9th International Conference on Through-life Engineering Service
3-4 November 2020, Cranfield UK

Probabilistic Remaining Useful Life Prediction Based on Deep Convolutional Neural Network

Zhibin Zhao[a,b], Jingyao Wu[a], David Wong[b], Chuang Sun[a], Ruqiang Yan[a,*]

[a]*School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an and 710049, China*
[b]*the Centre for Health Informatics and Department of Computer Science, University of Manchester, Manchester and M13 9PL, UK*

* Corresponding author. *E-mail address:* yanruqiang@xjtu.edu.cn

Abstract

Remaining useful life (RUL) prediction plays a vital role in prognostics and health management (PHM) for improving the reliability and reducing the cycle cost of numerous mechanical systems. Deep learning (DL) models, especially deep convolutional neural networks (DCNNs), are becoming increasingly popular for RUL prediction, whereby state-of-the-art results have been achieved in recent studies. Most DL models only provide a point estimation of the target RUL, but it is highly desirable to have associated confidence intervals for any RUL estimate. To improve on existing methods, we construct a probabilistic RUL prediction framework to estimate the probability density of target outputs based on parametric and non-parametric approaches. The model output is an estimate of the probability density of the target RUL, rather than just a single point estimation. The main advantage of the proposed method is that the method can naturally provide a confidence interval (*aleatoric* uncertainty) of the target prediction. We verify the effectiveness of our constructed framework via a simple DCNN model on a publicly available degradation simulation dataset of turbine engines. The source codes will be released at https://github.com/ZhaoZhibin/Probabilistic_RUL_Prediction.

Keywords: Remaining useful life; Probabilistic prediction; Deep convolutional neural networks

1. Introduction

The quantity and range of condition monitoring data is increasingly rapid, with data collected from a wide range of sensors and from different industrial systems. This, in combination with modern big data techniques, has led to the popularity of Prognostics and Health Management (PHM) both in the research community and within industry [1]. One important ingredient of PHM is Remaining Useful Life (RUL) prediction – the estimate of how long a system or machine may continue to operate unhindered. RUL plays a vital role in preventing unexpected system downtime, which may result in a huge cost or even casualties [2].

According to a systematic review paper published by Lei et al. [3], the RUL prediction methods can be classified into three categories: data-driven model-based methods, physics model-based methods, and hybrid methods. Given the complexity of modern industrial systems, and the quantity of associated monitoring data, data-driven model-based methods have become increasingly popular and shown successful results in RUL prediction.

Traditional machine learning algorithms, such as support vector machine [4] and tree-based approaches [5], need hand-crafted features as their inputs, which limits the performance improvement in the big data era. Alternatively, deep learning (DL) [6], which allows automatic feature extraction without extra domain knowledge, has become one of the most promising data-driven model-based approaches.

Many DL models have been applied to the RUL prediction, such as deep autoencoders (DAEs) [7], deep convolutional neural networks (DCNNs) [8], and deep recurrent neural networks (DRNNs) [9]. However, most of these DL models only provide a point estimation of the RUL, which means that the prediction uncertainty is often neglected.

In reality, any RUL prediction is subject to uncertainty caused by factors such as input uncertainty imported by errors of measurements or operating conditions also called *aleatoric* uncertainty, and model uncertainty introduced by the representation capability of models, also called *epistemic* uncertainty [10].

The quantification of the prediction uncertainty is important for assisting decision making, including identifying alternatives in PHM systems. Without prediction uncertainty, it would be impossible for decision-makers to quantify the trustworthiness of DL model results.

Despite the importance of prediction uncertainty quantification, there is limited research covering this drawback of DL-based RUL prediction methods. Recently, Liu et al. [11] and Jason et al. [12] applied the bootstrap method to different neural networks for

uncertainty quantification. However, this method requires the original training dataset to be repeatedly resampled and is time-consuming. Peng et al. [13], Wang et al. [14] and Mathias et al. [15] introduced the Bayesian DL framework based on probabilistic programming and used variational inferences to estimate the model parameters as well as the uncertainty. These approaches only model *epistemic* uncertainty of parameters. Kim et al. [16] systematically modeled two types of uncertainties embedded in prognostics, but they only assumed Gaussian noise in degradation processes.

