Data-driven predictive maintenance and time-series applications

Kefalas, M.

Chapter 5

Uncertainty Quantification in RUL Estimation

In Chapter 4, we introduced AutoML to the reader, discussed its significance to the data-driven domain generally and, specifically, its role in PHM. We further showcased through performing experiments on a specific simulated dataset from the aerospace industry that AutoML with traditional ML methods in its search space outperforms or is comparable to pre-selected traditional ML techniques and standard NNs while being outperformed by specifically designed NNs.

This chapter[1] will touch upon a topic of high significance in PHM and specifically data-driven PdM, namely uncertainty quantification (UQ). The motivation behind this chapter lies in the fact that in addition to the RUL prediction, one needs to assess also the confidence of that prediction. This is especially crucial in operations-critical and safety-critical applications, where an indication of the remaining time until failure should be as confident as possible. Otherwise, for example, unnecessary downtime of the asset might occur. Furthermore, by accounting for the uncertainties, the researcher or end-user can determine if, for example, the training data is not representative of the task or is too noisy (i.e., measurement uncertainties, operating environment uncertainties, future load uncertainties, input data uncertainties) or if the selected model is poorly selected (i.e., underparameterized NN).

To address this, we propose a technique for uncertainty quantification (UQ) based on Bayesian deep learning. The hyperparameters of the framework are tuned using a novel bi-objective Bayesian hyperparameter optimization (HPO) method with two objectives: predictive performance and predictive uncertainty. The method also integrates the data pre-processing steps into the HPO stage, models the RUL as a Weibull distribution, and returns the survival curves

[1]Contents of this chapter are based on [116]; Marios Kefalas, Bas van Stein, Mitra Baratci, Asteris Apostolidis, and Thomas Bäck. An End-to-End Pipeline for Uncertainty Quantification and Remaining Useful Life Estimation: An Application on Aircraft Engines. In 2022 7th European Conference of the Prognostics and Health Management Society, Turin, Italy, July 2022. PHM Society.

of the monitored assets to allow informed decision-making. We validate this method on the widely used C-MAPSS dataset.

5.1 Introduction

Prognostics and health management (PHM) includes multiple methodologies and functions as a decision support tool that aims at minimizing maintenance costs and predicting when a failure could occur by the assessment, prognosis, diagnosis, and health management of engineered systems [230]. The core of PHM is failure prognostics. Failure prognostics refers specifically to the phase involved with predicting future behavior and the system's useful lifetime left in terms of current operating state and the scheduling of required maintenance actions to maintain system health [227]. This useful lifetime left is often called the remaining useful life (RUL) [230] and is defined as the length from the current time and operating state to the end of the useful life [208] (for more information on the RUL see Section 3.1). The notice of pending equipment failure allows for sufficient lead-time so that necessary decisions, personnel, equipment, and spare parts can be organized and deployed, thus minimizing equipment downtime and repair costs. By leveraging RUL estimation[2], industries, such as aerospace, maritime, and energy, can improve maintenance schedules to avoid catastrophic failures and consequently save lives and costs [253]. The industry has to also assure that its asset utilization is optimum by guaranteeing a timely - but not premature - maintenance. Furthermore, this practice promotes sustainability as the use of spare parts is optimum and no useful life is wasted.

As has already been mentioned in previous chapters, the estimation of the RUL can be done in various ways. *Model-based*, *data-driven*, and *hybrid methods* are the most prominent approaches [230], and in general, all methods make some use of the sensor data of the equipment and/or maintenance history. For more details on these methods, we refer the reader to Section 3.3 and Section 4.1. It is worth mentioning here, though, that amongst the three said methods, data-driven methods are relatively easier to develop as they do not need (a lot of) expert or domain knowledge to develop the model, rendering them domain-agnostic and easily transferable between domains, and because of the plethora of tools that have been and are being developed. The previous make data-driven methods available to a broader audience of researchers and end-users.. They can require, however, large amounts of data.

Data-driven approaches either fall under the category of classic machine learning (ML) algorithms (such as random forests (RF)) [253, 195] or the more recently proposed deep neural networks (DNNs) [94, 146, 260]. In both cases, though, the estimation of the RUL is a challenging problem. The remaining useful life is not merely a target variable that can be predicted from sensor measurements, but it is a variable that needs to be inferred from a longer trend

[2]In this work we will be using the terms RUL *prediction* and RUL *estimation* interchangeably, unless otherwise stated.

of degradation patterns and when those begin to occur. In this view, and due to the advances in the general field of artificial intelligence (AI), deep learning (DL) and DNNs have proven to be a successful candidate to the RUL estimation task [138, 25, 112, 35, 177, 237]. One significant advantage of DNNs lies in their ability to learn features from raw data automatically and extract patterns that can enhance the RUL estimation accuracy [25, 237]. DNNs owe their success to their representational power and their capacity to learn sets of hierarchical features from simpler features due to their deep, multilayer architectures [98]. However, most of the state-of-the-art DL approaches used in prognostics provide mainly point estimates to their RUL predictions [177, 35, 26]. This is because DNNs do not inherently quantify the uncertainty associated with their predictions but instead treat their weights and biases as deterministic values. These predictions, though, are uncertain since they are prone to noise and wrong model inference (see Section 5.3.4).

Specifically, there are two sources of uncertainty, namely *epistemic* (or model) uncertainty and *aleatory* (or data) uncertainty [96]. The former occurs due to inadequate knowledge, data, and representational capacity of the model and the latter due to the inherent uncertainty of the data distribution [35, 4]. Additionally, from the nature of epistemic uncertainty we can see that it is a *reducible* part of the (total) uncertainty of a modeling process, as it can be reduced on the basis of additional information. On the contrary, aleatory uncertainty is an *irreducible* part of the (total) uncertainty, due to the inherently random effects in the data-generating process [96]. Most problems in engineering involve both sources of uncertainties. However, it may be difficult to distinguish whether a particular uncertainty should be put in the aleatory category or the epistemic category, in the modeling phase [120].

The lack of a measure of uncertainty, however, can lead to overly confident decisions [35, 74]. When it comes, for example, to cost-critical or safety-critical applications, it is necessary to know how much confidence a DL method has on its prognostic results and even more so when it comes to the RUL estimation [177, 26, 25, 35]. In addition, even though DNNs output predictive probabilities (e.g., image classification), these probabilities are falsely interpreted as model confidence [74]. For example, the probability of the softmax on the final layer of a neural network (NN) will not reflect if the network has knowledge of the input (see also adversarial examples [217]). Additionally, decision-making based on a single-point estimate is error-prone and leaves no room for the decision-maker to make an actionable choice [177]. When such an uncertainty estimate is available (see also Section 5.2) it is often the case that end-users and decision-makers need to choose by lacking broader information, such as distribution of predictions or other statistics that can assist the logistics further.

Furthermore, the end-user or researcher is faced with a multitude of decisions around the hyperparameters of the pre-processing of the data (e.g., label construction for RUL data), and of the learning algorithm (e.g., the number of layers in a DNN). Hyperparameters are not learnt but have to be set a-priori, and they have a large impact on the predictive performance

of a method but also uncertainty. On top of that, there can be hyperparameter configurations that allow low prediction error but have (relatively) large uncertainty and vice versa. In such scenarios, where trade-offs exist, it is vital to move towards a more *user-centric* approach, where the end-user can decide which hyperparameter configuration to adopt based on the criticality of the task. As such, hyperparameters need to be considered carefully both in terms of model accuracy and the uncertainty estimates.
The aforementioned statements motivate our main research question: Can we propose an automated framework for configuring RUL prediction models which are highly accurate and have less estimation uncertainty?
More specifically, our contributions are as follows:

1. We automatically optimize the hyperparameters of the Bayesian deep learning model through a Bayesian multi-objective optimization algorithm, jointly minimizing the RUL prediction error and the combined aleatory and epistemic uncertainties of the estimations. The reasoning behind this is that in certain tasks, there can be conflicts between these two objectives, as we briefly mentioned previously.

2. Together with the model hyperparameters, we further optimize the hyperparameters which are specific to the task of RUL estimation (the RUL label construction, see also Section 5.3), which is known to have an effect on the algorithmic performance [195]. We provide a thorough, end-to-end approach that can further assist researchers and end-users for offline RUL estimation.

