A Bayesian Optimization AdaBN-DCNN Method With Self-Optimized Structure and Hyperparameters for Domain Adaptation Remaining Useful Life Prediction

JIALIN LI1 AND DAVID HE2
1School of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China
2Department of Mechanical and Industrial Engineering, The University of Illinois at Chicago, Chicago, IL 60607, USA
Corresponding author: David He (davidhe@uic.edu)

ABSTRACT The prediction of remaining useful life (RUL) of mechanical equipment provides a timely understanding of the equipment degradation and is critical for predictive maintenance of the equipment. In recent years, the applications of deep learning (DL) methods to predict equipment RUL have attracted much attention. There are two major challenges when applying the DL methods for RUL prediction: (1) It is difficult to select the prediction model structure and hyperparameters such as network depth, learning rate, batch size, and etc. (2) The developed prediction model is domain dependent, i.e., it can only give good prediction performance in one data domain (one particular type of working conditions and fault modes). In order to meet the challenges, a novel RUL prediction method developed using a deep convolutional neural network (DCNN) combined with Bayesian optimization and adaptive batch normalization (AdaBN) is presented in this paper. The proposed RUL prediction model is validated by the turbofan engine degradation simulation dataset provided by NASA. The prediction results show that the proposed prediction model provides better prediction results than model structures obtained by random search and grid search. The results also show that the domain adaptation capability of the prediction model has been improved.

INDEX TERMS Remaining useful life prediction, Bayesian optimization, adaptive batch normalization, domain adaptation.

I. INTRODUCTION
In recent years, mechanical equipment has become more and more complicated with the development of new sciences and technologies. Therefore, the prognostics and health management (PHM) of mechanical equipment is becoming a more attractive topic. PHM provides a comprehensive management solution such as remaining useful life (RUL) prediction for maintaining the health of mechanical systems [1]. Over recent years, the research on developing data-driven RUL prediction methods has attracted much attention [2], [3]. Deep learning (DL) methods as a type of data-driven methods can achieve nonlinear mapping between training data and targets. Papers on developing DL based RUL prediction approaches have reported its superiority over other traditional data-driven approaches such as statistical approaches and so on [4], [5]. However, there are some challenges when using DL for RUL prediction such as network structure selection, hyperparameter optimization, and domain dependent etc. The purpose of the proposed method is to address these challenges.

Ren et al. [6] proposed a novel feature extraction method spectrum-principal-energy-vector to obtain the eigenvector. A deep convolutional neural network was used to predict bearing RUL. Yu et al. [7] proposed an autoencoder based on a bidirectional RNN to convert multi-sensor data into low-dimensional data in an unsupervised way. It was then used to construct a one-dimensional HI to reflect the degradation process of the device. Because learning features from a fixed window size may result in changes in local features and therefore may affect the prediction results, Zhao et al. [8] proposed a RUL prediction method based on the trend feature of the total time series representing degradation. An empirical mode decomposition (EMD) method was used to decompose and reconstruct the signals to obtain trend features. And then,
the conditional neural processes were used to evaluate the trend features. Finally, the RUL was predicted using the best trend features as an input to the RNN. In order to extract as much information as possible from the observed sequence, Elsheikh et al. [9] proposed a bidirectional hand-shaking long short-term memory (LSTM) to predict the RUL of a turbofan engine. Wu et al. [10] applied vanilla LSTM for RUL estimation of engineering systems, and proposed a dynamic differential technique to extract inter-frame information to enhance the cognitive ability of the degradation process model. Rigamonti et al. [11] used echo state networks (ESNs) to predict the RUL of industrial components such as turbofan engine. The innovation of the proposed method is to use a separate ESN memory capacity to aggregate ESNs results in a dynamic process and an additional ESN to estimate RUL by the mean variance estimation (MVE) method. Yang et al. [12] proposed an Elman neural network (ENN) based method to predict the RUL of an ultrasonic motor. The principal component analysis (PCA) was used to extract the motor degradation index from the monitoring data. An improved particle swarm optimization algorithm (IPSO) was used to improve the prediction accuracy of the ENN. Li et al. [13] used short-time Fourier transform (STFT) to extract time-frequency domain information from the rolling bearing dataset. A convolutional neural network (CNN) was used to achieve multi-scale feature fusion and intelligent RUL prediction. Zhu et al. [14] applied wavelet transform (WT) to obtain useful information, and then used a bilinear interpolation method to reduce the features. A multiscale convolutional neural network (MSCNN) model structure was proposed to save global and local information simultaneously to predict the RUL.

Among the DL based methods for mechanical equipment RUL prediction mentioned above, two important aspects related to model development need to be addressed: (1) selection of the model structure and hyperparameters (e.g. learning rate, batchsize, momentum etc.), and (2) domain adaptability of the developed model. The network structure and hyperparameter selection methods include: manual search, grid search, random search, and advanced optimization methods [15]. Most of the existing approaches develop and train the prediction model using manual search strategy [16], [17]. The model structure and training hyperparameters were chosen manually based on an understanding of the prediction methods and the experience of previous researchers. The grid search method is to equidistantly select several parameter values within the range to develop the models separately. Then the model with the best predicted result is selected as the target model. The references [8]–[10] applied the grid search method to select the optimal structure of the model. The random search method is to randomly select several sets of parameters within the range for model development. It has been empirically and theoretically proven that the random search is more effective for the hyperparameter optimization than for the grid search [18]. The domain adaptability of the prediction model is also an important performance quality indicator for model evaluation. In order to improve the domain adaptability of the developed prediction model, many attempts have been made by the researchers. Transfer learning (TL) [19], [20] strategy has proven to be effective for improving the adaptability of the model in many cases. A sufficient number of degraded samples for data-driven prediction are difficult to obtained, Zhang et al. [21] proposed a bi-directional long short-term memory (BLSTM) based TL method to predict RUL. The prediction model trained on one dataset can be used to predict the relevant target dataset by fine-tuning the TL strategy. In addition, adaptive batch normalization (AdaBN) [22], [23] algorithm can also be used to improve the domain adaptability of the prediction model. Initially, batch normalization (BN) was used to help stochastic gradient descent (SGD) optimization by adjusting the distribution of each layer of the output in the network [24]. It was later found that domain adaptation can be achieved by retraining the offset parameters in the BN unit.

