Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 118 (2023) 116–121

www.elsevier.com/locate/procedia

16th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME '22, Italy

Uncertainty-aware remaining useful life prediction for predictive maintenance using deep learning

Quy Le Xuan[a,*], Yeremia G. Adhisantoso[a], Marco Munderloh[a], Jörn Ostermann[a]

[a]*Institut für Informationsverarbeitung, Leibniz Universität Hannover, Appelstr. 9A, 30167 Hannover, Germany*

* Corresponding author. Tel.: +49 511 762-19581; Fax: +49 511 762-5333; *E-mail address:* lexuan@tnt.uni-hannover.de

Abstract

Reliably predicting Remaining Useful Life (RUL) is crucial for reducing asset maintenance costs. Deep learning emerges as a powerful data-driven method capable of predicting RUL based on historical operating data. However, standard deep learning tools typically do not account for the uncertainty inherent in prediction tasks. This paper presents an uncertainty-aware approach that predicts not only the RUL but also the associated confidence interval, capturing both aleatoric and epistemic uncertainty. The proposed approach is evaluated on publicly available datasets of aircraft turbofan engines, showing its ability to estimate accurate RUL and well-calibrated uncertainties that are robust to out-of-distribution data.

Keywords: Deep learning; Uncertainty quantification; RUL prediction; Predictive maintenance; PHM systems; Hyper-deep ensemble

1. Introduction

The need for developing effective maintenance strategies has arisen since the beginning of the industry. In principle, the main objective of maintenance is to reduce unplanned downtime as well as to improve the overall equipment effectiveness. Despite great improvements over the last decades, efficient maintenance management still poses a challenge in the manufacturing community. Two traditional maintenance strategies, namely reactive and preventive maintenance, are known to be not really effective. They perform maintenance either only after a failure occurred or always at a fixed regular rate without considering the health state of the target system. In contrast, with the modern predictive maintenance strategy, one aims to schedule maintenance according to the prediction of the remaining useful life (RUL) of the target system based on its historical operating data such that maintenance can be efficiently performed in a proactive manner.

RUL of an asset is defined as the length of time left until the asset reaches its end of life (EOL), i.e. when it can no longer operate properly and needs to be repaired. Typically, RUL is given in the number of operating cycles. The increased availability of condition monitoring data and the recent breakthroughs in Artificial Intelligence are the key factors that make data-driven solutions for RUL prediction extremely promising. In this context, deep learning (DL) has been shown to be a powerful data-driven method capable of predicting the RUL of an asset given its historical operating data collected by multiple sensors [1, 2, 3, 4]. However, standard DL tools typically do not take the uncertainty inherent in RUL prediction tasks into account.

In this work, we present a novel DL-based approach for uncertainty-aware RUL prediction that predicts not only the RUL but also outputs the associated confidence interval capturing both aleatoric and epistemic uncertainties [5] of the RUL prediction. To this end, we propose: 1) to train probabilistic models to output both the mean and the log variance of the predicted RUL with a novel alternate training scheme; 2) to deploy hyper-deep ensemble [6] that utilizes the diversity resulting from combining multiple models defined by different hyperparameters and weight initializations. We evaluate the performance of our proposed method in comparison with other existing state-of-the-art methods on the benchmark dataset CMAPSS [7]. Experiment results have shown the superior performance of our proposed method in

10.1016/j.procir.2023.06.021

terms of both prediction accuracy and quality of the uncertainty quantification, especially for out-of-distribution data.

2. Related work

With impressive successes achieved recently in multiple fields such as computer vision and natural language processing, DL has also been seen as a promising tool to address the task of RUL prediction. Researchers from both academic and industrial communities have shown an increased interest in solving this challenging task using diverse deep neural network (DNN) architectures, such as deep convolutional neural networks (DCNN) [1, 2], long short-term memory [3], or autoencoder [4]. The main advantage of DL-based approaches is the ability to learn the direct mapping from historical condition monitoring data to the target RUL without the need for manual feature engineering that requires domain knowledge. Nevertheless, most of the existing DL-based approaches can only perform a point estimation without providing any information about how certain they are with their prediction. This hinders the application of DL-based approaches for safe-critical systems in practice.

