An Uncertainty-Aware ED-LSTM for Probabilistic Suffix Prediction

Henryk Mustroph Michel Kunkler Stefanie Rinderle-Ma

Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany
{henryk.mustroph, michel.kunkler, 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 a single, most likely suffix. However, if the future course of a process is exposed to uncertainty or has high variability, the expressiveness of a single suffix prediction can be limited. To address this limitation, we propose probabilistic suffix prediction, a novel approach that approximates a probability distribution of suffixes. The proposed approach is based on an Uncertainty-Aware Encoder-Decoder LSTM (U-ED-LSTM) and a Monte Carlo (MC) suffix sampling algorithm. We capture epistemic uncertainties via MC dropout and aleatoric uncertainties as learned loss attenuation. This technical report provides a detailed evaluation of the U-ED-LSTM's predictive performance and assesses its calibration on four real-life event logs with three different hyperparameter settings. The results show that i) the U-ED-LSTM has reasonable predictive performance across various datasets, ii) aggregating probabilistic suffix predictions into mean values can outperform most likely predictions, particularly for rare prefixes or longer suffixes, and iii) the approach effectively captures uncertainties present in event logs.

Keywords: Probabilistic Suffix Prediction, Epistemic and Aleatoric Uncertainties, Encoder-Decoder LSTM

1 Introduction

In recent years, predicting the future course of a running business process (BP) using machine learning models has gained considerable attention in the field of Predictive Process Monitoring (PPM) [16]. Many well-performing PPM approaches use neural network (NN) architectures and have been criticized for acting as black-box approaches, lacking interpretability of their predictions [26]. Developing approaches that combine high accuracy and interpretability has therefore been acknowledged as a primary challenge in PPM [4]. Recent works have contributed towards more transparency in predictions by developing approaches that predict an entire sequence of remaining events, known as suffix prediction [3, 6, 9, 15, 21, 24, 25, 28]. Current suffix prediction approaches

have focused on predicting a single most likely suffix. In certain domains, the future course of a business process is often subjected to uncertainties and high variability, making it unlikely that it will match exactly with the predicted most likely suffix. Consider a drug development process in a pharmaceutical company: The process involves risks and uncertainties. Unforeseen events and unpredictable human influences can affect the remaining sequence of events. The company aims to predict the suffix, e.g., to plan resources, estimate the remaining time, and gauge the potential market approval of the drug. However, focusing solely on the most likely suffix may overlook alternative, plausible event sequences. By considering other possible suffixes, the pharmaceutical company can account for uncertainty in decision-making and improve its risk management.

Machine learning (ML) distinguishes epistemic and aleatoric uncertainties [11]. Epistemic uncertainties are reducible and stem from a lack of knowledge, e.g., training data. Aleatoric uncertainties, conversely, are irreducible. For instance, in the context of a business process, they can stem from external factors beyond the control of the organization running the process, such as delays in deliveries from external stakeholders or the involvement of humans in process execution. In this work, instead of predicting a single most likely suffix, we consider epistemic and aleatoric uncertainties to predict a probability distribution of suffixes. In line with the term probabilistic learning, which has been used in machine learning and statistics to emphasize that not a single target is learned, but a target distribution [14], we refer to our approach as *probabilistic suffix prediction*. We achieve *probabilistic suffix prediction* by training an *Uncertainty-Aware Encoder-Decoder Long Short-Term Memory* (U-ED-LSTM) NN and an *MC suffix sampling* algorithm. The U-ED-LSTM captures epistemic uncertainties by using MC dropout and aleatoric uncertainties as learned loss attenuation [7, 8, 12]. The MC suffix sampling algorithm can be outlined as follows: Multiple MC trials are conducted, where in each trial, the U-ED-LSTM is used to sample a suffix. The suffix sampling in one MC trial is performed auto-regressively, i.e., the suffix is generated iteratively by sampling one event after another until an end-of-sequence (EOS) token is sampled. We sample all event attributes from probability distributions obtained from the U-ED-LSTM for each event. Using three different hyperparameter settings, we evaluate the U-ED-LSTM predictive performance on four real-life datasets. Additionally, we added results from comparable models from the literature to demonstrate that the U-ED-LSTM has reasonable predictive performance. Furthermore, the probabilistic suffix prediction results are evaluated more thoroughly by comparing them with the most likely suffix prediction and assessing the model's calibration. The results show that i) the U-ED-LSTM exhibits reasonable predictive performance across various datasets, ii) aggregating probabilistic suffix predictions into mean values can outperform most likely predictions, particularly for rare prefixes or longer suffixes, and iii) assessing the calibration on the predicted remaining time (continuous event attributes) shows that our approach can capture temporal uncertainties given in the training data.

The technical report is outlined as follows: Sec. 2 covers preliminaries, Sec. 3 describes our probabilistic suffix prediction framework, Sec. 4 presents the evaluation, Sec. 5 discusses related approaches, and Sec. 6 concludes the work.

2 Preliminaries

This section introduces general uncertainty concepts, how to model uncertainty in NN, and a definition of suffix and remaining time prediction of BPs.

2.1 Uncertainty in Machine Learning

ML distinguishes epistemic and aleatoric uncertainties. [11] defines and describes both types of uncertainties. Epistemic uncertainty is referred to as "uncertainty due to a lack of knowledge about the perfect predictor" [11] and is reducible. Epistemic uncertainty can be further divided into approximation uncertainty and model uncertainty. Approximation uncertainty refers to a lack of data for selecting appropriate parameters for a predictor model and can generally be reduced by obtaining more training samples. Model uncertainty refers to a model's insufficient approximation capabilities and can be reduced by training models with a higher capacity. There is ongoing debate regarding how epistemic uncertainty should be captured, with one possibility being the use of probability distributions [11]. Aleatoric uncertainty is irreducible as it stems from inherently random effects in the underlying data. Aleatoric uncertainty is "appropriately modeled in terms of probability distributions" [11] and can henceforth be learned in a probabilistic model.

Uncertainty-Aware Neural Networks (NN). For NNs, two common approaches for estimating a model's uncertainty in its prediction are Bayesian approximation and ensemble learning-based techniques [1]. Bayesian approximation can be conducted with Bayesian Neural Networks (BNNs). BNNs assume their weights follow probability distributions, allowing a posterior distribution to be inferred, which can be used to quantify uncertainty. In most cases, obtaining an analytical solution for the posterior distribution is intractable due to neural networks' high non-linearity and dimensionality. Even techniques for approximating the posterior distribution, such as Markov Chain Monte Carlo or Variational Inference (VI) methods, can still be computationally expensive [1, 7]. Ensemble techniques, on the other hand, achieve uncertainty quantification by aggregating the predictions of multiple models. This can also become computationally expensive, especially when numerous complex models are involved [1].

Epistemic Uncertainty using Dropout as a Bayesian Approximation. Using Dropout during training and inference at every weight layer in an NN can be a simple and computationally efficient variational inference method for Bayesian approximation of a posterior distribution [7]. This approach is referred to as *Monte Carlo (MC) dropout* because the posterior distribution $p(W|X,Y)$ is approximated with a variational distribution $q_\theta(W)$. Masked weights are sampled from the variational distribution $\hat{W} \sim q_\theta(W)$, where θ denotes the set of the variational distribution's parameters (weights and bias terms) to be optimized. In practice, a dropout mask is often sampled from a Bernoulli distribution $z \sim \text{Bernoulli(p)}$, where p denotes the dropout probability. The dropout mask is then applied on the NN's weight matrices W such that $\hat{W} = W\,\text{diag}(z)$. During training, the L_2 regularization on the NN parameters θ ensures the method aligns with a probabilistic framework.

Heteroscedastic Aleatoric Uncertainty as Learned Loss Attenuation. Heteroscedastic models assume that observation noise, which follows a probability distribution, can vary with the input data x. This input-dependent observation noise, denoted as $\sigma(x)$, captures aleatoric uncertainty arising from inherent randomness in the data-generating process. To explicitly model this irreducible uncertainty, NNs are extended to directly learn $\sigma(x)$. This is commonly done assuming that the observation noise follows a Normal distribution. To learn this observation noise using an NN $f^W(\cdot)$ parameterized by weights W, an additional output neuron $f_\sigma^W(x)$ is added to the (mean) output neuron $f_y^W(x)$. However, in cases where both the inputs x and the predicted outputs $f_y^W(x)$ are constrained to be strictly positive (e.g., time durations), assuming that the observation noise follows a Log-Normal distribution may be more appropriate. This is equivalent to assuming that the observation noise is usually distributed over the input data in the log-transformed space, i.e., $\ln(x)$. Training the standard deviation of a probability distribution by including it in the loss function is referred to as learned loss attenuation [12]. In the regression case, the adapted loss function over N training samples with target y can be written as the negative log-likelihood of the underlying probability density function[1]:

$$\mathcal{L}_{con} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left(\frac{(y_i - f_y^W(x_i))^2}{f_\sigma^W(x_i)^2} + \log(f_\sigma^W(x_i)^2)\right) \tag{1}$$

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 [12]. Therefore, means and variances can also be learned on the predicted logits. Since the logits are passed in the Softmax function, MC integration has to be applied, i.e., averaging the cross-entropy loss of multiple draws from the logits distributions. We denote the number of MC trials with T, the categorical classes with C, and the ground truth class with c: $\hat{z}_{i,t} = f_y^W(x_i) + f_\sigma^W(x_i)\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$.

$$\mathcal{L}_{cat} = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{1}{T}\sum_{t=1}^{T}\left(\exp(\hat{z}_{i,t,c} - \log(\sum_{c'}^{C}(\exp(\hat{z}_{i,t,c'}))))\right)\right) \tag{2}$$

The combination of epistemic uncertainty quantification using MC dropout and aleatoric uncertainty quantification via learned loss attenuation was first proposed by [12], and this approach can be applied to any NN architecture.