To address the two issues encountered in developing DL based prediction methods mentioned above, this paper proposes a RUL prediction method based on deep convolutional neural network (DCNN) combined with Bayesian optimization [25] and AdaBN algorithm. The Bayesian optimization method is used to automatically select the network structure and hyperparameters. The AdaBN algorithm is used to improve the ability of predictive models to adapt to different data domains (DDs). The main contributions of this paper are summarized as follows:

1. This paper provides an optimization strategy for prediction network structure and hyperparameters based on data-driven method. The Bayesian optimization method is used to realize the self-selection of the network structure and hyperparameters. Compared with manual search and grid search, the proposed method can develop a RUL prediction model with better performance in a shorter time.

2. In this paper, the BN unit is added in the DCNN, and the prediction tasks of different DDs are adapted by modifying the offset parameters of the BN unit. Compared with the commonly used domain adaptation algorithm-TL, the AdaBN method requires fewer parameters to be fine-tuned in different DDs, which means that the calculation effort is reduced and the adaptation time is shortened.

The rest of the paper is organized as follows. In Section II, related background knowledge is provided. The proposed method is explained in Section III. In Section IV, the C-MAPSS dataset and validation results are described. Finally, Section V concludes the paper.

II. BACKGROUNDS

A. BAYESIAN OPTIMIZATION

The data-driven deep learning methods are more attractive because they are not affected by the physical structure of the system and can be developed rapidly. However, the hyperparameters of the deep learning network such as
within the range of values is global exploration. In Step 2, the distribution probability of \( f(x_{i+1}) \) is calculated according to the observation points \( f(x_{1:i}) \) and the rule of joint Gaussian distribution. Then, the position of the next observation point can be obtained in combination with the acquisition function. In Step 3, calculate \( f(x_{i+1}) \) and update the prediction function \( f_{i+1}^* \). Finally, repeat Steps 2 and 3. The hyperparameter set corresponding to the minimum value in all observation points is the best set.

Next, Gaussian process priors with Gaussian noise are introduced in Section II-A-1), kernel function options in Section II-A-2), and acquisition functions options in Section II-A-3).

1) GAUSSIAN PROCESS PRIORS

The Gaussian process (GP) [27] is a set of random variables so that any finite numbers of random variables have a joint Gaussian distribution. The GP can be represented entirely by the mean function and covariance function. The mean function \( m(x) \) and covariance function \( k(x, x') \) are expressed in Equations (2) and (3) as:

\[
m(x) = E[f(x)]
\]

\[
k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))]
\]

Therefore, \( f(x) \) obeys the GP can be expressed as:

\[
f(x) \sim GP[m(x) , k(x, x')]
\]

Fig. 1 shows a simple one-dimensional (1-D) GP with 3 observations.

The function \( f(x) \) is a GP, and the three observation points are \( x_1^*, x_2^*, \) and \( x_3^* \), respectively. So the joint distribution of the variables \( f(x_1^*) \), \( f(x_2^*) \), and \( f(x_3^*) \) is a Gaussian distribution. The solid black line in Fig. 1 is the mean of the function \( f(x) \) predicted by the GP through the observation point. The shaded area represents the 95% confidence interval for the Gaussian distribution of the function \( f(x) \). It ranges from \( m(x) - 1.96\sigma(x) \) to \( m(x) + 1.96\sigma(x) \), which \( m(x) \) and \( \sigma(x) \) are prediction GP mean and standard deviation. \( x_1 \) and \( x_2 \) are two randomly selected points in the range of \( x \), and the dotted line is a normal distribution of the \( f(x) \) value. \( x^+ \) is the point

\[
\text{Algorithm 1: Bayesian Optimization}
\]

**Input:** Within the range, randomly select \( i \) sets of hyperparameters \( x_i \). Each set contains \( m \) hyperparameters \( h_{p_1:m} \in x_i \), black-box function \( f \), and the number of optimization iterations \( n \).

**Step 1.** Global exploration:
- Calculate observations \( f_{i+1} = f(x_{i+1}) \), i.e., \( D_{i+n} = [x_{i+1}, f(x_{i+1})] \).
- Predict black-box function \( f_{i+1}^* \sim N(m(x_{i+1}), K) \), where \( K \) is shown in Equation (6).

**Step 2.** Determine the next observation point \( x_{i+1}^* \):
- According to the principle of GP distribution, \( f(x_{i+1}) \) and \( f(x_{i+1}) \) obey the joint Gaussian distribution as:
  \[
f(x_{i+1}) \sim N[m((x_{i+1})], [K])
\]
- The probability distribution of \( f(x_{i+1}) \) can be easily obtained as:\[P(f(x_{i+1})|D_{i+n})=N(\mu(x_{i+1}), \sigma^2(x_{i+1}))\], where \( \mu(x_{i+1}) = k(x_{i+1}, x_{i+1}) - \sigma(x_{i+1}) \) and \( \sigma(x_{i+1}) = k(x_{i+1}, x_{i+1})^{-1}k(x_{i+1}, x_{i+1}) \).
- \( x_{i+1} \) is determined by acquisition function \( \text{Acq}(x) \), i.e., \( x_{i+1} = \text{argmax}_{x \in X} \text{Acq}(x) \). A lower confidence bound acquisition function can be used: \( \text{AcqLcb}(x_{i+1}) = K(x_{i+1}, x_{i+1}) - \sigma(x_{i+1}) \).
- The next observation point can be calculated as \( x_{i+1} = \text{argmax}_{x \in X} \text{Acq}(x) \).

