix encoding step is required to obtain relevant inputs for the model. Two prefix encoding techniques are used in this paper.

- **Prefixes padded** [23]: This type of encoding, which is used in the LSTM model, consists of fixing the prefix's maximum length. Then, shorter prefixes are padded with zeros. In this study, we take the length of the longest trace as the maximum prefix length. An example of this technique is presented in Figure 1, top table. In this example, we suppose that the maximum length is 5, which is the length of the trace. Then, the prefix of length three is padded with two rows of zeros.
- **Prefixes flexible**: This technique encodes all events present in the prefix without using any padding method. Therefore, the length of the feature matrix is equal to the length of the prefix length. It is designed explicitly for the GGNN because this model can handle graphs of different sizes. Indeed, there is a pooling operation at the output of the GGNN to map graphs with different sizes into graphs embedding of the same size. Figure 1, bottom table, presents an example of prefixes flexible technique. We notice that the event e_2 and e_3 refer to the same activity b. Hence, they have similar activity representations. However, the time-based features are different since they do not occur at the same time. The number of lines is dictated by the number of events in the prefix.

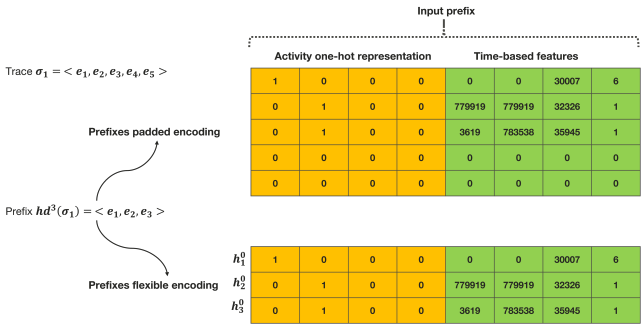

Figure 1: Example of prefix encoding methods used in this paper. The prefixes padded encoding is used in the LSTM model. The GGNN model uses the prefixes flexible encoding. The three first time-based features are computed in seconds. They are normalized using *min-max* normalization during the training phase.

3.3 Graph Neural Networks for the RCT prediction

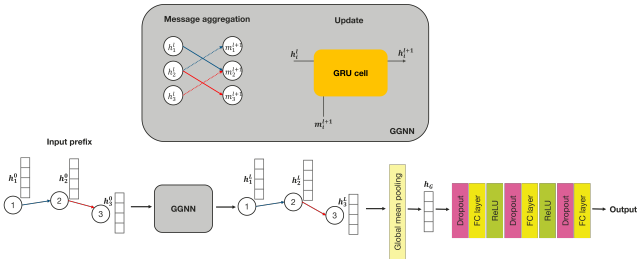

Figure 2: Architecture of GGNN model for the RCT prediction.

This section presents the GGNN model we propose to compare with the baseline (LSTM). We do not present the architecture of the LSTM model used in [26] because this model is well-known. The GGNN model is proposed by Li et al. [13]. The model is constituted of two phases (see Figure 2). The input graph whose node features are initialized by the prefix encoding method presented in Figure 1 first goes through a message aggregation phase in which each node i aggregates information from its neighboring nodes $\mathcal{N}(i)$.

$$m_i^{l+1} = \sum_{j \in \mathcal{N}(i)} e_{ji} \times \Theta \times h_j^l \tag{1}$$

where l is used to identify the iteration (layer), m_i^{l+1} is the message associated to node i at iteration $l+1$, e_{ji} is the edge weight and Θ is the trainable parameters. Then, the hidden state h_i of node i is updated by an update function U. In the case of GGNN, the function U is a Gated Recurrent Unit (GRU) cell.