2.2 Suffix Prediction

We define an event log $EL := \{t^{(1)}, t^{(2)}, ..., t^{(L)}\}$ as a set of cases, where L denotes the total number of cases. A case is a sequence of events denoted by $t^{(l)} := \langle e_1, e_2, ..., e_M \rangle$, where M is the number of events in case l. An event is a tuple of event attributes, denoted $e_m := (a_m, t_m, (d_{m_1}, ..., d_{m_k}))$. In this work, we assume that an event has at least two attributes: i) An event label a_m, which

[1]For a detailed derivation of the loss function, see [2].

links the event to a class of event types, and ii) a timestamp attribute t_m, which expresses the time an event happened. Additional event attributes are denoted as $(d_{m_1}, ..., d_{m_k})$. We assume that event attributes are either categorical or continuous. A case can be split into several prefix and suffix pairs. A prefix is defined as $p_{\leq k} := \langle e_1, e_2, ..., e_k \rangle$, with $1 \leq k < M$. A suffix is defined as $s_{>k} := \langle e_{k+1}, \ldots, e_M \rangle$. Suffix prediction involves predicting a suffix $\hat{s}$ based on an input prefix $p_{\leq k}$. The remaining time of a case, given a prefix $p_{\leq k}$, can be defined as $t := \sum_{j=1}^{M} t_{k+j}$ which represents the sum of the durations of all events in the suffix $s_{>k}$ until case completion.

3 Probabilistic Suffix Prediction Framework

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

3.1 Uncertainty-Aware Encoder-Decoder LSTM

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

Data Pre-processing 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. For instance, 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. Missing values for continuous event attributes are encoded as 0. For all encoder, decoder input, and decoder output continuous event attributes, when assuming that the observation noise over these attributes (as modeled through learned loss attenuation) follows a Normal distribution, we apply standard scaling, excluding the raw timestamp. For all decoder input and output continuous event attributes, when assuming that the observation noise follows a Log-Normal distribution, we first transform the attributes into log-space by applying the natural logarithm function as $\ln(1+x)$, ensuring that only positive values are passed to the logarithm. After this step, we apply standard scaling to the log-transformed values. Following [28], 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. This allows multiple prefixes, regardless of the actual prefix length, to be concatenated into a single batch tensor.

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. For every event attribute with K unique category classes, we add an additional NaN class and an

unknown class (a category class not present in the training data). 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 heuristic (see [28]).

Model Architecture. The U-ED-LSTM employs an encoder-decoder architecture of LSTMs [10] same as in [25]. LSTMs are well-suited for handling sequential data and have been proven effective for suffix prediction [4]. 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 activity and time (see [15, 25]). Since the encoder-decoder LSTM is aware of epistemic uncertainty, both LSTMs consist of stochastic LSTM cells, which are stochastic since they apply MC dropout for Bayesian approximation adopted from the [27]. Additionally, to enable the encoder-decoder LSTM to model aleatoric uncertainty, each output is represented by two neurons instead of a single one: one neuron predicts the mean (or logit), and the other predicts the associated standard deviation. Fig. 1 illustrates the U-ED-LSTM architecture with two-layer LSTM cells and one fully connected (FC) layer with two output neurons. [8] have proposed a different dropout variant in which the same dropout mask is applied across all time steps in an RNN, since naive dropout has been shown to be ineffective in Recurrent NNs. This variant is called variational (MC) dropout which is applied during training to the U-ED-LSTM. This is illustrated by the colored arrows in Fig. 1, where identical colors indicate the use of the same MC dropout mask across time steps.

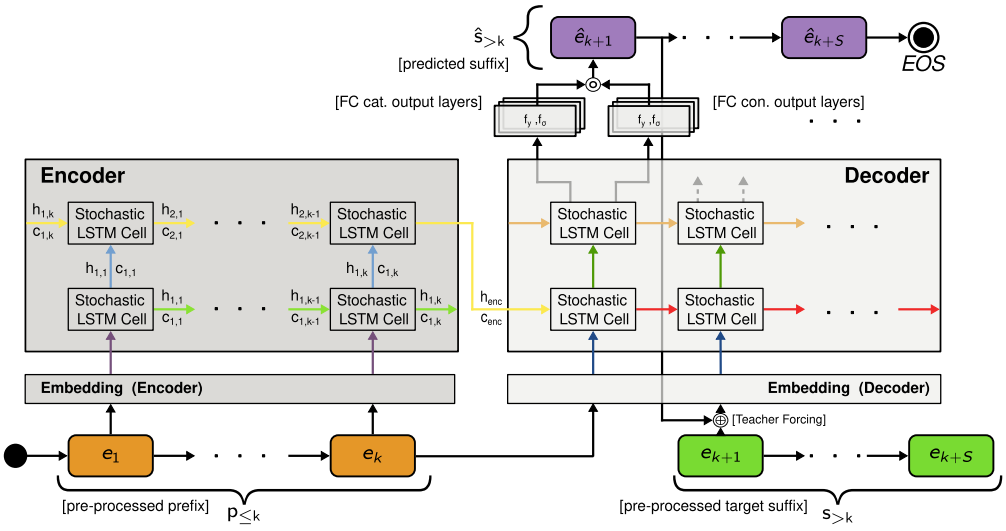

Figure 1: U-ED-LSTM Architecture and Training Pipeline

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 the latent vector tuple from the encoder along with the last event from the prefix. 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 event from the target suffix or the last predicted event based on a predefined probability. Then the decoder autoregressively predicts S events. For the event log attribute, the decoder has a fully connected output layer. A predicted event consists of the concatenation of all its predicted event attributes. Similar to the encoder, we sample the decoder's masked weights from its variational distribution $\hat{W}_{dec} \sim q_{\theta_{dec}}(W_{dec})$. For a given time step $s = 0, 1, \ldots, S-1$, where e_{k+s} denotes the current event and (h_{k+s}, c_{k+s}) the current latent vector tuple, the next event and updated latent vector tuple is predicted as follows: $f^{\hat{W}_{dec}}(e_{k+s}, (h_{k+s}, c_{k+s})) = (\hat{e}_{k+(s+1)}, (h_{k+(s+1)}, c_{k(s+1)}))$.

Loss Functions. To train the U-ED-LSTM, we use two distinct attenuated loss functions, one for continuous and another for categorical event attributes. The loss is calculated for a batch of N prefix-suffix pairs, $\{p^{(i)}_{\leq k}, s^{(i)}_{>k}\}_{i=1}^{N}$, where each predicted suffix has a fixed sequence length S.

For continuous event attributes, the decoder predicts a mean value $\hat{y} := f^{\hat{W}_{dec}}_{con_y}$ and the log-variance $\hat{v} := log(\hat{\sigma}^2) := f^{\hat{W}_{dec}}_{con_\sigma}$, which is common in practice for numerical stability (see [12]). The loss function based on Eq. 1 is implemented as follows:

$$\begin{aligned}\mathcal{L}_{con} &= \frac{1}{N \times S} \sum_{i=1}^{N} \sum_{s=0}^{S-1} \frac{1}{2} \left(\frac{(y^{(i)}_{k+(s+1)} - \hat{y}^{(i)}_{k+(s+1)})^2}{\hat{\sigma}^{2(i)}_{k+(s+1)}} + \log(\hat{\sigma}^{2(i)}_{k+(s+1)}) \right), \\ &= \frac{1}{N \times S} \sum_{i=1}^{N} \sum_{s=0}^{S-1} \frac{1}{2} \left(\exp(-\hat{v}^{(i)}_{k+(s+1)})(y^{(i)}_{k+(s+1)} - \hat{y}^{(i)}_{k+(s+1)})^2 + \hat{v}^{(i)}_{k+(s+1)} \right)\end{aligned} \tag{3}$$

For categorical event attributes, the decoder predicts a mean logit vector with a logit value for each category class $\hat{l} := f^{\hat{W}_{dec}}_{cat_y}$ and the variance vector with a variance for each logit value $log(\hat{\sigma}^2) := f^{\hat{W}_{dec}}_{cat_\sigma}$. Then we apply MC integration and average the coss-entropy loss (CEL) of multiple draws from the logits distribution: $\hat{z} = \hat{l} + \hat{\sigma}\, \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, I)$. The loss function based on Eq. 2 is implemented as follows:

$$\begin{aligned}\mathcal{L}_{cat} &= \frac{1}{N \times S} \sum_{i=1}^{N} \sum_{s=0}^{S-1} -\log \left(\frac{1}{T} \sum_{t=1}^{T} \left(\exp(\hat{z}^{(i)}_{k+(s+1),t,c} - \log \sum_{c'}^{C} \exp(\hat{z}^{(i)}_{k+(s+1),t,c'}) \right) \right), \\ &= \frac{1}{N \times S} \sum_{i=1}^{N} \sum_{s=0}^{S-1} \left(\frac{1}{T} \sum_{t=1}^{T} \mathrm{CEL}(y^{(i)}_{k+(s+1)}, \hat{z}^{(i)}_{k+(s+1),t}) \right)\end{aligned} \tag{4}$$