**Step 3.** Calculate \( f(x_{i+1}) \) and update prediction function \( f_{i+1}^* \):
- Use \( x_{i+1}^* \) determined in Step 2 to compute the new observations \( f_{i+1} = f(x_{i+1}) \).
- Predict black-box function \( f_{i+1}^* \sim N(m(x_{i+1}), K) \).

**Step 4.** For \( i = 1, 2, \ldots \) do
- Repeat Step 2 and Step 3.

**Output:** Predicted black-box function \( f_{i+1}^* \); the smallest observation \( f^+ \) and the corresponding hyperparameter set \( x_{\text{best}} \).
at which the \( f(x) \) value is the minimum for all observation points.

A commonly used squared exponential covariance function is shown in Equation (5). More other covariance functions are described in details in Section II-A-2. It can be known from Equation (5) that when the two points \( x_1 \) and \( x_2 \) are close, the \( k(x_1, x_2) \) value approaches 1, i.e., the mutual influence of the two points is larger. On the contrary, when the two points are far away, the \( k(x_1, x_2) \) value approaches 0, i.e., the interaction between the two points is weak.

\[
k(x_i, x_j) = \exp\left(-\frac{1}{2} (x_i - x_j)^2\right)
\] (5)

When Gaussian priors are performed on \( n \) observation points, it is known \( D_{1:n} = \{x_{1:n}, f(x_{1:n})\} \), and the function values \( f(x_{1:n}) \) are obey to a multivariate normal distribution, i.e., \( f(x_{1:n}) \sim N(m(x_{1:n}), K) \). And the kernel matrix of multivariate normal distribution is expressed as:

\[
K = \begin{bmatrix}
k(x_1, x_1) & \cdots & k(x_1, x_n) \\
\vdots & \ddots & \vdots \\
k(x_n, x_1) & \cdots & k(x_n, x_n)
\end{bmatrix}
\] (6)

The optimization is achieved by continuously increasing the observation points and performing repeating the Gaussian process prior until the prediction model is close to the real function \( f(x) \). Therefore, the choice of the location of the observation point \( x \) is crucial. It is known that there has \( n \) observation points \( D_{1:n} = \{x_{1:n}, f(x_{1:n})\} \). Then one needs to consider the next observation point \( x_{n+1} \) and the corresponding function value \( f(x_{n+1}) \). According to the principle of GP distribution, \( f(x_{1:n}) \) and \( f(x_{n+1}) \) obey the joint Gaussian distribution as:

\[
\begin{bmatrix}
f(x_{1:n}) \\
f(x_{n+1})
\end{bmatrix} \sim N \left( m \left( \begin{bmatrix} x_{1:n} \\ x_{n+1} \end{bmatrix} \right), \begin{bmatrix} K & k_s \\ k_s^T & k(x_{n+1}, x_{n+1}) \end{bmatrix} \right)
\] (7)

where \( k_s \) can be expressed as:

\[
k_s = [k(x_{n+1}, x_1), k(x_{n+1}, x_2) \cdots k(x_{n+1}, x_n)]
\] (8)

According to the mention above, the probability distribution of \( f(x_{n+1}) \) can be easily obtained as:

\[
P(f(x_{n+1} | D_{1:n}, x_{n+1})) \sim N(\mu(x_{n+1}), \sigma^2(x_{n+1}))
\] (9)

where \( \mu(x_{n+1}) \) and \( \sigma^2(x_{n+1}) \) can be expressed as:

\[
\begin{align*}
\mu(x_{n+1}) &= k_s^T K^{-1} f(x_{1:n}) \\
\sigma^2(x_{n+1}) &= k(x_{n+1}, x_{n+1}) - k_s^T K^{-1} k_s
\end{align*}
\] (10)

The derivation of the above equations is based on a noise-free environment. However, a noise-free environment in practical applications is rare. For example, applying a Gaussian process to fit the relationship between DLE and hyperparameters. Setting the same hyperparameters for multiple trials may result in different results. This shows that it is a noisy data environment. As shown in Equation (12), \( y \) is the signal after noise addition, where noise \( \epsilon \sim N(0, \sigma^2_{\text{noise}}) \).

And the kernel matrix of multivariate normal distribution in Equation (6) is changes as shown in Equation (13).

\[
y = f(x) + \epsilon
\]

\[
K = \begin{bmatrix}
k(x_1, x_1) & \cdots & k(x_1, x_n) \\
\vdots & \ddots & \vdots \\
k(x_n, x_1) & \cdots & k(x_n, x_n)
\end{bmatrix} + \sigma^2_{\text{noise}} I
\] (13)

Similarly, the probability distribution of \( f(x_{n+1}) \) in a noisy environment is also changed, and can be expressed as Equations (14-16) as:

\[
P(f(x_{n+1} | D_{1:n}, x_{n+1})) \sim N(\mu(x_{n+1}), \sigma^2(x_{n+1}) + \sigma^2_{\text{noise}})
\] (14)

\[
\mu(x_{n+1}) = k_s^T [K + \sigma^2_{\text{noise}} I]^{-1} f(x_{1:n})
\] (15)

\[
\sigma^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - k_s^T [K + \sigma^2_{\text{noise}} I]^{-1} k_s
\] (16)

2) KERNEL FUNCTION OPTIONS

The kernel function [28] has a crucial influence on the GP, which can affect the smoothness of the fitted curve. Equation (5) shows the standard squared exponential kernel. It can be seen from the equation that the points \( x_i \) and \( x_j \) with the same distance have the same covariance for different data environments. Therefore, when faced with a variety of complex prediction tasks, the standard squared exponential kernel is actually a little naive. An effective improvement method is to add hyperparameters in the kernel function to play a better role in a variety of data environments. The vector \( \theta \) can be applied to parameterize the kernel function, which can be expressed as \( k(x_i, x_j | \theta) \). In general, the parameterization of the kernel function is adjusted by parameters the signal standard deviation \( \sigma_l \) and the characteristic length scale \( \sigma_l \). The characteristic length scale briefly defines the distance that the input value becomes irrelevant to the response value. By defining \( \theta_1 = \log \sigma_l \) and \( \theta_2 = \log \sigma_l \), the parameters \( \sigma_l \) and \( \sigma_l \) can be enforced greater than 0.