$$h_i^{l+1} = GRU(m_i^{l+1}, h_i^l) \tag{2}$$

This message-passing process is repeated L times so that the information from one node is propagated to all other nodes in the graph. The hyper-parameter L is similar to the number of convolution layers in the CNN. The GGNN model is used in [41] to predict the next activity. In this study, we applied the model to resolve the RCT prediction problem. The input prefix is considered as a graph whose nodes are events presented in the prefix, and the edges represent the precedence relations between the events. Furthermore, edges are distinguished into three types:

- *Forward*: edges from one to a new activity within a case, e.g., $\langle e_1, e_2 \rangle$ in case 1 of Table 1. This edge is showed in blue in Figure 2.
- *Backward*: edges from one to an activity that has been performed in a case, e.g., $\langle e_8, e_9 \rangle$ in case 2.
- *Repeat*: edges between two events associated with the same activity, e.g., $\langle e_2, e_3 \rangle$ in case 1. This edge is the red one in Figure 2.

At the output of the GGNN block, we obtain the node embeddings. They are then passed through a readout function to get the graph embedding. We use the *global mean pooling* function to do this. Finally, the graph embedding goes through fully connected layers with the activation function ReLU to make a prediction.

4 EVALUATION

4.1 Datasets

In this study, we use three event log datasets.

- **Helpdesk** contains event logs from a ticketing management process of the help desk of an Italian software company. The logs are available at [19].
- **BPIC20** contains events pertaining to two years of travel expense claims [33].
- **EMS3141** dataset presents a real manufacturing process for assembling electronic boards in the automotive company Vitesco Technologies.

Table 2: Statistics of event logs used in the study

Event log	Num. activities	Avg. case len.	Max. case len.	Avg. case duration (days)	Max. case duration (days)	Min. case duration (days)	Variants
Helpdesk	10	4.66	15	40.85	59.99	30.64	207
BPIC20	15	5.49	24	11.62	368.19	1.06	64
EMS3141	35	28.76	44	7.01	80.87	0.69	296

All datasets were preprocessed, and only complete traces were extracted. Table 2 presents the statistics of the processed event logs. The datasets are then split into two sets according to the start time of each case. The first 2/3 of traces are used to train the model and the last 1/3 of traces as the test set. Regarding the input for the RCT prediction problem, only prefixes with a minimum length of two are considered because the GGNN model requires a graph input that contains at least two nodes.

4.2 Experimental setup

Our experiment, which is implemented with Pytorch, consists of two tasks. The first is to compare the performance of the direct and recursive approaches in predicting the RCT. The second task is to evaluate the performance of the GGNN model against the baseline model (LSTM). Regarding the training step, we use 20% of the training set for the validation. In addition, we add dropout layers and use the early stopping technique to prevent the overfitting problem. The dropout percentage is a hyperparameter that is fine-tuned during the training phase. Other hyperparameters we consider in the tuning process are the batch size, the learning rate, the number of layers, and the dimension of hidden layers. We use Bayesian optimization for quick tuning. Once optimized hyperparameters are found, the model is retrained. This process is repeated 5 times in order to have consistent results against the randomness of neural networks. Regarding the loss function and the metric to assess the model performance, we use the well-known metric Mean Absolute Error (MAE). More details about the hyperparameter tuning and the training process are presented at https://github.com/duongtoan261196/RemainingCycleTimePrediction.

4.3 Direct vs recursive approaches for the RCT prediction

This section presents an experiment to compare the performance of recursive and direct approaches in the RCT prediction problem. For that, we use the LSTM model from [26], which was developed in the recursive approach. Then, we build a direct approach of LSTM to compare with. The architecture of the LSTM-direct model is presented in Figure 3. The model includes k layers of LSTM with the dropout layer to prevent over-fitting. k is a hyperparameter, and c_0, h_0 are the initial cell and hidden states.

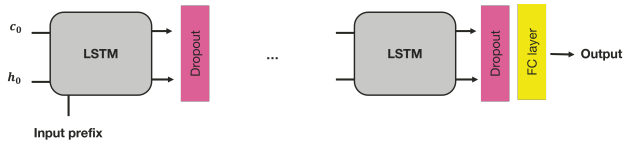

Figure 3: LSTM model for the RCT prediction.