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

$$\mathcal{L}_{total}(\theta_{enc}, \theta_{dec}) = \sum w_{con} \mathcal{L}_{con} + \sum w_{cat} \mathcal{L}_{cat} + \lambda(\|\theta_{enc}\|_2^2 + \|\theta_{dec}\|_2^2) \tag{5}$$

3.2 MC Suffix Sampling Algorithm

The MC Suffix Sampling Alg. 1, which is similar to other MC sampling approaches for sequence predictions [23, 29], approximates a posterior distribution of suffixes. In particular, Alg. 1 uses MC dropout as a Bayesian approximation to sample epistemic uncertainty, similar to [29], and draws samples from probability distributions learned via loss attenuation, following the approach of [23]. A suffix is sampled in each MC trial. Similar to [29], we employ variational MC dropout on the encoder. First, the prefix is passed into the encoder to obtain a latent vector tuple. Then, the decoder samples a suffix auto-regressively. Unlike during training, naive MC dropout is applied to the decoder during inference, as each event is predicted and used to sample the resulting event individually. Event attributes are sampled differently depending on whether they are continuous or categorical. For continuous event attributes, the event attribute values are directly drawn from a Normal distribution with the predicted means and variances. For categorical event attributes, the logit values are first drawn from Normal distributions and then passed through a Softmax function to obtain a categorical distribution. In a subsequent step, a value for the categorical attribute is drawn from this distribution. The auto-regressive 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.

Algorithm 1 MC Suffix Sampling - Probabilistic Suffix Prediction

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, $p_{\leq k} = \langle e_1, e_2, ..., e_k \rangle$: prefix

function MCSuffixSampling(T, M, p, $p_{\leq k}$)
  $\tilde{S} \leftarrow \emptyset$ ▷ Set of MC sampled probabilistic suffixes.
  for $t = 1$ to T **do**
    $\tilde{s}_{>k} \leftarrow \langle\rangle$ ▷ Sampled events of one probabilistic suffix.
    $\hat{W}_{enc} \leftarrow$ VariationalDropout(W_{enc}, p)
    $(h_{enc}, c_{enc}) \leftarrow f_{enc}^{\hat{W}_{enc}}(p_{\leq k})$
    $\tilde{e} \leftarrow e_k$
    $i \leftarrow 1$
    repeat
      $\hat{W}_{dec} \leftarrow$ NaiveDropout(W_{dec}, p)
      $(\hat{y},\ \hat{\sigma}^2), (\hat{l},\ \hat{\sigma}^2), (h, c) = f_{dec}^{\hat{W}_{dec}}(\tilde{e}, (h_{enc}, c_{enc}))$
      $\tilde{y}_{con} \sim \mathcal{N}(\hat{y}, \hat{\sigma}^2)$
      for $j = 1$ to $|\hat{l}|$ **do**
        $\tilde{v}_j \sim$ Categorical(Softmax($\hat{l}_j$))
        $\tilde{y}_{\text{cat}}^{(j)} \leftarrow \tilde{v}_j$
      end for
      $\tilde{e} \leftarrow \tilde{y}_{con} \cup \tilde{y}_{cat}$
      $\tilde{s}_{>k} \leftarrow \tilde{s}_{>k} \circ \tilde{e}$
      $i \leftarrow i + 1$
    until $i = M$ or GetActivity($\tilde{e}$) $= EOS$
    $\tilde{S} \leftarrow \tilde{S} \cup \{\tilde{s}_{>k}\}$
  end for
  return $\tilde{S}$
end function

4 Evaluation

The evaluation presents the predictive performance and calibration results of the U-ED-LSTM under three different hyperparameter settings, using standard suffix prediction and calibration metrics on four real-life datasets. We compare the most likely prediction generated by the U-ED-LSTM with the aggregated mean prediction of all sampled suffixes. This comparison demonstrates that, in certain cases, the aggregated mean prediction exceeds that of the most likely prediction. Additionally, we report predictive performance results from other ED-LSTMS and Transformers used for suffix prediction from existing literature, evaluated on the same datasets. Since the reported models were not re-implemented, the comparison is not meant to be direct but to provide an intuition of how other approaches perform, demonstrating that the U-ED-LSTM achieves reasonable predictive performance. Furthermore, we assessed the calibration of the U-ED-LSTM to show that the model can capture uncertainties in the event logs. Our implementation and evaluation are publicly available. [2]

Datasets. We evaluated the U-ED-LSTM predictive performance on four real-life data sets. The Helpdesk[3] dataset is an event log from a ticket management system from an Italian software company. The Sepsis[4] dataset represents the pathway of patients diagnosed with Sepsis through a hospital. The BPIC-2017[5] dataset is a loan application process from a Dutch bank and has been investigated in the Business Process Intelligence Competition (BPIC) 2017. The PCR[6] dataset contains process logs from laboratory SARS-CoV-2 RT-PCR tests over one year. Properties for each dataset are presented in Tab. 1. The datasets were split at the case level into training and testing sets using an 80%-20% ratio, following the approach in [13]. The training set was further divided into a training and validation subset: 65% of the original (80% training) data was used for model training, and 15% was used for validation during training, resulting in a 65%-15%-20% training-validation-testing data split. This splitting of the training data into training and validation data is also common in other suffix prediction approaches [15, 25, 28].

Table 1: Dataset Properties

Dataset	Cases	Events	Variants	Activities	Mean–SD Case Length	Mean–SD Case Duration	Cat. Event Attr.	Con. Event Attr.
Helpdesk	4580	21 348	226	14	4.66 – 1.18	40.86 – 8.39 (days)	12	4
Sepsis	1049	15 214	845	16	14.48 – 11.47	28.48 – 60.54 (days)	26	8
BPIC17	31 509	1 202 267	15 930	26	38.16 – 16.72	21.90 – 13.17 (days)	9	9
PCR	6166	117 703	1213	8	19.09 – 3.37	19 872 – 27 864 (sec.)	2	4

Training and Sampling. We trained our U-ED-LSTM model on an NVIDIA GTX 4090 GPU. The implementation allows users to flexibly select the encoder's input event attributes and the decoder's input and output (prediction) event attributes. Several training optimization techniques were implemented to achieve the best possible predictive performance. During training, the U-ED-LSTM predicts S events for each prefix in the batch, from which the loss is calculated. This

[2] Repository: https://github.com/ProbabilisticSuffixPredictionLab/Probabilistic_Suffix_Prediction_U-ED-LSTM_pub

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

[4] Sepsis: https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

[5] BPIC-17: https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b

[6] PCR: https://doi.org/10.5281/zenodo.11617408

approach optimizes its sequence-to-sequence predictions, similar to, e.g., the Complete Remaining Trace Prediction (CRTP) method [9, 28, 28]. We set $S = 5$, optimizing the U-ED-LSTM for sequence-to-sequence predictions while reducing error propagation in cases where the model predicts incorrect events early in the suffix. However, in the original CRTP method, S is set flexibly since a suffix of the same length as the target suffix is predicted for each prefix. We applied probabilistic teacher forcing, where the last predicted or target suffix event is randomly taken as input for the next event prediction [25]. The initial teacher forcing probability was set to 0.8, meaning that 80% of the input events came from the target suffix. This ratio gradually decreased starting from 20% of the training epochs onward. Since our loss function consists of the sum of multiple attribute losses, referred to as Multi-Task-Learning, we implemented a task-balancing algorithm called GradNorm [5]. GradNorm dynamically adjusts the gradient magnitudes and tunes the weight coefficient vectors w_{con} and w_{cat} after each optimization step based on the relative importance of each event attribute on the overall loss. To balance an appropriate level of Bayesian variational approximation to measure epistemic uncertainty during inference while also maintaining good predictive performance, we set the MC dropout rate to a constant p = 0.1 during training, following the recommendation in [27]. The MC sampling algorithm ran on an AMD Ryzen 9 7950 CPU. For each prefix in our testing datasets, we conducted 1000 MC sampling trials with an MC dropout probability of again p = 0.1 [27]. We allow only the predicted continuous event attribute values to be greater than zero and the *case elapsed time* to increase. When a lower *case elapsed time* value than the one from the previous event is sampled, the value from the last event is taken.

Hyperparameter Settings. Three different U-ED-LSTMs with varying hyperparameter settings were trained and compared for each dataset. In the **first setting**, we trained the U-ED-LSTM with 2-layer encoder and decoder LSTMs, along with a fully-connected (FC) layer in the decoder containing separate mean and variance heads for each output event attribute. We assumed normal distributed continuous event attributes and noise and used all event attributes as input features for the encoder and input and output features for the decoder. In the **second setting**, we trained the U-ED-LSTM with a 4-layer encoder and decoder LSTMs, along with an FC layer in the decoder containing separate mean and variance heads for each output event attribute. We assumed normal distributed continuous event attributes and noise. All event attributes were used as input features for the encoder, while only the activity and time attributes were used as input and output features for the decoder. In the **third setting**, we trained the U-ED-LSTM with a 4-layer encoder and decoder LSTMs, along with an FC layer in the decoder containing separate mean and variance heads for each output event attribute. We assumed log-normal distributed continuous event attributes and noise. All event attributes were used as input features for the encoder, while only the activity and time attributes were used as input and output features for the decoder. The remaining hyperparameters were set as follows: In each hyperparameter setting, the encoder and decoder LSTM have a hidden size of 128. We used the standard Adam optimizer in the first setting, while the more advanced AdamW optimizer was applied in the second and third settings. Learning rates were set, in each setting, the same, to 1×10^{-4} for the smaller Helpdesk and PCR datasets,

1×10^{-5} for the medium-sized Sepsis dataset, and 1×10^{-6} for the larger BPIC-17 dataset. After extensive testing, we observed that larger datasets benefited from lower learning rates. A batch size of 128 was used for all datasets except for the BPIC-17, which required a batch size 256 during training. Although a batch size of 128 might also produce good results for BPIC-17, computational constraints necessitated using a larger batch size to reduce training time. All models were trained for 200 epochs in the first and 100 epochs in the second and third settings, without early stopping but with continuous monitoring of validation set performance. Across all settings and datasets, the learning curves decreased during the first 100 epochs and stagnated. Based on this observation, and to reduce training time, we consistently train models for 100 epochs in the second and third settings. Additionally, we applied a L^2 regularization parameter of $\lambda = 1 \times 10^{-4}$. Table 2 summarizes the hyperparameters in each setting.