3. We adopt a user-centric approach that allows the user to estimate the RUL based on the model output, as it promotes a more interpretable RUL decision. We demonstrate how survival curves can provide the end-user with information regarding the RUL and its confidence.

4. We evaluate our multi-objective hyperparameter optimization (HPO) approach against a single-objective HPO by taking the harmonic mean (HM) of the objectives. Our approach is validated on two subsets of the widely used C-MAPSS dataset [3].

The rest of the chapter is organized as follows. In Section 5.2, we present related work in this field and in Section 5.3, the proposed method and its modules are introduced. In Section 5.4 we present the dataset used and discuss the experimental results. Finally, in Section 5.5 we conclude and discuss the limitations of our framework and suggest future work.

5.2 Related Work

The field of PHM has been widely credited in the past years with numerous contributions from researchers. Academic interest, industrial applications, as well as the scientific challenge

of developing methods to forecast a failure, have been the driving forces. While model-based prognostic methods, such as Kalman filters and their variants [84, 110], take into account the modeling and data uncertainty, only a few studies in the data-driven domain address this matter, despite its importance [26]. Touching upon the previous statement, in this section, we will present related work in the context of uncertainty quantification (UQ) for the RUL estimation, attending only to data-driven approaches.

From the traditional ML methods, only Gaussian process regression (GPR) [185] (also known as Kriging) addresses UQ. GPR is a stochastic interpolation method where unseen locations of a stochastic process are estimated as a linear function of observed values. It can further be understood as a form of Bayesian inference. Specifically, GPR places a Gaussian prior over the functions that could have generated the observed data. Using Bayes's theorem by combining the Gaussian prior and the Gaussian likelihood function (for tractability), we get the predictive distribution for a new value. However, GPR might not be the optimal model for some data, e.g., if the data does not come from a Gaussian process, or the dimensionality is high. Furthermore, the data generating the predictions are not learnt automatically as in DL but need proper pre-processing (e.g., feature extraction), and also GPR variance is known that it can be over-optimistic [52].

In this view, from the data-driven approaches, we will only review recent work that adopted a DL solution. We made this decision because, as also mentioned in Section 5.1, DL is becoming prominent in data-driven prognostics, as well as there has recently been a lot of attention on UQ for DL [74, 28, 171, 4]. This collection is by no means exhaustive. We refer the interested reader to [230] and [128] for a more thorough overview of related work on PHM.

Epistemic Uncertainty The work by Peng et al. [177] is a recent data-driven example of UQ in prognostics. The authors present a DL approach from a Bayesian viewpoint to address the confidence of their RUL predictions and implement the Bayesian approximation using Monte Carlo Dropout (MC Dropout) [74] (see also Section 5.3.4). Kraus et al. [127] dealt with epistemic uncertainty in prognostics using variational inference (see also Section 5.3.4) and combine DL with notions from survival analysis to increase the intepretability of the estimation. In the same domain, Wang et al. [237] used MC Dropout to estimate the epistemic uncertainty of a recurrent convolutional neural network (RCNN) for the RUL estimation. However, none of the previous studies touched upon aleatory uncertainty.

Aleatory Uncertainty Zhao et al. [259], addressed the aleatory uncertainty by using a deep convolutional neural network (DCNN) through a shortened version of the ResNet [86] and assumed that the target RUL values follow a Gaussian distribution with parameters μ and σ being the network's outputs. They also adopted a non-parametric approach by combining the predicted RUL from the network with quantile regression, predicting this way multiple RUL at

different quantile levels. However, this approach did not take into account epistemic uncertainty and, to the extent of our knowledge, there was no HPO.

Epistemic and Aleatory Uncertainties Caceres et al. considered in [35] both epistemic and aleatory uncertainties. They used an explicit form of variational inference to account for the epistemic uncertainty and addressed the aleatory uncertainty by a probabilistic output layer parameterized by a Gaussian distribution and further performed HPO through grid search. In the same manner, Kim et al. [118], and Li et al. [139] designed RUL frameworks by taking into account the effects of both epistemic and aleatory uncertainties. They both used MC dropout to address the epistemic uncertainties. Kim et al. [118] addressed the aleatory uncertainty by a probabilistic output layer parameterized by a Gaussian distribution and assumed a monotonically decreasing relationship between the aleatory uncertainty and RUL, and further performed HPO on the number of hidden layers amongst other hyperparameters. Li et al. [139] modeled aleatory uncertainty by a probabilistic output layer following various types of lifetime distributions (Weibull, Gaussian, and Logistic). Benker et al. [25] adopted a Bayesian neural network and addressed both uncertainties as well, but took into account the aleatory uncertainty post-training. They further quantified the epistemic uncertainty using a Hamiltonian Monte Carlo method, a more efficient variant of the Markov Chain Monte Carlo (MCMC) methods in high dimensional spaces.
These recent studies have made a great contribution to the field of data-driven prognostics by proposing methods to account for and quantify the uncertainty of their predictions. Nonetheless, there remain perspectives to consider. In more detail, most of the literature reviewed ([259, 139, 25]) did not state any form of HPO and those that did ([177, 35, 118, 26]), did not optimize necessary hyperparameters in the pre-processing stage and used less efficient HPO techniques (e.g., grid search). What is more, the reviewed methods that perform some form of HPO used *only* the RUL prediction error as the only criterion to guide the HPO, as opposed to also taking into account the epistemic and aleatory uncertainties. Lastly, in our literature review, we did not come across any methods that allow the end-user to make an informed RUL prediction based on information output by the model.

5.3 Proposed Method

Our method works by training a Bayesian deep learning model on training data U presented in the form of multivariate time-series (see also Definition 3.1 in Section 3.1 for details in the notation). The steps of our method are summarized as:

1. Data pre-processing by removing any redundant signals, normalizing the remaining sensor values and performing a sliding window transformation.

2. Target-RUL construction to allow supervised learning.

3. Modeling using a Bayesian deep learning model and taking into account the uncertainty of the predictions.

4. Hyperparameter optimization of the hyperparameters of steps 1,2, and 3.

5.3.1 Pre-Processing

The pre-processing stage has already been introduced in Chapter 4. We refer the reader to Section 4.3.1 for a more in-detail discussion. In brief, pre-processing time-series data involves sensor selection and value normalization. Sensor selection involves filtering the available data from sensor measurements which, for example, either do not exhibit any correlation with the target or have strong correlations with other sensors. Furthermore, even if no correlation is present, but the sensor values do not exhibit any variation, these features can often be discarded as they do not add any valuable information.
Pre-processing also involves normalizing the available data to mitigate any effect that different ranges of values or large deviations can have in the subsequent learning phase. Two of the most often used normalization methods are Z-normalization and Min-max normalization (see also Section 4.3.1 for more details on these two normalization methods).
In this chapter, as a next step, for each $\boldsymbol{X_u}$ (see also Definition 3.1 in Section 3.1 for details in the notation), we perform a sliding window transformation with a sequence of length w (window size), in order to enclose the inputs into multidimensional sequential data, which are to be considered as one sample. This transformation allows one to increase the number of training data, standardize the sample input lengths, and accelerate model training [35]. *For this work, the window size w is treated as pre-processing hyperparameter.*

5.3.2 Target-RUL Construction

We would like to tackle this problem as a regression problem. For the target-RUL construction we followed the same process described in Chapter 4. We refer the reader to Section 4.3.2 for a more in-detail discussion. In brief, there are two popular ways to create these labels, namely *linear* and *piece-wise linear* methods [195]. The former interprets the RUL in the strictest sense, as time to failure. Thus, every time-step is mapped to a value equal to $EoL - t$, where t is the current time-step. This approach, however, implies that the health of the system degrades linearly with usage [195]. The latter reflects the fact that initially the degradation is negligible, and after a specific point in time, it becomes more evident (see Figure 4.2 for an example). The point after which the RUL degrades linearly is called the *reflection point* [94]. This, way we can construct an RUL curve for each $u \in U$, by mapping each rolling window to the RUL

at the end of that window. For this work, the type of label creation method and the reflection point are treated as pre-processing hyperparameters.