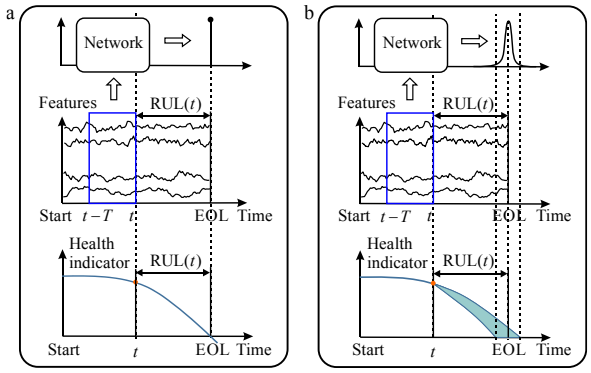


Fig. 1. RUL prediction with point estimation (a) and uncertainty-aware method (b).

To quantify the uncertainty for RUL prediction, Zhao et al. [8] recently proposed a probabilistic ResNet-based DCNN, which models the prediction uncertainty using a parametric as well as a non-parametric approach. With the parametric approach, the uncertainty is modeled in the form of the standard deviation of the conditional target distribution that is assumed to be Gaussian. In contrast, the non-parametric approach aims to predict multiple quantile levels of the target conditional distribution by optimizing the sum of the corresponding quantile regression losses. The main limitation of [8] is that the work only captures aleatoric uncertainty.

Bayesian neural networks (BNNs) [9] are known as a theoretically well-founded framework to account for epistemic uncertainty in neural networks. To capture epistemic uncertainty, in BNNs one treats model parameters as random variables and places some prior distribution over them. However, due to a large number of parameters in DNNs as well as a large number of samples in the training dataset, exact Bayesian inference is computationally intractable such that approximation approaches are typically employed instead.

Monte Carlo Dropout (MCD) [10] is one of the most efficient methods to implement the approximation of Bayesian inference for DNNs based on a technique called dropout. While the standard dropout is only applied for training DNNs as a regularization technique to avoid overfitting, MCD applies dropout also at test time. Each forward pass at test time will result in the output of a randomly sampled and thinned network, such that combining the outputs of multiple forward passes will allow the prediction uncertainty to be quantified. Recently, MCD has also been proposed to apply for quantifying uncertainty in RUL prediction. For example, Biggio et al. [11] analyzed the performance of MCD in comparison with several Deep Gaussian-based alternatives for the task of uncertainty-aware RUL prediction. However, only a relatively simple network architecture composed of fully connected (FC) layers was employed for MCD.

3. Background

Let us assume we have a dataset of N input-target pairs

$$\mathcal{D} = \left\{ \left(\boldsymbol{X}^{(i)}, y^{(i)} \right) : \boldsymbol{X}^{(i)} \in \mathbb{R}^{M \times T}, y^{(i)} \in \mathbb{R} \right\}_{i=1}^{N}. \quad (1)$$

Each input sample $\boldsymbol{X}^{(i)}$ is a 2D matrix including operating data recorded by M sensors for the last $T-1$ time steps and the current one indexed by T:

$$\boldsymbol{X}^{(i)} = \left({}^{1}\boldsymbol{x}^{(i)}, {}^{2}\boldsymbol{x}^{(i)}, \ldots, {}^{M}\boldsymbol{x}^{(i)} \right), \quad (2)$$

where ${}^{m}\boldsymbol{x}^{(i)} = \left({}^{m}x_1^{(i)}, {}^{m}x_2^{(i)}, \ldots, {}^{m}x_T^{(i)} \right)^{\mathrm{T}}$ with $m \in \{1, 2, \ldots, M\}$. Each target sample is a scalar representing the actual RUL of our system computed from the current time step.

Point estimation: In a typical supervised learning setup, given dataset $\mathcal{D}$, the task is to learn a mapping

$$f_{\boldsymbol{\theta}} : \mathbb{R}^{M \times T} \rightarrow \mathbb{R}$$
$$\boldsymbol{X} \mapsto f_{\boldsymbol{\theta}}(\boldsymbol{X}) = \hat{y}, \quad (3)$$

with parameters $\boldsymbol{\theta}$, such that the empirical risk w.r.t a pre-defined loss function L

$$\mathbb{E}_{(\boldsymbol{X},y)\sim\hat{p}_{\text{data}}}[L(f_{\boldsymbol{\theta}}(\boldsymbol{X}), y)] = \frac{1}{N}\sum_{i=1}^{N} L\left(f_{\boldsymbol{\theta}}(\boldsymbol{X}^{(i)}), y^{(i)} \right) \quad (4)$$

is minimized [12]. Here, $\hat{p}_{\text{data}}$ denotes the empirical distribution defined by the training data. For point estimation, Euclidean loss or squared error (SE) is typically used as the loss function. It has been shown that the empirical risk in Eq. (4) (with SE being the loss function) is minimized when the predicted output for a given input $\boldsymbol{X}$ is equal to the expectation of the target data with respect to the conditional probability distribution $\hat{p}_{\text{data}}(y \mid \boldsymbol{X})$ [13].