The improved squared exponential kernel: the results of parameterizing the kernel function in Equation (5) are as follows:

\[
k(x_i, x_j | \theta) = \sigma^2_l \exp\left(-\frac{(x_i - x_j)^2}{\sigma^2_l} \right)
\] (17)

where \( \sigma_l \) is the characteristic length scale, and \( \sigma_l \) is the signal standard deviation.

In addition, other kernel functions commonly used for Bayesian optimization are the Matern class. The kernel Matern 3/2 and Matern 5/2 are defined as:

\[
k(x_i, x_j | \theta) = \sigma^2_f (1 + \sqrt{3} \frac{r}{\sigma_l} \exp(-\sqrt{3}\frac{r}{\sigma_l}))
\] (18)

\[
k(x_i, x_j | \theta) = \sigma^2_f (1 + \sqrt{5} \frac{r}{\sigma_l} + \frac{5r^2}{3\sigma_l^2} \exp(-\sqrt{5}\frac{r}{\sigma_l}))
\] (19)

where \( r = \sqrt{(x_i - x_j)^2} \).
3) ACQUISITION FUNCTIONS OPTIONS

The Bayesian optimization process is accomplished by continuously adding observation points and performing Gaussian process priors. Therefore, the choice of observation points will directly affect the performance of the GP, further affecting the distribution of the predicted function. A “goodness” observation point can shorten the number of Gaussian process iterations, so that the Gaussian posterior distribution is closer to the actual function. Based on the observation points \( D_{1: n} = \{x_{1:n}, f(x_{1:n})\} \) and the posterior distribution function \( Q \), the acquisition function is applied to find the next observation point \( x_{\text{next}} \).

As shown in Equation (20), the next observation point \( x_{\text{next}} \) at which the acquisition function \( \text{Acq}(x) \) has the maximum value, i.e., the point having the best potential to make the \( x_{\text{best}} \).

\[
x_{\text{next}} = \arg\max_{x \in X} \text{Acq}(x) \quad (20)
\]

There are many types of acquisition functions [29], and the three commonly used include confidence bound criteria, probability of improvement (PI), and expected improvement (EI).

The confidence bound criteria contains: lower confidence bound (LCB) and upper confidence bound (UCB). The LCB is used to find the minimum value of the function, and the UCB is used to find the maximum value. They can be expressed as:

\[
\text{Acq}_{\text{LCB}}(x) = k\sigma_Q(x) - \mu_Q(x) \quad (21)
\]
\[
\text{Acq}_{\text{UCB}}(x) = k\sigma_Q(x) + \mu_Q(x) \quad (22)
\]

where \( \mu_Q \) is the posterior mean, \( \sigma_Q \) is the posterior standard deviation, and \( k \geq 0 \).

The PI is to maximize the probability of improvement over the minimum value of all observation points. When using the PI method to find the next point, it is a pure exploitation process compared to balanced “exploitation” and “exploration”. In order to increase its “exploration” ability, the parameter \( \xi \) is added in the equation. The PI can be expressed as:

\[
\text{Acq}_{\text{PI}}(x) = P_Q(f(x) < f(x^+) - \xi) = \frac{\Phi\left(\frac{f(x^+) - \mu_Q(x) - \xi}{\sigma_Q(x)}\right)}{\Phi\left(\frac{-\xi}{\sigma_Q(x)}\right)} \quad (23)
\]

where \( \Phi(\cdot) \) is the cumulative distribution function of the standard normal, \( x^+ \) is the point with the minimum \( f \) value among all observation points, and \( \xi \) is a trade-off parameter slightly greater than 0.

The EI method considers not only the possibility of improvement, but also the room for improvement. So the “exploration” capability of the EI method is more than the PI method. The EI can be expressed as:

\[
\text{Acq}_{\text{EI}}(x) = E_Q(\max (0, f(x^+) - f(x) - \xi)) = \begin{cases} 
(f(x^+) - \mu_Q(x) - \xi) \Phi(Z) + \sigma_Q(x) \phi(Z), & \sigma_Q(x) > 0 \\
0, & \sigma_Q(x) = 0 
\end{cases} \quad (24)
\]

where \( \phi(\cdot) \) is the probability density function of the standard normal, and \( Z = \left(\frac{f(x^+) - \mu_Q(x) - \xi}{\sigma_Q(x)}\right) \).

To further compare the three acquisition functions, the GP with three acquisition functions is applied to find \( x_{\text{best}} \) in Equation (25) as:

\[
y = 2 - \left[ e^{-((x-2)^2)} + e^{-((x-0.5)^2)} + \frac{1}{x^2 + 1} \right] \quad (25)
\]

The GP prediction process with the three acquisition functions is shown in Fig. 2. The three columns (a), (b), and (c) correspond to the three acquisition functions LCB, PI, and EI. The initial observation points are set at \( x = -1 \) and \( x = 7 \), and the subsequent 7 observation points are obtained according to the three acquisition functions, corresponding to graphs Fig. 2(a)-2 to Fig. 2(a)-8, Fig. 2(b)-2 to Fig. 2(b)-8, and Fig. 2(c)-2 to Fig. 2(c)-8. Each graph in Fig. 2 contains two parts: the GP (upper) and the acquisition function curve (lower). The red solid line in the upper part is the objective function \( y \), the black dotted line is the mean of the Gaussian posterior distribution, and the green shaded area represents the 95% confidence interval of the predicted distribution.

