A Simple and Calibrated Approach for Uncertainty-Aware Remaining Time Prediction

Keyvan Amiri Elyasi[1][0009−0007−3016−2392], Han van der Aa[2][0000−0002−4200−4937], and Heiner Stuckenschmidt[1][0000−0002−0209−3859]

[1] Data and Web Science Group, University of Mannheim, Germany
{keyvan,heiner}@informatik.uni-mannheim.de
[2] Faculty of Computer Science, University of Vienna, Austria
han.van.der.aa@univie.ac.at

Abstract. Business processes are typically supported by information systems that log execution data, enabling the prediction of remaining time for ongoing process instances. Deep learning models are often used for this task due to their accuracy, but they only provide point estimates, without accounting for uncertainty. This limits their reliability, as decision-making often benefits from prediction intervals. Uncertainty quantification techniques can help by estimating both expected values and uncertainty. However, existing techniques are often poorly calibrated, computationally expensive, or not adaptable to different deep learning models. This paper examines these challenges and proposes a simple, efficient solution using Laplace approximation and calibrated regression. Our approach distinguishes between model and data uncertainty, integrates easily with any deep learning model, and can be applied to pre-trained networks. Benchmarking on 10 real-world event logs shows that our method matches state-of-the-art performance while significantly reducing training and inference time. This makes it a strong yet simple baseline for uncertainty-aware remaining time prediction in business processes.

Keywords: Remaining time prediction · Uncertainty quantification · Predictive process monitoring · Deep learning.

1 Introduction

Predictive process monitoring (PPM) leverages recorded event data to forecast how ongoing instances of a business process will progress to completion. A key task in PPM is predicting the remaining time of these instances to provide reliable estimates for customers and stakeholders. It also enables timely interventions to prevent service level agreement (SLA) violations, and support resource allocation and scheduling decisions [17, 26].

Remaining time prediction is often done using deterministic models that only provide point estimates (i.e., the expected remaining time) without measuring

their own uncertainty. However, decision-makers prefer prediction intervals and need information about uncertainty [27]. For example, in a supply chain setting, it is considerably more helpful to learn that an anticipated delivery of raw materials should occur in the next 7 to 10 days, with 95% confidence, rather than just receiving the information that the expected delivery date is in 8 days. This issue is especially relevant in deep learning models, which, despite their high accuracy [20, 26], operate as black boxes and often overestimate their confidence, making their predictions less reliable for decision-making [15, 28].

To address this, various uncertainty quantification techniques have been proposed for neural networks [1, 8]. The key challenge is to simultaneously estimate both model (epistemic) and data (aleatoric) uncertainty, while providing well-calibrated predictions. Common techniques include Monte Carlo (MC) dropout [7] for epistemic uncertainty and heteroscedastic regression [10] for aleatoric uncertainty, both of which have been applied to PPM tasks like remaining time prediction [27, 28]. Although combining these techniques [10] can estimate both types of uncertainty, they have four key limitations. First, MC dropout requires architectural modifications, limiting its applicability to certain deep learning models [11]. Second, it requires sampling (i.e., multiple forward passes in the neural network), introducing significant computational overhead during inference and validation [14, 18]. Third, uncertainty-aware models are not necessarily more accurate than deterministic ones. Finally, both MC dropout and heteroscedastic regression often produce miscalibrated prediction intervals [11, 12] that fail to capture the true remaining time.

To overcome these limitations, we propose an approach that combines Laplace approximation [6] with calibrated regression [11] in the context of PPM. It transforms a pre-trained deterministic neural network into a Bayesian model, allowing uncertainty estimation without reducing accuracy. The approach has two steps. First, we apply Laplace approximation to estimate epistemic and aleatoric uncertainty while keeping the predicted point estimate from the original model. By restricting Laplace approximation to the final linear layer, our approach remains flexible, allowing process analysts to test different deep learning architectures to find the best fit for their data. In the second step, we use calibrated regression to refine uncertainty estimates. Specifically, we improve the standard calibrated regression technique by introducing a process-aware variant that learns separate recalibration models for ongoing process instances of different lengths. This accounts for variations in model error across different stages of process execution and ensures well-calibrated prediction intervals. We evaluate our approach against state-of-the-art baselines and show that it achieves comparable performance while significantly reducing computational costs. This makes it a simple yet effective baseline for uncertainty-aware remaining time prediction.

The paper is structured as follows: Section 2 motivates probabilistic remaining time prediction, which is formally defined in Section 3. Our proposed approach is detailed in Section 4 and evaluated in Section 5. Section 6 reviews related work, and finally, Section 7 summarizes our findings and outlines directions for future research.

2 Motivating Example

To motivate the benefits of accurate probabilistic remaining time predictions, consider a logistics process that is subject to a SLA stating that 95% of packages must be delivered within 5 days of ordering. For four packages currently in the system, Table 1 shows their actual remaining time to delivery (in days), as well as the predictions made by three different prediction models.

Table 1. Ground truth and predictions for remaining time to delivery, PE: Point Estimate, ϵ: error, PI: Prediction Interval (95% confidence), IW: Interval Width.

#	Remaining time	model **A**		model **B**			model **C**		
		PE	ϵ	PI	ϵ	IW	PI	ϵ	IW
P1	6.0	4.5	−1.5	[5.0 , 9.0]	+1.0	4.0	[4.0 , 5.0]	−1.5	1.0
P2	4.0	4.2	+0.2	[3.0 , 9.0]	+2.0	6.0	[3.8 , 6.2]	+1.0	2.4
P3	3.0	3.9	+0.9	[0.5 , 3.9]	−0.8	3.4	[0.8 , 3.2]	−1.0	2.4
P4	4.4	5.8	+1.4	[3.6 , 4.8]	−0.2	1.2	[3.3 , 4.5]	−0.5	1.2

Although all models have the same mean absolute error (1.0 days), their usefulness when it comes to recognizing at-risk packages varies considerably:

- Model **A** is a deterministic model that produces point estimates. It fails to detect the true delay for *P1* while potentially triggering unnecessary corrective actions for *P4* (by falsely predicting a delay). Additionally, it lacks uncertainty estimation, displaying the same confidence level for all predictions, thus not giving any indication about the likelihood that this estimate will be correct or how the expected deviation could be.
- Model **B** is a probabilistic model, for which the prediction intervals (PI) with 95% confidence are reported. It correctly predicts a delay for *P1* and on-time delivery for *P3* and *P4*, while maintaining well-calibrated predictions—ensuring all actual dates fall within its prediction intervals. It also signals high uncertainty (i.e., wider interval) for less reliable predictions (e.g., for *P2*). This enables decision-makers to intervene for *P1* while consulting experts or deferring action on *P2* until more data becomes available.
- Model **C** is also probabilistic but offers a desirable advantage over model B: it produces narrower prediction intervals. However, *P1*'s actual delivery date falls outside the predicted range. While this toy example includes only four packages, the issue becomes more significant if the pattern generalizes—e.g., if the prediction intervals for 95% confidence level fail to capture the ground truth in 25% of cases, the model is overconfident. Moreover, the model's largest error coincides with its lowest uncertainty, indicating that uncertainty estimates fail to reflect actual errors. This inconsistency reduces trust in its predictions for decision-making.