In this paper, we aim to simplify the process of *aleatoric* uncertainty quantification and provide a probabilistic RUL prediction framework with sufficient flexibility to work with existing DL models. The framework estimates the probability density of target outputs using both parametric and non-parametric approaches. Inspired by [17,18], the parametric approach estimates parameters of the hypothetical distribution based on maximum likelihood estimation (MLE), and the non-parametric approach predicts multiple RULs to estimate a posterior RUL distribution via quantile regression.

To demonstrate the approach, we apply it to a DCNN. The DCNN we used is a shortened and modified version of ResNet [19], named MResNet9. Six metrics for evaluating the performance of predicting ground-truth RUL and one metric for evaluating the performance of probabilistic RUL prediction are constructed, and we observe that MResNet9 can achieve acceptable results in the probabilistic RUL prediction framework and *aleatoric* uncertainty quantification based on experimental verification.

The rest of this paper is organized as follows: Section 2 describes the details of the point estimation, the probabilistic RUL prediction framework, and the DCNN architecture. In Section 3, we perform experimental verification on a public dataset. Finally, Section 4 summarizes the main conclusions of this paper.

2. Methodology

In this section, we first review point estimation methods, and then introduce the probabilistic RUL prediction framework and its implementation details for deep learning models. After that, we further describe the DCNN architecture used in the probabilistic RUL prediction method.

2.1. Point estimation

Mathematically, the real RUL, y_t, at the timestamp t can be formulated as:

$$y_t = f(x_t;\theta) + \varepsilon_t \tag{1}$$

where $f(x_t;\theta)$ denotes a representation function with the input data x_t and learned parameters θ, such as a deep neural network and ε_t is the additional noise. The input data, x_t, may contain multiple historical observations, $z_t, \dots z_{t-l+1}$, (l denotes the length of the sliding window) and each historical observation may be high-dimensional, containing measurements related to machine conditions, such as vibration sensors and temperatures.

A common method to predict RULs is to assume that the noise $\varepsilon_t \sim \mathcal{N}(0,\sigma^2)$ follows the same Gaussian distribution at all timestamps. Under this assumption, the parameters θ of the deep neural network can be estimated by solving the following optimization problem using the training samples $\{x_i, y_i\}_{i=1}^N$:

$$\theta^* = \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N (y_i - f(x_i;\theta))^2 \tag{2}$$

After solving the above optimization problem and obtaining the estimated parameters θ^*, the predicted RUL at the timestamp t is given by $\hat{y}_t = f(x_t;\theta^*)$, which is a point estimate of y_t.

2.2. Probabilistic RUL prediction framework

Probabilistic RUL prediction at time t can be expressed as modeling the conditional distribution of the RUL y_t:

$$\mathbb{P}(y_t|x_t) \longrightarrow \mathbb{P}(y_t|z_t, \dots z_{t-l+2}, z_{t-l+1}) \tag{3}$$

It is worth mentioning that the input data, x_t, represents multiple historical observations, $z_t, \dots z_{t-l+1}$ with a sliding window l. After obtaining the conditional distribution, the point estimate RUL, $\hat{y}_t$, can be obtained by the distribution expectation:

$$\hat{y}_t = \mathbb{E}[y_t|x_t] \tag{4}$$

Under the above definition, the main challenge becomes how to model the conditional distribution via designing a neural network $f(x_t;\theta)$ that incorporates historical observations and RULs.

Inspired by [17,18], we establish a probabilistic RUL prediction framework that (i) estimates parameters of a posterior RUL distribution based on MLE and (ii) generates an estimation of a posterior RUL distribution by a non-parametric method estimating multiple RULs using quantile regression [20]. Benefiting from the flexibility of a neural network, that is the number of network outputs can adapt to the target requirement, we can directly model the parameters of the hypothetical

distribution in the parametric approach and concerned quantile points in the non-parametric approach.