5.3.3 Modeling

As mentioned in Section 5.1, amongst the data-driven methods employed for prognostics DNNs have proven to be good candidates due to their representational power [138, 25, 112, 35, 177, 237]. In general, shallow learning methods are not designed for large-scale datasets and, more importantly, need extensive feature engineering efforts [262]. In this view, we decided to employ DL to address the RUL estimation problem. As this task is based on sequential data (multivariate time-series), we decided to use recurrent layers and specifically gated recurrent unit (GRU) layers as the model base due to their lower complexity and similarly good performance in modeling long dependencies [139], when compared to long short-term memory (LSTM) layers.

5.3.4 Uncertainty Quantification

As discussed briefly in Section 5.1 predictions made by neural networks are inherently uncertain, as they are prone to noise and/or wrong model inference. At the same time, however, NNs treat their weights and biases as deterministic values. This results in NNs being overly confident, even when they should *not* be. In general, there are two sources of uncertainty. In the context of NN, the epistemic and aleatory uncertainties can be considered by putting a prior on model parameters or the outputs. The latter means assuming that the model outputs follow a specific distribution, such as Weibull. The former can be addressed by treating the weights and biases (we jointly note them as W) of the network as random variables, defining a prior over them, and then using Bayesian inference to learn the posterior distributions of the network's weights [177, 262, 35] as:

$$p(W|X,Y) = \frac{p(Y|X,W)p(W)}{p(X,Y)}, \tag{5.1}$$

where X, Y are the training data and their labels, respectively. The posterior distribution on the network's parameters is, however, computationally intractable even for NNs of any practical size, as the number of parameters is very large and the functional form of a NN does not allow for exact integration [28, 74, 35]. Moreover, the denominator in Equation 5.1 is unavailable in closed form or requires exponential time to compute [27].

A large part of ongoing research is focused on approximating such posterior distributions [26]. Amongst these, prominent methods are Markov Chain Monte Carlo (MCMC) methods and its variants, and variational inference (VI) [26, 262, 27, 35]. The former, generally, converge slowly and are computationally expensive for large datasets or complex models. Instead, variational inference solves the same problem by using optimization techniques rather than sampling methods like MCMC [27]. Specifically, variational inference sidesteps the difficulty mentioned above

altogether by defining an approximate variational distribution $q(W)$ from a distributional family $\mathbb{D}$, that is the best approximation to the exact posterior $p(W|X,Y)$, with respect to the Kullback-Leibler (KL) divergence. This means that,

$$q^*(W) = \underset{q(W)\in\mathbb{D}}{\arg\min}\, KL(q(W) \,||\, p(W|X,Y)), \tag{5.2}$$

where $KL(q(W) \,||\, p(W|X,Y))$ is defined as:

$$KL(q(W) \,||\, p(W|X,Y)) = \mathbb{E}_{q(W)}\left[\log \frac{q(W)}{p(W|X,Y)}\right] \tag{5.3}$$

However, because Equation 5.2 is intractable[3] VI maximizes instead what is called the evidence lower bound (ELBO), which is defined as:

$$ELBO(q(W)) = \mathbb{E}\left[\log(p(X,Y|W)\right] - KL(q(W) \,||\, p(W)) \tag{5.4}$$

In turn, though, exactly maximizing Equation 5.4 is computationally prohibitive. To address this, variational inference can be divided into methods that implicitly use model uncertainties, such as MC Dropout [74] and methods that explicitly model weight parameters as probability distributions such as Bayes-by-Backprop [28, 35, 262].

In this work, we have decided to use MC Dropout to model the **epistemic uncertainty** due to its simplicity, scalability, and computational efficiency compared to other Bayesian deep learning approaches [74, 118]. It is implemented through gradient-based learning methods and stochastic regularization techniques, which are widely available in existing DL libraries [177]. MC Dropout is, in essence, regular dropout applied at both training and inference steps. The addition of dropout between every layer can switch off some portion of neurons in each layer and generate random predictions as samples from a probability distribution that is considered equivalent to performing approximate VI. In more detail, MC Dropout showed that by choosing a specific form of an approximate distribution q, as a distribution over matrices whose columns are randomly set to zero, the VI in a NN can be interpreted as performing one forward pass through the NN with dropout. For more details on MC Dropout, see [74] and the accompanying appendix.

We should note here that there is a current debate as to the validity of MC Dropout being Bayesian [35, 262, 171]. In [171], Osband et al. highlighted that a shortcoming of MC Dropout is that the dropout rate does not depend on the data, which translates into the fact that employing dropout for posterior approximation cannot say anything about a set of data being observed once or more times. This, of course, can have significant implications in support of reliable uncertainty quantification and consequently deserves attention. As this work was

[3]See [27] page 6 for details.

mainly devoted to the usage of bi-objective HPO and user-centric approach, we have decided to address this *highly relevant but challenging issue* in future work.

Finally, in order to model the **aleatory uncertainty**, inspired by [154], we further assume that the RUL values follow a Weibull distribution, the reason being that Weibull is extensively employed in survival and reliability analysis to model times-to-failure. Moreover, it is simple, but also expressive, being able to take various forms, such as the exponential distribution [139]. The probability density function (PDF) of the 2-parameter Weibull that we used is defined as:

$$f(x) = \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{b-1} e^{-(x/\alpha)^{\beta}}\ , \tag{5.5}$$

for $x \geq 0,\ \alpha, \beta \in (0, +\infty)$, where α is the scale parameter and β the shape parameter of the distribution.

In this view and to adopt a user-centric approach for the RUL estimation (3$^{\text{rd}}$ contribution), the output layer of the DNN (see Section 5.3.3) will output the parameters of the Weibull distribution, α, β. This is a more user-centric approach, as for a sample input (e.g., a sequence of sensor values), the end-user is presented with the parameters that govern the distribution of the times-to-failure. This allows for more informative and interpretable decision-making in subsequent steps. The end-user can decide himself what statistics or percentiles (e.g., the mean-time-to-failure (MTTF)) to use as the point estimate of the RUL and the overall knowledge of the distribution of failure times can allow decision-makers to reason if the results are plausible or not. This contrasts with most methods that return a point-estimate to the end-user.

5.3.5 Hyperparameter Optimization

The optimization of hyperparameters enhances the performance of a machine learning algorithm, and thus, HPO is considered an important step in developing AI and ML frameworks. Various methods and algorithms are available for HPO, such as grid search (GS), random search (RS), evolutionary algorithms (EA), and Bayesian optimization (BO) [68]. In this study, a *bi-objective* variant of a state-of-the-art BO algorithm, namely *Mixed-integer Parallel Efficient Global Optimization* (MIP-EGO), is chosen due to its efficiency for optimizing expensive problems [231]. MIP-EGO is based on efficient global optimization (EGO), also known as Bayesian optimization (BO). The algorithm uses random forest (RF) models to handle mixed integer data and mixed integer evolution strategies (MIES) as internal optimizer. The bi-objective variant of MIP-EGO uses the S-metric hyper-volume (see also Section 5.4.3) improvement infill criterion to select new candidate solutions.

In order to perform the HPO of the Bayesian deep learning and the problem-specific preprocessing hyperparameters by jointly optimizing the prediction error and uncertainty address (1$^{\text{st}}$ and 2$^{\text{nd}}$ contributions), MIP-EGO is set to determine the hyperparameter values that

minimize simultaneously the pointwise root mean squared error (RMSE) and the uncertainty by optimizing the bi-objective function described in Algorithm 5.1[4]. In more detail, MIP-EGO will evaluate different configurations h_p by pre-processing the data and training a DNN (lines 1 and 2). In lines $5-15$ the trained network is used to make predictions on each sample of the validation set (size m) (see Section 5.4.2 for information on the validation set) by multiple passes R which output different α, β at each pass using MC Dropout (see Section 5.3.4). To determine the RUL estimate for an input sample, we calculated the median of the predicted α values ($\bar{\alpha}$) and the median of the predicted β values ($\bar{\beta}$) (line 11) and used the mean-time-to-failure (MTTF) of the Weibull distribution with parameters the calculated medians (line 13). The choice of the MTTF was to reduce the selection bias to any statistic and the choice of median to counteract effects of possible outliers. Of course, any other statistic could be used here. The mean-time-to-failure is defined as: $MTTF(\alpha, \beta) = \alpha\Gamma(1 + 1/\beta)$, where Γ is the gamma function. For the over all point-wise performance, f_1, we calculated the RMSE between the predicted RUL (over all the instances) and the ground truth values (line 16). To determine the uncertainty for an input sample, we calculated the standard deviation of the predicted α values ($\widehat{\alpha}$) and the standard deviation of the predicted β values ($\widehat{\beta}$) (line 12) and averaged the two values (line 14). For the overall uncertainty f_2, we calculated the average over all the uncertainties (line 17).