Table 2: Hyperparameter Settings

Hyperparameter	**Setting 1**	**Setting 2**	**Setting 3**
U-ED-LSTM layers	2	4	4
FC layers in decoder	1	1	1
Assumed distribution for con. event attr.	Normal	Normal	Log-Normal
Encoder features	All	All	All
Decoder features	All	Activity & Time	Activity & Time
Hidden size	128	128	128
Optimizer	Adam	AdamW	AdamW
Learning rate	1×10^{-4} – 1×10^{-6}	1×10^{-4} – 1×10^{-6}	1×10^{-4} – 1×10^{-6}
Batch size	128 (BPIC17 256)	128 (BPIC17 256)	128 (BPIC17 256)
Epochs	200	100	100
MC Dropout Probability (Train/ Test)	0.1	0.1	0.1
Weight-Decay/ Regularization	1×10^{-4}	1×10^{-4}	1×10^{-4}

4.1 Predictive Performance

The evaluation demonstrates the predictive performance of our U-ED-LSTM under different hyperparameter settings across multiple datasets. Therefore, the most likely prediction generated by the U-ED-LSTM is compared with the aggregated mean prediction of all sampled suffixes from the probabilistic approach. The most likely suffix prediction is obtained by auto-regressively sampling the most probable event label at each step until the *EOS* token is reached, following the approach used in previous works [3, 9, 15, 24, 25, 28]. Additionally, we report results from other models in the literature without re-implementation and with different hyperparameter settings. These results are not intended for direct performance comparison but rather to indicate that the predictive performance of our models is reasonable. In future work, we plan to re-implement the approaches proposed in the literature to enable a thoroughly reliable and transparent comparison of results using the best-performing hyperparameter settings.

Metrics. We adopted three commonly used evaluation metrics for suffix prediction, the *Damerau-Levenshtein Similarity* (DLS) metric to assess the activity sequence prediction[3, 18, 21, 25, 28] , the *Mean Average Error* (MAE) of the remaining time predictions [3, 24, 25, 28], and the *Mean Average Error (MAE)* of the suffix length prediction. Here, we used a holdout test set, did not prune the datasets, started the evaluation from a prefix length of 1, i.e., $p_{\leq k}, \forall k \geq 1$, and implemented

the metrics in the following manner: The DLS on the event labels is defined as a normalized DLS distance $\text{DLS}(\hat{s}, s) := 1 - \frac{\text{DL}(s,\hat{s})}{\max(|s|,|\hat{s}|)}$ where s and $\hat{s}$ denote the actual 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. We obtain the DLS for the most likely suffix prediction by comparing the predicted suffix with the ground truth suffix. For the probabilistic suffix prediction, we calculate the DLS for all MC samples and take the mean. Since we preprocessed the event logs by adding the *case elapsed time* and a *event elapsed time* attributes to the event logs, we can obtain a remaining time prediction in two ways: By summing the predicted *event elapsed times* of a case or by taking the *case elapsed time* value of the last event in a case. We implemented both. We calculated the mean remaining time for the probabilistic suffix prediction and compared that mean aggregation with the ground truth value. Other works [3, 24] observed that some suffix prediction approaches fail at predicting the right suffix length. Therefore, we also measured the MAE between the true and the predicted suffix length, the *suffix length MAE*. We took the mean suffix length of the MC samples for probabilistic suffix prediction.

Predictive Performance Results. The results of the most likely and aggregated mean of all sampled suffixes from the probabilistic approach using the U-ED-LSTM with different hyperparameter settings, as well as results from existing ED-LSTM and Transformer-based approaches from the literature, are presented in Tab. 3 for suffix length and suffix event label predictions, and in Table 4 for remaining time predictions. The results, detailed by prefix and suffix lengths, for each hyperparameter setting, comparing the most likely predictions to the aggregated mean of all sampled suffixes from the probabilistic approach, are shown in Figure 2 for **Setting 1**, Figure 3 for **Setting 2**, and Figure 4 for **Setting 3**.

As existing models from the literature and their predictive performance results, the following are selected: MM-Pred from [15], an ED-LSTM model evaluated on the Helpdesk and BPIC-17 datasets. The authors used a 70%-10%-20% train-validation-test split. The model consists of a 2-layer encoder and decoder LSTMs with a hidden size 32 and a dropout rate of 0.2 for regularization during training. Additionally, MM-Pred includes a component called the "Modulator", which learns the significance of each event label and lifecycle transition attribute and passes this information as an additional feature to the decoder. ED-LSTM-GAN from [25] was evaluated on the Helpdesk and BPIC-17 datasets. The authors used a 70%-10%-20% train-validation-test split. The model consists of a 5-layer encoder and decoder LSTMs with a hidden size 32 as a generator and an FC layer as the discriminator as part of the GAN architecture. Training is conducted for 500 epochs, with early stopping applied after 30 epochs. RMSprop is the optimizer, with a 5×10^{-5} learning rate. Probabilistic teacher forcing is used with a ratio of 0.1. AE (inspired by [15, 25]) and AE-GAN (inspired by [25]) from [13] were evaluated on the Helpdesk, Sepsis, and BPIC-17 datasets. An 80%-20% train-test split was used. Both models consist of 4-layer encoder and decoder LSTMs with a hidden size of 128 and are trained for 400 epochs, with early stopping applied after 50 epochs. The Adam optimizer is used with a learning rate of 1×10^{-4}, and a dropout rate of 0.3 is applied for regularization. SuTraN, along with an ED-LSTM re-implementation of [25], from

[28], was evaluated on the BPIC-17 dataset. The authors used a 55%-35%-25% train-validation-test split. The ED-LSTM architecture consists of a 4-layer encoder and decoder LSTMs with a hidden size 64. A batch size of 128 is used, and models are trained for 200 epochs with early stopping based on the validation set performance. The Adam optimizer is applied with an initial learning rate of 2×10^{-4}, adjusted dynamically via an exponential learning rate scheduler with a decay factor of 0.96. A 1×10^{-4} weight decay is used, and teacher forcing is employed during training. Similarly to [4], we observe that results from other works are difficult to compare directly, as they often differ not only in hyperparameter settings but also in several different aspects, such as varying data preprocessing strategies (e.g., pruning of event labels), the exclusion of short prefix lengths, differences in evaluation methodology (e.g., holdout test sets versus cross-validation), and inconsistent metric implementations (e.g., comparing the predicted suffix to a single ground truth versus a set of possible suffixes). However, we report the predictive performance results from the abovementioned models to provide an intuition of whether the U-ED-LSTM demonstrates reasonable and competitive predictive performance.

Based on the results for the suffix length MAE and the DLS of suffix event labels reported in Table 3, it can be concluded that the second hyperparameter setting yielded the best performance for the U-ED-LSTM on the Helpdesk and BPIC-17 datasets, especially for the results of the probabilistic approach. Notably, for the Helpdesk dataset, the DLS for suffix event label prediction improved significantly, from 0.53 to 0.82 for the most likely prediction and from 0.44 to 0.65 for the aggregated mean of sampled suffixes. These results are comparable to those reported in the literature, where DLS values for most likely predictions range from 0.84 (-0.02) (ED-LSTM-GAN [25]) to 0.87 (-0.05) (MM-Pred [15]). Similar improvements are observed in the MAE of suffix length predictions for the Helpdesk dataset when comparing Setting 1 to Setting 2. One possible reason Setting 2 provided the best results on the Helpdesk dataset could be the deeper architecture with a 4-layer encoder and decoder LSTMs. This enhanced the U-ED-LSTM's ability to abstract higher-level patterns, capture more complex dependencies, and generalize better to longer suffixes. This is particularly evident when comparing the DLS for suffix event label prediction at a rare suffix length of 10: In Setting 2 (Fig. 3), the DLS is 0.4, representing a 100% improvement over Setting 1 (Figure 2), which is 0.2. Moreover, comparing Setting 2 with Setting 3 on all datasets instead of some results from the Sepsis dataset, it can be seen that using a normal distribution compared to a log-normal distribution for loss attenuation also outperformed the predictive performance for the suffix length and suffix event label prediction. The model's learning process was more stable in the normal distribution settings, resulting in a smoother convergence across all output event attributes during training. The third setting yielded the best suffix length MAE and the best DLS for suffix event label predictions of the U-ED-LSTM on the Sepsis dataset, achieving a DLS of 0.18 based on the aggregated mean of all sampled suffixes from the probabilistic approach, surprisingly, outperforming the most likely prediction. However, while this is still a relatively poor result, it is comparable to those reported in the literature, such as 0.14 (+0.04) for AE-GAN [13] and 0.22 (-0.04) for AE [13]. These results suggest that suffix prediction for the Sepsis dataset is challenging,