Table 3 presents the average prediction errors over different prefix lengths of the LSTM-recursive and LSTM-direct models on two public datasets *Helpdesk* and *BPIC20*. The table shows that LSTM-direct gives a better prediction for both datasets. The last column shows the percentage of error we could reduce by applying the direct approach instead of the recursive approach. These results confirm our initial arguments in Section 1. Moreover, we observe from the experiment that the recursive approach takes more time to predict than the direct one. This is because, on the one hand, the recursive model must predict the next event until a stopping criterion is reached. On the other hand, the direct model predicts in one shot. For the next experiments, we test only the direct approaches.

Table 3: Average MAE (days) over all prefix lengths of the prediction by the LSTM-recursive and LSTM-direct model

Dataset	LSTM recursive	LSTM-direct	Gain in %
Helpdesk	5.783	**5.666**	2.02
BPIC20	3.431	**3.269**	4.72

4.4 GGNN vs LSTM for the RCT prediction

The main contribution of this study is the use of graph neural networks for RCT prediction. This section compares the performance of the GGNN presented in Figure 2 and the LSTM model in Figure 3. Table 4 shows that the GGNN model outperforms the LSTM for the *Helpdesk* dataset. Concerning the *BPIC20* dataset, the two models perform nearly the same, with only 0.18% difference in the MAE. Figure 4 shows the prediction error for each prefix length. The figure does not show prefix lengths for which the number of samples is very small. It can be seen from Figure 4a that the prediction errors reduce when the prefix length increases. This result is reasonable because the longer prefix gives more information to predict the RCT. However, this seems not to be the case for the *BPIC20* dataset (Figure 4b). Indeed, the error goes up for prefixes of length 6. Actually, in the *BPIC20* dataset, the number of samples for prefixes from the length of 6 is much lower than those for shorter prefixes. Hence, the model is trained less for these prefixes. Overall, these results indicate that the GGNN model may be applied to the RCT prediction problem to achieve better prediction.

Table 4: Average MAE (days) over all prefix lengths of the prediction by the LSTM and GGNN model

Dataset	LSTM	GGNN	Gain in %
Helpdesk	5.666	**5.345**	5.67
BPIC20	**3.269**	3.275	-0.18

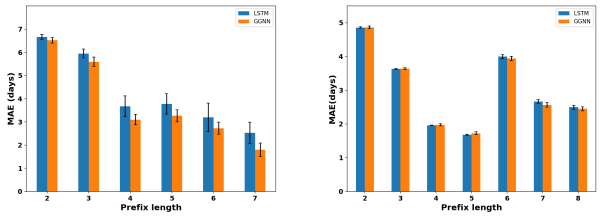

Figure 4: MAE values (days) of different prefix lengths for two public event logs: *Helpdesk* (a), *BPIC20* (b). The error bars at the top of each bar represent the standard deviations of the metric.

4.5 Application of RCT prediction in a real manufacturing process

After comparing the two models on public datasets, we apply them to a real-life process of the automotive company Vitesco Technologies. The process is composed of different steps to assemble electronic boards from printed circuit boards (PCBs). Figure 5 presents the process related to the product family EMS3141 by a directly-follows graph. The assembly process consists of two consecutive phases: Front-end (FE) in blue and Back-end (BE) in yellow. More details about this process can be found in [7]. The dataset cannot be shared for confidentiality reasons. Compared to the two public datasets, the *EMS3141* dataset is more complex in terms of the number of activities, case length, and the number of variants (see Table 2). For the RCT prediction problem, we only consider products that have completed the FE phase and are currently in the BE phase. Because products that have completed the FE phase are stocked in an intermediate area while waiting to be processed in the BE phase, and the waiting time varies significantly depending on the production status and several other factors. This uncertainty can therefore degrade the performance of predictive models.