5.4 Experimental Setup and Results

We are interested in investigating the existence and trade-offs between the RUL prediction error and the prediction uncertainty when using bi-objective HPO, and to examine the advantages that can be gained compared to using a single-objective variant. Furthermore, we show how the proposed method can be more user-centric compared to the current techniques. Datasets and experimental results are described in this section.

5.4.1 Data

In this study, we use the widely used C-MAPSS benchmark dataset [3]. The dataset was released in 2008 [198] and it has been used in the field of PHM ever since, to develop techniques and methods for estimating the RUL [183, 128]. It is a simulated turbofan engine degradation dataset from NASA's Prognostics Centre of Excellence[5]. The dataset consists of four subsets: FD001, FD002, FD003, and FD004, each of each exhibits a different number of operating conditions and fault modes. In this work, we used datasets FD001 and FD003, which exhibited

[4]Please note that some of the notations in the pseudocode of Algorithm 5.1 differ from the notations in the pseudocode of the original publication [116]. We did this for clarity, as well as for consistency among the chapters of this thesis.

[5]`https://ti.arc.nasa.gov/tech/dash/groups/pcoe/`

Algorithm 5.1: Bi-objective Function

Data: X, V, hp, R ; `# Training data, validation data, hyperparameter configuration, sample size of MC Dropout passes`

Result: f_1, f_2 ; `# RMSE, uncertainty`

$X', V', Y_{X'}, Y_{V'} \leftarrow \text{Pre\_processing}(X, V, hp)$; `# Data pre-processing and RUL creation for the training and validation data (see Sections 5.3.1 and 5.3.2)`

$M \leftarrow \text{DNN}(X', V', Y_{X'}, Y_{V'}, hp)$; `# Model training`

$m \leftarrow |V'|$;

$RUL \leftarrow [\,]$; $Var \leftarrow [\,]$; `# Initializing empty lists for RUL and uncertainty`

for $i \leftarrow 1$ **to** m **do**

  $A \leftarrow [\,]$; $B \leftarrow [\,]$;

  for $j \leftarrow 1$ **to** R **do**

    $\alpha, \beta \leftarrow M(V_i)$; `# Predicting using trained DNN through MC Dropout (see Section 5.3.4)`

    $A \leftarrow A\,\alpha$; $B \leftarrow B\,\beta$; `# Appending` α `and` β `into A and B lists`

  end

  $\bar{\alpha} \leftarrow median(A)$; $\bar{\beta} \leftarrow median(B)$; `# Median values of A and B`

  $\widehat{\alpha} \leftarrow std(A)$; $\widehat{\beta} \leftarrow std(B)$; `# Standard deviations of A and B`

  $RUL \leftarrow RUL\ \mathbb{E}[Weibull(\bar{\alpha}, \bar{\beta})]$; `# Appending calculated RUL into RUL list`

  $Var \leftarrow Var\ mean([\widehat{\alpha}, \widehat{\beta}])$; `# Appending average between` $\widehat{\alpha}, \widehat{\beta}$ `to Var list`

end

$f_1 \leftarrow RMSE(RUL, Y_{V'})$; `# Root mean squared error`

$f_2 \leftarrow mean(Var)$; `# Average value of Var`

the same number of operating conditions but different number of fault modes. Each of these datasets is arranged in an $n \times 26$ matrix where n corresponds to the number of data points (samples) in each unit and 26 is the number of columns/features. Each row is a snapshot of data taken during a single operating time cycle. Regarding the 26 features, the 1st represents the engine number, the 2nd represents the operational cycle number. Features $3-5$ represent the operational settings, and features $6-26$ represent the 21 sensor values. Engine performance can be significantly affected by the three operating settings. More information about these 21 sensors can be found in [170]. What is more, each subset exhibits a different number of faults (see Table 5.1).

Each of these subsets are further split into training set and test set (see Table 5.1 for details). For each engine trajectory within the training sets, the last data entry corresponds to the end-of-life (EoL) of the engine, i.e., the moment the engine is declared unhealthy or in failure status. The test sets contain data up to some time before the failure and the aim here is to predict the RUL for each of the test engines.

These multivariate time-series are from a different engine i.e., the data can be considered to be from a fleet of engines, of the same type though, and each trajectory is assumed to be the life-

cycle of an engine. Every engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition.

To compare the model performance on the test data, we need some objective performance measures. In this study, we us the Root Mean Square Error (RMSE) [260, 146, 143], defined as: $RMSE = \sqrt{1/n \sum_{i=1}^{n} d_i^2}$, where $d_i = \widehat{RUL_i} - RUL_i$, $\widehat{RUL_i}$ is the estimated RUL and RUL_i is the ground truth RUL for instance (engine) i, respectively.

Table 5.1: FD001 and FD003 C-MAPSS dataset details.

Dataset	**FD001**	**FD003**
Train trajectories	100	100
Test trajectories	100	100
Operating conditions	1	1
Fault conditions	1	2
Max train trajectory (cycles)	362	525
Min train trajectory (cycles)	128	145
Max test trajectory (cycles)	303	475
Min test trajectory (cycles)	31	38
Training samples	20631	24720

5.4.2 Experimental Setup

The experiments[6] were executed on 10 *NVIDIA Tesla T4* GPUs, of 16GB, *GDDR6* memory. Source code has been developed in `Python` V3.8.8[7]. Experimental time was around 3-5 days (wall clock time), per dataset.

We began by randomly selecting 80% of units from the training set and using the remaining 20% as the validation set to select the hyperparameters. We then randomly truncate the trajectories of the validation set at five different locations such that five different cases are obtained from each trajectory following [153]. The truncation is needed to replicate the dedicated test data, i.e., trajectories up to some time before the failure. Note here, however, that we did not use any information from the dedicated test set. Minimum truncation is 5% of the total life, and maximum truncation is 96% of the total life. We continued with the pre-processing of the training and validation sets. In more detail, we normalized the data transforming the 3 operational settings and 21 sensor values to the range $[-1, 1]$ (min-max normalization) and discarded any of them that have zero variance. Constant values do not provide any useful degradation information for determining the RUL.

[6]The source code of the experiments can be found at `https://github.com/MariosKef/RULe`.

[7]We used `tensorflow`(2.5.0), `scikit-learn`(0.24.1), `pandas`(1.2.3), `numpy`(1.19.5).

For the next steps of the pre-processing and data transformation (sliding window and RUL target construction), as well as for the DNN training, we performed HPO to select their optimal hyperparameter values that optimize simultaneously the pointwise RMSE and the uncertainty, in order to address our 1st and 2nd contributions (see Section 5.3.5). The tuned hyperparameters and their respective ranges can be seen in Table 5.2. Note that the search space contains not only integer variables but also categorical ones. We executed the hyperparameter optimization (see Section 5.3.5) with a budget of 300 function evaluations (of which 100 are initial configurations sampled with the latin hypercube sampling (LHS) method). Moreover, the MIP-EGO configurator is set to evaluate 10 configurations per step in parallel for FD001 and 9 configurations for dataset FD003[8].

Following the hyperparameter optimization phase, we are presented with a two-dimensional set of points showing the RMSE and UQ on the validation set. Each point corresponds to a specific hyperparameter configuration. By considering only the non-dominated solutions, we end up with (an approximation to) the Pareto front. The Pareto front is set of points, which cannot be improved with respect to one objective without making another objective worse [62] (see blue points in Figure 5.1). The non-dominated set of solutions delivers hyperparameter configurations which allow us to view the trade-offs between the RMSE and the UQ. We can subsequently pre-process and train on the *entirety* of the training data (training and validation) using the configurations corresponding to the points on the Pareto front and finally test our method on the dedicated test set. During this stage, we use Algorithm 5.1 by inputting as X the entire training set, V the dedicated test set, and h_p the configuration corresponding to the selected point from the Pareto front.

Additionally, we used the Adam optimizer [119] with a clip value of 0.5, $R = 30$ for the number of MC Dropout passes, and trained for 100 epochs with early-stopping (patience = 5). Finally, since we want our DNN to learn the relationship between the input sequences and the Weibull parameters, we used as a loss function the *negative log-likelihood of the* 2*-parameter Weibull distribution* [247, 154] to train the network.