Mean-variance estimation: The actual remaining useful life of a system/component depends not only on the historical operating data and its current health state but also on many other factors, such as future operating conditions or future load. These kinds of influence factors are typically unknown as well as unpredictable at the time when the RUL prediction is performed. Therefore, the RUL y_t of a system at time t, given the measured historical operating data $\boldsymbol{X}_t$, has to be considered as a random variable capturing the randomness in the real (but unknown) data-generating process, which is commonly referred to as aleatoric or data uncertainty [5, 14]. In order to account for this data uncertainty, instead of learning the mapping which produces a single predicted

RUL $\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{X})$ for a given input $\boldsymbol{X}$, it is more desirable to model the conditional target distribution $p(y \mid \boldsymbol{X})$. Let us assume that $p(y \mid \boldsymbol{X})$ is a Gaussian distribution given by

$$p(y|\boldsymbol{X}) = \mathcal{N}\left(y; \hat{\mu}(\boldsymbol{X}), \hat{\sigma}^2(\boldsymbol{X})\right). \tag{5}$$

Here, $\hat{\mu}$ and $\hat{\sigma}$ are the mean and variance, respectively. Both of them are modeled to be a function of the input $\boldsymbol{X}$. Our task is now to learn the mapping

$$\begin{aligned} \boldsymbol{g}_{\boldsymbol{\theta}} : \mathbb{R}^{M\times T} &\to \mathbb{R}\times\mathbb{R}^{+} \\ \boldsymbol{X} &\mapsto \left({}^{1}g_{\boldsymbol{\theta}}(\boldsymbol{X}), {}^{2}g_{\boldsymbol{\theta}}(\boldsymbol{X})\right) = \left(\hat{\mu}, \hat{\sigma}^2\right), \end{aligned} \tag{6}$$

with the model parameters $\boldsymbol{\theta}$, such that the negative log-likelihood computed over all data samples

$$\begin{aligned} \mathrm{NLL} &= -\frac{1}{N}\sum_{i=1}^{N} \log p\left(y^{(i)} \mid \boldsymbol{X}^{(i)}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \frac{\log \hat{\sigma}^2\left(\boldsymbol{X}^{(i)}\right)}{2} + \frac{\left(y^{(i)} - \hat{\mu}\left(\boldsymbol{X}^{(i)}\right)\right)^2}{2\hat{\sigma}^2\left(\boldsymbol{X}^{(i)}\right)} + \text{const.} \end{aligned} \tag{7}$$

is minimized. Intuitively, training a model to minimize the NLL w.r.t the model parameters $\boldsymbol{\theta}$ can also be interpreted as minimizing the Kullback-Leibler divergence representing the dissimilarity of the empirical distribution p_{data}, defined by the training data and the model distribution p. Note that this method of mean-variance estimation would reduce to the conventional point estimation method described in the previous section under the assumption that the variance of the conditional target distribution is a constant.

4. Method

4.1. Quantifying aleatoric uncertainty

As discussed previously, aleatory uncertainty can be quantified by minimizing the NLL loss when training a predictive neural network that outputs both the mean and variance of the conditional target distribution. In practice, training such a model suffers from numerical instability when the variance $\hat{\sigma}^2$ is close to zero. To avoid this problem, we train our model to predict the log variance $s(\boldsymbol{X}) = \log \hat{\sigma}^2(\boldsymbol{X})$ instead of $\hat{\sigma}^2(\boldsymbol{X})$. The corresponding loss function can then be formulated as

$$\mathcal{L}_{\text{NLL}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \frac{s\left(\boldsymbol{X}^{(i)}\right)}{2} + \frac{\left(y^{(i)} - \hat{\mu}\left(\boldsymbol{X}^{(i)}\right)\right)^2}{2\exp\left(\hat{s}\left(\boldsymbol{X}^{(i)}\right)\right)} + \frac{\alpha}{2}\|\boldsymbol{\theta}\|_2^2. \tag{8}$$

In [8] and [15], the authors propose to add an L2 regularization term of the variance to the NLL loss function with the aim to learn small variances. In this work, we do not use this kind of variance decay. We observed that this variance decay might impact the desired calibration property of the predicted variances. Instead, we propose to add an L2 regularization term of model weights to the loss function. Varying the weight decay coefficient α allows us to train different models with different effective complexities, which is crucial for the method to quantify epistemic uncertainty as described in the following section.