Figure 6 presents the prediction error for each prefix length between the LSTM and the GGNN model. The results clearly show that the GGNN model outperforms the LSTM. The average error over all prefix lengths is 1.267 hours for LSTM and 0.339 hours for GGNN, respectively. That is 73.2% reduction on the prediction error

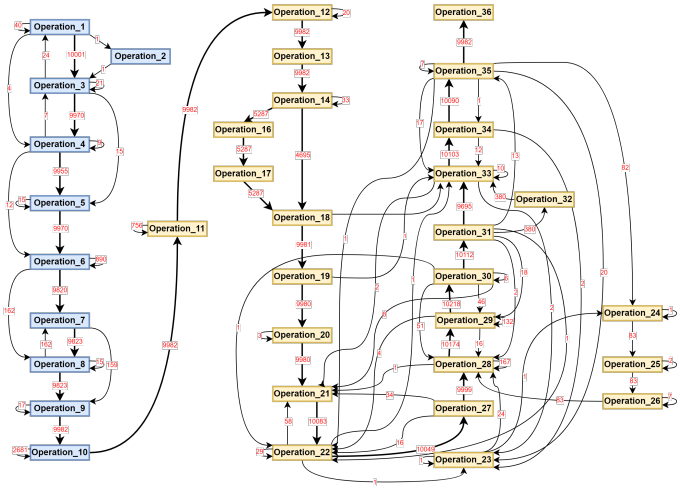

Figure 5: Directly follows graph generated for the EMS3141 logs. The FE phase is in blue, and the BE is in yellow. The label on each edge represents the number of products that have taken the corresponding transition. Operation IDs in nodes are anonymized for confidentiality reasons.

of GGNN versus LSTM. This outperformance is clearly superior compared to the results obtained with *Helpdesk* and *BPIC20*. This also shows that the GGNN model works better with more complex processes.

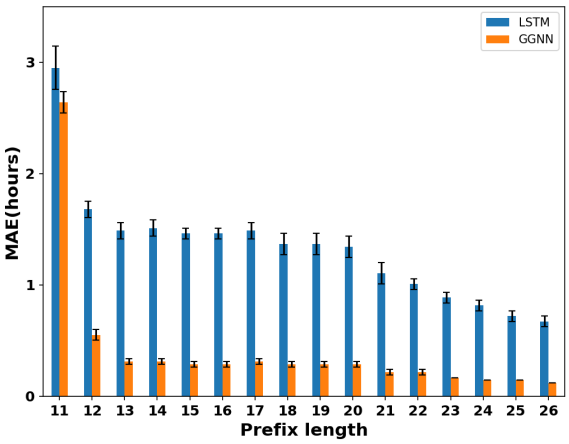

Figure 6: MAE values (hours) of different prefix lengths for the EMS3141 dataset. The error bars at the top of each bar represent the standard deviations of the metric.

5 CONCLUSION AND FUTURE WORKS

This paper presents the first use of GNN to predict the RCT from a given prefix of an ongoing process instance. The paper first demonstrates through experimentation that the direct approach is more efficient than the recursive approach regarding prediction errors and computation time. Then, we compare the GGNN model with the existing LSTM approach, which is the best performing so far for this task [37]. Results on two public event logs and a real-life process show that GGNN outperforms the LSTM model in most scenarios. In particular, the GGNN works a lot better with highly complex processes.

For future research, we intend to develop more GNN variations for the RCT prediction problem. Moreover, we will also focus on the task of prefix encoding and event encoding to improve the prediction. Regarding the training process, the relative error, i.e., Mean Absolute Percentage Error (MAPE), can be used to train and evaluate the performance of prediction models.

All source code and public datasets that can be used to reproduce the reported results are available at https://github.com/duongtoan261196/RemainingCycleTimePrediction.