likely due to high variability and many unique cases that do not follow a general pattern. In the Sepsis dataset, wide interquantile ranges (IQRs) can be observed across all settings in Figure 2, Figure 3, and Figure 4. The DLS of the aggregated mean consistently falls near the center of the distribution, probably indicating that a diverse set of sampled suffixes with different suffix lengths contribute equally to DLS. For the BPIC-17 dataset, the probabilistic approach achieved the best results regarding suffix length MAE and DLS for suffix event labels in Setting 2, even outperforming the most likely prediction. This improvement can be attributed to the potentially long suffixes present in BPIC-17 cases. The probabilistic method performs better in predicting suffix lengths, contributing to a lower DLS score due to more accurate sampling of the EOS token. This result demonstrates the advantage of using a probabilistic approach for predicting long suffixes, where uncertainty and variability naturally increase with each additional (false) predicted event. For the PCR dataset, the first setting provided the best performance of the U-ED-LSTM in terms of both suffix length MAE and DLS for suffix event label predictions. One possible reason why the 2-layer setting outperformed the 4-layer could be the relatively small size and low complexity of the PCR dataset. The PCR dataset represents a highly automated process managed by a workflow engine, resulting in less variability than the other datasets. Since the most likely suffix was obtained

Table 3: Predictive Performance (Categorical): Suffix Length MAE and Suffix Event Labels DLS

Method	**Suffix Length MAE**				**Suffix Event Labels DLS**			
	Helpdesk	Sepsis	BPIC17	PCR	Helpdesk	Sepsis	BPIC17	PCR
Own Results								
Most likely - Setting 1	0.96	27.59	13.74	**1.48**	0.53	0.1	**0.35**	**0.83**
Probabilistic - Setting 1	0.74	6.84	14.29	1.98	0.44	0.14	0.28	0.59
Most likely - Setting 2	**0.36**	8.8	40.83	3.49	**0.82**	0.11	0.21	0.67
Probabilistic - Setting 2	0.38	6.83	**11.42**	3.63	0.65	0.12	0.31	0.54
Most likely - Setting 3	0.54	26.96	40.78	4.37	**0.82**	0.09	0.16	0.62
Probabilistic - Setting 3	0.53	**6.16**	33.71	4.43	0.52	**0.18**	0.2	0.54
ED-LSTM from Lit.								
MM-Pred [15]	-	-	-	-	0.87	-	0.3	-
ED-LSTM-GAN [25]	-	-	-	-	0.84	-	0.34	-
AE [13]	-	-	-	-	0.86	0.22	0.14	-
AE-GAN [13]	-	-	-	-	0.86	0.14	0.07	-
ED-LSTM [28]	-	-	-	-	-	-	0.32	-
Transformer from Lit.								
SuTraN [28]	-	-	-	-	-	-	0.38	-

from auto-regressively sampling the mode activity with the highest softmax probability, it can be expected to have the best DLS. Interestingly, this is not true for small prefix sizes in the Sepsis and BPIC-17 datasets in Setting 2.

A similar conclusion regarding the optimal hyperparameter setting per dataset for the U-ED-LSTM can be drawn for the continuous event time attributes, as shown in Table 4. Interestingly, the assumed log-normal distribution returned significantly worse results than the normal distribution for all datasets except for the remaining time of the last event MAE in the Sepsis dataset.

This observation suggests that while the log-normal distribution offers certain theoretical benefits (e.g., modeling non-negative values), it may not be suitable for suffix prediction. Our observations indicate that the log-normal distribution performs well when the target values are close to each other (i.e., exhibit low variance) but perform poorly when the distribution is broader. This issue was particularly relevant in our case. Although the remaining times (sum and last) across datasets were measured initially in days (except for the PCR dataset), we trained the model using the logarithm of time values in seconds, followed by standard normalization. Consequently, after destandardizing and exponentiating the outputs for evaluation, the predictions became highly sensitive to sampled outliers, often resulting in extreme and unrealistic values. This outcome is surprising, as we expected the log-normal distribution to provide improved precision due to its ability to model non-negative continuous values, which is the case for time event attributes. Our

Table 4: Predictive Performance (Continuous): Remaining Time MAE

Method	**Remaining Time Sum MAE**				**Remaining Time Last MAE**			
	Helpdesk	Sepsis	BPIC17	PCR (sec.)	Helpdesk	Sepsis	BPIC17	PCR (sec.)
Own Results								
Most likely Setting 1	11.21	38.09	11.76	**159.19**	18.35	29.21	**9.95**	9340.1
Probabilistic Setting 1	**8.74**	31.21	12.45	165.91	14.02	34.3	10.83	19237.2
Most likely Setting 2	11.76	34.5	10.73	170.87 sec.	**9.1**	29.88	10.75	**8871.74**
Probabilistic Setting 2	9.58	**31.18**	**10.62**	170.44	10.99	31.41	14.4	12281.35
Most likely Setting 3	261.64	147.15	10.68	180.18 sec.	553.82	**24.54**	2887.58	411704.04
Probabilistic Setting 3	5789.51	135.05	1387.15	15262002.78	557.32	24.87	4448.17	412893.69
ED-LSTM from Lit.								
MM-Pred [15]	-	-	-	-	-	-	-	-
ED-LSTM-GAN [25]	6.21	-	13.95	-	-	-	-	-
AE [13]	3.83	735.04	69.51	-	-	-	-	-
AE-GAN [13]	3.88	187.12	100.19	-	-	-	-	-
ED-LSTM [28]	-	-	8.44	-	-	-	-	-
Transformer from Lit.								
SuTraN [28]	-	-	5.5	-	-	-	-	-

probabilistic approach can obtain better remaining time (sum) predictions for smaller prefixes and, in total, in the Helpdesk and Sepsis data sets. This might be because suffixes from small prefix lengths have higher uncertainty. Interestingly, the opposite is the case with the BPIC-17 dataset. Since the BPIC-17 dataset has the most extended case length, obtaining only 1000 MC samples might not have been sufficient. We noticed that the algorithm often moves into highly uncertain regions in our MC sampling approach. This usually leads to sampling from high variances, resulting in considerable *event elapsed time* and *case elapsed time* values, inflating the mean aggregations. Furthermore, it can be seen in Fig. 4 that the values of the learned log-normal distributed loss attenuation are sensitive to outlier predictions, which greatly influence the aggregated mean of all sampled suffixes.

Overall, Setting 2 proved to be the best hyperparameter configuration. The probabilistic approach's aggregated mean of all sampled suffixes often outperforms the most likely prediction,

particularly for short prefixes and long suffixes (high variability, high uncertainty), especially for continuous event attribute predictions. Assuming a normal distribution-based loss attenuation for noise on continuous input data yielded better results than a log-normal distribution.