Comparing the search for \( x_{\text{best}} \) point process of the three acquisition functions, the methods LCB and PI are all “sink” in the local minimum. However, there is a difference between the two methods. The PI method can jump out to search for other ranges after several times around the local minimum, while the LCB method is always searching for the \( x_{\text{best}} \) point around the local minimum. Compared with the first two methods, the results of the PI method are ideal. The Gaussian process with 9 observation points selected by PI method can find the best point.

B. DEEP CONVOLUTIONAL NEURAL NETWORK

The DCNN [30] is a stack by several CNN layers. The CNN is derived from the neocognitron model proposed by Japanese scholar Fukushima in 1980 [31]. The development of neocognitron model was inspired by the animal’s visual cortex, which consists of the S-layer (consist of simple cells) and C-layer (consist of complex cells). The theory of the S-layer and C-layer continue to evolve into convolution layer and pooling layer in the CNN. In the next 20 years, however, CNN has not been greatly developed due to the limitations of computing power at that time and the rise of algorithms such as HMM and support vector machine (SVM).

Until the upsurge of deep learning research caused by Hinton [32] in 2006, the CNN has regained widespread attention. And with the rapid development of computing devices, the recognition accuracy of CNN method on large-scale visuals far exceeds other methods. The algorithm based on CNN has won many awards in the ImageNet large scale visual recognition challenge (ILSVRC) since 2012, such as AlexNet, ZFNet, VGGNet, GoogLeNet, and ResNet. The CNN can use two-dimensional data as input, and also has the characteristics of weight sharing. This paper applies CNN to fuse multi-sensor signals to predict the RUL of mechanical equipment. The CNN typically
FIGURE 2. Influence of acquisition function on Gaussian process: (a) lower confidence bound, (b) probability of improvement, and (c) expected improvement.
includes convolution operation, activation operation, pooling operation, and dropout operation for overfitting prevention.

1) CONVOLUTIONAL OPERATION AND ACTIVATION OPERATION

The convolution operation [33] consists of input data, convolution kernels, and the resulting feature maps. Fig. 3 shows a two-dimensional (2D) convolution operation in which the input data is 2D data consisting of signals collected by \( n \) sensors.

The convolution filter extracts multiple matrixes by sliding over the input data. The size of the data extracted each time by convolutional filter is the same as the size of the convolution kernel. The three shaded portions in the input data are the three adjacent data extractions of the sliding convolutional filter. The element-wise multiplication was performed between the each data matrix extracted by the convolution filter and the convolution kernel. Then all the elements in the result are summed and plus an offset to obtain an element in feature map. As the convolutional filter slides continuously to extract the data matrix, a complete feature map can be obtained. By repeating the above convolution operation on the \( m \) convolution kernels, \( m \) feature maps can be obtained.

As shown in Fig. 3, the vertical axis of the input data is the length of the time sequence \( L_s \), and the horizontal axis is signals collected by \( n \) sensors. The convolution filter and the convolution kernel are the same size, i.e., \( \text{filter} \_\text{size} = \text{kernel} \_\text{size} = \{k_1, k_2\} \). The sliding step of the convolution filter is \( [S_1, S_2] \). The number of convolution kernels and the number of feature maps are equal to \( m \). The size of the feature map is \( [(L_s - k_1)/S_1 + 1, (n - k_2)/S_2 + 1] \).

As described above, it is known that the larger the sliding step of the convolution filter, the smaller the feature map obtained. In order to increase the size of feature map and extract the edge information of the input data, the input edge padding process is performed. If the padding dimension is \( p \), the feature map size obtained after data edge padding is \( [(L_s - k_1 + 2p)/S_1 + 1, (n - k_2 + 2p)/S_2 + 1] \). Therefore, the obtained feature map size can be adjusted by the padding dimension \( p \). It should be noted that the obtained feature map size should be an integer. The data used for padding is usually 0, which is called zero-padding. Other data can also be used for padding the input data. The mathematical expression of the convolution operation and the activation operation are as follows:

\[
\begin{align*}
    h_{ij}^m &= \text{sum} \left( W_m \odot F_{ij}^c \right) + b_m \quad (26) \\
    y_{ij}^m &= \varphi(h_{ij}^m) \quad (27)
\end{align*}
\]

where \( W_m \) is the \( m \)-th kernel matrix, \( F_{ij}^c \) is the data matrix extracted by the sliding convolutional filter, \( \odot \) is the element-wise multiplication, \( b_m \) is the offset of the \( m \)-th convolution kernel, \( h_{ij}^m \) is the \( m \)-th feature map, \( i \) and \( j \) are the stride of the convolutional filter in the vertical and horizontal directions, \( \text{sum}(\&) \) is a process to sum all of elements in \( \& \), \( \varphi(\cdot) \) is an activation function, and \( y_{ij}^m \) is the output of activation.

2) DROPOUT OPERATION AND POOLING LAYER

The purpose of dropout operation [34] for neurons in the hidden layer is to avoid the network overfitting. It randomly makes the hidden layer neurons to zero based on the dropout probability \( d_p \). When the dropout operation is not performed, all parameters of the network are fine-tuned at each epoch of training. Multiple iterations training under the same network structure resulted in a very low training loss, but the testing results are usually not as expected. By randomly dropout part of the neurons in the hidden layer, the numbers of the neurons involved in each training epoch are different. The dropout layer is usually placed after the convolution layer. As shown in Fig. 4, the output of the activation function is randomly zeroed according to the dropout probability \( d_p \). The role of the dropout layer is to randomly reset some values in the feature map to zero. The symbol \( \odot \) in Fig. 4 represents the neuron reset to zero.

