




Probabilistic Suffix Prediction of Business Processes

Michel Kunkler^{*} , Henryk Mustroph^{*} , Stefanie Rinderle-Ma 

TUM School of Computation, Information and Technology

Technical University of Munich

Garching, Germany

{michel.kunkler, henryk.mustroph, stefanie.rinderle-ma}@tum.de

Abstract—Suffix prediction of business processes forecasts the remaining sequence of events until process completion. Current approaches focus on predicting the most likely suffix, representing a single scenario. However, when the future course of a process is highly uncertain and variable, a single scenario may have limited predictive value. To address this limitation, we propose probabilistic suffix prediction, a novel approach that returns a set of sampled suffixes. The method is based on an uncertainty-aware encoder-decoder LSTM combined with a Monte Carlo suffix sampling algorithm. We capture epistemic uncertainty via MC dropout and aleatoric uncertainty as learned loss attenuation. Comparisons with two other uncertainty-aware PPM approaches across four datasets demonstrate that our probabilistic suffix prediction approach achieves reasonable predictive performance while it allows estimating prediction intervals for multiple objectives within a single model.

Index Terms—Probabilistic Suffix Prediction, Epistemic & Aleatoric Uncertainty, Encoder-Decoder LSTM

I. INTRODUCTION

In recent years, predicting the future course of a running business process (BP) using machine learning (ML) models has gained considerable attention in the field of Predictive Process Monitoring (PPM) [1]. Many PPM approaches have focused on predicting a single objective, such as the next activity, remaining time, or the process outcome [2]. Recent works have developed approaches that predict an entire sequence of remaining events, known as suffix prediction using neural networks (NNs) [3]–[10]. Current suffix prediction approaches predict a single most-likely suffix. However, in domains such as healthcare or manufacturing, the future course of a business process can be highly uncertain and variable. In such situations, given the low likelihood that the true suffix matches the most likely prediction, one may be more interested in exploring possible suffix scenarios and obtaining prediction intervals for relevant aspects.

Consider a simple repair process of a service center as depicted in Fig. 1, where the repair activity fails frequently, meaning that it must be repeated until the quality control is passed. Suppose that an organization nevertheless aims to provide customers with predictions regarding the return date of their parts. Providing a single return date might not be practical because it would often be missed, possibly leading to customer dissatisfaction. By instead offering a time range with a given coverage probability (e.g., 75%), the organization can reflect

inherent uncertainties while still providing useful information to the customer. Assume further that the organization wants to inform customers about the expected range of repair costs, which depends on the number of repair attempts required. Accordingly, the organization would likely aim to provide a prediction interval for the number of repair cycles that a part might undergo, which could be extracted from the same suffixes to be consistent with the time range.

Hence, this work aims at learning uncertainties from historical data and to take these uncertainties into account for predicting and sampling possible suffix scenarios, from which multiple prediction intervals can be obtained. In ML, uncertainties are often categorized into epistemic and aleatoric. Epistemic uncertainty stem from a lack of knowledge, e.g., a lack of training data, and are defined as reducible. Aleatoric uncertainty, conversely, are irreducible. In the context of a business process, they can, e.g., stem from external factors such as delays in deliveries from external stakeholders or from human involvement in the process execution. Both forms of uncertainty can be learned jointly with NNs [11]. However, no existing approach has yet incorporated epistemic and aleatoric uncertainties for suffix prediction of business processes. We refer to our approach as probabilistic suffix prediction which we achieve by first training an uncertainty-aware encoder-decoder long short-term memory (U-ED-LSTM) NN on event log data. The U-ED-LSTM captures epistemic uncertainty by applying MC dropout and aleatoric uncertainty by learned loss attenuation [11]–[13]. For a given prefix, we then sample suffixes with our Monte Carlo suffix sampling algorithm (MC-SA). The MC-SA draws suffix samples by using the U-ED-LSTM and takes aleatoric and epistemic uncertainty into account. For generating a suffix, the MC-SA algorithm samples events until completion in an autoregressive fashion. The algorithm uses a novel combination of dropout as a Bayesian approximation and sampling from the via loss attenuation learned probability distributions to realize event attributes in an autoregressive fashion.

We show that i) our probabilistic suffix prediction approach yields multiple prediction intervals from sampled suffixes and ii) achieves competitive predictive performance against single-objective methods. The paper is outlined as follows: Sec. II covers preliminaries, Sec. III describes our probabilistic suffix prediction framework, Sec. IV presents the evaluation, Sec. V discusses related approaches, and Sec. VI concludes the work.

^{*}These authors contributed equally.

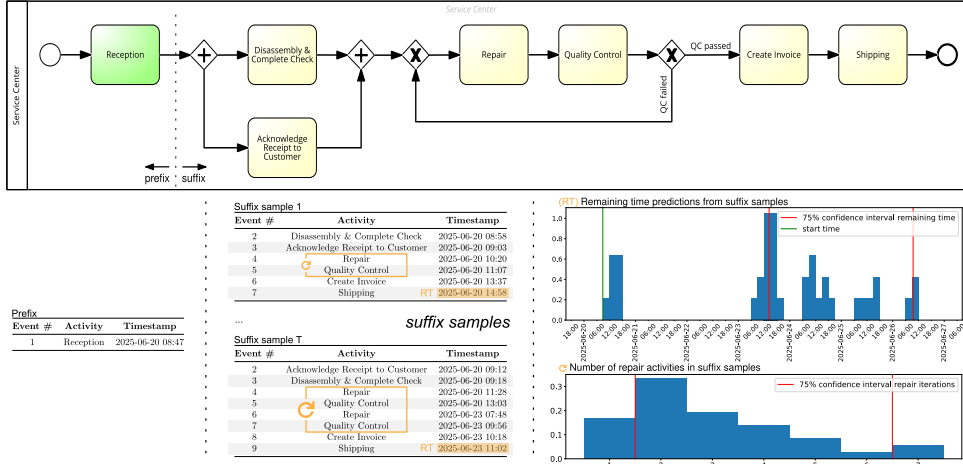


Fig. 1. Suffix samples of a repair process and histograms of the remaining time and number of repair activities with a 75% prediction interval

II. PRELIMINARIES

This section introduces uncertainty in machine learning, followed by suffix prediction of business processes.