ACKNOWLEDGMENTS

This project is supported by ANITI through the French "Investing for the Future – P3IA" program under the Grant agreement n^{o}ANR-19-P3IA-0004.

REFERENCES

[1] Ahmad Aburomman, Manuel Lama, and Alberto Bugarín. 2019. A Vector-Based Classification Approach for Remaining Time Prediction in Business Processes. *IEEE Access* 7 (2019), 128198–128212. https://doi.org/10.1109/ACCESS.2019.2939631

[2] Alfredo Bolt and Marcos Sepúlveda. 2014. Process Remaining Time Prediction Using Query Catalogs. In *Business Process Management Workshops*, Niels Lohmann, Minseok Song, and Petia Wohed (Eds.). Springer International Publishing, Cham, 54–65.

[3] Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. 2019. Learning Accurate LSTM Models of Business Processes. In *Business Process Management*, Thomas Hildebrandt, Boudewijn F. van Dongen, Maximilian Röglinger, and Jan Mendling (Eds.). Springer International Publishing, Cham, 286–302.

[4] Andrea Chiorrini, Claudia Diamantini, Alex Mircoli, and Domenico Potena. 2022. Exploiting Instance Graphs and Graph Neural Networks for Next Activity Prediction. In *Process Mining Workshops*, Jorge Munoz-Gama and Xixi Lu (Eds.). Springer International Publishing, Cham, 115–126.

[5] Alexandre Checoli Choueiri, Denise Maria Vecino Sato, Edson Emilio Scalabrin, and Eduardo Alves Portela Santos. 2020. An extended model for remaining time prediction in manufacturing systems using process mining. *Journal of Manufacturing Systems* 56 (2020), 188–201. https://doi.org/10.1016/j.jmsy.2020.06.003

[6] Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, and Fredrik Milani. 2018. Predictive Process Monitoring Methods: Which One Suits Me Best?. In *Business Process Management*, Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke (Eds.). Springer International Publishing, Cham, 462–479.

[7] Le Toan Duong, Louise Travé-Massuyès, Audine Subias, and Nathalie Barbosa Roa. 2021. Assessing product quality from the production process logs. *The International Journal of Advanced Manufacturing Technology* 117, 5 (2021), 1615–1631.

[8] Weiguang Fang, Yu Guo, Wenhe Liao, Karthik Ramani, and Shaohua Huang. 2020. Big data driven jobs remaining time prediction in discrete manufacturing system: a deep learning-based approach. *International Journal of Production Research* 58, 9 (2020), 2751–2766.

[9] Francesco Folino, Massimo Guarascio, and Luigi Pontieri. 2012. Discovering Context-Aware Models for Predicting Business Process Performances. In *On the Move to Meaningful Internet Systems: OTM 2012*, Robert Meersman, Hervé Panetto, Tharam Dillon, Stefanie Rinderle-Ma, Peter Dadam, Xiaofang Zhou, Siani Pearson, Alois Ferscha, Sonia Bergamaschi, and Isabel F. Cruz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 287–304.

[10] Nitin Harane and Sheetal Rathi. 2020. *Comprehensive Survey on Deep Learning Approaches in Predictive Business Process Monitoring.* Springer International Publishing, Cham, 115–128. https://doi.org/10.1007/978-3-030-38445-6_9

[11] Muhammad Asjad Khan, Hung Le, Kien Do, Truyen Tran, Aditya Ghose, Khanh Hoa Dam, and Renuka Sindhgatta. 2018. Memory-Augmented Neural Networks for Predictive Process Analytics. *CoRR* abs/1802.00938 (2018). arXiv:1802.00938 http://arxiv.org/abs/1802.00938

[12] Wolfgang Kratsch, Jonas Manderscheid, Maximilian Röglinger, and Johannes Seyfried. 2021. Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. *Business & Information Systems Engineering* 63, 3 (2021), 261–276.