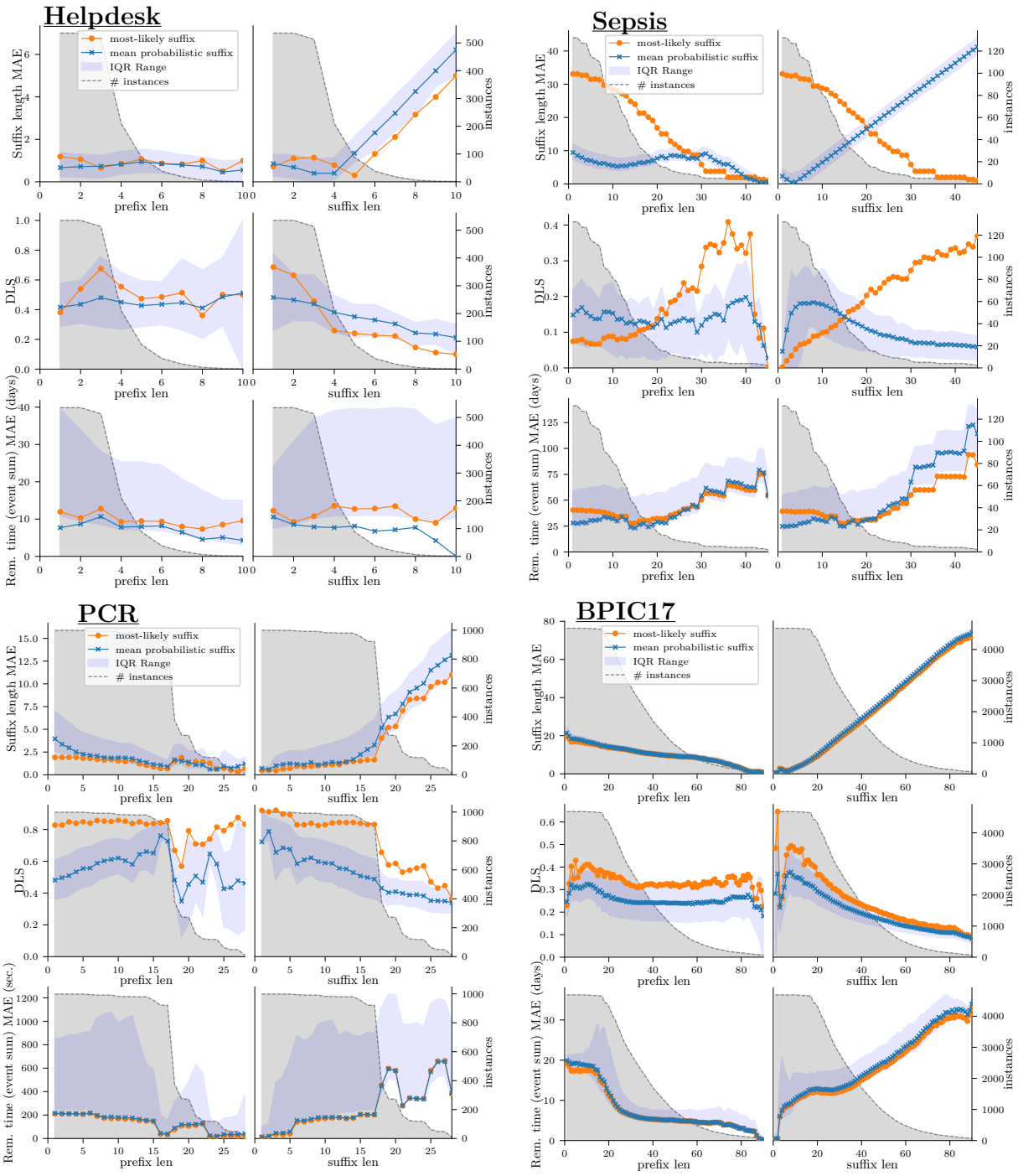

Figure 2: Predictive Performance - Setting 1

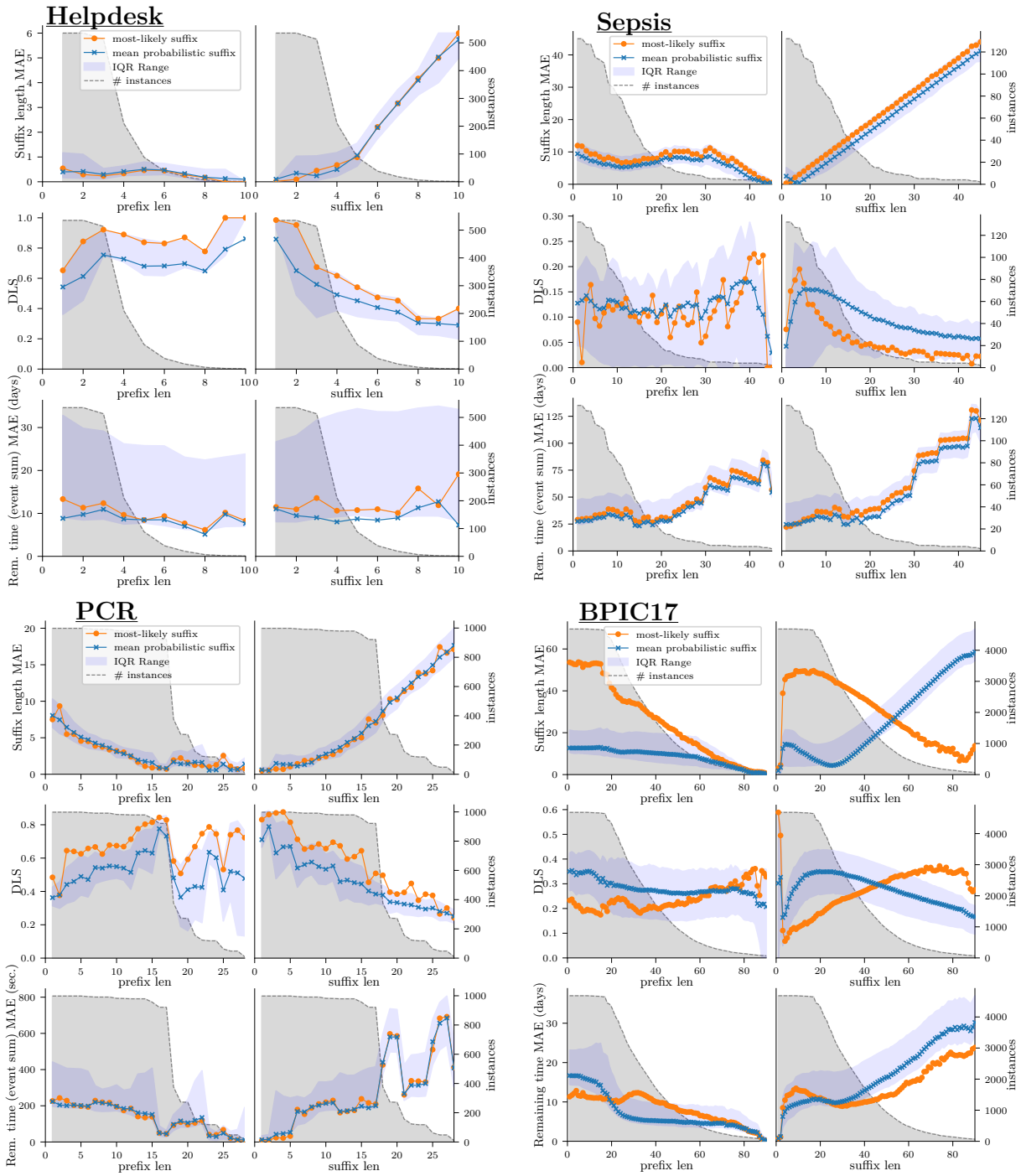

Figure 3: Predictive Performance - Setting 2

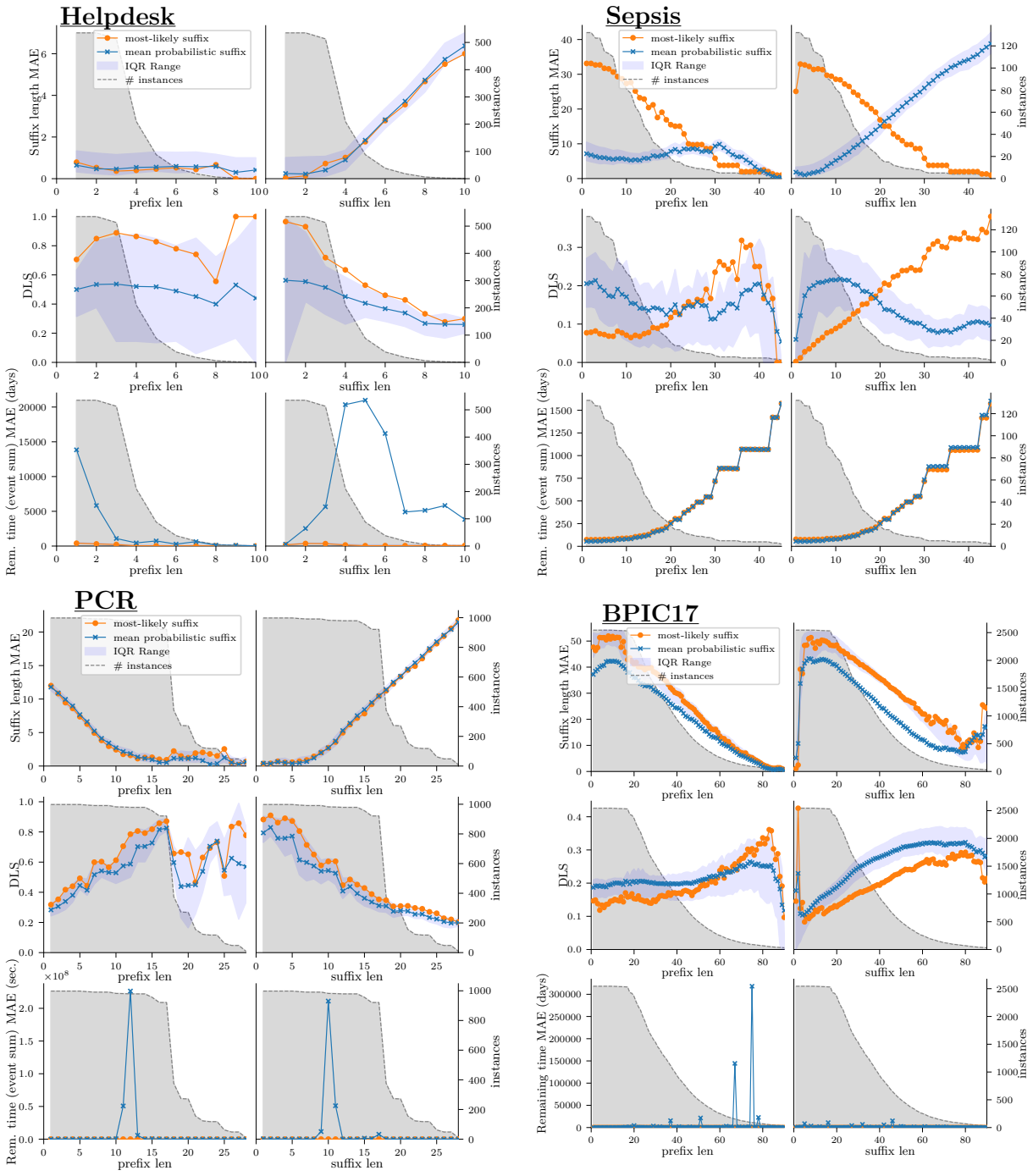

Figure 4: Predictive Performance - Setting 3

4.2 Calibration Results

We evaluated the calibration of the U-ED-LSTM to demonstrate that the model can capture the variability in the respective event logs.