Next to highlighting the benefits of probabilistic models over deterministic ones, this illustration reveals key qualities that such models should have: (1) their prediction intervals should include the actual value, (2) their uncertainty—reflected by interval width—should correlate with actual errors, and (3) their prediction intervals should be narrow, without compromising the other two qualities.

3 Probabilistic remaining time prediction

This section covers key concepts used throughout the paper. We first define the problem of probabilistic remaining time prediction in Section 3.1 and then introduce the two main types of uncertainty—epistemic and aleatoric—in Section 3.2.

3.1 Problem Definition

We assume that process execution data is stored in an event log, which is a collection of traces. Each trace captures the sequence of events recorded for a single process instance, i.e., $\psi = \langle e_1, e_2, ..., e_n \rangle$, where each event is a tuple $e = (a, t, \Delta)$, with a its activity, t its timestamp, and Δ an attribute-value map that contains optional data payload. For a trace $\psi = \langle e_1, e_2, ..., e_n \rangle$, the partial execution trace, $hd^k(\psi) = \langle e_1, ..., e_k \rangle$ represents its prefix of length $k \in [1, n-1]$.

Remaining time prediction is typically framed as a regression problem with three main steps: feature extraction, training, and inference. In the feature extraction step, prefixes of varying lengths are generated from each completed trace in the event log to represent different stages of execution. Each prefix is then mapped to a feature set $\mathbf{x}$ with a target value y, representing the remaining time until the trace is completed. These prefixes are treated as independent training examples, forming the training dataset $\mathcal{D} = (\mathbf{x}, y)$.

During training, a function $y = f_{\mathbf{w}}(\mathbf{x})$ is learned to predict the remaining time from the feature set. Deep neural networks are commonly used for this due to their high predictive accuracy [20, 26]. Training finds the optimal network weights, $\hat{\mathbf{w}}$, by minimizing a loss function. During inference, the same feature extraction process is applied to an unseen prefix to generate a predictive feature set $\mathbf{x}^*$. The trained network with optimal weights then estimates the expected remaining time as $\mu_{\mathbf{x}^*} = f_{\hat{\mathbf{w}}}(\mathbf{x}^*)$. However, this yields only a point estimate without accounting for predictive uncertainty. To address this, we define probabilistic remaining time prediction as follows.

Probabilistic remaining time prediction. We treat the remaining time as a continuous random variable and estimate its posterior probability distribution, $P(y^* \mid \mathbf{x}^*, \mathcal{D})$. This distribution captures the likelihood of the remaining time for an unseen prefix (y^*), given its predictive features ($\mathbf{x}^*$) and the training dataset $\mathcal{D}$. We refer to this as the *posterior distribution* of the remaining time. Probabilistic remaining time prediction integrates an uncertainty quantification technique into a deep learning model to estimate this distribution.

Although the posterior distribution can take any form, it is commonly assumed to follow a Gaussian distribution: $P(y^* \mid \mathbf{x}^*, \mathcal{D}) \sim \mathcal{N}(\mu_{\mathbf{x}^*}, {\sigma_{\mathbf{x}^*}}^2)$. Probabilistic remaining time prediction involves estimating its mean ($\mu_{\mathbf{x}^*}$) and standard deviation ($\sigma_{\mathbf{x}^*}$), where the mean represents the expected remaining time, and the standard deviation quantifies predictive uncertainty. This uncertainty stems from two sources—epistemic and aleatoric—as discussed next.

3.2 Epistemic vs. Aleatoric Uncertainty

This section defines epistemic (model) and aleatoric (data) uncertainty and reviews existing techniques for estimating them.

Epistemic uncertainty. Epistemic uncertainty reflects a prediction model's limited knowledge of the process that generated the training data. Event logs capture only part of possible process behavior, with most prefixes representing common process behavior. As a result, the model may exhibit high variance or overfit to frequent patterns, leading to errors when predicting rare but critical cases such as delays. Likewise, if the process changes over time (i.e., concept drift), the model may produce unreliable predictions for unseen prefixes. A robust probabilistic model recognizes its limited knowledge about out-of-data examples and quantifies it as epistemic uncertainty when encountering unfamiliar situations [25].

Aleatoric uncertainty. Aleatoric uncertainty captures the noise inherent in observations, reflecting the ambiguity and variability in remaining time y given input features $\mathbf{x}$ [25, 28]. It stems from noisy event logs or unknown factors not represented in the input features. Accurate remaining time prediction depends on estimating both processing and waiting times, which are influenced by unknown factors. For instance, waiting times may result from resource contention, synchronization between resources (within or across processes), batching, and task handovers [13]. Capturing these factors requires inter-case encoding [22], which is absent in leading remaining time prediction approaches [2–4, 17].

Uncertainty quantification (UQ). The main objective of probabilistic remaining time prediction is to estimate total uncertainty ($\sigma_{\mathbf{x}^*}$) by combining epistemic ($\sigma^e_{\mathbf{x}^*}$) and aleatoric ($\sigma^a_{\mathbf{x}^*}$) uncertainty, as shown in Equation 1.

$$\sigma_{\mathbf{x}^*} = \sqrt{(\sigma^e_{\mathbf{x}^*})^2 + (\sigma^a_{\mathbf{x}^*})^2} \tag{1}$$

Epistemic uncertainty can be reduced by collecting more data, particularly in low-density regions of the feature space (e.g., traces from infrequent process behaviors). In contrast, aleatoric uncertainty, which reflects process complexity, cannot be reduced by collecting more data. Therefore, an effective UQ technique should not only provide an accurate estimate of total uncertainty but also distinguish between these two types of uncertainty.