[13] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. *ICLR* 117 (2016).

[14] Andreas Metzger, Philipp Leitner, Dragan Ivanović, Eric Schmieders, Rod Franklin, Manuel Carro, Schahram Dustdar, and Klaus Pohl. 2015. Comparing and Combining Predictive Business Process Monitoring Techniques. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 2 (2015), 276–290. https://doi.org/10.1109/TSMC.2014.2347265

[15] Alfonso Eduardo Márquez-Chamorro, Manuel Resinas, and Antonio Ruiz-Cortés. 2018. Predictive Monitoring of Business Processes: A Survey. *IEEE Transactions on Services Computing* 11, 6 (2018), 962–977. https://doi.org/10.1109/TSC.2017.2772256

[16] Nicolo Navarin, Beatrice Vincenzi, Mirko Polato, and Alessandro Sperduti. 2017. LSTM networks for data-aware remaining time prediction of business process instances. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Honolulu, HI, USA, 1–7. https://doi.org/10.1109/SSCI.2017.8285184

[17] Dominic A Neu, Johannes Lahann, and Peter Fettke. 2022. A systematic literature review on state-of-the-art deep learning methods for process prediction. *Artificial Intelligence Review* 55, 2 (2022), 801–827.

[18] Patrick Philipp, Rafael X. Morales Georgi, Jürgen Beyerer, Sebastian Robert, and Jürgen Beyerer. 2019. Analysis of Control Flow Graphs Using Graph Convolutional Neural Networks. In *2019 6th International Conference on Soft Computing & Machine Intelligence (ISCMI)*. IEEE, Johannesburg, South Africa, 73–77. https://doi.org/10.1109/ISCMI47871.2019.9004296

[19] M Polato. 2017. Dataset belonging to the help desk log of an Italian Company. *University of Padova, Padova* 1 (2017). https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

[20] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. 2014. Data-aware remaining time prediction of business process instances. In *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, IEEE, Beijing, China, 816–823.

[21] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. 2018. Time and activity sequence prediction of business process instances. *Computing* 100, 9 (2018), 1005–1031.

[22] Mahsa Pourbafrani, Shreya Kar, Sebastian Kaiser, and Wil M. P. van der Aalst. 2022. Remaining Time Prediction for Processes with Inter-case Dynamics. In *Process Mining Workshops*, Jorge Munoz-Gama and Xixi Lu (Eds.). Springer International Publishing, Cham, 140–153.

[23] Efrén Rama-Maneiro, Juan Vidal, and Manuel Lama. 2021. Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Transactions on Services Computing* 1 (2021), 1–1.

[24] Efrén Rama-Maneiro, Juan C Vidal, and Manuel Lama. 2021. Embedding Graph Convolutional Networks in Recurrent Neural Networks for Predictive Monitoring. *arXiv preprint arXiv:2112.09641* 1 (2021).

[25] Arik Senderovich, Chiara Di Francescomarino, Chiara Ghidini, Kerwin Jorbina, and Fabrizio Maria Maggi. 2017. Intra and Inter-case Features in Predictive Process Monitoring: A Tale of Two Dimensions. In *Business Process Management*, Josep Carmona, Gregor Engels, and Akhil Kumar (Eds.). Springer International Publishing, Cham, 306–323.

[26] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. 2017. Predictive Business Process Monitoring with LSTM Neural Networks. In *Advanced Information Systems Engineering*, Eric Dubois and Klaus Pohl (Eds.). Springer International Publishing, Cham, 477–492.

[27] Farbod Taymouri and Marcello La Rosa. 2020. Encoder-decoder generative adversarial nets for suffix generation and remaining time prediction of business process models. *arXiv preprint arXiv:2007.16030* 1 (2020).