Metrics. To evaluate the calibration of remaining time predictions (sum and last), we used the Probability Integral Transform (PIT). Given the predicted distribution of remaining times gener-

ated by the U-ED-LSTM for each test case, obtained by sampling multiple suffixes, we constructed an empirical cumulative distribution function (CDF) for each prediction. For a given test case i, let $\hat{t}_{i,j}$ be the remaining time prediction of the j-th sampled suffix, and let t_i be the ground-truth remaining time. The normalized PIT value per test case u_i is computed as: $u_i = \frac{1}{T}\sum_{j=1}^{T} \mathbf{1}\{\hat{t}_{i,j} \leq t_i\}$, where $T = 1000$ is the total number of MC samples and $\mathbf{1}\{\cdot\}$ is the indicator function. The calculation is the same for the remaining time sum and last. After computing the set of PIT values $u := \{u_1, \dots u_{D_{test}}\}$, PIT plots were constructed. The x-axis represents the PIT values between $[0, 1]$, and the y-axis shows the probability density. The PIT plots can have different shapes, each indicating different model calibration: A uniform distribution (i.e., a flat line at density 1) indicates perfect calibration. The predicted variance matches the true variability in the ground truth data exactly. A U-shape suggests that the model predicts too little variance, and outliers get underestimated. For instance, if the true variability of remaining times spans ± 5 days around the mean, but the model only predicts a variance of ± 3 days, it will consistently fail to capture outliers. A bell shape indicates that the model predicts too much variance. For instance, if the true variability of remaining times spans ± 5 days around the mean, and the model predicts a much higher variance of ± 10 days, it will consistently predict, over all samples per case, remaining times below and above the ground truth. A slope-shaped PIT plot indicates a systematic bias in the model's predictions. In such cases, no reliable conclusions about the predicted variance can be drawn. The PIT plots of the U-ED-LSTM, across all hyperparameter settings, for all datasets are depicted in Fig. 5 (**Setting 1**), Fig. 6 (**Setting 2**), Fig. 7 (**Setting 3**).

Calibration Results. For the Helpdesk dataset, the model is quite well calibrated for predicting the remaining time for both sum and last in Setting 2, demonstrating its ability to predict a wide range of remaining times across all test cases. In contrast, Setting 1 shows that the U-ED-LSTM tends to predict values with a bit more variance across test cases. However, the model tends to predict smaller remaining time values sum and last compared to the ground truths (values tend to be greater than 0.5 on the x-axis). In Setting 3, a systematic bias (slope at 0) is visible. The model tends to predict values that are too large. Nevertheless, for the Helpdesk dataset, the U-ED-LSTM, especially in Setting 2, can capture the variability in the remaining time (sum and last) from the test dataset quite well. For the Sepsis dataset, the U-ED-LSTM consistently exhibits U-shaped PIT plots across all settings. This suggests that the model underestimates the variance, predicting much less variability than in the ground truth values across all test cases. This contrasts the Helpdesk dataset, where the model tends to overestimate variance. These observations align closely with the predictive performance results and are consistent with the characteristics of the datasets. The Helpdesk dataset contains lower inherent variability and shorter suffix lengths. Therefore, the model tends to predict too much variance, whereas for the Sepsis dataset, which contains high variability, the model underestimates it. Nevertheless, among the evaluated settings, Setting 2 yielded the best calibration for Sepsis. The calibration performance for the BPIC-17 dataset is generally poor. Clear systematic bias is observed in the remaining time predictions (sum and last). Among all configurations, Setting 2 provides the most balanced calibration, although a

minor bias remains. Specifically, in Setting 2, the remaining time sum is slightly underestimated, while the remaining time last is somewhat overestimated. In Setting 1 of the U-ED-LSTM on the PCR dataset, the PIT plot exhibits a bell-shaped distribution for the remaining time sum, indicating that the model's predicted variance is much higher than the true variability of remaining time values. The sloped PIT plot for the remaining time last reveals a systematic bias, with the U-ED-LSTM consistently overestimating the true remaining times last. However, calibration for both remaining time types shows improvement in Setting 2. Overall, the results suggest that in Setting 2, the model's calibration for the PCR dataset aligns reasonably well.