A. Uncertainty in Machine Learning

ML distinguishes epistemic and aleatoric uncertainties. Epistemic uncertainty is referred to as “uncertainty due to a lack of knowledge about the perfect predictor” [14] and is reducible. There is an ongoing debate regarding how epistemic uncertainty should be captured, with one possibility being the use of probability distributions [14]. Aleatoric uncertainty is considered irreducible as it stems from inherently random effects in the underlying data. Aleatoric uncertainty is “appropriately modeled in terms of probability distributions” [14] and can henceforth be learned in a probabilistic model.

Epistemic Uncertainty using Dropout as a Bayesian Approximation. Using dropout at every activation layer (or variants such as drop-connect) in an NN during training and inference can be a simple and computationally efficient method for approximating Variational Inference, thereby enabling a Bayesian approximation of the posterior predictive distribution [12]. This approach is referred to as *Monte Carlo (MC) dropout* because the posterior predictive distribution $p(y|x, X, Y)$ is approximated via MC sampling, using multiple stochastic forward passes where at each forward pass, a dropout mask is sampled from a variational distribution $q_\theta(W)$ and applied on the activations, which approximates the posterior predictive distribution. In practice, a dropout mask is often sampled from a Bernoulli distribution $z \sim \text{Bernoulli}(1-p)$, where p denotes the dropout probability. The dropout mask is then applied to the NN’s activations. During training, applying L2 regularization to the NN parameters θ corresponds to a Gaussian prior over its parameters, enabling the overall objective to approximate the evidence lower bound in Variational Inference.

Heteroscedastic Aleatoric Uncertainty as Learned Loss Attenuation. Heteroscedastic models assume the observation

noise σ can vary with the input x . This noise $\sigma(x)$ can represent aleatoric uncertainty arising from inherent randomness in the data-generating process. The model explicitly quantifies this irreducible uncertainty by learning $\sigma(x)$. Assuming that the noise $\sigma(x)$ is normally distributed on an output neuron $f_y^W(x)$ of the NN, where $f^W(\cdot)$ denotes the NN with its weights W , then the noise can be learned as an additional output neuron $f_\sigma^W(x)$. Training the standard deviation of a normal distribution $f_\sigma^W(x)$ by including it in the loss function is referred to as learned loss attenuation (see [11]).

In the case of classification, NNs typically employ the Softmax function, which already outputs a categorical probability distribution. However, this probability distribution might not capture model uncertainties [11]. Therefore, means and variances should also be learned on the predicted logits to capture model uncertainties.

B. Suffix Prediction

We define an event log $EL := \{t^{(1)}, t^{(2)}, \dots, t^{(L)}\}$ as a set of cases, where L denotes the total number of cases. A case is a sequence of events denoted by $t^{(l)} := \langle e_1, e_2, \dots, e_M \rangle$, where M is the number of events in case l . An event is a tuple of continuous and categorical event attributes, denoted $e_m := (a_{con}, a_{cat})$. In this work, we assume that an event has at least one categorical attribute: An event label, which links the event to a class of event types; and one continuous attribute: a timestamp attribute, which expresses the time at which an event happened. A case can be split into several prefix and suffix pairs. A prefix is defined as $p_{\leq k} := \langle e_1, e_2, \dots, e_k \rangle$, with $1 \leq k < M$. A suffix is defined as $s_{> k} := \langle e_{k+1}, \dots, e_M \rangle$. Suffix prediction involves predicting a suffix \hat{s} based on an input prefix $p_{\leq k}$.

III. PROBABILISTIC SUFFIX PREDICTION FRAMEWORK

This section presents the probabilistic suffix prediction framework consisting of the U-ED-LSTM model and the MC-SA.

A. Uncertainty-Aware Encoder-Decoder LSTM

The U-ED-LSTM implementation comprises the data preparation, model architecture, and loss functions for training.

Data Preparation and Embedding. Given an event log, we first apply feature engineering techniques to the events' timestamp attribute to derive additional features for the U-ED-LSTM. We introduce a *case elapsed time* attribute, representing the time elapsed since the first event in the case, an *event elapsed time* attribute, representing the time since the last event within the same case (with the value set to 0 for the first event), a *day of the week* attribute, and a *time of day* attribute. The latter two features are incorporated due to the potential influence of periodic trends on the future course of a process. E.g., in a company that operates only on weekdays, when an activity is completed on Friday evening, the next activity is unlikely to occur before Monday. We apply standard scaling to all continuous event attributes, except for the raw timestamp, and encode missing values as 0. Following [10], we also apply input padding to facilitate batch training: Each case is padded with zeros at the beginning to a fixed length, determined by the maximum case length in the event log, excluding the top 1.5% of the longest cases. For every categorical event attribute with K unique category classes, we add an additional NA (not available) class for missing values and an unknown class (category class not present in the training data). After the data pre-processing, all categorical event attributes are embedded using an embedding layer stack that maps each categorical event attribute into a vector of fixed dimensionality. The embedding layer is defined as a learnable weight matrix of size $(K+2) \times D$, where $D = \min(600, \text{round}(1.6(K+2)^{0.56}))$ is the chosen embedding dimension, following a common rule of thumb [10].