Baseline

We also performed a baseline experiment to evaluate the bi-objective hyperparameter approach. Our baseline differs from the work we reviewed in Section 5.2, as none of the related work took into account the joint optimization of the RMSE and the uncertainty. Our baseline transforms the bi-objective optimization problem into a single-objective by minimizing the harmonic mean (HM) of the RMSE and uncertainty, as:

$$HM = \frac{2}{RMSE^{-1} + Uncertainty^{-1}} \tag{5.6}$$

[8]This was a result of GPU availability. In any case, this did not affect the validity of the computations.

For this task we used the single-objective MIP-EGO, which uses the so-called Moment-Generating Function (MGF) based infill-criterion [239] to select new candidate solutions. Moreover, the MIP-EGO configurator is set to evaluate 10 configurations per step in parallel for FD001 and FD003, for a maximum of 300 function evaluations. We used this baseline in order to investigate the benefits of using the bi-objective HPO compared to the single-objective approach. The reason of taking the HM compared to e.g., the arithmetic mean, is because it is less susceptible to fluctuation of the observations, thus making it a more ideal baseline for this first study.

5.4.3 Hypervolume Indicator

To compare the bi-objective HPO approach to the single-objective approach based on the HM we decided to use the hypervolume indicator (HVI). The HVI or S-metric [263] is the hypervolume in the objective space $\mathbb{R}^m$ that is dominated by the Pareto points bounded by a reference point $y_{ref} \in \mathbb{R}^m$. The reason for choosing the HVI as a measure of comparison is that it is intuitive, as dominating a large part of the objective space is desirable. Furthermore, the HVI is widely used in evaluating the performance of various multi-objective optimization algorithms.

Table 5.2: Hyperparameters in the model development for the C-MAPSS dataset.

Type	**Hyperparameter**	**Search Space**
Pre-processing	Sliding window size	[20, 50]
	Reflection point (percentage of total life)	[25, 75]
	Initial RUL value	[110, 130]
	RUL degradation style	['linear', 'nonlinear']
DNN	Number of recurrent layers	[1, 3]
	Number of dense layers	[1, 3]
	Number of neurons per layer	[10, 100]
	Activations	['tanh', 'sigmoid']
	Recurrent dropout rate	[1e-5, 0.9]
	Dropout rate	[1e-5, 0.9]
	Output activations	['softplus', 'exp']
	Learning rate	[1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 2e-5]
	Batch size	[32,64,128]

5.4.4 Results and Discussion

Having generated the Pareto front of the hyperparameter configurations (see Section 5.4.2) we selected each configuration, trained on the entirety of the dataset and made inferences about

both the training and (dedicated) test data.

Figures 5.1 and 5.2 show in blue circles the Pareto front of the hyperparameter configurations performance on the validation sets of datasets FD001 and FD003, respectively. The red triangles depict the results on the dedicated test set (dominated solutions might exist). The number next to each point represents the hyperparameter configuration giving rise to that specific solution and are shown here to manifest how the solutions' topology changes when validated on the dedicated test set.

In order to see if the neural network can learn from the data, in Figures 5.3 and 5.4, we show the evolution, over time, of the Weibull PDFs, of units 2 and 9 from the FD001 and FD003 training data, respectively. We do this by plotting the Weibull PDFs per time-index of the units' data. For this task, we used the models which returned the lowest RMSE on the dedicated test sets of FD001 and FD003 (points with green shade in Figures 5.1 and 5.2). In Figures 5.3 and 5.4, we can see that as the time-index of the data increases (darker-red shades in the legend), the PDFs variance decreases. Even though the distributions' variance does not initially seem to be monotonically decreasing, as we approach the end-of-life of the assets (darker-red shades), we can see that the variance decreases, giving more mass to the expected time-to-failure, and that the expected time-to-failure approaches 0. This is a desirable property as it indicates that the model can learn the correct failure dynamics because the more time-steps have passed, the more data has been collected, and consequently, there is more degradation information, especially near the end-of-life of the asset.

In Figures 5.5 and 5.6 we show the evolution of the HVI per a maximum of 300 function evaluations between the bi-objective and single-objective HPO. To be able to compare the HVI of the single-objective approach to the bi-objective approach, we calculated the HVI of the Pareto efficient solutions of the RMSE and uncertainty as pre-images of the HM. Furthermore, we normalized both objectives to $[0, 1]$ and used as $y_{ref} = (1.1, 1.1)$.

We can see from the two figures that the HVI of the single-objective approach and the bi-objective approach plateau to the same final HVI, albeit the bi-objective approach reaches the plateau in fewer iterations, on FD001, whereas on FD003, the single-objective approach reaches the plateau in slightly fewer iterations than the bi-objective method. The HVI might indicate that the harmonic mean manages to also identify a balance between the objectives and can be used as an alternative to the bi-objective HPO. The seemingly smaller number of function evaluations of the single-objective approach in the figures, compared to the bi-objective approach, is simply an artifact of infeasible configurations that were discarded by the single-objective MIP-EGO.

Examining Figures 5.1 and 5.7 we can see that the bi-objective approach returned more hyperparameter configurations lying on the Pareto front (7 blue points on Figure 5.1) compared to the single-objective approach (6 blue points on Figure 5.7). Even though the number is marginally larger, this suggests that the bi-objective approach might be more suitable for iden-

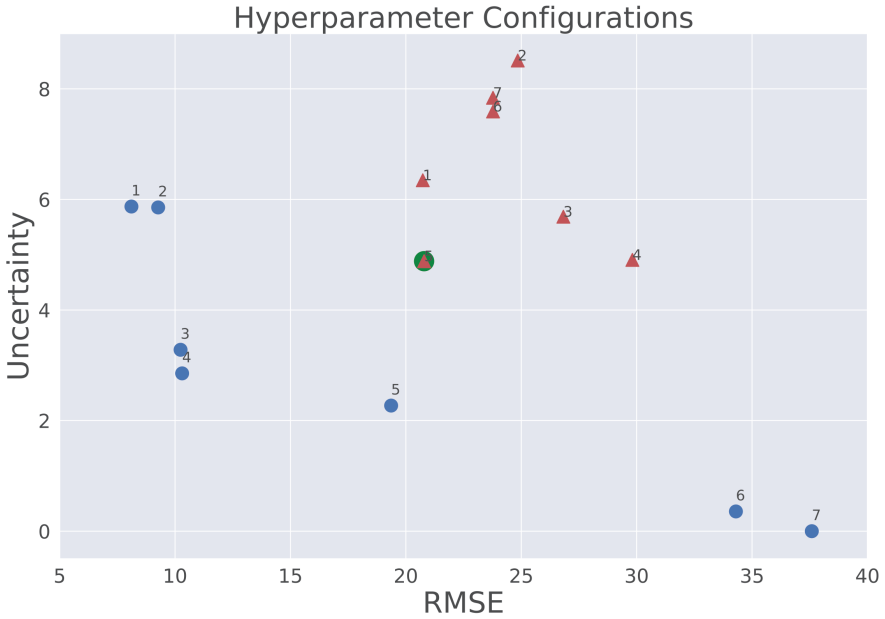

Figure 5.1: RMSE-UQ points corresponding to the hyperparameter configurations on FD001 using the bi-objective approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set. The green point represents the model with the lowest RMSE on the dedicated test set.

tifying a larger and more diverse set of hyperparameters. Moreover, it is interesting to see that the configurations returned from the two HPO methods (blue points in Figures 5.1 and 5.7) present similar values of uncertainty, even though more than 80% of the configurations of the single-objective HPO exhibit uncertainty lower than 2, with that number being around 29% for the bi-objective HPO. Regarding RMSE, however, we observe the inverse trend. In the bi-objective method, more than 70% of the returned configurations result in RMSE lower than 20, with this number being 50% in the single-objective approach. In addition, we can see that the performance of the resulting hyperparameters (blue points) on the dedicated test set (red triangles) differs between the two figures. Firstly, in the bi-objective approach, the performances on the dedicated test set per hyperparameter configuration are clustered together when compared to the single-objective approach in Figure 5.7 where the points are spread out more, especially in the uncertainty axis. Secondly, in the bi-objective method, the RMSE and uncertainty values of the dedicated test set lie in the range of $[20.73, 29.82]$ and $[4.88, 8.51]$, respectively. In the single-objective method these ranges are $[25.97, 37.51]$ and $[0, 7.93]$, respectively, for the RMSE and uncertainty. It is interesting to see that the bi-objective HPO returned better scores for