[28] Farbod Taymouri and Marcello La Rosa. 2020. Encoder-Decoder Generative Adversarial Nets for Suffix Generation and Remaining Time Predication of Business Process Models. *CoRR* abs/2007.16030 (2020). arXiv:2007.16030 https://arxiv.org/abs/2007.16030

[29] Farbod Taymouri, Marcello La Rosa, and Sarah M. Erfani. 2021. *A Deep Adversarial Model for Suffix and Remaining Time Prediction of Event Sequences.* Proceeding of SIAM, Virtual conference, 522–530. https://doi.org/10.1137/1.9781611976700.59 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611976700.59

[30] Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. 2019. Outcome-Oriented Predictive Process Monitoring: Review and Benchmark. *ACM Trans. Knowl. Discov. Data* 13, 2, Article 17 (mar 2019), 57 pages. https://doi.org/10.1145/3301300

[31] Wil MP Van der Aalst, M Helen Schonenberg, and Minseok Song. 2011. Time prediction based on process mining. *Information systems* 36, 2 (2011), 450–475.

[32] Sjoerd van der Spoel, Maurice van Keulen, and Chintan Amrit. 2013. Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital. In *Data-Driven Process Discovery and Analysis*, Philippe Cudre-Mauroux, Paolo Ceravolo, and Dragan Gašević (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 60–83.

[33] Boudewijn van Dongen. 2020. BPI Challenge 2020: Domestic Declarations. 4TU.ResearchData. fDataset. *BPI Challenge 2020* 1 (2020). https://doi.org/10.4121/uuid:3f422315-ed9d-4882-891f-e180b5b4feb5

[34] B. F. van Dongen, R. A. Crooy, and W. M. P. van der Aalst. 2008. Cycle Time Prediction: When Will This Case Finally Be Finished?. In *On the Move to Meaningful Internet Systems: OTM 2008*, Robert Meersman and Zahir Tari (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 319–336.

[35] Jarne Vandenabeele, Gilles Vermaut, Jari Peeperkorn, and Jochen De Weerdt. 2022. Enhancing Stochastic Petri Net-based Remaining Time Prediction using k-Nearest Neighbors. *arXiv preprint arXiv:2206.13109* 1 (2022).

[36] Ishwar Venugopal, Jessica Töllich, Michael Fairbank, and Ansgar Scherp. 2021. A Comparison of Deep-Learning Methods for Analysing and Predicting Business Processes. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, IEEE, Shenzhen, China, 1–8.

[37] Ilya Verenich, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Irene Teinemaa. 2019. Survey and Cross-Benchmark Comparison of Remaining Time Prediction Methods in Business Process Monitoring. *ACM Trans. Intell. Syst. Technol.* 10, 4, Article 34 (jul 2019), 34 pages. https://doi.org/10.1145/3331449

[38] Ilya Verenich, Hoang Nguyen, Marcello La Rosa, and Marlon Dumas. 2017. White-Box Prediction of Process Performance Indicators via Flow Analysis. In *Proceedings of the 2017 International Conference on Software and System Process* (Paris, France) *(ICSSP 2017)*. Association for Computing Machinery, New York, NY, USA, 85–94. https://doi.org/10.1145/3084100.3084110

[39] Nur Ahmad Wahid, Taufik Nur Adi, Hyerim Bae, and Yulim Choi. 2019. Predictive business process monitoring–remaining time prediction using deep neural network with entity embedding. *Procedia Computer Science* 161 (2019), 1080–1088.

[40] Qiqi Wang, Hongjie Zhang, Cheng Qu, Yu Shen, Xiaohui Liu, and Jing Li. 2021. RLSchert: An HPC Job Scheduler Using Deep Reinforcement Learning and Remaining Time Prediction. *Applied Sciences* 11, 20 (2021), 2076–3417. https://doi.org/10.3390/app11209448

[41] Sven Weinzierl. 2021. Exploring gated graph sequence neural networks for predicting next process activities. In *International Conference on Business Process Management*. Springer, Springer International Publishing, Cham, 30–42.