Model Architecture. The U-ED-LSTM employs an encoder-decoder (ED) architecture based on LSTMs [15], similar as in [7]. LSTMs are well-suited for handling sequential data and have been proven effective for suffix prediction in business processes [2]. Additionally, ED architectures offer flexibility by decoupling tasks between the encoder and decoder and by handling different input and output event features: the encoder can focus on summarizing the prefix and can take all event attributes as input, while the decoder leverages these representations to predict target event attributes, e.g., only the event labels and time features (see [6], [7]). The U-ED-LSTM implementation allows users to flexibly select the encoder's input event attributes, as well as the decoder's input and output event attributes. For both the encoder and decoder, we use LSTMs with stochastic LSTM cells, which apply dropout as a Bayesian approximation, as in [16]. Rather than employing single output neurons to represent either a mean or logit value, each output is now represented by two neurons: one for the mean (or logit) and a second for the standard deviation. Fig. 2 illustrates the U-ED-LSTM architecture during training. Naive dropout has shown to be ineffective in recurrent NNs such as LSTMs [13]. Therefore, [13] have proposed a different dropout variant in which the same dropout mask is applied across all

time steps in an RNN, referred to variational dropout. We apply variational dropout in our U-ED-LSTM, indicated by the colored arrows in Fig. 2.

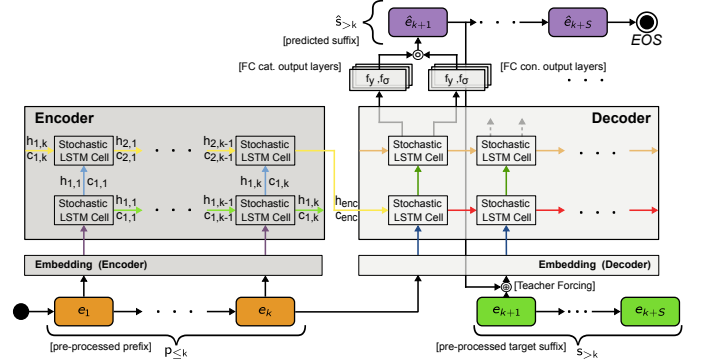


Fig. 2. Exemplary U-ED-LSTM model and training pipeline

The **encoder** processes input prefixes to compress the event sequence information into a fixed-length representation, known as a latent vector. More formally, we define the encoder as a function $f^{\hat{W}_{enc}}(\cdot)$, with masked weights sampled from the encoder's variational distribution $\hat{W}_{enc} \sim q_{\theta_{enc}}(W_{enc})$. For a given input prefix $p_{\leq k}$, a latent vector tuple is predicted: $f^{\hat{W}_{enc}}(p_{\leq k}) := (h_k, c_k)$. Thereby, h_k and c_k represent the last hidden and cell state in the encoder.

The **decoder** receives at the first timestep the latent vector from the encoder (h_{enc}, c_{enc}) and the last event from the prefix e_k . It then updates the latent vector and generates a new event. At each subsequent timestep, the model uses the previously updated latent vector tuple and a previous event. During training, teacher forcing is applied, selecting either the previous event from the target suffix or the last predicted event randomly. Then the decoder autoregressively predicts S events, where S is a predefined fixed sequence length. Similar to the encoder, we sample the decoders masked weights from its variational distribution $\hat{W}_{dec} \sim q_{\theta_{dec}}(W_{dec})$. For a given time step $s = 0, 1, \dots, S-1$, we define e_{k+s} as the current event, (h_{k+s}, c_{k+s}) as the current latent vector tuple, and the prediction of the next event and updated latent vector tuple as follows: $f^{\hat{W}_{dec}}(e_{k+s}, (h_{k+s}, c_{k+s})) := (\hat{e}_{k+(s+1)}, (h_{k+(s+1)}, c_{k+(s+1)}))$. A predicted event consists of the concatenation of all its event attributes. These can be categorical or continuous, such that the predicted event is defined as: $\hat{e}_{k+(s+1)} := (\{f^{\hat{W}_{dec}}_{con_1}(\cdot), \dots, f^{\hat{W}_{dec}}_{con_D}(\cdot)\}, \{f^{\hat{W}_{dec}}_{cat_1}(\cdot), \dots, f^{\hat{W}_{dec}}_{cat_K}(\cdot)\})$, where D the number of continuous and K the number of categorical attributes. The continuous attribute predictions consist of a mean prediction \hat{y} and a log-variance prediction \hat{v}_{con} , $f^{\hat{W}_{dec}}_{con}(\cdot) := (\hat{y}, \hat{v}_{con})$. The categorical attribute predictions consist of a mean \hat{l} and a log-variance vector \hat{v}_{cat} , where each element in the respective vectors represents a mean and a log-variance value of a categorical class, $f^{\hat{W}_{dec}}_{cat}(\cdot) := (\hat{l}, \hat{v}_{cat})$. The log-variance is predicted, rather than the variance itself, to ensure numerical stability [11].

Loss Functions. To train the U-ED-LSTM, we use two distinct attenuated loss functions, one for continuous and another one for categorical event attributes. The loss is calculated for a batch of N prefix-suffix pair training samples, $\{p_{\leq k}^{(i)}, s_{> k}^{(i)}\}_{i=1}^N$. The continuous loss function is implemented as follows:

$$\mathcal{L}_{con} = \frac{1}{N \times S} \sum_{i=1}^N \sum_{s=0}^{S-1} \frac{(y_{k+(s+1)}^{(i)} - \hat{y}_{k+(s+1)}^{(i)})^2}{2 \cdot \exp(\hat{v}_{k+(s+1)}^{(i)})} + \frac{1}{2} \hat{v}_{k+(s+1)}^{(i)} \quad (1)$$

For categorical event attributes, we calculate the loss using MC integration by averaging the cross-entropy loss (CEL) over multiple samples drawn from the logits distribution. $\hat{z} = \hat{l} + \hat{\sigma} \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, I)$. The categorical loss function is implemented as follows:

$$\mathcal{L}_{cat} = \frac{1}{N \times S} \sum_{i=1}^N \sum_{s=0}^{S-1} \frac{1}{T} \sum_{t=1}^T \text{CEL}(y_{k+(s+1)}^{(i)}, \hat{z}_{k+(s+1),t}^{(i)}) \quad (2)$$