Alternately training scheme: Our empirical observations suggest that the standard training scheme that aims to learn the mean and the log variance jointly by directly optimizing the NLL suffers from poor prediction performance. Such a standard joint training scheme often drives our models to undesired sub-optimal local minimums. To overcome this problem, we propose to train predictive models to learn the mean and the log variance in an alternate manner (see Algorithm 1). For each training epoch, we first update the weights of the encoder and the mean-head parts towards optimizing the MSE loss. After that, we update the weights for the logvar-head part towards optimizing the NLL loss while freezing the encoder and the mean-head parts (see Fig. 3). In the following, the methods with models trained according to the alternate training scheme will be denoted with the suffix -A.

Algorithm 1: **An alternate training epoch**

Input: Training set $\mathcal{D}$ of batched input-target pairs, network `net`
Output: Updated network weights $\boldsymbol{\theta}$
Function: `alternately_training_epoch(net)`:

```
θ := (θ_encoder, θ_mean_head, θ_logvar_head);   # weights of the last epoch
Foreach (input, target) in D:
    pred_mean, pred_logvar = net(input);        # forward pass
    loss = MSE(pred_mean, target);
    Compute ∇_θ loss and update weights θ_encoder, θ_mean_head;
    Freeze encoder and mean-head parts of the network;
    loss = NLL(pred_mean, target);
    Compute ∇_θ loss and update weights θ_logvar_head;
return θ updated
```

4.2. Quantifying epistemic uncertainty

Basically, the mapping represented by a neural network trained to fit a given set of observed data is a deterministic function. This is also true even for the case in which our network is designed to account for the aleatoric uncertainty as described in section 4.1. The reason for this is that, after training, each network's parameter will take a single fixed value. However, in principle, there might exist multiple possible models with different combinations of the parameters that are able to well explain the observed data. This kind of uncertainty in model parameters is commonly referred to as epistemic uncertainty [5, 14].

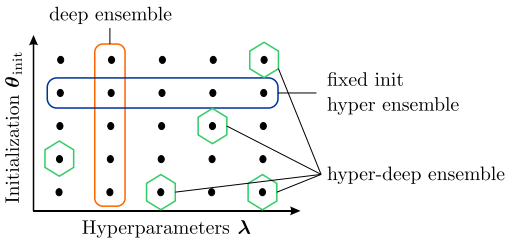


Fig. 2. Difference between deep ensemble and hyper-deep ensemble. Each point represents a network trained with weights' initialization ${}^{i}\boldsymbol{\theta}_{\text{init}}$ and hyperparameter ${}^{j}\boldsymbol{\lambda}$. Deep ensemble considers models with the same hyperparameter configuration but trained with different weights' initializations. Fixed-init-hyper ensemble considers models with different hyperparameters but trained using a fixed weights' initialization. Hyper-deep ensemble subsumes both aforementioned methods by exploiting two sources of diversity, namely the variation of hyperparameters and weights' initialization [6].

Deep ensembles-based approaches [16, 17, 18, 6], provide an efficient way to realize the approximation of Bayesian inference for DNNs. Using deep ensembles, one samples the posterior weight distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$ by training multiple models with different weights' initialization. The models trained this way are empirically shown to be diverse in both

weight and function space, such that combining these models can significantly improve the prediction performance in terms of the accuracy as well as the quantification of uncertainty [17]. In this work, we propose to apply hyper-deep ensemble [6] - an enhanced version of deep ensembles - to quantify the epistemic uncertainty for the underlying task of RUL prediction. The main idea of hyper-deep ensembles is to additionally exploit the diversity resulting from combining neural networks defined by different hyperparameters. In this work, the hyperparameter search is done by employing Optuna [19] - a Bayesian hyperparameter optimization framework. The selection of models from the set of the model candidates, resulting from varying weight initializations and hyperparameters, to build the desired ensemble is realized by means of a so-called hyper-ens selection algorithm [6]. Using hyper-ens, the ensemble members are iteratively selected from the set of model candidates with replacement to maximize a performance metric on a validation set until a pre-defined number of candidates are selected or no further performance improvement is possible [6]. See Fig. 2 for an illustration of the difference between deep ensemble and hyper-deep ensemble.