Bayesian neural networks (BNNs) estimate epistemic uncertainty by treating network weights ($\mathbf{w}$) as probabilistic variables and learning their posterior distribution, $p(\mathbf{w}|\mathcal{D})$. Predictions are generated by averaging outputs over all possible weights [6, 7]. However, exact inference in BNNs is often intractable, so MC dropout [7] is used as an approximation. Dropout is a regularization technique in neural networks where, during training, random neurons are ignored with probability p to prevent overfitting. During inference, dropout is turned off, and all neurons are used, with their activations scaled by p to compensate for the reduced number of active neurons during training. In MC dropout, instead of turning off dropout during inference, multiple stochastic forward passes are made with dropout enabled to estimate the average and variance of predictions [7].

Learning aleatoric uncertainty requires estimation of observation noise (i.e., the difference between observed remaining time and its predicted value). Heteroscedastic regression [10] models the observation noise as a Gaussian distribution $\epsilon \sim \mathcal{N}(0, \varepsilon^2(\mathbf{x}^*))$, with $\varepsilon(\mathbf{x}^*)$ learned from input features. This approach modifies standard neural network training by adjusting the output layer to predict both the expected remaining time and the log-variance of the noise, and by adapting the loss function to account for varying noise levels. This encourages a strong correlation between uncertainty and prediction errors [10].

MC dropout and heteroscedastic regression can be combined to quantify both types of uncertainty [10] and have been applied to PPM tasks [28]. However, this approach can produce overconfident predictions, may reduce accuracy compared to deterministic models, and is limited to specific deep learning architectures. Additionally, it incurs high computational costs due to sampling during inference and validation. To overcome these issues, we propose a simple, efficient, and calibrated UQ approach as outlined in the next section.

4 Approach

Figure 1 outlines our two-step approach: estimating predictive uncertainty via Laplace approximation and refining it with calibrated regression. Given an event log, we first generate prefixes and extract features to construct the training dataset ($\mathcal{D}$) as described in Section 3. A deterministic neural network (DNN) is then trained to produce point estimates of the remaining time. Instead of training a probabilistic model from scratch [10, 28], we apply Laplace approximation to convert the DNN into a Bayesian model that captures epistemic and aleatoric uncertainty. To refine the predicted uncertainty, we use the validation set ($\mathcal{D}_v \subset \mathcal{D}$) to learn recalibration models for prefixes of different lengths. At inference, we extract features for a running instance, predict the remaining time using the DNN, estimate uncertainty via the Laplace model, and refine it using the appropriate recalibration model. The following sections detail Laplace approximation (Section 4.1) and calibrated regression (Section 4.2).

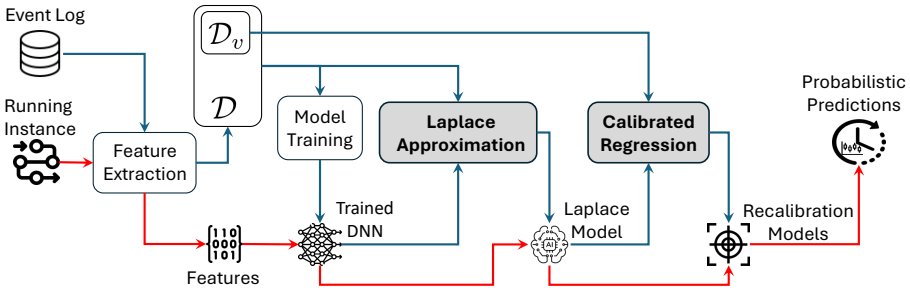

Fig. 1. Overview of our probabilistic remaining time prediction approach.

4.1 Laplace Approximation

We apply Laplace approximation (LA) [6] to estimate the predictive uncertainty of a pre-trained neural network $f_{\mathbf{w}}$. For epistemic uncertainty, LA treats the network weights ($\mathbf{w}$) as probabilistic variables, and approximates $p(\mathbf{w}|\mathcal{D})$ as a Gaussian distribution. The mean of this distribution is the optimal weights ($\hat{\mathbf{w}}$) obtained from deterministic training. The covariance (Σ) is estimated as the negative inverse Hessian of the loss function evaluated at optimal weights [6]. For aleatoric uncertainty, LA assumes the observation noise follows a Gaussian distribution with zero mean and constant standard deviation across all prefixes: $\epsilon \sim \mathcal{N}(0, \varepsilon^2)$. The noise level ($\varepsilon$) is inferred by maximizing the likelihood of the observed data given the optimal network weights [6].

We treat the weights of the last linear layer of the network probabilistically, keeping all other weights fixed at their optimal deterministic values. In this setup, the posterior distribution of remaining time follows a Gaussian distribution: $P(y^* \mid \mathbf{x}^*, \mathcal{D}) \sim \mathcal{N}(\mu_{\mathbf{x}^*}, {\sigma_{\mathbf{x}^*}}^2)$. The expected value of remaining time is given by the deterministic network: $\mu_{\mathbf{x}^*} = f_{\hat{\mathbf{w}}}(\mathbf{x}^*)$. The standard deviation represents the total uncertainty, which includes both epistemic and aleatoric uncertainty components (see Equation 1). These components are estimated through LA as shown in Equation 2, where $\mathbf{J}(\mathbf{x})$ is the Jacobian matrix of network weights, $\mathbf{I}$ is the identity matrix, and $^\intercal$ denotes the transpose operation [6].

$$\sigma^e_{\mathbf{x}^*} = \sqrt{\mathbf{J}(\mathbf{x})^\intercal \Sigma \mathbf{J}(\mathbf{x})} \quad \sigma^a_{\mathbf{x}^*} = \varepsilon \mathbf{I} \tag{2}$$

LA is an efficient UQ technique involving two simple steps: computing the Hessian of the loss function, and inferring constant observation noise. Since LA is applied only to the last linear layer, it can be used with any pre-trained neural network, regardless of its architecture. Unlike MC dropout [28], which is limited to recurrent and convolutional networks, our approach offers more flexibility, enabling uncertainty estimation with other deep learning architectures [2, 3].