The total loss consists of a weighted sum of continuous losses and categorical losses, weighted by weight coefficient vectors w_{con} and w_{cat} and the L_2 regularization term of the encoder's and decoder's parameters (θ). The total loss is implemented as follows:

$$\mathcal{L} = \sum_{k=1}^K w_{con}^k \mathcal{L}_{con}^k + \sum_{d=1}^D w_{cat}^d \mathcal{L}_{cat}^d + \lambda(\|\theta_{enc}\|_2^2 + \|\theta_{dec}\|_2^2) \quad (3)$$

B. MC Suffix Sampling Algorithm

The MC-SA draws suffix samples from our U-ED-LSTM. This algorithm is specifically designed to generate multiple samples from uncertainty-aware event sequence prediction methods, such as probabilistic suffix prediction. Unlike time series forecasting, event sequence prediction is less commonly addressed in the AI domain. However, it plays a central role in business process management, which makes the proposed MC-SA particularly suitable for process sequence forecasting. It combines dropout as a Bayesian approximation during inference to take epistemic uncertainty into account and the via loss attenuation learned probability distributions to consider aleatoric uncertainty. It is formalized in Alg. 1 and can be outlined as follows: At each MC trial, a new suffix is sampled. For sampling a suffix, the prefix is first passed into the encoder to obtain a latent vector tuple (h_{enc}, c_{enc}) . We apply variational dropout already on the encoder to account for potential uncertainties in encoding the prefix into the latent space. The decoder is then employed for sampling a suffix in an autoregressive fashion. At the first step, the latent vector tuple and the last event from the prefix are taken as input. The event attributes of the first event of the suffix are then sampled, with categorical and continuous attributes handled differently: For continuous event attributes, the event attribute values are drawn from normal distributions with the predicted mean and variance pairs. For categorical event attributes, the logit values of the individual categories are first drawn from normal distributions and then passed through a Softmax

Algorithm 1 MC Suffix Sampling

Require: $T \in \mathbb{N}$: number of MC samples, $M \in \mathbb{N}$: max. suffix length to be sampled, $p \in [0, 1]$: dropout probability, $p_{\leq k} = \langle e_1, e_2, \dots, e_k \rangle$: prefix

```

1: function MCSUFFIXSAMPLING( $T, M, p, p_{\leq k}$ )
2:    $\tilde{S} \leftarrow \emptyset$  ▷ Set of sampled suffixes
3:   for  $t = 1$  to  $T$  do
4:      $\tilde{s}_{> k} \leftarrow \langle \rangle$  ▷ Current suffix
5:      $\tilde{W}_{enc} \leftarrow \text{VariationalDropout}(W_{enc}, p)$ 
6:      $(h_{enc}, c_{enc}) \leftarrow f_{enc}^{\tilde{W}_{enc}}(p_{\leq k})$ 
7:      $\tilde{e} \leftarrow e_k$ 
8:      $i \leftarrow 0$ 
9:     repeat
10:       $\tilde{W}_{dec} \leftarrow \text{NaiveDropout}(W_{dec}, p)$ 
11:       $\tilde{e}_{k+1}, (h, c) \leftarrow f_{dec}^{\tilde{W}_{dec}}(\tilde{e}, (h_{enc}, c_{enc}))$ 
12:       $\tilde{a}_{con}, \tilde{a}_{cat} \leftarrow \tilde{e}_{k+1}$ 
13:      for  $j = 1$  to  $|D|$  do
14:         $\tilde{y}, \tilde{v}_{con} \leftarrow \tilde{a}_{con}^{(j)}$ 
15:         $\tilde{a}_{con}^{(j)} \sim \mathcal{N}(\tilde{y}, \exp(\tilde{v}_{con}))$ 
16:      end for
17:      for  $j = 1$  to  $|K|$  do
18:         $\tilde{l}, \tilde{v}_{cat} \leftarrow \tilde{a}_{cat}^{(j)}$ 
19:         $\tilde{a}_{cat}^{(j)} \sim \text{Categorical}(\text{Softmax}(\mathcal{N}(\tilde{l}, \exp(\tilde{v}_{cat}))))$ 
20:      end for
21:       $\tilde{e} \leftarrow (\tilde{a}_{con}, \tilde{a}_{cat})$ 
22:       $\tilde{s}_{> k} \leftarrow \tilde{s}_{> k} \circ \tilde{e}$ 
23:       $i \leftarrow i + 1$ 
24:    until  $i = M$  or  $\text{GetActivity}(\tilde{a}_{cat}) = \text{'EOS'}$ 
25:     $\tilde{S} \leftarrow \tilde{S} \cup \{\tilde{s}_{> k}\}$ 
26:  end for
27:  return  $\tilde{S}$ 
28: end function

```

function to obtain a categorical distribution. In a subsequent step, the categorical class is drawn from this distribution. The autoregressive prediction of the next event continues until either *EOS* is predicted or the predefined maximum sequence length M is reached. The algorithm returns \tilde{S} , a set of sampled suffixes with size T . It combines aspects of two other autoregressive sampling algorithms for time series data, namely that of [17], which also uses variational MC dropout on the encoder and naive dropout on the decoder for sampling the next timestamp, and that of [18], which samples from trained probability distributions.

IV. EVALUATION

In this section, we first describe the experimental setting, i.e., how we conducted the training, and which data sets and hyperparameters we chose. We compared the predictive performance of our probabilistic suffix prediction approach against most-likely suffix predictions from our U-ED-LSTM model and against two other uncertainty-aware PPM approaches from the literature. Our implementation and models are publicly available¹.

Datasets. We evaluated on one synthetic and three real-life data sets. The synthetic data set resembles the repair shop process as depicted in Fig. 1. The data set and its generating code are available in our repository¹. The Helpdesk² dataset is an event log from a ticket management system from an Italian software company. The Sepsis³ dataset represents the pathway of patients diagnosed with Sepsis through a hospital.