4.3. Combining aleatoric and epistemic uncertainty

For a given ensemble of K networks under consideration $\mathcal{E} = \left\{ \boldsymbol{g}_{\boldsymbol{\theta}^j \mid \boldsymbol{\theta}^j_{\text{init}}, \boldsymbol{\lambda}^j} \right\}_{j=1}^{K}$ and a given input $\boldsymbol{X}$, let $\left(\hat{\mu}_j, \hat{\sigma}_j\right) = \boldsymbol{g}_{\boldsymbol{\theta}^j \mid \boldsymbol{\theta}^j_{\text{init}}, \boldsymbol{\lambda}^j}\left(\boldsymbol{X}\right)$ be the mean and variance of RUL predicted by the j-th member of the ensemble, which is defined by the hyperparameters $\boldsymbol{\lambda}^j$ and trained with the weights' initialization $\boldsymbol{\theta}^j_{\text{init}}$. The combined prediction for the mean and the variance that captures both the aleatoric and epistemic uncertainty can then be computed as follows (cf. [16, 14]):

$$\hat{\mu}_{\mathcal{E}} = \frac{1}{K}\sum_{j=1}^{K} \hat{\mu}_j, \tag{9}$$

$$\hat{\sigma}_{\mathcal{E}} = \frac{1}{K}\sum_{j=1}^{K} \hat{\sigma}_j^2 + \frac{1}{K}\sum_{j=1}^{K} \hat{\mu}_j^2 - \left(\frac{1}{K}\sum_{j=1}^{K} \hat{\mu}_j\right)^2. \tag{10}$$

Note that each model in the ensemble $\mathcal{E}$ can be interpreted as being sampled from the weight posterior $p\left(\boldsymbol{\theta} \mid \mathcal{D}\right)$ in the context of BNNs. The combined prediction of the mean and the variance as given in Eq. (9) and (10) are actually the mean and variance of the predictive posterior distribution (marginal) $p\left(y \mid \boldsymbol{X}, \mathcal{D}\right) = \int_{\Theta} p(y \mid \boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) \mathrm{d}\boldsymbol{\theta}$, which is approximated by Monte Carlo integration [9].

4.4. Network architecture

Fig. 3 visualizes our proposed network architecture for the task of uncertainty-aware RUL prediction. The basis of the architecture is inspired by the DCNN proposed in [2], which also deals with RUL prediction for aircraft turbofan engines but without accounting for the inherent prediction uncertainty. The reason for choosing this architecture is that it is a simple but efficient one that achieved state-of-the-art performance on the CMAPSS dataset [2, 20]. The main components of the proposed network architecture are a CNN module and two FC modules. The CNN module is composed of five 2D-convolutional layers followed by a flatten and dropout layer. It is responsible for extracting a feature vector of a desired length from the input. As a reminder, the input includes multivariate time series of fixed length representing historical operating data of the asset of interest. The feature vector extracted by the CNN module is then fed into two separate FC modules that output the predicted mean RUL and the associated log variance representing the aleatoric uncertainty (see section 4.1). Note that the convolutional layers used here are two-dimensional, meaning that their kernels are shifted along both dimensions of their input map to compute the corresponding feature map. Each kernel, however, is one-dimensional. This allows learning features along the temporal dimension separately from each time series, instead of learning features blurred across multiple time series by using 2D kernels.

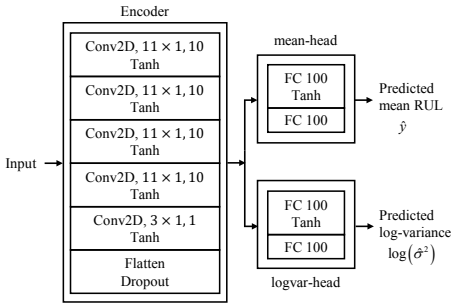


Fig. 3. Network architecture. In the first part of the CNN module, four convolutional layers are stacked, each of them having 10 kernels of size 11×1, followed by a Tanh activation function. The output of each of these four convolutional layers is 10 feature maps with the same dimensions as those of the input $\boldsymbol{X}$. The final convolutional layer with kernel size 3×1 combines the previous feature maps, along the channel dimension. The extracted features are then flattened and dropout before being passed onto two separate FC modules to output the predicted mean RUL and its corresponding variance.