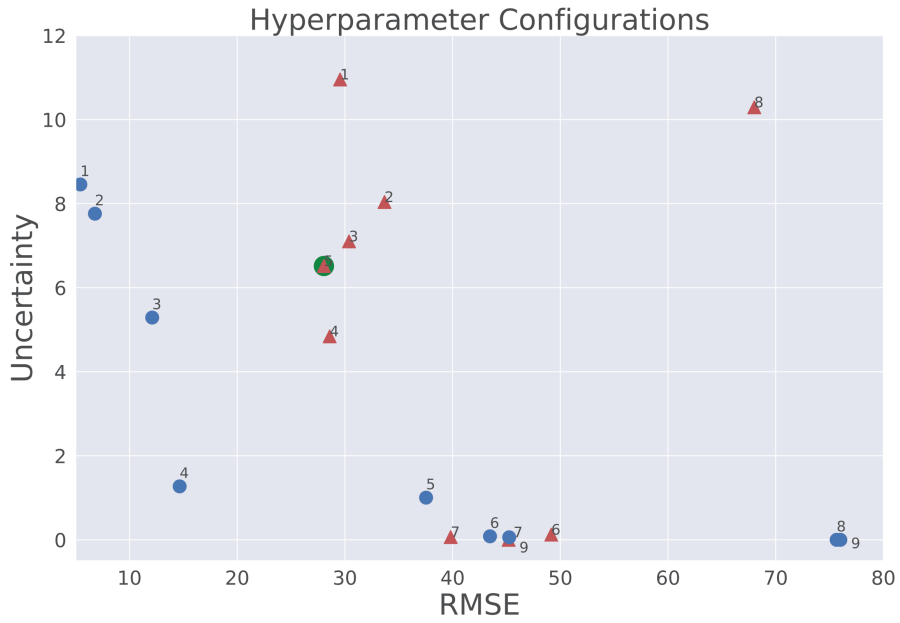

Figure 5.2: RMSE-UQ points corresponding to the hyperparameter configurations on FD003 using the bi-objective approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set. The green point represents the model with the lowest RMSE on the dedicated test set.

the RMSE and more "concentrated scores" for the uncertainty compared to the single-objective approach.

Regarding FD003 when examining Figures 5.2 and 5.8 we can see that the bi-objective approach returned, again, a larger number of hyperparameter configurations lying on the Pareto front (9 blue points on Figure 5.2) compared to the single-objective approach (7 blue points on Figure 5.8). Even though the number is marginally larger, this suggests, like previously, that the bi-objective approach might be more suitable for identifying a larger and more diverse set of hyperparameters. In the bi-objective method, around 44% of the returned configurations result in RMSE lower than 20, with this number being around 57% in the single-objective approach. Nevertheless, we observe that the hyperparameter configurations from the bi-objective approach returned overall configurations with lower levels of uncertainty compared to the single-objective method. Specifically, more than 66% of the configurations on the bi-objective HPO result in uncertainty that is less than 2, with this number being around 43% in the single-objective HPO. Regarding the resulting hyperparameters' performance (blue points) on the dedicated test set (red triangles), there are no apparent differences between the two methods' topologies.

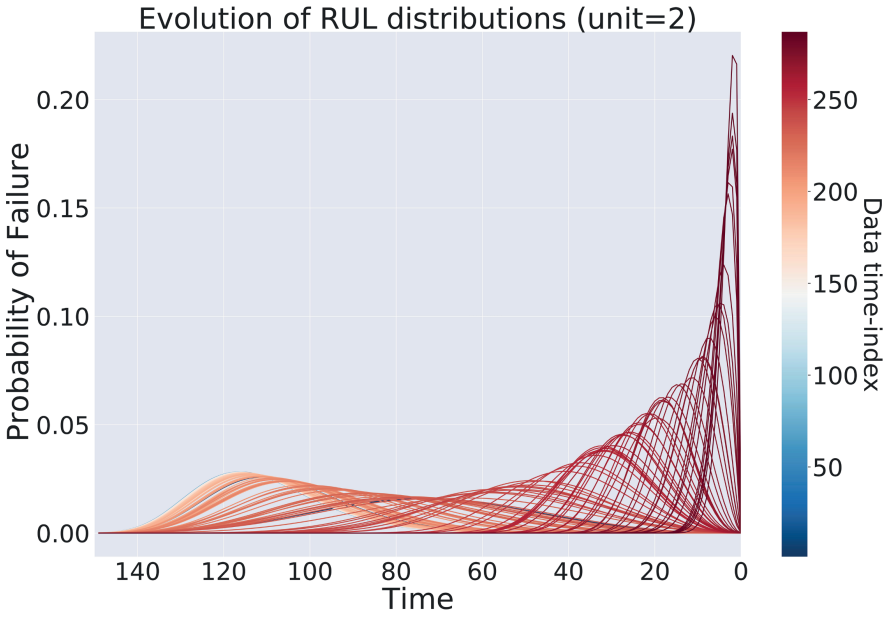

Figure 5.3: Evolution of Weibull distributions of unit 2 from FD001. Blue shades indicate the start of the unit's trajectory and red shades the end. Note that the x-axis is inverted for clarity.

Lastly, in the bi-objective method, the RMSE and uncertainty values of the dedicated test set lie in the range of $[28.05, 68.01]$ and $[0, 10.96]$, respectively. In the single-objective method, these ranges are $[23.82, 50.76]$ and $[0.14, 18.53]$, respectively, for the RMSE and uncertainty. This shows that for this dataset, the bi-objective method returned lower uncertainty values, but the single-objective approach returned RMSE values that lie in a more favorable range, thus indicating no clear winner.

From the previous results, we conclude that the usage of bi-objective HPO can reveal interesting trade-offs between the RMSE and uncertainty. Additionally, the results show that even though the bi-objective approach can return more configurations on the Pareto front, the single-objective HPO is also a good alternative for this task. The differences in the experimental findings between the two datasets might be justified by the the fact that FD003 has 2 simulated fault conditions compared to FD001. In addition, we cannot rule out that the maximum allowable number of function evaluations or training epochs might have affected the findings, as more epochs might allow the network to learn more. More function evaluations of the HPO, on the other hand, will explore a larger part of the hyperparameter configuration space which might uncover more promising configurations.

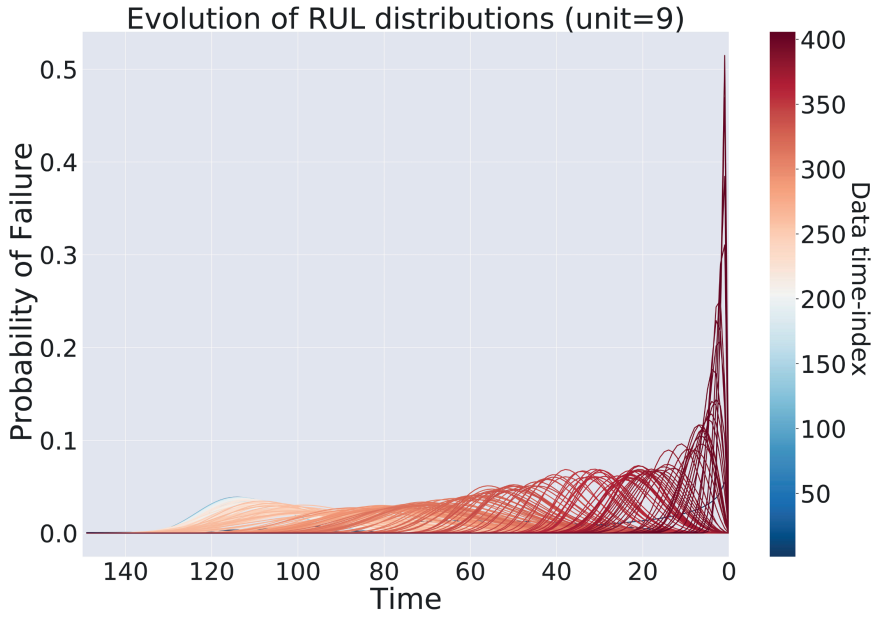

Figure 5.4: Evolution of Weibull distributions of unit 9 from FD003. Blue shades indicate the start of the unit's trajectory and red shades the end. Note that the x-axis is inverted for clarity.