¹ Repository: https://github.com/ProbabilisticSuffixPredictionLab/Probabilistic_Suffix_Prediction_U-ED-LSTM_pub

² Helpdesk: <https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb>

³ Sepsis: <https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460>

TABLE I
DATASET PROPERTIES

Dataset	cases	events	variants	event labels	mean-sd case length	mean-sd case duration (days)	cat. event attr.	con. event attr.
Helpdesk	4 580	21 348	226	14	4.66 - 1.18	40.86 - 8.39	12	4
Sepsis	1 049	15 214	845	16	14.48 - 11.47	28.48 - 60.54	26	8
BPIC-17	31 509	1 202 267	15 930	26	38.16 - 16.72	21.90 - 13.17	9	9
Repair Shop	9 896	79 874	16	7	8.07 - 2.14	2.27 - 1.42	1	4

The BPIC-17⁴ dataset is a loan application process from a Dutch bank and has been investigated in the Business Process Intelligence Competition (BPIC)-17. Properties for each dataset are presented in Tab. I. The datasets were split at the case level into training and testing sets using an 80%-20% ratio, following the approach in [19]. The training data was further divided into a training and validation set, resulting in an overall 65%-15%-20% training-validation-testing data split. We conducted our evaluation on the test set, starting from a prefix length of 1, i.e., $p_{\leq k}, \forall k \geq 1$.

Training and Sampling. We trained our U-ED-LSTM model on an NVIDIA GTX 4090 GPU. Several training optimization techniques were implemented: Since sequence-to-sequence training is more effective than single-event training for suffix prediction [9], we calculated the loss and optimized the model using sequences of $S = 5$ events. Using a fixed sequence length S performed better in our approach and also reduced the impact of early prediction errors in the suffix. Furthermore, we applied teacher forcing (open-loop training), i.e., selecting either the last predicted or target event as input for the next event prediction [7]. We set the initial teacher forcing probability to 0.8, meaning that 80% of the input events came from the target suffix, and then decreased the ratio from 20% of the training epochs onward. Since suffix prediction is a multi-task learning problem involving different categorical and numerical event attributes, we implemented a task-balancing algorithm called GradNorm [20]. GradNorm dynamically tunes the weight coefficient vectors w_{con} and w_{cat} of the loss terms after each optimization step based on the relative importance of each event attribute on the overall loss. We set the MC dropout rate to a constant value of $p = 0.1$ and sampled $T = 1000$ times.

Hyperparameter Setting. For each dataset, three different U-ED-LSTMs with varying hyperparameter settings were trained and compared. The results are described in detail in our technical report [21]. Here we present the hyperparameter setting of the U-ED-LSTM with the best evaluation results. We trained the U-ED-LSTM with four LSTM layers for the encoder and decoder, along with fully-connected layers for the mean and variance of each output event attribute. All event attributes were used as input features for the encoder, while only the *event label* (i.e., activity) and time attributes (i.e., *case and event elapsed time*) were used as input and output features for the decoder. The remaining hyperparameters were set as follows: Both the encoder and decoder have a hidden size of 128. We used the AdamW optimizer with a learning

rate of 10^{-4} for the Repair Shop 10^{-5} for Helpdesk and Sepsis, and 10^{-6} for the larger BPIC-17 dataset. We used a batch size of 128, except for BPIC-17, where a larger batch size of 256 was needed to speed up training. All models were trained for 100 epochs without early stopping, but with continuous monitoring of performance on the validation set. The regularization parameter was set to $\lambda = 10^{-4}$.

Reimplemented Approaches. For obtaining prediction intervals for suffix lengths, we reimplemented the LSTM model based suffix prediction approach from [5]. Their approach implicitly models uncertainty in predicting next-event labels by random sampling from the categorical distributions produced by the LSTM. While random sampling could also be applied to other suffix prediction approaches, we adopted the approach of [5], as they are the only ones who explicitly proposed random sampling from categorical distributions and evaluated this scenario. For comparison, we could also consider a beam-search strategy as in [7]. However, beam search does not capture uncertainty in the same way as random sampling and typically generates only a limited number of suffixes (fewer than ten). Additionally, we reimplemented the uncertainty-aware LSTM model for remaining time prediction from [16]. This approach was chosen because it captures both epistemic and aleatoric uncertainties within the same framework proposed by [11] and enables the construction of remaining-time prediction intervals.

For both reimplemented approaches, we used the same datasets and data preparation (i.e., preprocessing, splitting, and embedding), training-related hyperparameter settings (i.e., optimizer, learning rate, epochs, and batch size), and sampling iterations (i.e., 1000), as employed for training our U-ED-LSTM and sampling with our MC-SA. We reimplemented the *full-shared* LSTM approach from [5], which employs one shared LSTM layer, and one LSTM and fully-connected layer for each predicted event attribute. The input event attributes include *event labels*, *resources*, and *case elapsed time*, while the model predicts suffixes in an autoregressive fashion. Unlike conventional suffix prediction methods, which select the event label via the arg-max operation on the softmax output, their approach samples the next event label from the categorical distribution derived from the softmax function at each step (random-sampling), thereby implicitly considering uncertainties [11]. We reimplemented the LSTM model in accordance with the specifications provided in [5], using a hidden size of 50 and batch normalization as regularization between the shared and task-specific LSTMs. The reimplemented approach proposed by [16] uses two LSTM layers with MC dropout and learned loss attenuation. The *event labels* and the *case elapsed*

⁴BPIC-17: <https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b>

time are used as input as well as predicted output. As the target, we used the *case elapsed time*. We reimplemented the LSTM model with the parameters specified in [16], namely a hidden size of 10, a fixed dropout rate of $p = 0.1$, and the L2 regularization parameter $\lambda = 0.1$.

A. Mean Predictions

We evaluated the predictive performance of point predictions (i.e., mean aggregations) from our probabilistic suffix samples and compared them against most-likely suffix predictions obtained from our U-ED-LSTM and against the two reimplemented approaches from the literature [5], [16].