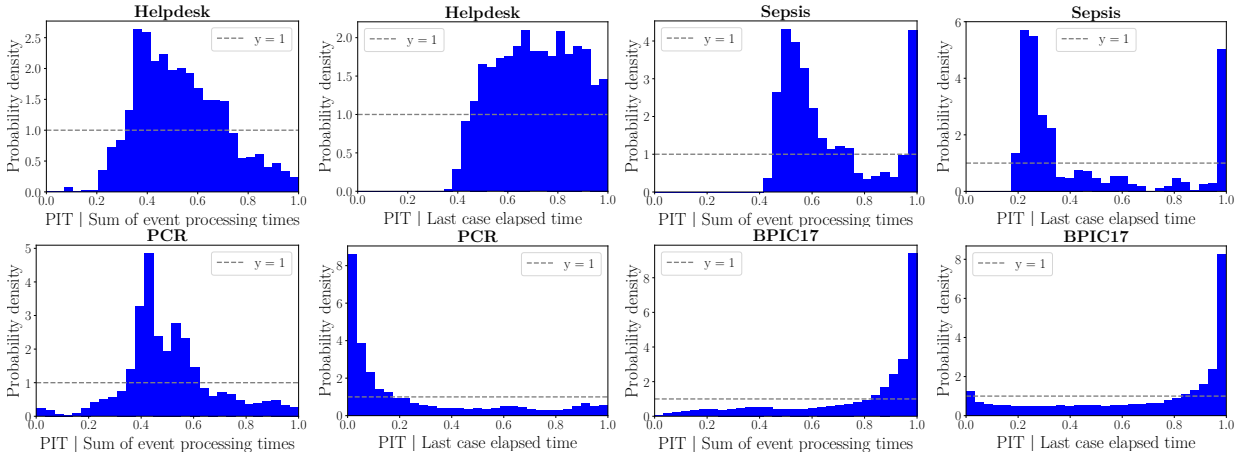

Figure 5: Model Calibration of Remaining Time Predictions - Setting 1

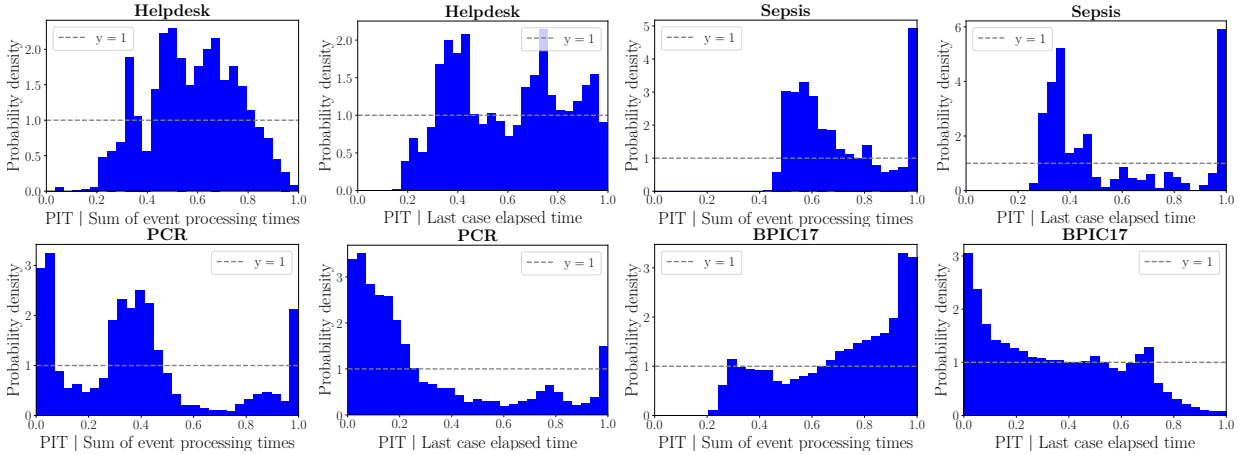

Figure 6: Model Calibration of Remaining Time Predictions - Setting 2

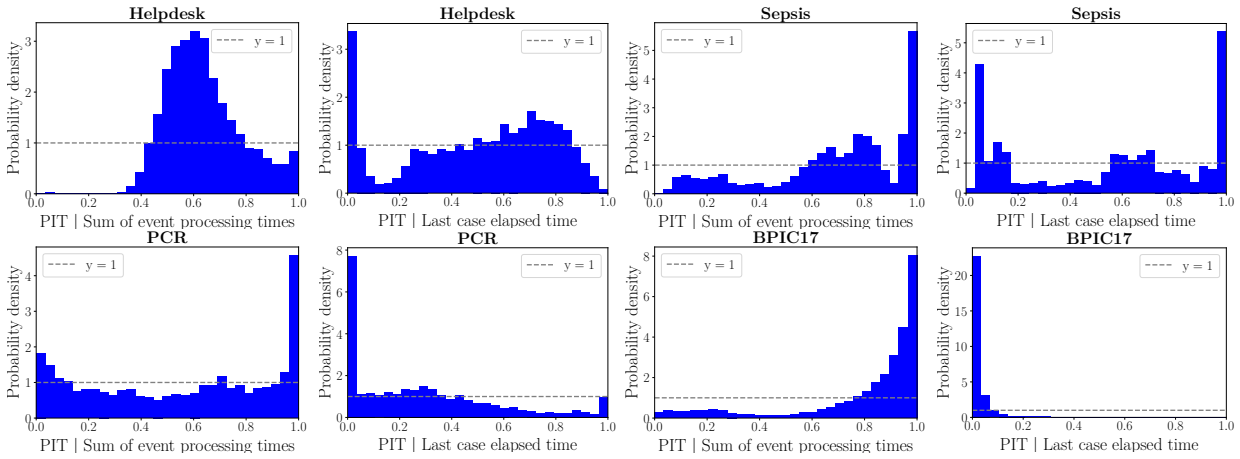

Figure 7: Model Calibration of Remaining Time Predictions - Setting 3

5 Related Work

Suffix Prediction. Current suffix prediction approaches focus on predicting the most likely suffix and improving predictive performance. The methods differ in the models used, predicted event attributes, and strategies to enhance training. Early suffix prediction approaches use LSTMs [3, 6, 9, 24]. Predictive performance is improved by using encoder-decoder LSTMs [15, 25]. More recent encoder-decoders are enriched by more complex NN architectures such as combined General Recurrent Units, Graph NNs, and attention [21] or transformers [28]. Recently, LLMs have been used for suffix prediction [18], 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 [6, 18, 21] and lifecycle transitions [15]. Other approaches predict the sequence of activities and time attributes [9, 24, 25, 28], and resource information [3]. Special training considerations are applied to improve predictive performance. [25], for example, introduce teacher forcing and enhance its encoder-decoder LSTM with adversarial training to improve performance and robustness. [9, 28] proposed CRTP, demonstrating that models trained this way outperform those optimized for single-event prediction. For testing, [3] try random sampling from categorical distributions against an arg-max strategy to derive the best matching activities in a suffix. Similar to our approach, they observe better performance in suffix length prediction.

Uncertainty in PPM. For remaining time and next activity predictions, combined epistemic and aleatoric uncertainty for NNs is applied to PPM by [27]. [20] 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, [17] introduces Conformalized MC dropout, leveraging uncertainty and conformal predictions to construct prediction intervals for the next activity prediction to improve interpretability. However, they do not evaluate their approach on open-source, real-world datasets.

In [19, 22], Bayesian Networks are used to predict the sequence of activities, but Bayesian networks cannot handle large and complex data.

6 Conclusion

In this technical report, we presented an approach for *probabilistic suffix prediction* that leverages our *U-ED-LSTM* and *MC suffix sampling* algorithm and an extensive performance evaluation of the proposed model. Our approach captures epistemic uncertainty via MC dropout and aleatoric uncertainty as learned loss attenuation. No other work has yet addressed incorporating epistemic and aleatoric uncertainties for suffix predictions of business processes. Probabilistic suffix prediction can offer enhanced reliability and transparency by generating a distribution over possible future sequences rather than a single deterministic outcome. For instance, instead of predicting a single remaining time or a fixed number of activity loop executions, the model can provide a range of possible values and their associated probabilities.

Future Work. We demonstrated the predictive performance and calibration of our U-ED-LSTM. However, to further improve the performance and calibration of our approach, we i) further experiment with different hyperparameters, ii) try to improve the log-normal distribution of assumed observation noise for loss attenuation to obtain results such as in [23], iii) try different methods to measure epistemic uncertainty such as deep ensembles instead of MC dropout, iv) choose different NN architectures for sequence predictions, especially for long-range sequences such as transformers. In the evaluation, we only assessed activity sequence and remaining time predictions. Since our approach can predict all event attributes, evaluating additional attributes could yield further insights into the model's predictive performance.

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul W. Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Inf. Fusion*, 76:243–297, 2021. doi: 10.1016/J.INFFUS.2021.05.008.

[2] Christopher M Bishop. Mixture density networks. 1994.

[3] Manuel Camargo, Marlon Dumas, and Oscar González Rojas. Learning accurate LSTM models of business processes. In *Business Process Management - BPM*, pages 286–302, 2019. doi: 10.1007/978-3-030-26619-6_19.

[4] Paolo Ceravolo, Marco Comuzzi, Jochen De Weerdt, Chiara Di Francescomarino, and Fabrizio Maria Maggi. Predictive process monitoring: concepts, challenges, and future research directions. *Process Science*, 1(1):2, 2024.

[5] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning - ICML*, pages 794–803, 2018.

[6] Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. Predicting process behaviour using deep learning. *Decis. Support Syst.*, 100:129–140, 2017. doi: 10.1016/J.DSS.2017.04.003.

[7] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning - ICML*, pages 1050–1059, 2016. URL http://proceedings.mlr.press/v48/gal16.html.

[8] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems - NeurIPS*, pages 1019–1027, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/076a0c97d09cf1a0ec3e19c7f2529f2b-Abstract.html.

[9] Björn Rafn Gunnarsson, Seppe vanden Broucke, and Jochen De Weerdt. A direct data aware LSTM neural network architecture for complete remaining trace and runtime prediction. *IEEE Trans. Serv. Comput.*, 16(4):2330–2342, 2023. doi: 10.1109/TSC.2023.3245726.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[11] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110(3):457–506, 2021. doi: 10.1007/S10994-021-05946-3.

[12] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems - NeurIPS*, pages 5574–5584, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html.

[13] István Ketykó, Felix Mannhardt, Marwan Hassani, and Boudewijn F. van Dongen. What averages do not tell: predicting real life processes with sequential deep learning. In *ACM/SIGAPP Symposium on Applied Computing - SAC*, pages 1128–1131, 2022. doi: 10.1145/3477314.3507179.

[14] Nadja Klein. Distributional regression for data analysis. *Annual Review of Statistics and Its Application*, 11, 2024. doi: 10.1146/annurev-statistics-040722-053607.

[15] Li Lin, Lijie Wen, and Jianmin Wang. Mm-pred: A deep predictive model for multi attribute event sequence. In *SIAM International Conference on Data Mining - SDM*, pages 118–126, 2019. doi: 10.1137/1.9781611975673.14.

[16] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. Predictive monitoring of business processes. In *Advanced Information Systems Engineering - CAiSE*, pages 457–472. Springer, 2014. doi: 10.1007/978-3-319-07881-6_31.

[17] Nijat Mehdiyev, Maxim Majlatow, and Peter Fettke. Augmenting post-hoc explanations for predictive process monitoring with uncertainty quantification via conformalized monte carlo dropout. *Data Knowl. Eng.*, 156:102402, 2025. doi: 10.1016/J.DATAK.2024.102402.

[18] Vincenzo Pasquadibisceglie, Annalisa Appice, and Donato Malerba. LUPIN: A LLM approach for activity suffix prediction in business process event logs. In *International Conference on Process Mining - ICPM*, pages 1–8, 2024. doi: 10.1109/ICPM63005.2024.10680620.

[19] Stephen Pauwels and Toon Calders. Bayesian network based predictions of business processes. In *Business Process Management Forum - BPM Forum*, pages 159–175. Springer, 2020. doi: 10.1007/978-3-030-58638-6_10.

[20] Pietro Portolani, Alessandro Brusaferri, Andrea Ballarino, and Matteo Matteucci. Uncertainty in predictive process monitoring. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems - IPMU*, pages 547–559, 2022. doi: 10.1007/978-3-031-08974-9_44.

[21] Efrén Rama-Maneiro, Juan Carlos Vidal, Manuel Lama, and Pablo Monteagudo-Lago. Exploiting recurrent graph neural networks for suffix prediction in predictive monitoring. *Computing*, 106(9):3085–3111, 2024. doi: 10.1007/S00607-024-01315-9.

[22] Simon Rauch, Christian M. M. Frey, Ludwig Zellner, and Thomas Seidl. Process-aware bayesian networks for sequential event log queries. In *International Conference on Process Mining - ICPM*, pages 161–168, 2024. doi: 10.1109/ICPM63005.2024.10680678.

[23] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36 (3):1181–1191, 2020. doi: 0.1016/j.ijforecast.2019.07.001.

[24] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In *Advanced Information Systems Engineering - CAiSE*, pages 477–492, 2017. doi: 10.1007/978-3-319-59536-8_30.

[25] Farbod Taymouri, Marcello La Rosa, and Sarah M. Erfani. A deep adversarial model for suffix and remaining time prediction of event sequences. In *SIAM International Conference on Data Mining - SDM*, pages 522–530, 2021. doi: 10.1137/1.9781611976700.59.

[26] Ilya Verenich, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Irene Teinemaa. Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *ACM Trans. Intell. Syst. Technol.*, 10(4):34:1–34:34, 2019. doi: 10.1145/3331449.

[27] Hans Weytjens and Jochen De Weerdt. Learning uncertainty with artificial neural networks for predictive process monitoring. *Appl. Soft Comput.*, 125:109134, 2022. doi: 10.1016/J.ASOC.2022.109134.

[28] Brecht Wuyts, Seppe K. L. M. vanden Broucke, and Jochen De Weerdt. Sutran: an encoder-decoder transformer for full-context-aware suffix prediction of business processes. In *International Conference on Process Mining - ICPM*, pages 17–24, 2024. doi: 10.1109/ICPM63005.2024.10680671.

[29] Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at uber. In *IEEE International Conference on Data Mining Workshops - ICDM*, pages 103–110, 2017. doi: 10.1109/ICDMW.2017.19.