Using the expected value ($\mu_{\mathbf{x}^*}$) and the total uncertainty ($\sigma_{\mathbf{x}^*}$) estimated by LA, prediction intervals for remaining time is computed as $\mu_{\mathbf{x}^*} \pm \text{z-score} \times \sigma_{\mathbf{x}^*}$, where the z-score is determined from the standard normal table based on the confidence level. A well-calibrated prediction ensures that the prediction intervals contains the true remaining time with the expected probability. However, LA makes simplifying assumptions, such as probabilistic weights only in the last layer and constant observation noise, which may lead to miscalibrated uncertainty estimates. To mitigate this, we apply post-hoc calibrated regression [11], training auxiliary models to adjust the uncertainty estimates provided by LA.

4.2 Calibrated Regression

The second step in our approach applies calibrated regression to refine the uncertainty estimates from LA. We first define calibration and introduce the standard approach for calibrated regression [11]. Then, we explain how we extend this approach by learning separate recalibration models for prefixes of different lengths.

Calibration. Calibration measures how well the prediction intervals align with observed remaining times [5]. For a confidence level p, the prediction interval coverage probability, $PICP(p)$, measures the proportion of observed remaining times falling within the model's predicted interval [9]. Calibration is commonly evaluated using a calibration plot, with $PICP(p)$ on the y-axis and $p \in [0, 1]$ on the x-axis. The area between this plot and the parity line ($PICP(p) = p$), known as the miscalibration area (MA), reflects the model's calibration performance. A perfectly calibrated model satisfies $PICP(p) = p$ and has a miscalibration area of zero [5]. For instance, the 95% credible prediction intervals should ideally contain the true value in 95% of prefixes. The miscalibration area ranges between 0 and 0.5.

Standard calibrated regression. Calibrated regression begins by creating a calibration dataset. For prefixes in the validation set, we form a set of tuples $(\mathbf{x}_t, y_t, \mu_{\mathbf{x}_t}, \sigma_{\mathbf{x}_t}) \in \mathcal{D}_v$, where y_t is the true remaining time, and $\mu_{\mathbf{x}_t}$ and $\sigma_{\mathbf{x}_t}$ are the predicted mean and standard deviation. These values are then used to calculate the cumulative distribution function (CDF) value, $F_t(y_t) : \mathbb{R} \to [0, 1]$, for each tuple. Next, we compute the empirical probability $\hat{P}(p)$, which is the fraction of true values y_t that fall below the corresponding quantile $F_t^{-1}(p)$ across all t. This is repeated for different confidence levels p, forming a calibration dataset with pairs $(F_t(y_t), \hat{P}(F_t(y_t)))$. Each pair consists of a predicted CDF value and its corresponding empirical probability [11].

Calibrated regression learns a recalibration model $\mathcal{R} : [0, 1] \to [0, 1]$ to adjust the CDF values so that they match the empirical probability. Kuleshov et al., [11] proposed *isotonic regression* to train model $\mathcal{R}$, as it ensures higher predicted probabilities correspond to higher empirical probabilities. A similar approach involves learning a constant *scaling factor* λ such that the adjusted standard deviation $\sigma'_{\mathbf{x}_t} = \lambda\sigma_{\mathbf{x}_t}$ minimizes miscalibration area for prefixes in the validation set. In this paper, we experiment with both approaches. We first apply calibrated regression to adjust total uncertainty and then scale epistemic and aleatoric uncertainties proportionally to maintain their original ratio.

The standard recalibration approach assumes consistent prediction errors across prefix lengths, ignoring the iterative nature of remaining time prediction. However, as a process unfolds, the model updates its predictions using longer prefixes, and errors should generally decrease with more available information. This means shorter prefixes are more prone to miscalibration and require stronger adjustments. To address this, we propose a process-aware calibrated regression that trains a separate recalibration model for prefixes of similar lengths.

Process-aware calibrated regression. We extend standard calibrated regression by partitioning prefix lengths into multiple groups and learning a separate recalibration model for each group. In the simplest case, all prefixes form a single group, reducing our approach to the standard recalibration. At the other extreme, each prefix length forms its own group, but this risks sparsely populated calibration datasets, leading to unreliable calibration. To balance this, we group similar prefix lengths together.

Partitioning algorithm: Given the validation set $\mathcal{D}_v$, the sorted list of all prefix lengths $\mathcal{S}$ and the maximum number of groups $\mathcal{G}$, we create a partition over $\mathcal{S}$ and use it to create multiple calibration datasets from $\mathcal{D}_v$. The set of all prefixes of length k in $\mathcal{D}_v$ is denoted as $\mathcal{D}_v^k$. Our algorithm proceeds in two steps:

1. We initialize an empty group g_ℓ and set $\overline{g_\ell}$ to count the number of prefixes from $\mathcal{D}_v$ that their corresponding length is assigned to g_ℓ. We then assign the smallest length $k \in \mathcal{S}$ to g_ℓ, update $\overline{g_\ell} = \overline{g_\ell} + \mid \mathcal{D}_v^k \mid$, and remove k from $\mathcal{S}$. We continue assigning lengths to the same group until $\overline{g_\ell} \leq \mid \mathcal{D}_v \mid / \mathcal{G}$. Once this condition is met, g_ℓ is finalized, and we create a new group. This process repeats until all prefix lengths are assigned, forming the partition $\mathit{\Pi} = \{g_\ell \mid \ell \in L\}$, where L is the set of group indices.
2. For each group $\pi \in \mathit{\Pi}$, we obtain $\{\mathcal{D}_v\}^\pi = \bigcup_{k \in \pi} \mathcal{D}_v^k$, the set of all prefixes with lengths in π. We then construct a separate calibration dataset for $\{\mathcal{D}_v\}^\pi$ using the standard calibrated regression approach.

The output of *partitioning algorithm* is a set of calibration datasets. For each dataset, we learn a recalibration model $\mathcal{R}_\pi$ and use it to adjust the total uncertainty for its corresponding prefixes in the validation set following the standard calibrated regression approach. After applying all recalibration models, we compute their aggregated performance on the validation set. This process is repeated by calling the *partitioning algorithm* for different values of $\mathcal{G} \in \{1, 10, 20, ..., 100\}$ to find the partition that minimizes the sparsification error, as defined next.

Sparsification error: The sparsification error measures how well a model's uncertainty estimate correlates with its prediction error. It is computed by sorting predictions by their uncertainty and progressively removing subsets with the highest uncertainty (1% at a time). The mean absolute error on the remaining predictions is then plotted against the fraction of removed examples, forming the error curve. A monotonically decreasing error curve indicates that the model assigns higher uncertainty to less accurate predictions. In contrast, randomly removing subsets results in a nearly flat random curve. The Area Under the Random Gain curve (AURG) is the area between the error curve and the random curve, quantifies the sparsification error [18]. A higher AURG indicates lower sparsification error, meaning the predicted uncertainties provide a useful estimate of the model's error.