Metrics. To evaluate the mean predictions, we adopted three commonly used evaluation metrics for suffix prediction: The *Damerau-Levenshtein Similarity* (DLS) metric to assess the predicted event labels, also known as activity sequence prediction [5], [7], [8], [10], [22] and the *Mean Average Error* (MAE) of the remaining time prediction [4], [5], [7], [10]. Previous studies [4], [5] showed that some suffix prediction methods struggle to predict the correct suffix length. To address this, we also evaluated the suffix length prediction by calculating the MAE. The most-likely suffix prediction was obtained by auto-regressively generating a suffix by sampling the most-likely event label at each step until the *EOS* token is reached, similar to other works [4]–[7], [9], [10]. The DLS on the event labels is defined as a normalized DLS distance $DLS(\hat{s}, s) := 1 - \frac{DL(s, \hat{s})}{\max(|s|, |\hat{s}|)}$ where s and \hat{s} denote the true and predicted sequence of event labels. Informally, $DLS = 1$ expresses that the two sequences are identical, while a $DLS = 0$ expresses that the two are entirely dissimilar. For the most-likely suffixes, we computed the DLS by comparing the prediction to the ground truth suffix. For the probabilistic column in Tab. II, we calculated the DLS for each sampled suffix and took the mean. To obtain the most-likely remaining time MAE, we calculated both the last *case elapsed time* and the sum of the *event elapsed times* of the most-likely suffix, compared each with the ground truth remaining time, and used the better of the two. For the probabilistic remaining time MAE, we calculated the mean of the last *case elapsed time* and the mean of the sums of the *event elapsed times* across all sampled suffixes and compared it to the ground truth, and used the better of the two. For the most-likely suffixes, we compared their lengths with the ground truth lengths. For the probabilistic, we computed the length of each sampled suffix, took the mean, and compared it to the ground truth lengths.

Results. All mean prediction results are depicted in Tab. II. An overview of the results and comparisons across different hyperparameter settings of our approach is provided in a technical report [21]. For the Helpdesk test data, our model achieved a DLS score of 0.82 when the most-likely suffix is obtained and a score of 0.65 when suffix samples from our MC-SA are aggregated. In comparison, the approach of [5] achieved a DLS of 0.66 for random-sampling (probs.) and 0.67 when the arg-max (most-likely) next event label is used. The results from our approach and from the approach

from [5] are comparable; our technical report [21] shows that deeper and more advanced models tend to produce the best results. For the Sepsis test data, our approach achieved only DLS score of 0.12, while the simpler model of [5] achieved a better DLS of 0.33 at random sampling and even 0.37 for the arg-max suffix. For the BPIC-17 test data, our MC-SA approach achieved a DLS score of 0.31, which outperforms the DLS score from the most-likely suffixes. The improved DLS score of the probabilistic approach may be attributed to the lower MAE in suffix length, as many most-likely suffixes sampled in this dataset can exceed 50 events. The approach of [5] achieved a DLS of 0.11 for random-sampling and 0.1 for using arg-max. For the Repair Shop test data, the DLS of our probabilistic approach with 0.75 is slightly lower than that of the most-likely predictions with 0.85, yet both outperform the results achieved by the model of [5]. While we also achieve

TABLE II
PREDICTIVE PERFORMANCE

Metric	Dataset	Our approach		[16]		[5]	
		most likely	prob.	most likely	prob.	most likely	prob.
suffix event labels DLS \uparrow	Helpdesk	0.82	0.65	-	-	0.67	0.66
	Sepsis	0.11	0.12	-	-	0.37	0.33
	BPIC-17	0.21	0.31	-	-	0.10	0.11
	Repair	0.85	0.75	-	-	0.69	0.67
suffix length MAE \downarrow	Helpdesk	0.36	0.38	-	-	0.90	0.65
	Sepsis	8.80	6.83	-	-	1.90	2.82
	BPIC-17	40.83	11.42	-	-	45.57	45.51
	Repair	1.05	1.38	-	-	2.10	2.16
remaining time (days) MAE \downarrow	Helpdesk	9.10	10.99	11.73	17.50	-	-
	Sepsis	34.50	31.18	29.48	32.12	-	-
	BPIC-17	10.73	10.62	10.44	12.53	-	-
	Repair	0.70	0.90	0.97	1.13	-	-

reasonable results for remaining time prediction, a notable finding is that our probabilistic approach can outperform the most-likely predictions. In contrast, this does not hold for the model of [16], where the most-likely predictions even slightly outperform the probabilistic ones of our approach on the Sepsis and BPIC-17 datasets. For the Helpdesk test data, the remaining time MAE is 10.99 days for the probabilistic and 9.1 days, outperforming the results of [16]. For the Sepsis test data, our approach reached a remaining-time MAE of 31.18 days for the probabilistic, which is slightly higher, but still close to the best MAE of 29.48 days achieved by the most-likely variant of [16]. For the BPIC-17 test data, the MAE was 10.62 days for our probabilistic approach, essentially matching the best result of 10.44 days from the most-likely variant of [16].

B. Prediction Intervals

Secondly, we assess the accuracy of prediction intervals for remaining time and suffix length intervals. For that, we compared the empirical coverage of the prediction intervals obtained from our suffix samples with the coverage of the prediction intervals for remaining time predictions obtained from the implementation of [16] and for suffix lengths obtained from the implementation of [5].