5. Experiments and Results

5.1. Metrics

Root-mean-squared error (RMSE): Let d_i be the difference between the predicted RUL $\hat{y}_i$ and the actual RUL y_i of the i-th data sample:

$$d_i = \hat{y}_i - y_i. \tag{11}$$

The RMSE for a dataset with N samples can be formulated as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2}. \tag{12}$$

NASA's scoring function: Another metric that is also commonly used to evaluate the accuracy of RUL prediction is a so-called scoring function proposed by NASA [21]:

$$s = \sum_{i=1}^{N} s_i, \text{ with } s_i = \begin{cases} \exp\left(\frac{-d_i}{13}\right) - 1, & \text{if } d_i < 0 \\ \exp\left(\frac{d_i}{10}\right) - 1, & \text{if } d_i \geq 0, \end{cases} \tag{13}$$

where d_i is the prediction error as given in Eq. (11). In contrast to RMSE, the scoring function s is not symmetric and penalizes over-estimation more than under-estimation.

Table 1. Experimental results on four sub-datasets of CMAPSS.

Method	Sub-dataset FD001					Sub-dataset FD003				
	RMSE↓	Score↓	NLL↓	QL-0.1↓	QL-0.9↓	RMSE↓	Score↓	NLL↓	QL-0.1↓	QL-0.9↓
PE	13.11±1.24	322±116	–	–	–	12.86±0.34	331±39	–	–	–
B1-PE [2]	12.61±0.19	274±24	–	–	–	12.64±0.14	284±27	–	–	–
B2-PE [8]	13.26±n/a	291±n/a	–	–	–	12.66±n/a	277±n/a	–	–	–
MVE-A	12.62±0.64	263±51	2.93±0.07	1.96±0.06	2.32±0.10	12.49±0.25	324±21	2.90±0.06	2.41±0.04	2.00±0.19
B-MVE [8]	12.48±n/a	242±n/a	n/a	3.85±n/a	3.89±n/a	12.71±n/a	303±n/a	n/a	4.80±n/a	2.91±n/a
MCD-A	12.32±0.39	237±33	2.88±0.04	1.92±0.06	**2.22±0.09**	12.06±0.20	**271±18**	2.87±0.06	2.26±0.05	2.00±0.20
DE-A	12.27±0.38	246±35	2.88±0.03	1.93±0.04	2.26±0.08	12.50±0.28	322±21	2.89±0.06	2.39±0.04	2.02±0.18
HDE-A	**12.05±0.17**	**234±14**	**2.84±0.01**	**1.80±0.02**	**2.25±0.02**	**11.78±0.10**	278±9	**2.78±0.01**	**2.23±0.05**	**1.81±0.01**
	Sub-dataset FD002					Sub-dataset FD004				
PE	21.68±1.57	5250±1785	–	–	–	25.59±2.01	11202±5454	–	–	–
B1-PE [2]	22.36±0.32	10412±544	–	–	–	23.31±0.39	12466±853	–	–	–
MVE-A	20.01±1.03	4678±2988	3.47±0.04	3.67±0.21	3.36±0.13	22.47±0.49	6426±826	3.61±0.04	4.62±0.14	3.64±0.17
MCD-A	20.15±0.96	3781±1678	3.48±0.04	3.62±0.20	3.37±0.17	22.46±0.48	6418±833	3.58±0.04	4.60±0.15	3.64±0.16
DE-A	19.54±1.16	3657±1453	3.46±0.02	3.64±0.10	3.32±0.15	21.94±0.33	5491±672	3.55±0.02	4.39±0.15	3.58±0.06
HDE-A	**18.41±0.06**	**2551±93**	**3.42±0.01**	**3.41±0.01**	**3.22±0.03**	**21.60±0.13**	**4734±191**	**3.52±0.01**	**4.12±0.04**	**3.55±0.02**

Negative Log-Likelihood (NLL): We use NLL as a further metric to jointly evaluate the accuracy of the predicted mean as well as the quality of the predicted variance. NLL measures how well the distribution modeled by the neural network fits the actual distribution defined by the data samples. We report NLL according to Eq. (7) except for the constant term.

Quantile losses (QL-0.1, QL-0.9): Let $y^{(i)}$ be the actual RUL and $y_\tau^{(i)}$ the predicted RUL at quantile level τ. The corresponding quantile loss [22] can be then formulated as

$$\text{QL-}\tau = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - y_\tau^{(i)}\right)\left(\tau - \mathbf{1}\left(y^{(i)} < y_\tau^{(i)}\right)\right), \qquad (14)$$

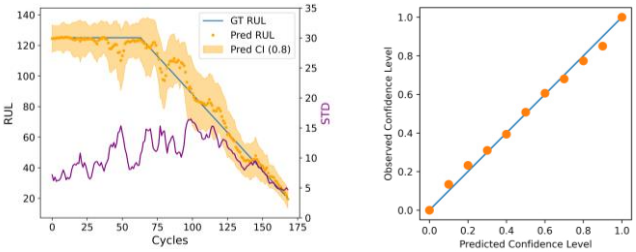
where $\mathbf{1}(A)$ is an indicator function that equals 1 (or 0) if A is true (or false).