Parametric approach: Given a specific distribution, the MLE-induced loss is used as a loss function to optimize the corresponding parameters of the representation function. For simplicity, a common way to model *aleatoric* uncertainties is to use Gaussian distribution, which means that we assume ε_t is the additional Gaussian noise. Given training samples $\{x_i, y_i\}_{i=1}^N$, the neural network, $f(x_i;\theta)$, outputs the mean μ_i and the standard deviation σ_t of Gaussian distribution. Then we apply the negative log-likelihood of all training samples to construct the following optimization problem:

$$\theta^* = \operatorname{argmin}_\theta -\frac{1}{N}\sum_{i=1}^N \log \mathbb{P}(y_i|\mu_i,\sigma_i)$$
$$= \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \left(\frac{(y_i-\mu_i)^2}{2\sigma_i^2} + \frac{\log(\sigma_i^2)}{2}\right) \quad (5)$$

The standard deviation σ_i should be strictly positive. Furthermore, a large σ_i is undesirable, especially when the RUL is short. To address the first constraint, we apply the softplus activation function to ensure the positivity. To address the second constraint, similar to [16], we add a variance decay term in the final loss function, which is reformulated as:

$$\theta^* = \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \left(\frac{(y_i-\mu_i)^2}{2\sigma_i^2} + \frac{\log(\sigma_i^2)}{2} + \lambda\sigma_i^4\right) \quad (6)$$

where λ is the trade-off parameter controlling the importance of the variance decay term. We can use the mean μ_t as the predicted RUL and the standard deviation σ_t to evaluate the uncertainties at the timestamp t.

The main disadvantage of the parametric approach is that it is distribution-specific. If the assumed distribution cannot represent the real-life uncertainty distribution, the performance will be sub-optimal.

Non-parametric approach: When it is difficult to determine a specific distribution, a non-parametric approach is possible. For the non-parametric approach, multiple predicted RULs at different quantile levels can be obtained by quantile regression. Given the real RUL y_i and the neural network $f(x_i;\theta)$ output $\hat{y}_i^q$ at a quantile level $q \in [0,1]$, the quantile regression loss is defined as:

$$L_q(y_i,\hat{y}_i^q) = q(y_i - \hat{y}_i^q)^+ + (1-q)(\hat{y}_i^q - y_i)^+ \quad (7)$$

where $(y_i)^+ = \max(0, y_i)$ means that we only maintain the positive part and let the negative part equal to zero. With that, given multiple quantile levels $Q = \{q_1,\ldots,q_M\}$, the final optimization problem using the quantile loss of all training samples can be formulated as:

$$\theta^* = \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \sum_{m=1}^M L_{q_m}(y_i,\hat{y}_i^{q_m}) \quad (8)$$

For example, given different quantile levels $Q = \{0.1,0.5,0.9\}$, we can use the quantile $\hat{y}_t^{0.5}$ as the predicted RUL and other two quantiles $[\hat{y}_t^{0.9}, \hat{y}_t^{0.1}]$ as interval estimation with 80% confidence interval at the timestamp t.

Although quantile regression is distribution-free, it cannot obtain the whole distribution. That is the non-parametric approach is not additive for a certain period.

2.3. DCNN architecture

In the above probabilistic RUL prediction framework, one important detail is left unanswered, that is how to construct a deep neural network $f(x_t;\theta)$ incorporating historical observations and RULs. Among different deep neural networks, DCNN possessing a strong ability of representative learning has become increasingly popular and achieve significant success in a wide range of fields. Meanwhile, the ResNet proposed by He [19], is one of the most popular backbones in DCNNs, and its deep version is the first DCNN exceeding the human baseline accuracy. Therefore, in this paper, we constructed a MResNet9 which is a shortened and modified version of ResNet.