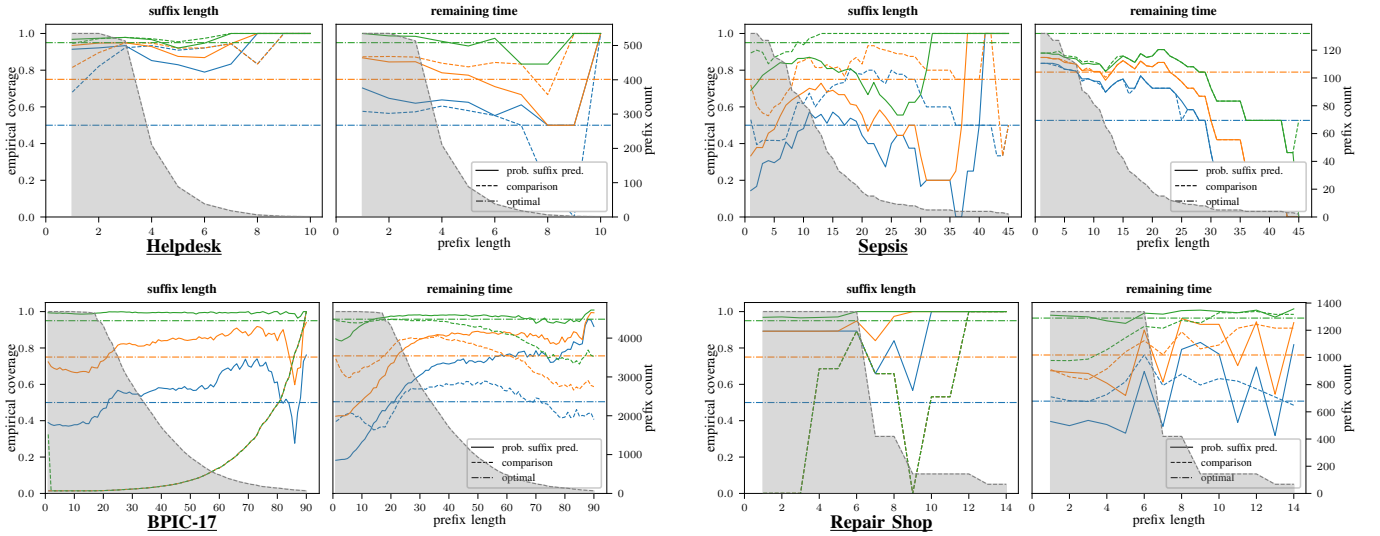


Fig. 3. Empirical coverages

Prediction Intervals. We used our approach to predict 50%, 75%, and 95% prediction intervals of remaining times and sequence lengths based on the samples obtained from our MC-SA and compared it with the same intervals that we obtained for the suffix lengths from using the approach from [5] and for the remaining time by using the approach from [16]. Fig. 3 shows for the individual prefix lengths the average empirical coverages, measured as the proportion of ground truth suffixes contained within the predicted confidence. The plots show that for all but the synthetic repair shop data set, the empirical coverage clearly increases with the nominal level, indicating that our approach can be used to predict multiple levels of confidence. Our model shows overcoverage for both metrics on Helpdesk and undercoverage for Sepsis. For BPIC-17 and Sepsis, the most pronounced undercoverage is shown in the suffix lengths for small prefix lengths, indicating that the model predicts *EOS* tokens too frequently. The aggregated coverages aggregated across all prefix lengths are shown in Tab. III. Prediction intervals are useful for estimating ranges of possible outcome scenarios, which can provide more valuable information than predicting only an expected outcome. Moreover, when combined with a process model or domain knowledge, suffix length confidence intervals may allow for computing count intervals.

V. RELATED WORK

Suffix Prediction. Current suffix prediction approaches focus on predicting accurate most likely suffixes. The methods differ in the models used, predicted event attributes, and strategies to enhance training. Early suffix prediction approaches use LSTMs [3]–[5], [9]. Predictive performance is improved by using encoder-decoder LSTMs [6], [7]. More recent encoder-decoders are enriched by more complex NN architectures such as combined General Recurrent Units, Graph NNs, and attention [8] or transformers [10]. Recently, LLMs have been

TABLE III
EMPIRICAL COVERAGES

Metric	Dataset	Our approach			[16] / [5]		
		50%	75%	95%	50%	75%	95%
suffix length	Helpdesk	0.91	0.94	0.97	0.83	0.90	0.97
	Sepsis	0.37	0.55	0.80	0.54	0.72	0.93
	BPIC-17	0.47	0.75	0.99	0.03	0.03	0.04
	Repair	0.88	0.91	0.98	0.42	0.42	0.42
remaining time	Helpdesk	0.65	0.84	0.98	0.57	0.87	1.0
	Sepsis	0.64	0.77	0.85	0.71	0.77	0.81
	BPIC-17	0.47	0.57	0.94	0.47	0.76	0.93
	Repair	0.45	0.69	0.96	0.57	0.72	0.81

used for suffix prediction [22], facing challenges such as lack of interpretability or not all prefixes can simultaneously be passed into a prompt. In addition, existing approaches can be categorized based on predicted event attributes. Some approaches predict only the sequence of activities [3], [8], [22] and lifecycle transitions [6]. Other approaches predict the sequence of activities and time attributes [4], [7], [9], [10], and resource information [5]. Special training considerations are applied to improve predictive performance. [7], for example, introduce teacher forcing and enhance its encoder-decoder LSTM with adversarial training to improve performance and robustness. For testing, [5] try random sampling from categorical distributions against an arg-max strategy to derive the best matching activities in a suffix.

Uncertainty in PPM. For remaining time and process outcome predictions, combined epistemic and aleatoric uncertainty for NNs is applied to PPM by [16]. [23] applies and compares deep ensemble and MC dropout in attention-based NNs for the next activity prediction. Both approaches aim to improve single-event prediction performance and show how uncertainty and prediction accuracy correlate. Most recently, [24] introduces Conformalized MC dropout, leveraging uncertainty and conformal predictions to construct prediction

intervals for the processing time of the next event prediction. However, they do not evaluate their approach on open-source datasets.