In summary, our process-aware approach finds the best partitioning of prefix lengths and trains separate recalibration models for prefixes with similar lengths.

5 Evaluation

This section presents the experiments used to evaluate our approach. Table 2 summarizes the characteristics of the 10 publicly available event logs used as a basis for this. In the remainder, Section 5.1 describes the experimental setup, followed by the results in Section 5.2. Our implementation and employed event logs are available in our project's public repository.[3]

[3] `https://github.com/keyvan-amiri/UQ4PPM`

Table 2. Characteristics of the employed event logs (time-related attributes in days).

Event log	Cases	Events	Activities	Variants	Case length		Case duration	
					Avg.	Max	Avg.	Max
BPIC20DD	10 500	56 437	17	99	5.37	24	11.5	469.2
BPIC20ID	6449	72 151	34	753	11.19	27	86.5	742.0
BPIC20PTC	2099	18 246	29	202	8.69	21	36.8	328.2
BPIC20RFP	6886	36 796	19	89	5.34	20	12.0	410.0
BPIC20TPD	7065	86 581	51	1478	12.25	90	87.4	1202.0
BPIC15-1	1199	52 217	398	1170	43.55	101	95.9	1486.0
BPIC13I	7554	65 533	13	2278	8.68	123	12.1	771.4
BPIC12	13 087	262 200	36	4366	20.04	175	8.6	137.2
Sepsis	1050	15 214	16	846	14.49	185	28.5	422.3
Helpdesk	4580	21 348	14	226	4.66	15	40.9	60.0

5.1 Experimental Setup

Data split. Traces in the event log are sorted chronologically based on their start timestamp. We use the first 80% of traces for training, while the remaining 20% are reserved for testing. Within the training set, the last 20% of traces are reserved as the validation set. We create prediction tasks by extracting all prefixes in the range $[2, n-1]$ per trace of length n in the training and test sets.

Deterministic neural network (DNN). We use the data-aware long short-term memory (LSTM) [17] due to its strong performance in a recent benchmark [20]. The model consists of two LSTM layers with 150 units, and a dropout layer with a dropout rate of 0.1 in between. The model is trained using the L1-loss and NAdam optimizer for up to 200 epochs, with early stopping patience of 50 epochs. A batch size of 64 is used during training.

Approach configurations. We first train the DNN and apply Laplace approximation (LA) to estimate epistemic and aleatoric uncertainty as described in Section 4.1. Since LA is applied only to the final linear layer, we can efficiently use a full-rank covariance matrix to compute the Hessian of the loss function. This avoids the need for less expressive diagonal or block-diagonal factorizations like Kronecker-factored approximate curvature and low-rank approximations. We compare two recalibration models to instantiate the process-aware calibrated regression step (Section 4.2), yielding two approach configurations, one using isotonic regression [11] (referred to as *LA+I*) and one using a learned constant scaling factor (*LA+S*).

Baselines. We benchmark our approach against the following baselines:

- A probabilistic remaining time prediction approach [28] using MC dropout combined with heteroscedastic regression (*DA+H*). We use a dropout probability of 0.1, and the concrete dropout variant (*CDA+H*), where the optimal dropout probability is learned. Default hyperparameter values [28] are used for training.
- Heteroscedastic regression (*HR*) [10]: We modify the last layer of the DNN to estimate aleatoric uncertainty, as explained in Section 3.
- Deep ensemble (*DE*) and bootstrapping ensemble with Heteroscedastic regression (*BE+H*) [12] train multiple models, with the ensemble size treated as a hyperparameter. We evaluate 3–10 members for *DE*, and 3–40 for *BE+H*.

Each *BE+H* model performs heteroscedastic regression and is trained on a random 25% of the training data.
- Classification and regression diffusion model (*CARD*) [9]: We integrate a denoising diffusion model into the DNN to estimate aleatoric uncertainty.

DA+H and *CDA+H* estimate both epistemic and aleatoric uncertainty. The other baselines estimate only aleatoric uncertainty; among them, *BE+H* and *CARD* are applied to a PPM task for the first time in this paper.

Performance metrics. We quantify the performance criteria using mean absolute error (MAE) for *accuracy*, miscalibration area (MA) for *calibration*, and AURG for *sparsification error*. We also compare the mean prediction interval width (MPIW) [5] to evaluate the tightness of prediction intervals.

5.2 Results

Table 3 summarizes the experimental results for probabilistic remaining time prediction across ten event logs and four performance metrics.

Additionally, we use the Wilcoxon signed-rank test to compare our approach against the baselines, identifying significant performance differences at confidence levels of 0.95, 0.90, and 0.80, with the results presented in Table 4. For example, the entry $\{(LA+I, LA+S > CARD) : 0.90\}$ shows that both *LA+I* and *LA+S* outperform CARD with 90% confidence. Since higher confidence results are valid at lower levels, redundant entries are omitted.

Overall, neither our approach nor any baseline consistently outperforms across all event logs and metrics. Below, we summarize the key findings.

Accuracy. Both configurations of our approach achieve the highest accuracy in 3 out of 10 event logs and perform close to the best in three others. Only *CARD* performs significantly worse than both, while the other baselines show no significant differences. Since our approach preserves the accuracy of the DNN, this suggests that existing baselines [28] do not consistently enhance accuracy and may sometimes reduce it.

Calibration and interval width. Both configurations of our approach significantly outperform the baselines in terms of calibration, with *LA+S* achieving better results than *LA+I*. UQ approaches such as MC dropout and heteroscedastic regression (*DA+H*, *CDA+H*, *HR*) tend to produce overly narrow prediction intervals that often fail to capture the true remaining time. In contrast, *LA+S* and *LA+I* consistently yield well-calibrated prediction intervals, as indicated by their low MA values in Table 3. However, this calibration often comes at the expense of wider intervals. While *CDA+H*, *BE+H*, and *HR* produce narrower intervals, they frequently exhibit poor calibration. Balancing interval width and calibration remains a core challenge in probabilistic remaining time prediction. Notably, when the predicted mean diverges from the actual remaining time, widening the interval becomes the only way to ensure coverage. Thus, leveraging a DNN with higher accuracy can mitigate this trade-off.