The architecture of MResNet9 is shown in Fig. 1. In the figure, the number before *Conv2d* is the size of the convolution kernel. The number after *Conv2d* is the quantity of the convolution kernels, and the number after *Fc* is the output dimension (in this case, a variable, Parameters). */2* means that the feature dimensions are halved. In addition, *BN* represents the Batch Normalization (BN) layer, *ReLU* represents a Rectified Linear Unit activation function, and *Dropout* represents the Dropout layer whose rate is set to 0.2. The final output depends on whether we wish the posterior to be a point estimate, a parametric distribution or a non-parametric method. For example, if a point estimation is required, the final output is the RUL. Alternatively, if a non-parametric method is used, the final output should be a set of quantiles. It is also worth mentioning that the convolutional operation does not work on the second dimensionality in this architecture.

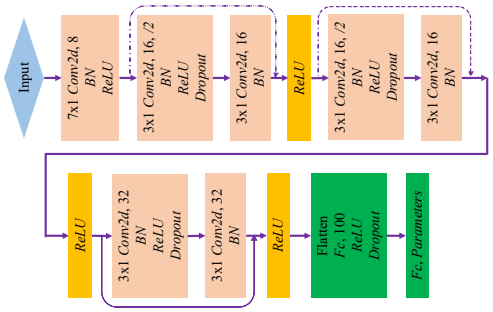

Fig. 1. The architecture of MResNet9. The solid line denotes that the dimension of input features is the same as that of output features, and the dashed line denotes that the dimension of input features is reduced by half to match that of output features.

3. Experimental verification

In this section, we use one publicly available turbofan engine degradation simulation dataset provided by the Prognostics CoE at NASA Ames to verify the performance of our constructed probabilistic RUL prediction framework. All the experiments are performed using the PyTorch library, running on the Ubuntu 16.04.6 GNU/Linux and GeForce GTX TITAN X.

3.1. C-MAPSS dataset

The C-MAPSS dataset is a turbofan engine degradation simulation dataset generated by C-MAPSS, a suitable platform that can record measurements from different sensors corresponding to predefined health-related parameters [21,22]. This dataset consists of four different sub-datasets simulated under different compositions of two working conditions and two fault modes. A total of 21 sensor measurements are collected in each sub-dataset that includes a fleet of engines. The detailed information is listed in Table 1. The aim is to predict the RULs of testing engine units (EU) via run-to-failure training EU. In this paper, we choose FD001 and FD003 with the same condition and different fault modes to verify the performance of our constructed probabilistic RUL prediction framework.

Table 1. Detailed information of the C-MAPSS dataset.

Sub-dataset names	FD001	FD002	FD003	FD004
Conditions	1	6	1	6
Fault modes	1	1	2	2
EU for training	100	260	100	249
EU for testing	100	259	100	248

3.2. Data pre-processing

1) *Sensor selection*: A total of 21 sensor measurements were selected, but there are some measurements whose variances are too small to provide any useful information. Thus, we drop any measurement whose variance was smaller than a constant threshold $1 * 10^{-6}$. The remaining set of 14 sensor measurements C contained sensors 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21 from the original dataset.

2) *Normalization*: For the stability of model training and testing, each sensor measurement is normalized using the following formulae:

$$S_j^{train-norm} = 2 * \frac{S_j^{train} - a_j^{train}}{b_j^{train} - a_j^{train}} - 1, j \in C$$

$$S_j^{test-norm} = 2 * \frac{S_j^{test} - a_j^{train}}{b_j^{train} - a_j^{train}} - 1, j \in C$$

where S_j^{train} and S_j^{test} denote measurements of the sensor j from training and testing sets, respectively. a_j^{train} and b_j^{train} represent the minimum value and the maximum value of measurements of sensor j from the training set. $S_j^{train-norm}$ and $S_j^{test-norm}$ are normalized measurements of the sensor j from training and testing sets, respectively.