The mathematical expression of the dropout operation is shown in Equation (28). The dropout operation is only used during the training process. However, all neurons are active when the network is used for testing. Therefore, the coefficient \( 1/(1 - d_p) \) was used to correct the difference between training and testing in dropout operation, as shown in Equation (29).

\[
\begin{align*}
    R &= \text{rand}(\text{size}(Y)) > d_p \quad (28) \\
    \tilde{Y} &= (R \odot Y)/(1 - d_p) \quad (29)
\end{align*}
\]

where \( \text{rand}(\cdot) \) function randomly generated numbers between 0 to 1, \( \text{size}(\cdot) \) extracts the dimensions of the matrix \( Y \), \( R \) is dropout mask containing only 0 and 1, \( \tilde{Y} \) is the output of dropout layer.
Following the dropout layer, the pooling layer [35] is used to reduce the dimension of the data and extract the data characteristics. The pooling filter size is \([k_3, k_4]\) and the sliding step of the pooling filter is \([s_3, s_4]\). Unlike the convolution operation, only the data extracted by the filter itself performs some pooling processing. Commonly used pooling operations are maximum pooling \((\text{Max} \text{ pooling})\) and average pooling \((\text{Average} \text{ pooling})\). The \text{Max} pooling selects the maximum value of the data extracted by the filter, and the \text{Average} pooling averages all the data extracted by the filter. So the number of output elements is equal to the number of filters in the pooling layer.

The mathematical expressions of the \text{Max} pooling operation and \text{Average} pooling operation are as follows:

\[
y_{ij}^p = \text{Max} \text{ pooling}(\mathbf{F}_{ij}^p)
\]

\[
y_{ij}^p = \text{Average} \text{ pooling}(\mathbf{F}_{ij}^p)
\]

where \(\mathbf{F}_{ij}^p\) is the data matrix extracted by the sliding pooling filter, \(y_{ij}^p\) is the \(p\)-th output of pooling layer, \(i\) and \(j\) are the stride of the pooling filter in the vertical and horizontal directions.

\section{C. Adaptive Batch Normalization}

A DL based RUL prediction model can be developed based only on data. The dataset used to develop a model with different working conditions and fault types is called the data domain (DD). The prediction models developed in different DDS are often not universal to each other. In order to make the developed prediction model have a better adaptability to various DDSs, this paper applies the AdaBN algorithm [36]. AdaBN is a novel transfer learning strategy based on batch normalization (BN). A BN layer normalizes each input channel across a mini-batch. It is usually placed between the convolution operation and the activation operation. The BN layer can speed up the network training process and reduce the sensitivity of the convolution operation on the output data. In the training process, all the samples are divided into several mini-batches for separate training. If the sample size is represented by \(s\) and the number of samples included in each mini-batch is represented by \(b\), then the number of batches \(n_b\) is equal to \(s/b\). As shown in Equation (32), the BN layer first normalizes the output of each channel by subtracting the mean of the mini-batch and dividing by standard deviation of the mini-batch. However, the normalization results are trapped in the linear activation interval of the sigmoid and tanh activation functions. Therefore, the normalized results shift by the learnable offset \(\beta^c\) and scale by the learnable scale factor \(\gamma^c\). Moreover, when \(\beta^c = \mu(X_{1:b}^c)\) and \(\gamma^c = \sigma^2(X_{1:b}^c)\), it is equivalent to keeping the original state of the input data \(X_j^c\) unchanged.

\[
\hat{X}_j^c = \frac{X_j^c - \mu(X_{1:b}^c)}{\sigma^2(X_{1:b}^c)} + \epsilon
\]

\[
Y_j^c = \gamma^c \cdot \hat{X}_j^c + \beta^c
\]

where \(X_j^c\) is \(j\)-th \((j = 1, \ldots, b)\) input in a mini-batch of channel \(c\), \(\mu(X_{1:b}^c)\) and \(\sigma^2(X_{1:b}^c)\) are the mean and variance of one mini-batch in channel \(c\), \(\epsilon\) is a very small positive number for improving numerical stability when \(\sigma^2(X_{1:b}^c)\) is very small, and \(\gamma^c, \beta^c\) are scale factor and offset.

The test data is not batch-processed and input into the network at one time. Therefore, \(\mu\) and \(\sigma^2\) in Equation (32) are replaced by the \(\mu_{\text{test}}\) and \(\sigma^2_{\text{test}}\) as shown in Equations (34-35).

\[
\mu_{\text{test}} = E(\mu_{1:n_b})
\]

\[
\sigma^2_{\text{test}} = \frac{n_b}{n_b - 1} E(\sigma^2_{1:n_b})
\]

where \(n_b\) is the number of batches in training process, and \(E(\ )\) is the average operation.

When the prediction DD is different but similar to the training DD, the transfer learning method is commonly used to retrain the network. The traditional transfer learning method retrain parts of the weights and bias in the network to adapt to the different DDSs. In this way, not a large amount of parameters need to be retrained. Only the retrained layers in the network play a predictive role. The AdaBN method applied in this paper keeps the weights and biases of the network unchanged, and only retrain the scale factor \(\gamma^c\) and offset \(\beta^c\) in the BN layer. Of course, \(\mu\) and \(\sigma^2\) should also be replaced with \(\mu_{\text{target}}\) and \(\sigma^2_{\text{target}}\).

\[
\hat{X}_j^c = \frac{X_j^c - \mu_{\text{target}}}{\sigma^2_{\text{target}} + \epsilon}
\]

\[
Y_j^c = \gamma^c_{\text{retrain}} \cdot \hat{X}_j^c + \beta^c_{\text{retrain}}
\]

where \(\mu_{\text{target}}\) and \(\sigma^2_{\text{target}}\) are mean and variance of target domain, \(\gamma^c_{\text{retrain}}\) and \(\beta^c_{\text{retrain}}\) are retrained scale factor and offset with target data.
of the researcher to select the best hyperparameters through multiple trials. Table 2 shows the parameters selected by Bayesian optimization method, including structural parameter and training parameters. The range of parameters is chosen empirically. The Bayesian algorithm selects the parameters with the best prediction result through multiple iterations within the set range.