Sparsification error. Both configurations of our approach achieve comparable sparsification error compared to the baselines that capture both epistemic

Table 3. Performance comparison for probabilistic remaining time prediction across 10 event logs. MAE and MPIW are measured in days. For all metrics except AURG, lower values indicate better performance.

		Baselines						**Our approach**	
Log	Metric	*DA+H*	*CDA+H*	*HR*	*DE*	*BE+H*	*CARD*	*LA+I*	*LA+S*
BPI20DD	MAE	5.38	6.32	4.13	5.92	4.70	4.50	**4.12**	**4.12**
	MA	0.34	0.41	0.23	0.30	0.20	0.05	0.11	**0.04**
	MPIW	**3.37**	4.05	5.47	7.31	6.67	16.56	19.44	13.03
	AURG	1.76	1.74	1.45	**2.05**	1.71	0.35	1.19	1.13
BPI20ID	MAE	14.62	17.09	15.40	17.26	**14.08**	22.26	21.72	21.72
	MA	0.16	0.06	0.28	0.11	0.11	0.06	**0.03**	**0.03**
	MPIW	37.59	85.76	**15.77**	95.11	37.67	139.63	96.09	95.87
	AURG	4.45	3.31	**8.36**	7.80	7.01	4.36	6.16	6.13
BPI20PTC	MAE	9.45	12.83	8.36	11.08	10.45	11.86	**7.73**	**7.73**
	MA	0.27	0.47	0.31	0.29	0.32	**0.05**	0.15	0.07
	MPIW	8.19	**5.13**	5.41	18.52	9.22	64.13	14.91	59.06
	AURG	4.56	4.77	4.52	5.43	**5.71**	3.15	3.18	3.06
BPI20RFP	MAE	**5.09**	6.30	5.59	7.35	6.62	5.71	5.19	5.19
	MA	0.25	0.28	0.28	0.32	0.30	**0.03**	0.10	0.05
	MPIW	6.50	5.52	**5.29**	8.06	6.18	25.5	21.09	16.00
	AURG	1.74	2.55	2.36	**2.61**	2.49	-0.71	1.04	0.85
BPI20TPD	MAE	**25.02**	25.98	26.01	27.12	27.80	30.37	27.79	27.79
	MA	0.15	0.08	0.31	**0.05**	0.19	**0.05**	0.34	**0.05**
	MPIW	174.13	121.91	**35.57**	109.36	45.86	129.46	41.35	131.35
	AURG	8.41	9.38	10.45	14.00	**14.45**	1.13	4.09	5.92
BPI15-1	MAE	30.29	32.23	26.93	**24.71**	27.23	100.43	25.28	25.28
	MA	0.22	0.26	0.24	0.11	0.15	0.39	**0.03**	0.10
	MPIW	402.67	**20.25**	37.52	144.43	72.38	182.0	103.12	156.59
	AURG	1.53	**12.61**	5.89	8.65	5.44	3.74	5.42	5.25
BPI13I	MAE	3.67	4.19	3.15	3.36	**3.06**	4.69	5.53	5.53
	MA	0.15	0.21	0.26	0.06	0.20	0.12	**0.04**	0.11
	MPIW	12.25	14.90	8.08	21.9	**8.05**	29.09	20.45	14.36
	AURG	0.84	1.08	0.27	0.25	0.54	**1.43**	0.21	1.22
BPI12	MAE	6.60	8.02	**5.86**	6.05	5.95	7.03	7.84	7.84
	MA	0.27	0.14	0.10	0.13	0.07	0.29	0.19	**0.05**
	MPIW	**8.63**	18.40	21.69	16.00	22.09	15.70	23.81	44.57
	AURG	**1.86**	1.59	1.63	1.58	1.80	0.34	0.33	0.46
Sepsis	MAE	16.35	18.08	**15.32**	15.44	15.69	24.50	15.73	15.73
	MA	0.11	0.37	0.07	**0.04**	0.13	0.18	0.21	0.06
	MPIW	17.29	**4.86**	50.95	43.1	66.45	81.07	98.15	51.30
	AURG	0.52	-4.52	4.87	4.80	3.53	**8.35**	1.53	2.24
Helpdesk	MAE	11.28	10.09	**9.45**	11.18	18.65	12.87	**9.45**	**9.45**
	MA	0.25	0.40	0.44	0.28	0.34	0.17	0.24	**0.12**
	MPIW	**7.82**	13.74	7.63	24.23	34.30	42.37	23.64	70.27
	AURG	-1.16	0.17	0.70	0.53	1.11	-0.08	1.44	**1.48**

Table 4. Statistically significant performance differences.

Metric	**Significant Performance Differences**
MAE	{(LA+I, LA+S > CARD): 0.90}
MA	{(LA+S > DA+H, CDA+H, HR, BE+H, DE): 0.95}, {(LA+I > HR): 0.95}, {(LA+I > CDA+H): 0.90}, {(LA+S > CARD, LA+I): 0.90}, {(LA+I > DA+H): 0.80}
MPIW	{(CDA+H, HR, BE+H > LA+I, LA+S): 0.95}, {(DE > LA+S): 0.95}, {(LA+I > CARD): 0.90}
AURG	{(HR, BE+H, DE > LA+I, LA+S): 0.95} , {(LA+S > CARD): 0.80}

and aleatoric uncertainty (*DA+H*, *CDA+H*). However, pure aleatoric baselines like *DE*, *BE+H* and *HR* perform significantly better. The heteroscedastic loss function in *HR*, enhances the correlation between estimated uncertainty and model error, while greater disagreement among ensemble members in *DE* and *BE+H* reflects greater uncertainty and indicates larger potential errors. Our findings suggest a trade-off between capturing epistemic uncertainty and minimizing sparsification error, which warrants further investigation.

We evaluated the relative contribution of epistemic uncertainty by computing its proportion in the total uncertainty, averaged across event logs. Both our configurations (*LA+I*, *LA+S*) and baselines (*DA+H*, *CDA+H*) allocate a similar share (17-18%) to epistemic uncertainty, leading to two insights. First, it validates the simplifying assumptions of LA: despite applying Bayesian inference only to the final layer and assuming constant aleatoric noise, our approach performs comparably to more complex methods like MC dropout and heteroscedastic regression. Second, it highlights that aleatoric uncertainty dominates probabilistic remaining time prediction, explaining why purely aleatoric baselines (*HR*, *BE+H*) still achieve strong performance.