3) *Sample generation*: A commonly used time window embedding method was applied to generate the training and testing samples. That is the input data, x_t, contains multiple historical observations $z_t, \ldots, z_{t-l+1}$, where k is the length of the sliding window. In this paper, we set l equal to 30. Finally, the dimension of each training sample was 1x30x14 which represents an input channel, window length, and sensor number respectively. In addition, the RUL labels of the training samples also had a huge impact on the testing performance. Two methods, including generating RUL labels by the percentages of the whole life [14] and generating RUL labels by a piecewise linear function with a constant RUL in the early stage [8,23], are often applied to the RUL prediction of C-MAPSS. In this paper, we simply adopted the latter one and set the constant RUL equal to 125.

3.3. Evaluation indicators

The performance was evaluated by Root Mean Squared Error (RMSE), Root Mean Squared Logarithmic Error (RMLSE), Mean Absolute Error (MAE), Median Absolute Deviation (MAD), and Score Function (SF). These metrics are defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\delta_k^2}, \delta_k = \hat{y}_k - y_k$$

$$\text{RMLSE} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(\log(y_k+1) - \log(\hat{y}_k+1)\right)^2}$$

$$\text{MAE} = \frac{1}{K}\sum_{k=1}^{K}|\delta_k|$$

$$\text{R2} = 1 - \sum_{k=1}^{K}\delta_k^2 / \sum_{k=1}^{K}(y_k - y^{mean})^2$$

$$\text{SF} = \sum_{k=1}^{K}\text{SF}_k, \ \text{SF}_k = \begin{cases} e^{-\delta_k/13} - 1, & \delta_k < 0 \\ e^{\delta_k/10} - 1, & \delta_k \geq 0 \end{cases}$$

where K is the number of testing samples, $\hat{y}_k$ denotes the predicted RUL, y_k denotes the real RUL of the sample k, and y^{mean} denotes the mean of real RULs respectively. RMSE is a commonly used metric in the RUL prediction. RMLSE will be useful when handling right skewed labels. MAE is more robust to outliers. R2 can eliminate the effect of dispersion of the original data. SF is a scoring

function used in the 2008 Prognostics and Health Management Data Challenge [21].

The performance of probabilistic RUL prediction is evaluated by the quantile loss at a predefined quantile level, q, denoted as QL-q (e.g., QL-0.9).

3.4. Experimental results

We trained the model using the Adam optimizer with initial learning rate 0.001, max epoch as 80, and mini-batch size as 256. In addition, a learning rate annealing method called 'step' was applied this decreased the learning rate via multiplying 0.1 in epochs 40 and 60. To avoid randomness, the final accuracy is the average of the last 10 epochs.

The final results of two datasets are shown in Table 2 and Table 3. M1 represents the point estimation. M2 represents the non-parametric approach based on quantile regression with the quantile levels $Q = \{0.1, 0.5, 0.9\}$. M3-1 represents the parametric approach based on Gaussian assumption with the trade-off parameter $\lambda = 1$, and M3-2 represents $\lambda = 10^{-5}$. It should be noted that the max epoch is 180 and declined epochs are 120 and 160 for M3-2, because a small λ will allow a relatively large variance which leads to the tardiness of the learning procedure. We can observe that the probabilistic RUL prediction framework can achieve considerable results and even higher accuracies than the point estimation.

Performance of the parametric distribution estimate is similar to the non-parametric method. This might be because that the true prediction uncertainty obeys a Gaussian distribution to a certain extent. Besides, QL-0.1 and QL-0.9 of M3-1 are much larger than those of M3-2, which results from a large trade-off parameter λ. In the next subsection, we would explain how the parameter λ affected the uncertainty quantification.

Table 2. Accuracy comparisons of three methods on FD001.

Datasets	FD001			
	M1	M2	M3-1	M3-2
RSME	13.26	13.08	**12.48**	12.99
RMLSE	0.1991	0.2093	**0.1936**	0.1999
MAE	9.853	9.619	**9.409**	9.954
R2	0.8905	0.8934	**0.9031**	0.895
SF	290.53	292.9	**242.3**	245.2
QL-0.1	-	2.154	3.848	**1.986**
QL-0.9	-	**2.288**	3.890	2.356

Table 3. Accuracy comparisons of three methods on FD003.