The structural parameter ‘network depth’ will be introduced below in conjunction with the framework graph of the proposed method. This paper uses a piecewise learning rate. First set an initial learning rate, then the learning rate gradually decreases by the amount determined as a certain number of training epochs multiplying with a certain factor. The number of epochs for dropping the learning rate is represented by ‘learning rate drop period’, and the factor for dropping the learning rate is represented by ‘learning rate drop factor’. The learning rate drop period and learning rate drop factor are set to 10 and 0.5 respectively. The training samples will be divided into several mini-batches for training. The sample size contained in each batch is represented by batchsize. The batchsize represents an integer number of samples. The standard gradient descent (SGD) algorithm with momentum is used to updates the network parameters (weights and biases) and minimizes the loss function. The SGD algorithm can oscillate along the steepest descending path to the optimum, and the momentum term can reduce this oscillation. Adding the L2 regularization term for the weights is one way to reduce over-fitting.

Fig. 5 shows the framework of the Bayesian optimization AdaBN-DCNN RUL prediction model with uncertain structures. The prediction method proposed in this paper is based on DCNN, which includes several convolutional layers, one pooling layer and two fully connected layers. The uncertain structure in the prediction model is determined by 3 dotted boxes. The number of dotted boxes represents the ‘network depth’ in Table 2. The same number of convolutional layers contain in the three dotted boxes.

And the number of dashed boxes that make up the prediction model is determined by Bayesian optimization. In order to improve the network’s ability to adapt to multiple DDs, BN units have been added in the convolutional layer and the fully connected layer. When dealing with different predicted DDs, only the green BN unit in Fig. 5 needs to be retrained to make a better prediction result. The filters of multiple convolutional layers in the same dashed boxes have the same size, stride and numbers. The filter size in the three dashed boxes are [6, 1], [4, 1] and [2, 1] respectively. The same number of filters is used for all the convolution layers in the same dashed box. The initial number of filters can be determined by Equation (38). The number of filters used for the convolution layer in the three dashed boxes are $\text{Num}_{\text{Filters}}$, $2 \times \text{Num}_{\text{Filters}}$, and $3 \times \text{Num}_{\text{Filters}}$ respectively.

$$\text{Num}_{\text{Filters}} = \text{round}(10 / \sqrt{n_d})$$  \hspace{1cm} (38)

where $n_d$ is the ‘Network depth’, round( ) is a function round to nearest integer, and $\text{Num}_{\text{Filters}}$ is the initial number of filters.

There is a pooling layer and two fully connected layers after the convolutional layers. The Average_pooling operation is used in the pooling layer. The pooling filter size is [1, 2], and the stride of the pooling filter is [1, 1]. The number of neurons in the two fully connected layers is 30 and 1 respectively. The activation function used in the prediction network is ReLU. The activation function is not shown in Fig. 5, and it should be placed behind the green BN unit. The output of the last fully connected layers is the predicted RUL.

Table 3 shows the overall flow of the proposed RUL prediction method. The purpose of the proposed method is
TABLE 3. An overall procedure of the proposed RUL prediction method.

| Algorithm 2: The proposed method |
|----------------------------------|
| **Purpose:** Construct a prediction model with self-optimizing structure, hyperparameter selection, and domain adaptability. |
| **Step 1. Data preparation:** |
| - Sliding time window processes training and test data into short segments of [30, 14]. |
| - Normalize training and test data with z-score. |
| - Save the processed training data \(x_{\text{train}}\) and test data \(x_{\text{test}}\). |
| **Step 2. AdaBN-DCNN model construction:** |
| - Set the structure of AdaBN-DCNN model as shown in Fig. 5. |
| - Input training data and hyperparameters into the model for training. The types and ranges of the hyperparameters are shown in Table 2. |
| - The trained AdaBN-DCNN model can be represented by \(f_{\theta_{\text{Ada}}, \beta_{\text{Ada}}, \lambda_1} \text{.} \) |
| **Step 3. Search for the best set of hyperparameters:** |
| - Randomly select one set of hyperparameters to train the AdaBN-DCNN model. Compute the test error \(E_{\text{test}}\). |
| - Perform Gaussian process regression and search for the next observation (A new set of hyperparameters need to be verified) based on acquisition function. |
| - Input the new set of hyperparameters into the AdaBN-DCNN model for training and compute a new test error. |
| - Repeat the above two steps until the set number of optimization steps is reached. |
| - Finally, record the best test error and corresponding hyperparameters. |
| **Step 4. Strategies for the new test data domain \(x'_{\text{test}}\):** |
| - The best hyperparameters have been selected in Step 3 to train the prediction model constructed in Step 2. |
| - When the test data domain changes, it is only necessary to apply the corresponding \(x'_{\text{train}}\) to retrain the scale factor and offset in the BN unit. No other parameters need to be changed in the model. |
| **Output:** A prediction model with hyperparameters optimized and the domain adaptability enhanced. |

TABLE 4. Description of the C-MAPSS dataset.

| Dataset Contents         | Sub-datasets |
|--------------------------|--------------|
|                         | FD001 | FD002 | FD003 | FD004 |
| Engines in training set  | 100   | 260   | 100   | 249   |
| Engines in test set      | 100   | 259   | 100   | 248   |
| Max/min cycles for training | 362/128 | 378/128 | 525/145 | 543/128 |
| Max/min cycles for test Operating condition | 303/31 | 367/21 | 475/38 | 486/19 |
| Fault modes              | 1     | 6     | 1     | 6     |
| TW length                | 30    | 21    | 36    | 18    |
| Training samples         | 17731 | 48559 | 21220 | 57016 |
| Test samples             | 100   | 259   | 100   | 248   |

FIGURE 6. A simplified diagram of the simulation engine in C-MAPSS.