5.2. Dataset

CMAPSS is an aircraft turbofan engine degradation dataset provided by NASA [7, 21]. It is one of the most popular benchmark datasets widely used to evaluate approaches addressing the RUL prediction problem. CMAPSS consists of four sub-datasets (FD001-4) including multivariate time series data of 21 sensors that represent the operating state of different components of a fleet of turbofan engines. The data is simulated under different operational conditions and fault modes using a model-based simulation program called Commercial Modular Aero-Propulsion System Simulation. Each sub-dataset contains one training and one test set, each of which includes data of 100 engine units. Each engine unit starts with different degrees of initial wear and has different manufacturing variation that is unknown. The training sets include run-to-failure data, i.e., the sensor readings until the engine units reach their end of life. In contrast, the test sets include data up to some time, before the system failure occurs. Further details of CMAPSS can be found in [7, 21, 2].

5.3. Implementation details

The data including multivariate sensor readings are normalized to be within the range $[-1, 1]$. The normalization is done by fitting a Min-Max-Scaler using the available training data. The fitted scaler is applied to the training data as well as to the testing data that is supposed to be not available when building models. To predict the RUL of an engine unit at a given time, we consider the operating data of the last T timesteps which are set according to the length of the shortest recorded trajectory in each subset.



(a) (b)

Fig. 4. (a) Ground-truth RUL (blue solid line) and RUL predictions (dark orange dots) with 0.1-0.9 confidence interval (light orange filled region) of an exemplary test engine unit in FD001; (b) Reliability diagram for evaluating the calibration of predicted uncertainty.

Models in this work are implemented in PyTorch. We use the official train/test split of CMAPSS as described in section 5.2 to build models and report their performance. For each subset, we further split the available training data into train (80%) and validation (20%) sets. For training models, mini-batch gradient descent with a batch size of 512 was applied. Moreover, we use the Adam optimizer with an initial learning rate η and a weight decay α, which are, along with the dropout rate p, the hyperparameters to be tuned. For hyperparameter optimization (HPO) we employed a Bayesian optimization framework called Optuna [19] with a TPE sampler. The search space was: $\eta \sim \log U[1\text{e-}5, 0.1)$, $\alpha \sim U[0, 0.01)$, $p \sim$ discrete $U(0, 1, 0.1)$. The number of trials for HPO was set to 100 across all settings. Furthermore, we set the number of training epochs to 200 and applied early stopping to avoid overfitting. The number of ensemble members was set to 5 for the ensemble methods. All the experiments were repeated 5 times with different seeds.

5.4. Results

The experimental results of the different methods under comparison are given in Table 1. The results of point estimation (PE, see section 3), mean-variance estimation (MVE-A, see section 4.1), Monte Carlo dropout (MCD-A, see section 2), deep ensemble (DE-A, see sections 4.2 and 4.3), hyper-deep ensemble (HDE-A, see sections 4.2 and 4.3) are reported according to our implementation, whereas those of the other

baselines B1-PE [2], B2-PE [8], and B-MVE [8] are taken from the referenced work. As can be seen here, ensemble methods including MCD-A, DE-A, HDE-A show better performance compared to the PE and MVE methods w.r.t both error-based and uncertainty-based metrics. This confirms that combining the predictions of multiple models enables a performance improvement in terms of both prediction accuracy and quality of uncertainty quantification. Especially, our proposed HDE-A method shows the best performance in most settings, indicating the superior benefit of utilizing the diversity resulting from the variation of hyperparameters as well as the weights' initialization.

Fig. 4a depicts the RUL predictions with the 0.1-0.9 confidence interval of an exemplary test engine unit in FD001. Following [23], we analyzed the calibration of the considered models regarding the predicted uncertainty using reliability diagrams, which are also commonly referred to as calibration curves. The reliability diagram visualizes the calibration by plotting the observed confidence level as a function of the predicted confidence level. Our calibration analysis indicates that all implemented models are well-calibrated since their calibration curves closely follow the 1:1 line (see Fig. 4b for an example).