Datasets	FD003			
	M1	M2	M3-1	M3-2
RSME	12.66	**11.90**	12.71	12.60
RMLSE	0.1821	**0.1671**	0.1820	0.1654
MAE	9.428	**8.636**	9.362	8.967
R2	0.8955	**0.9077**	0.8948	0.8965
SF	276.8	**251.0**	302.6	283.8
QL-0.1	-	**2.043**	4.796	2.350
QL-0.9	-	**1.800**	2.908	1.988

To further verify the performance of our probabilistic RUL prediction framework, we also compared our approach with other state-of-the-art methods, including Support Vector Regression (SVR) copied from [24], DCNN [8] (point estimation) and Bayesian DL [16]. We can observe that our proposed approach can achieve a slightly better performance than other methods in both datasets.

Table 4. Accuracy comparisons of other state-of-the-art methods, and accuracies of other methods are directly copied from the original paper.

Datasets	FD001		FD003	
Metrics	RMSE	SF	RMSE	SF
SVR	20.96	1381.5	21.05	1598.3
DCNN [8]	12.61	273.7	12.64	284.1
Bayesian DL [16]	**12.19**	267.2	12.07	409.4
MResNet9	12.48	**242.3**	**11.90**	**251.0**

3.5. Uncertainty estimation

To make the *aleatoric* uncertainty quantification more understandable, we show the [10%, 90%] prediction interval estimation (that is 80% confidence interval) of RUL (EU24 in FD003) using both parametric and non-parametric approaches in Fig. 1. All the methods provided a decreasing uncertainty over the cycle, which is very useful for the RUL prediction in practice, as the shorter RUL is associated with higher risk. As shown in Fig. 1 (a) and (b), when the trade-off parameter λ of the parametric approach is too large, the learning procedure will pay more attention to the variance decay term, which means that the model would prefer to output a relatively small variance. From the perspective of probability, a large λ means we assume a strong prior of the variance in the loss function, which might destroy the assumed parametric distribution.

4. Conclusion

We constructed a probabilistic RUL prediction framework using parametric and non-parametric approaches to estimate the probability density of target outputs based on a DCNN model. Our proposed framework can provide not only a point estimate of the RUL, but also the associated confidence interval of the RUL prediction. In

addition, one publicly available turbofan engine degradation simulation dataset was used to verify the performance of our solution via comparisons with other state-of-the-art methods.

Acknowledgements

This work was supported by Natural Science Foundation of China (No. 51835009, No. 51705398) and the Fundamental Research Funds for the Central Universities (xzy022019072).

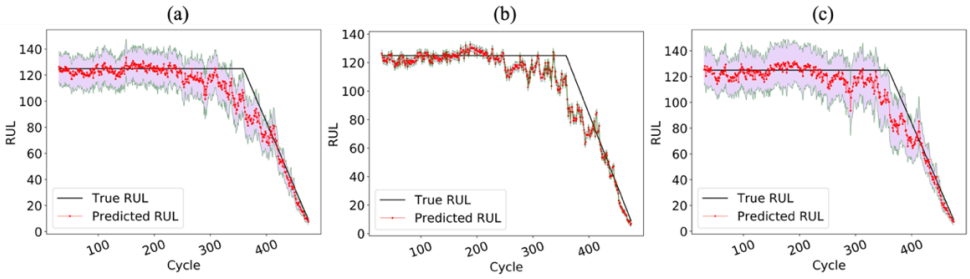

Fig. 1. The [10%, 90%] prediction interval estimation (80% confidence interval) of RUL of EU24 in FD003. (a) the parametric approach with $\lambda = 10^{-5}$; (b) the parametric approach with $\lambda = 1$; (c) the nonparametric approach.

References

[1] Z. Zhao, T. Li, J. Wu, C. Sun, S. Wang, R. Yan, X. Chen, Deep learning algorithms for rotating machinery intelligent diagnosis: An open source benchmark study, ISA Transactions. (2020).