5.4.5 Application

Next, we will demonstrate how the proposed method can allow a more user-centric and interpretable approach to end-users (3$^{\text{rd}}$ contribution). For this application, we used the models which returned the lowest RMSE on the *dedicated test sets* of FD001 and FD003. These points are indicated with a green marker on Figures 5.1 and 5.2. Specifically, since the trained network outputs the α and β parameters per input sample, the end-user can utilize this information to visualize, for example, the survival curves corresponding to each input sample, as well as other important information.

Survival curves are visualization methods from survival analysis that show the probability of an event *not* happening up to a point in time. In our case, this means that a failure has not occurred up to a point it time t (hence the asset will survive longer than t). A survival curve is defined as $1-$CDF, where CDF stands for the cumulative distribution function (in this case, the Weibull's CDF). For example in Figures 5.9 and 5.10 we plot the survival curves of test units $81, 4$ from the FD001 dataset and test units $28, 3$ from the FD003 dataset. For each test unit, we plot *all* the survival curves (shown within shaded areas for clarity) resulting from the multiple values of α and β that the network outputs through the MC Dropout, as well as the

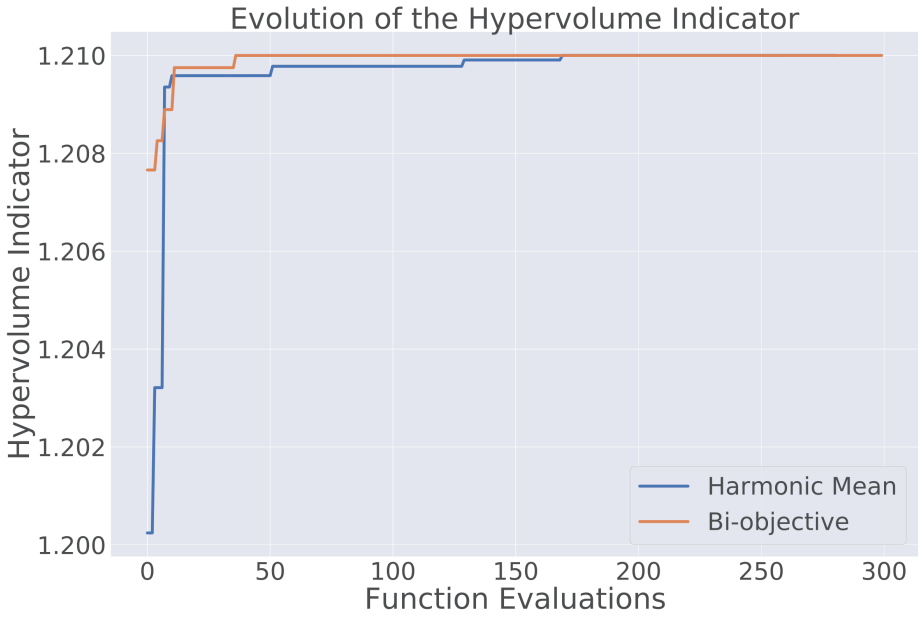

Figure 5.5: Evolution of the HVI of the bi-objective HPO and the single-objective HPO on FD001.

"median" curves that have as parameters the median values of the αs and βs, for a reference. This allows two things: the end-user can visually inspect the survival curves and, for instance, select a probability-of-survival threshold, based on one of them (e.g., the "median" curve), after which a unit should be maintained. Additionally, based on how wide the shaded areas are, the user can decide whether to employ the recommendation or proceed to further actions, such as further inspection by a field expert. For example, in Figure 5.10 the "median" survival curve of test unit 28 tells us that the probability of not having a failure up to time 100 from the current point in time (time 0) is about 80% and that this estimation is "more confident" compared to that of test unit 3, as the shaded area is less wide than the shaded area of test unit 3. Similarly, in Figure 5.9 the estimation of the survival curves of test unit 81 is "more confident" compared to that of test unit 4.

5.5 Discussions and Conclusions

In this work, we dealt with the remaining useful life (RUL) estimation using Bayesian deep learning by taking into consideration the uncertainty of the estimate together with the pre-

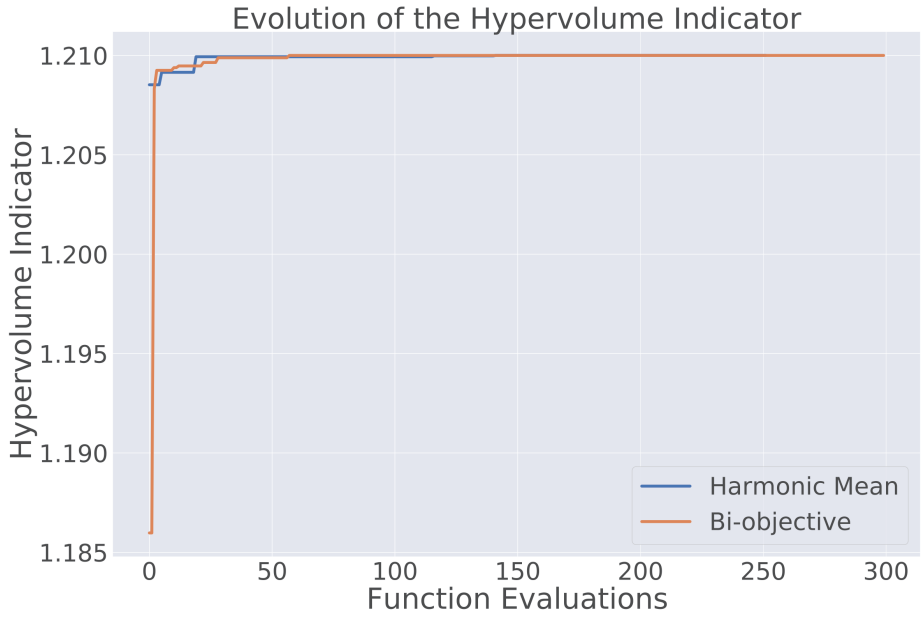

Figure 5.6: Evolution of the HVI of the bi-objective HPO and the single-objective HPO on FD003.

dicted point estimate. We investigated the first, to our knowledge, usage of bi-objective hyperparameter optimization (HPO) that minimizes simultaneously the pointwise RMSE and the uncertainty. In this direction, we optimized together with the hyperparameters of the neural network (NN) the hyperparameters that govern the pre-processing steps, delivering thus, an end-to-end, data-driven, pipeline for the (offline) RUL prediction. We validated our approach on two subsets of the widely used C-MAPSS dataset [3]. We, further, demonstrated how survival curves can provide the end-user with information regarding the RUL and its confidence. The experimental results indicate that, the bi-objective HPO might be more suitable for identifying a larger and more diverse set of hyperparameter configurations compared to the single-objective HPO that aggregates the two objectives through the harmonic mean (HM). However, both methods reach the same hypervolume indicator value of the Pareto front in, more or less, the same number of function evaluations and the findings did not indicate whether a method is more suitable for lower uncertainty or lower RMSE scores. Regarding the performance of the Pareto front configurations, when validated on the dedicated test sets, there was no clear winner between the two methods, although in the first examined case the RMSE values are better and the overall performance scores are clustered together. Overall, the results show that, for the examined cases, the bi-objective method is able to suggest more hyperparameter