Business Process Simulation. Predicting suffixes can also be performed via simulation techniques [25], [26], which are based on simulation models that can also be directly and automatically discovered from data. A main difference between suffix prediction and simulation approaches lies in how to overall system's state, e.g., the current resource availabilities or concurrently running instances, is modeled: Simulation approaches explicitly model the system's state, whereas suffix prediction approaches learn it implicitly, e.g., waiting times in a simulation model arise explicitly due to resources being busy or unavailable, whereas in PPM approaches, waiting times are emulated implicitly through learned patterns [27]. Using business process simulation approaches to generate suffixes starting from a given prefix would pose some challenges: First, the overall system's state at the last event of a prefix must be modeled, which may be done by using information from recent events. Second, the state of the prefix's process itself would need to be modeled, which may pose challenges, e.g., when the control flow is based on a Petri net, replaying the prefix on the Petri net could yield several possible markings (unless the Petri net has unique labeled transitions), resulting in multiple possible starting points for the simulation. The primary advantage of probabilistic suffix prediction lies in its minimal reliance on assumptions about the underlying process. In contrast, data-driven business process simulation approaches, e.g., a control-flow discovery algorithm, resources' roles and availabilities, or activity durations [25]. On the other hand, due to the explicit representation of the system's state, simulation models may have the potential to provide more exact results.

VI. CONCLUSION

In this work, we propose a probabilistic suffix prediction approach based on our U-ED-LSTM and a novel, event-sequence-specific MC-SA. Our approach learns and approximates epistemic and aleatoric uncertainties via MC dropout and leveraging learned loss attenuation. We demonstrate that our approach can be used to predict prediction intervals for multiple objectives based on a single set of sampled suffixes. Our results show that both the calibration of the approach with respect to a specific prediction interval and its overall predictive performance can often be improved. Hence, we aim to address calibrating prediction intervals and adapt methods for enhancing the predictive performances. E.g., different NN architectures, such as transformer architectures, as well as learning different distributions via loss attenuation, appear promising. Lastly, we see potential in applying probabilistic suffix prediction in data-driven business process simulation or in proactive conformance checking.

REFERENCES

- [1] F. M. Maggi, C. D. Francescomarino, M. Dumas, and C. Ghidini, "Predictive monitoring of business processes," in *CAiSE*, 2014, pp. 457–472.
- [2] P. Ceravolo, M. Comuzzi, J. De Weerd, C. Di Francescomarino, and F. M. Maggi, "Predictive process monitoring: concepts, challenges, and future research directions," *Process Science*, vol. 1, no. 1, p. 2, 2024.
- [3] J. Evermann, J. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," *Decis. Support Syst.*, vol. 100, pp. 129–140, 2017.
- [4] N. Tax, I. Verenich, M. L. Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *CAiSE*, 2017, pp. 477–492.
- [5] M. Camargo, M. Dumas, and O. G. Rojas, "Learning accurate LSTM models of business processes," in *BPM*, 2019, pp. 286–302.
- [6] L. Lin, L. Wen, and J. Wang, "Mm-pred: A deep predictive model for multi attribute event sequence," in *SDM*, 2019, pp. 118–126.
- [7] F. Taymouri, M. L. Rosa, and S. M. Erfani, "A deep adversarial model for suffix and remaining time prediction of event sequences," in *SDM*, 2021, pp. 522–530.
- [8] E. Rama-Maneiro, J. C. Vidal, M. Lama, and P. Monteagudo-Lago, "Exploiting recurrent graph neural networks for suffix prediction in predictive monitoring," *Computing*, vol. 106, no. 9, pp. 3085–3111, 2024.
- [9] B. R. Gunnarsson, S. vanden Broucke, and J. D. Weerd, "A direct data aware LSTM neural network architecture for complete remaining trace and runtime prediction," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 2330–2342, 2023.
- [10] B. Wuyts, S. K. L. M. vanden Broucke, and J. D. Weerd, "Sutran: an encoder-decoder transformer for full-context-aware suffix prediction of business processes," in *ICPM*, 2024, pp. 17–24.
- [11] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *NeurIPS*, 2017, pp. 5574–5584.
- [12] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016, pp. 1050–1059.
- [13] —, "A theoretically grounded application of dropout in recurrent neural networks," in *NeurIPS*, 2016, pp. 1019–1027.
- [14] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," *Mach. Learn.*, vol. 110, no. 3, pp. 457–506, 2021.
- [15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [16] H. Weytjens and J. D. Weerd, "Learning uncertainty with artificial neural networks for predictive process monitoring," *Appl. Soft Comput.*, vol. 125, p. 109134, 2022.
- [17] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *ICDM Workshops*, 2017, pp. 103–110.
- [18] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.
- [19] I. Ketykó, F. Mannhardt, M. Hassani, and B. F. van Dongen, "What averages do not tell: predicting real life processes with sequential deep learning," in *SAC*, 2022, pp. 1128–1131.
- [20] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *ICML*, 2018, pp. 794–803.
- [21] H. Mustroph, M. Kunkler, and S. Rinderle-Ma, "An Uncertainty-Aware ED-LSTM for probabilistic suffix prediction," *arXiv preprint arXiv:2505.21339*, 2025.
- [22] V. Pasquadibisceglie, A. Appice, and D. Malerba, "LUPIN: A LLM approach for activity suffix prediction in business process event logs," in *ICPM*, 2024, pp. 1–8.
- [23] P. Portolani, A. Brusaferrri, A. Ballarino, and M. Matteucci, "Uncertainty in predictive process monitoring," in *IPMU*, 2022, pp. 547–559.
- [24] N. Mehdiyev, M. Majlatow, and P. Fettke, "Augmenting post-hoc explanations for predictive process monitoring with uncertainty quantification via conformalized monte carlo dropout," *Data Knowl. Eng.*, vol. 156, p. 102402, 2025.
- [25] D. Chapela-Campa, O. López-Pintado, I. Suvorau, and M. Dumas, "SIMOD: automated discovery of business process simulation models," *SoftwareX*, vol. 30, p. 102157, 2025.
- [26] F. Meneghello, C. D. Francescomarino, C. Ghidini, and M. Ronzani, "Runtime integration of machine learning and simulation for business processes: Time and decision mining predictions," *Inf. Syst.*, vol. 128, p. 102472, 2025.
- [27] M. Camargo, D. Báron, M. Dumas, and O. G. Rojas, "Learning business process simulation models: A hybrid process mining and deep learning approach," *Inf. Syst.*, vol. 117, p. 102248, 2023.