[2] M. Kordestani, M. Saif, M.E. Orchard, R. Razavi-Far, K. Khorasani, Failure Prognosis and Applications—A Survey of Recent Literature, IEEE Transactions on Reliability. (2019) 1–21.

[3] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, J. Lin, Machinery health prognostics: A systematic review from data acquisition to RUL prediction, Mechanical Systems and Signal Processing. 104 (2018) 799–834.

[4] H.-Z. Huang, H.-K. Wang, Y.-F. Li, L. Zhang, Z. Liu, Support vector machine based estimation of remaining useful life: current research status and future trends, J Mech Sci Technol. 29 (2015) 151–163.

[5] Z. Li, K. Goebel, D. Wu, Degradation Modeling and Remaining Useful Life Prediction of Aircraft Engines Using Ensemble Learning, J. Eng. Gas Turbines Power. 141 (2018) 041008-041008–10.

[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature. 521 (2015) 436–444.

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

[8] X. Li, Q. Ding, J.-Q. Sun, Remaining useful life estimation in prognostics using deep convolution neural networks, Reliability Engineering & System Safety. 172 (2018) 1–11.

[9] K.T.P. Nguyen, K. Medjaher, A new dynamic predictive maintenance framework using deep learning for failure prognostics, Reliability Engineering & System Safety. 188 (2019) 251–262.

[10] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision?, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2017: pp. 5580–5590.

[11] D. Liu, W. Xie, H. Liao, Y. Peng, An Integrated Probabilistic Approach to Lithium-Ion Battery Remaining Useful Life Estimation, IEEE Transactions on Instrumentation and Measurement. 64 (2015) 660–670.

[12] J. Deutsch, D. He, Using Deep Learning-Based Approach to Predict Remaining Useful Life of Rotating Components, IEEE Transactions on Systems, Man, and Cybernetics: Systems. 48 (2018) 11–20.

[13] W. Peng, Z.-S. Ye, N. Chen, Bayesian Deep-Learning-Based Health Prognostics Toward Prognostics Uncertainty, IEEE Transactions on Industrial Electronics. 67 (2020) 2283–2293.

[14] B. Wang, Y. Lei, T. Yan, N. Li, L. Guo, Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery, Neurocomputing. 379 (2020) 117–129.

[15] M. Kraus, S. Feuerriegel, Forecasting remaining useful life: Interpretable deep learning approach via variational Bayesian inferences, Decision Support Systems. 125 (2019) 113100.

[16] M. Kim, K. Liu, A Bayesian Deep Learning Framework for Interval Estimation of Remaining Useful Life in Complex Systems by Incorporating General Degradation Characteristics, IISE Transactions. 0 (2020) 1–23.

[17] R. Wen, K. Torkkola, B. Narayanaswamy, D. Madeka, A Multi-Horizon Quantile Recurrent Forecaster, ArXiv:1711.11053 [Stat]. (2018).

[18] Y. Chen, Y. Kang, Y. Chen, Z. Wang, Probabilistic forecasting with temporal convolutional neural network, Neurocomputing. 399 (2020) 491–501.

[19] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 2016: pp. 770–778.

[20] R. Koenker, G. Bassett, Regression Quantiles, Econometrica. 46 (1978) 33–50.

[21] A. Saxena, K. Goebel, D. Simon, N. Eklund, Damage propagation modeling for aircraft engine run-to-failure simulation, in: 2008 International Conference on Prognostics and Health Management, 2008: pp. 1–9.

[22] A. Saxena, K. Goebel, Turbofan engine degradation simulation data set, NASA Ames Prognostics Data Repository. (2008).

[23] J. Wu, K. Hu, Y. Cheng, H. Zhu, X. Shao, Y. Wang, Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network, ISA Transactions. (2019).

[24] G. Sateesh Babu, P. Zhao, X.-L. Li, Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life, in: S.B. Navathe, W. Wu, S. Shekhar, X. Du, X.S. Wang, H. Xiong (Eds.), Database Systems for Advanced Applications, Springer International Publishing, Cham, 2016: pp. 214–228.