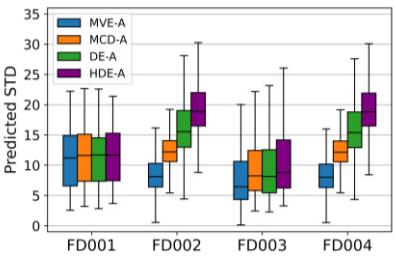


Fig. 5. Out-of-distribution analysis. Here, the models under consideration are trained on FD001 and evaluated on FD001, FD002, FD003, and FD004.

We further evaluate the robustness of the uncertainty quantification of different methods regarding out-of-distribution data. For that, we trained the models on FD001 and reported their predicted standard deviation on all sub-datasets FD001-4 (see Fig. 5). For FD001 and FD003, all methods under consideration output relatively small standard deviations (STDs). This is comprehensible since data of FD001 and FD003 are expected to be in-distribution due to the fact that they are simulated under the same operational condition [7]. For out-of-distribution data (FD002 and FD004, simulated under six operational conditions [7]), the predicted STDs of MVE-A are still small, indicating that their uncertainty quantification is over-confident, whereas larger STDs are observed for the ensemble methods (MCD-A, DE-A, HDE-A). This evidence again supports the importance of modeling epistemic uncertainty. Overall, it is obvious that the uncertainty estimation of the proposed HDE-A method is most indicative w.r.t the out-of-distribution data.

6. Conclusion

In this work, we proposed a novel uncertainty-aware framework for RUL prediction that is able to predict not only a single value for RUL like traditional DL-based methods but is also capable of providing the associated confidence interval capturing both aleatoric and epistemic uncertainty. The evaluations using the aircraft turbofan engine dataset CMAPSS showed superior performance of the proposed method in terms of both prediction accuracy and quality of the uncertainty quantification, especially for out-of-distribution data.

Acknowledgements

This work was supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK), Germany, within the framework of the project IIP-Ecosphere (project number 01MK20006A).

References

[1] G. S. Babu, P. Zhao and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *International conference on database systems for advanced applications*, 2016.

[2] X. Li, Q. Ding and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety,* vol. 172, p. 1–11, April 2018.

[3] S. Zheng, K. Ristovski, A. Farahat and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *International Conference on Prognostics and Health Management (ICPHM)*, 2017.

[4] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan and X. Chen, "Deep Transfer Learning Based on Sparse Autoencoder for Remaining Useful Life Prediction of Tool in Manufacturing," *IEEE Transactions on Industrial Informatics,* vol. 15, p. 2416–2425, April 2019.

[5] Y. Gal, "Uncertainty in deep learning," 2016.

[6] F. Wenzel, J. Snoek, D. Tran and R. Jenatton, "Hyperparameter Ensembles for Robustness and Uncertainty Quantification," *Advances in Neural Information Processing Systems,* January 2021.

[7] A. Saxena and K. Goebel, *Turbofan Engine Degradation Simulation Data Set,* 2008.

[8] Z. Zhao, J. Wu, D. Wong, C. Sun and R. Yan, "Probabilistic Remaining Useful Life Prediction Based on Deep Convolutional Neural Network," *SSRN Electronic Journal,* 2020.

[9] R. M. Neal, Bayesian learning for neural networks, Springer, 1996.

[10] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016.

[11] L. Biggio, A. Wieland, M. A. Chao, I. Kastanis and O. Fink, "Uncertainty-Aware Prognosis via Deep Gaussian Process," *IEEE Access,* vol. 9, p. 123517–123527, 2021.

[12] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, Cambridge, Massachusetts: The MIT Press, 2016.

[13] C. M. Bishop, "Mixture density networks," 1994.

[14] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," *arXiv:1703.04977,* 2017.

[15] M. Kim and K. Liu, "A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics," *IISE Transactions,* vol. 53, p. 326–340, 2020.

[16] B. Lakshminarayanan, A. Pritzel and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems,* vol. 30, 2017.

[17] S. Fort, H. Hu and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *arXiv preprint arXiv:1912.02757,* 2019.

[18] Y. Wen, D. Tran and J. Ba, "Batchensemble: an alternative approach to efficient ensemble and lifelong learning," *arXiv:2002.06715,* 2020.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019.

[20] O. Serradilla, E. Zugasti, J. Rodriguez and U. Zurutuza, "Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects," *Applied Intelligence,* January 2022.

[21] A. Saxena, K. Goebel, D. Simon and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *International Conference on Prognostics and Health Management*, 2008.

[22] R. Koenker and G. Bassett, "Regression Quantiles," *Econometrica,* vol. 46, p. 33, January 1978.

[23] V. Kuleshov, N. Fenner and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *International conference on machine learning*, 2018.