Ablation study. We evaluate the impact of process-aware calibrated regression via an ablation study comparing *LA+I* and *LA+S* against the uncalibrated uncertainty estimates of the *LA* model, as shown in Table 5. Since calibration does not influence MAE, it is excluded from this comparison. Both *LA+I* and *LA+S* enhance calibration performance, albeit typically at the cost of wider prediction intervals. An exception to this pattern is observed in the Sepsis log, where improved calibration is achieved without the typical trade-off, as prediction intervals become narrower rather than wider. In BPI20DD, BPI20RFP, and BPI13I logs, the original *LA* model already yields well-calibrated intervals, limiting the benefits of recalibration. Recalibration reduces sparsification error across most logs, with the most pronounced improvements observed in Helpdesk, BPI20DD, and BPI20TPD.

Table 5. Performance of Laplace approximation before calibrated regression.

Metric	BPI20DD	BPI20ID	BPI20PTC	BPI20RFP	BPI20TPD	BPI15-1	BPI13I	BPI12	Sepsis	Helpdesk
MA	0.07	0.15	0.28	0.06	0.41	0.15	0.04	0.32	0.15	0.37
MPIW	14.52	66.07	9.26	14.73	24.04	65.24	19.32	14.61	66.20	14.00
AURG	0.12	6.13	2.43	0.11	0.00	3.38	1.22	−0.49	2.11	−0.60

Sensitivity analysis. We analyze the sensitivity of MA, MPIW, and AURG to the maximum number of groups ($\mathcal{G}$) used for prefix lengths in process-aware calibrated regression. Table 6 reports the coefficient of variation (i.e., the ratio of standard deviation to mean) for each performance metric.

Table 6. Sensitivity analysis of process-aware calibrated regression using the coefficient of variation for each performance metric.

	Metric	BPI20DD	BPI20ID	BPI20PTC	BPI20RFP	BPI20TPD	BPI15-1	BPI13I	BPI12	Sepsis	Helpdesk
LA+I	MA	0.04	0.12	0.02	0.01	0.00	0.01	0.12	0.00	0.01	0.00
	MPIW	0.02	0.03	0.01	0.00	0.00	0.00	0.02	0.00	0.01	0.00
	AURG	0.32	0.03	0.11	0.25	0.53	0.11	0.22	0.82	0.08	1.72
LA+S	MA	0.06	0.14	0.12	0.07	0.01	0.08	0.09	0.03	0.07	0.08
	MPIW	0.06	0.05	0.23	0.08	0.04	0.04	0.06	0.01	0.08	0.01
	AURG	0.35	0.02	0.07	0.35	0.33	0.11	1.06	0.78	0.02	1.10

Both MA and MPIW metrics remain generally stable across varying values of $\mathcal{G}$, indicating that using separate recalibration models for different prefix lengths has limited impact on these metrics. Exceptions include BPIC20ID, BPIC20PTC, and BPIC13I, where moderate sensitivity is observed. Increasing the number of groups narrows the prediction intervals but worsens calibration in BPIC20ID. Both metrics deteriorate with more groups in BPIC20PTC, whereas in BPIC13I, both improve. In contrast, AURG exhibits high sensitivity to $\mathcal{G}$. Across most event logs, increasing $\mathcal{G}$ consistently reduces sparsification error. However, beyond a certain threshold, this improvement plateaus or slightly reverses. This trend supports the use of AURG as an objective for tuning calibrated regression. Overall, partitioning prefixes into a few length-based groups and applying separate recalibration models can effectively reduce sparsification error, though excessive partitioning often fails to yield further improvement.

Training and inference time. Training and inference times were averaged across 10 event logs using an NVIDIA RTX A6000 GPU. Training the DNN took 35 minutes, with our approach configurations adding only 2 minutes. In contrast, *DA+H* and *CDA+H* were up to 20 times more computationally expensive than the deterministic model, even surpassing ensembles (*DE*, *BE+H*). By avoiding sampling during validation, our approach significantly reduces training time compared to existing baselines for capturing both types of uncertainty. Inference times per prefix for *DA+H*, *CDA+H*, and *CARD* ranged from 35 to 440 milliseconds due to sampling, while our approach configurations and other baselines required only 0.10 to 0.55 milliseconds. The low inference time of our approach make it suitable for real-time probabilistic predictions.

Limitations and threats to validity. Our findings are limited to the 10 event logs used, and significance holds only within this scope. Additionally, we evaluated only LSTMs with sequential encoding; results may not generalize to other architectures or encodings.

In summary, our approach maintains the accuracy of the DNN while delivering well-calibrated prediction intervals. Our results demonstrate that efficient uncertainty estimation with LA and process-aware calibrated regression performs similarly to state-of-the-art baselines. Moreover, due to its efficiency and com-

patibility with any pre-trained DNN, our approach offers a simple yet powerful solution for probabilistic predictive analysis.

6 Related Work

This section provides a brief overview of related work on remaining time prediction and uncertainty quantification with neural networks.

Remaining time prediction. Approaches for predicting remaining time can be broadly classified into two categories: annotated process models and machine learning models [26]. In the first category, transition systems have been enhanced with queuing models [23] or machine learning models [19]. However, pure machine learning approaches generally outperform annotated process models, with neural networks achieving the highest accuracy [20, 26]. Various deep learning architectures have been proposed for this problem, including recurrent neural networks [4, 17], Transformers [3], and graph neural networks [2]. Nevertheless, these approaches only predict the expected remaining time and lack uncertainty estimates required for generating prediction intervals. Stochastic Petri nets [21] address this gap by providing probabilistic predictions, but they rely on process models and are generally less accurate than deep learning models [26]. Integrating uncertainty quantification into neural networks provides a more accurate solution for probabilistic remaining time prediction.

Uncertainty Quantification. Uncertainty quantification in deep learning has led to significant advancements in areas like computer vision [1,10] and reinforcement learning [1,7,14], particularly in safety-critical applications like self-driving cars and healthcare [1,8,18]. Various approaches have been proposed to estimate predictive uncertainty, including MC dropout [7], heteroscedastic regression [10], deep ensembles [18], Laplace approximation [6], diffusion models [9], and simultaneous quantile regression [25], with comprehensive reviews providing insights into these approaches and their applications [1,8]. Despite its success in other domains, uncertainty quantification remains largely unexplored in business process management. Existing approaches include probabilistic outcome prediction using gradient boosting decision trees [24], and probabilistic event processing time prediction with quantile regression forests [16] and MC dropout [15]. Notably, the only existing approach for probabilistic remaining time prediction combines MC dropout with heteroscedastic regression [27, 28].