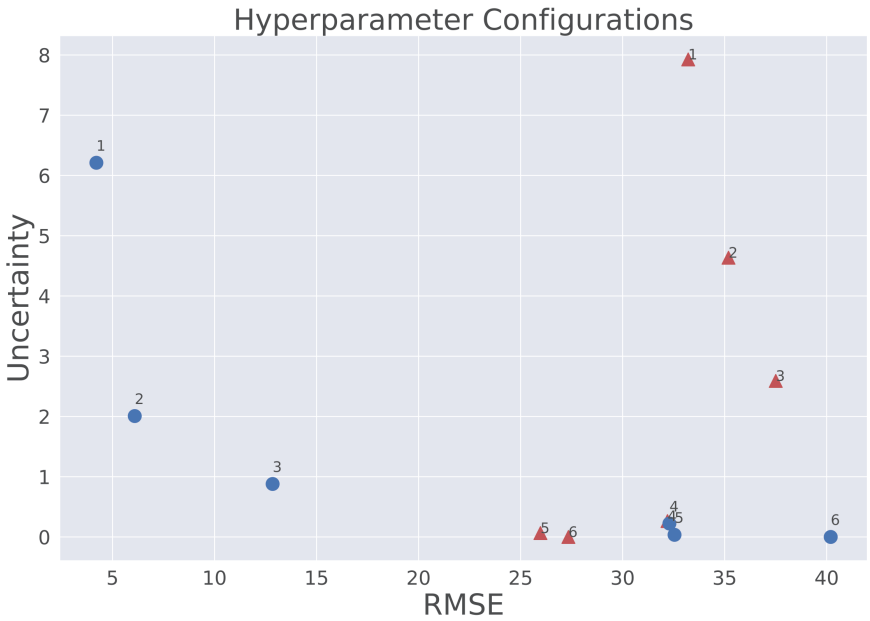

Figure 5.7: RMSE-UQ points corresponding to the hyperparameter configurations on FD001 using the harmonic mean approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set.

configurations and that the single-objective alternative is able to compete in terms of scores. This suggests that for a certain class of problems single-objective HPO methods are sufficient, allowing practitioners an ample selection of efficient single-objective HPO methods.

Concerning the limitations of our work, due to the high computational costs of running the experiments multiple times no statistical significance tests are performed. Despite that fact, our methodology is experimentally sound and suggests an alternative approach for HPO in PHM. Furthermore, as indicated, we are aware that there is a current debate as to the validity of Monte Carlo Dropout being Bayesian [171]. This could, in turn, make the corresponding predictive models problematic in support of reliable uncertainty quantification. As this work was mainly devoted to the usage of bi-objective hyperparameter optimization and user-centric approach, we have decided to address this highly relevant but challenging issue in future work. Future work should, in general, emphasize research on computationally efficient and accurate uncertainty quantification of DL models, as this will further open the road of AI applied in real-world applications.

Finally, we would be very interested in extending the bi-objective HPO to a many-objective context (> 2 objectives) to add more objectives, such as run-time, to find a compromise between

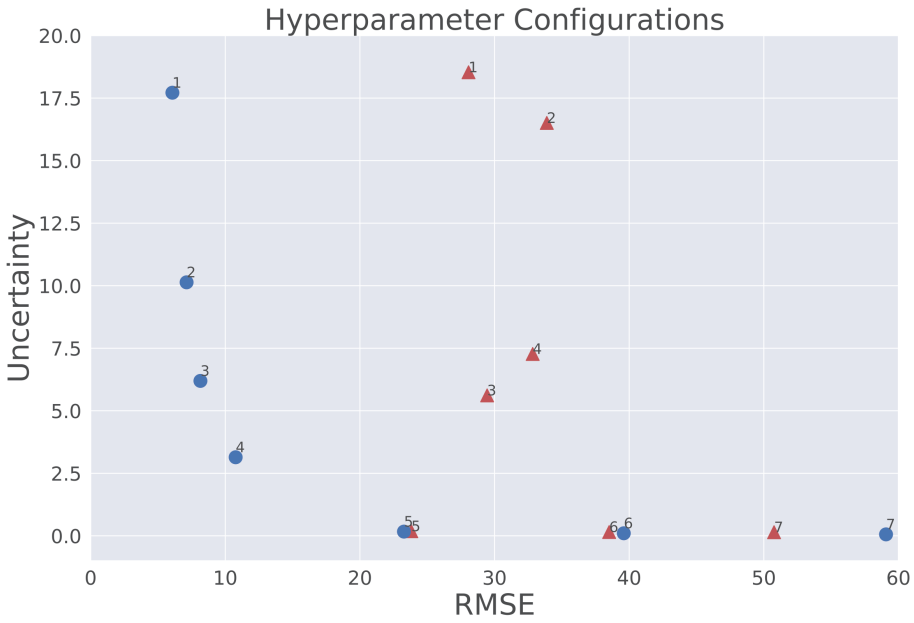

Figure 5.8: RMSE-UQ points corresponding to the hyperparameter configurations on FD003 using the harmonic mean approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set.

accuracy, uncertainty, and training time. The authors hope that multi-objective hyperparameter optimization methods become a new alternative, as it is not the case that a single-objective method can always capture the conflicting interests that exist in real-world problems.

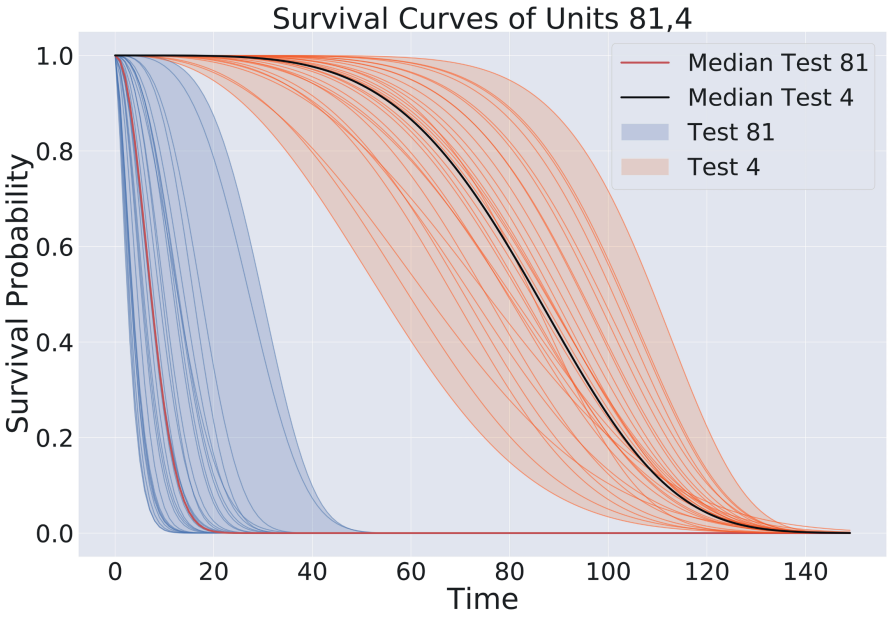

Figure 5.9: Survival curves of three units $81, 4$ from FD001. The shaded areas include *all* the survival curves from the multiple passes through MC Dropout.

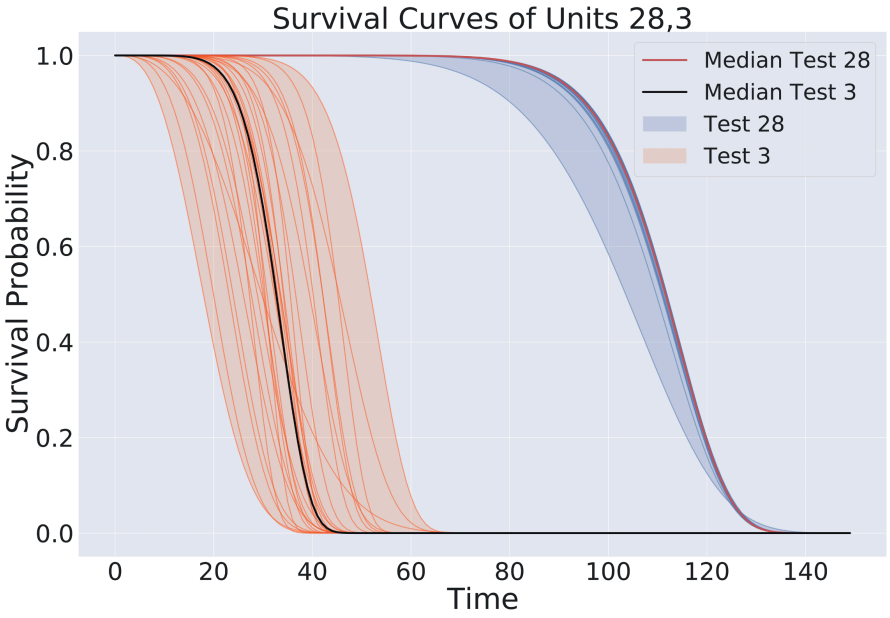

Figure 5.10: Survival curves of three units $28, 3$ from FD003. The shaded areas include *all* the survival curves from the multiple passes through MC Dropout.