7 Conclusion

In this paper, we proposed an uncertainty-aware remaining time prediction approach. Unlike deterministic models that only provide the expected values without uncertainty estimation, it estimates both epistemic and aleatoric uncertainty to generate well-calibrated prediction intervals. Specifically, we applied Laplace approximation to transform a pre-trained deterministic neural network into a Bayesian model and refined the uncertainty estimates using a process-aware calibrated regression technique. Benchmarking on 10 real-world event logs against

state-of-the-art approaches for probabilistic remaining time prediction, our approach delivers comparable performance while significantly reducing computational costs. Moreover, it preserves the accuracy of the pre-trained network and generates well-calibrated prediction intervals. Finally, our approach is compatible with any deep learning model, regardless of architecture. By combining flexibility, efficiency, and strong predictive performance, it offers a valuable tool for developing uncertainty-aware information systems that support decision-making in business processes.

This work opens several directions for future research. In calibrated regression step, we exclusively relied on the predicted expected remaining time, associated uncertainty, and ground truth values, without incorporating the predictive features. A natural extension is to leverage the pre-trained network to learn embeddings of prefixes, which can serve as input representations for learning recalibration models. We also plan to investigate alternative prefix grouping strategies, such as clustering based on process semantics. Additionally, we will explore various feature extraction techniques and deep learning architectures to assess whether enhancing predictive accuracy can reduce the trade-off between calibration and narrow prediction intervals. We also intend to investigate how information loss during feature extraction and the expressive power of different models affect epistemic and aleatoric uncertainty, as well as the quality of probabilistic predictions. Finally, we aim to apply our approach to classification tasks, such as probabilistic process outcome prediction.

References

1. Abdar, M., Pourpanah, F., Hussain, S., et al.: A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion **76**, 243–297 (2021)
2. Amiri Elyasi, K., van der Aa, H., Stuckenschmidt, H.: PGTNet: A process graph transformer network for remaining time prediction of business process instances. In: Guizzardi, G., Santoro, F., et al. (eds.) Advanced Information Systems Engineering. pp. 124–140. Springer Nature Switzerland (2024)
3. Bukhsh, Z.A., Saeed, A., Dijkman, R.M.: Processtransformer: Predictive business process monitoring with transformer network (2021)
4. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: Hildebrandt, T., van Dongen, B.F., et al. (eds.) Business Process Management. pp. 286–302. Springer International Publishing (2019)
5. Chung, Y., Char, I., Guo, H., et al.: Uncertainty toolbox: an open-source library for assessing, visualizing, and improving uncertainty quantification (2021)
6. Daxberger, E., Kristiadi, A., et al.: Laplace redux - effortless bayesian deep learning. In: NeurIPS. vol. 34, pp. 20089–20103. Curran Associates, Inc. (2021)
7. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: ICML. vol. 33, pp. 1050–1059 (2016)
8. Gawlikowski, J., Tassi, C.R.N., et al.: A survey of uncertainty in deep neural networks. Artificial Intelligence Review **56**(1), 1513–1589 (2023)
9. Han, X., Zheng, H., Zhou, M.: CARD: Classification and regression diffusion models. NeurIPS **35**, 18100–18115 (2022)

10. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: NeurIPS. vol. 30. Curran Associates, Inc. (2017)
11. Kuleshov, V., Fenner, N., Ermon, S.: Accurate uncertainties for deep learning using calibrated regression. In: ICML. vol. 35, pp. 2796–2804. PMLR (2018)
12. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: NeurIPS. vol. 30. Curran Associates, Inc. (2017)
13. Lashkevich, K., Milani, F., Chapela-Campa, D., Suvorau, I., Dumas, M.: Unveiling the causes of waiting time in business processes from event logs. Information Systems **126**, 102434 (2024)
14. Lockwood, O., Si, M.: A review of uncertainty for deep reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **18**(1), 155–162 (2022), number: 1
15. Mehdiyev, N., Majlatow, M., Fettke, P.: Communicating uncertainty in machine learning explanations: a visualization analytics approach for predictive process monitoring. In: World Conference on Explainable Artificial Intelligence. pp. 420–438. Springer (2024)
16. Mehdiyev, N., Majlatow, M., Fettke, P.: Quantifying and explaining machine learning uncertainty in predictive process monitoring: an operations research perspective. Annals of Operations Research pp. 1–40 (2024)
17. Navarin, N., Vincenzi, B., Polato, M., Sperduti, A.: LSTM networks for data-aware remaining time prediction of business process instances. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–7 (2017)
18. Nemani, V., Biggio, L., et al.: Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial. Mechanical Systems and Signal Processing **205**, 110796 (2023)
19. Polato, M., Sperduti, A., Burattin, A., Leoni, M.d.: Time and activity sequence prediction of business process instances. Computing **100**, 1005–1031 (2018)
20. Rama-Maneiro, E., Vidal, J.C., Lama, M.: Deep Learning for Predictive Business Process Monitoring: Review and Benchmark. IEEE Transactions on Services Computing **16**(1), 739–756 (2023)
21. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Information Systems **54**, 1–14 (2015)
22. Senderovich, A., Di Francescomarino, C., et al.: Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In: Business Process Management: BPM. pp. 306–323. Springer (2017)
23. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining for delay prediction in multi-class service processes. Information systems **53**, 278–295 (2015)
24. Shoush, M., Dumas, M.: When to intervene? prescriptive process monitoring under uncertainty and resource constraints. In: Di Ciccio, C., Dijkman, R., et al. (eds.) Business Process Management Forum. pp. 207–223. Springer (2022)
25. Tagasovska, N., Lopez-Paz, D.: Single-model uncertainties for deep learning. In: NeurIPS. vol. 32. Curran Associates, Inc. (2019)
26. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: 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)
27. Weytjens, H., De Weerdt, J.: Learning uncertainty with artificial neural networks for improved remaining time prediction of business processes. In: International Conference on Business Process Management. pp. 141–157. Springer (2021)
28. Weytjens, H., De Weerdt, J.: Learning uncertainty with artificial neural networks for predictive process monitoring. Applied Soft Computing **125**, 109134 (2022)