IV. VALIDATION OF THE PROPOSED METHOD

A. C-MAPSS DATASET DESCRIPTION

The simulated turbofan engine run to failure data was used to validation the proposed prediction model. The simulated data was produced using a model based simulation program C-MAPSS developed by NASA [37]. This section provides a detailed description of 5 parts: simulated signal selection, time window (TW) processing, data normalization, RUL function correction, and model evaluation method. As shown in Fig. 6, the structure of the simulated turbofan mainly consists of 5 parts: fan, low pressure compressor (LPC), high pressure compressor (HPC), combustor, high pressure turbine (HPT), and low pressure turbine (LPT). Symbols N1 and N2 in Fig. 6 represent the core shaft and the fan shaft, respectively.

The C-MAPSS datasets contains multiple sets of run to failure data from different engine states. It is divided into 4 sub-datasets according to the simulated operating conditions and fault modes. Each sub-dataset contains independent training data and test data which consists of several sets of data from different engine run to failure health conditions. For example, FD001 contains 100 sets of training engine data and 100 sets of test engine data. In order to test all the data in the test dataset, the TW length should be less than or equal to the minimum number of cycles in the test dataset. More details on the C-MAPSS dataset are given in Table 4.

The description of the 21 collected engine simulation sensor outputs is provided in Table 5. As the operating state of the engine changes, some of the collected data show a significant trend of increasing (↑) or decreasing (↓), while others show a fluctuating trend (∼). In order to meet the prediction requirements, only 14 simulation output signals with a significant trend were selected as inputs to the predictive model.

14 signals with regular trends in Table 5 were selected to evaluate the RUL of the engine. The RUL of the engine is measured in cycles. Each data point in each cycle of the engine contains 14 features. Normally, one would use the 14 features as inputs to predict the RUL. Since the signals don’t change consistently with RUL of the engine and the time series data cannot directly fed into a CNN, the method proposed uses a period of time data to predict the RUL of the
engine as the RUL of the last time point in this period. The sliding TW method was used to cut an entire time series data into several segments. The time series length is $T_x$, the TW length is $L_{tw}$, and the sliding step of the TW is $S_l$. Then the obtained number of data segments is $(T_x - L_{tw})/S_l + 1$.

As a result of the pre-processing, 14 engine-related features were generated from the 14 selected sensor output signals. The values of these features vary from zero to hundreds. In order to use these features uniformly as input to the predictive model, it is necessary to normalize the feature values. In this paper, the $z$-score normalization processing was applied to process each feature separately, as shown in Equation (39):

$$y_n = (x_n - \mu_x)/\sigma_x$$  \hspace{1cm} (39)

where $x_n$ is the $n$-th feature vector, $\mu_x$ is the average of feature vector $x_n$, $\sigma_x$ is the standard deviation of feature vector $x_n$, and $y_n$ is the normalized data.

The first engine of the FD001 training set contains 192 cycles. The selected 14 engine related features were normalized as shown in Fig. 7(a). It can be seen that the 14 signals either increase or decrease with engine degradation. The predictive model predicts the RUL of the engine based on changes in parameters during engine degradation. However, it can be seen from Fig. 7 that the features remain relatively unchanged at the beginning of the engine operation. Based on this observation, the standard output of the prediction model was modified. The commonly used linear RUL function was replaced by the piecewise linear RUL function. According to the literature [38], the maximum service life of the engine is set to 125. As shown in Fig. 7(b), a piecewise linear RUL function is obtained by replacing $RUL > 125$ in the linear RUL function with 125.

To measure the RUL prediction accuracy, both late prediction and early prediction errors have to be considered. When the predicted RUL is greater than the actual RUL, it is called late prediction, and the opposite is called early prediction. In order to evaluate the predictive ability of the model, two commonly used evaluation methods are adopted in this paper, as shown in Equations (40) and (41). Comparing the two evaluation methods, it can be found that $E_{RMSE}$ in Equation (40) has the same penalty for both the late prediction and the early prediction, while $E_{score}$ in Equation (41) has a greater penalty for the late prediction.

$$E_{RMSE} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (y^p_i - y_i)^2}$$  \hspace{1cm} (40)

$$E_{score} = \sum_{i=1}^{m} d_i$$  \hspace{1cm} (41)

where $E_{RMSE}$ is the root mean square error of the prediction result, $E_{score}$ is a prediction result evaluation method, $y^p_i$ is the predicted RUL of $i$-th test engine, $y_i$ is the actual RUL of $i$-th test engine, and $m$ is the number of test engines.

### B. RESULTS AND ANALYSIS

1) **PREDICTION RESULTS WITH BAYESIAN OPTIMIZATION**

The advantage of the proposed method is that it can optimize the structure and select hyperparameters by itself. Therefore, only the range of the parameters needs to be determined during network training, as shown in Table 2. As shown in Fig. 5, the network includes three types of layers: convolutional layer, pooling layer, and fully connected layer. The filter sizes of the convolutional layers in the three dashed boxes are [6, 1, 4] and [2, 1], respectively. All the filters have the same stride [2, 1]. The pooling filter size is [1, 2], and the stride of the pooling filter is [1, 1]. The number of neurons in the two fully connected layers are 30 and 1. The standard gradient descent (SGD) algorithm with momentum was used to update the network parameters. The maximum number of training epochs was set to 40. The number of epochs for dropping the learning rate was represent by ‘learning rate drop period’.

### TABLE 5. 21 sensor outputs of the simulation engine running.

| #  | Symbol | Description                   | Units | Trend |
|----|--------|-------------------------------|-------|-------|
| 1  | T2     | Total Temperature at fan inlet | °R    | ~     |
| 2  | T24    | Total temperature at LPC outlet | °R | ↑     |
| 3  | T30    | Total temperature at HPC outlet | °R | ↑     |
| 4  | T50    | Total temperature LPT outlet | °R | ↑     |
| 5  | P2     | Pressure at fan inlet         | psia  | ~     |
| 6  | P15    | Total pressure in bypass-duct | psia  | ~     |
| 7  | P30    | Total pressure at HPC outlet | psia | ↑     |
| 8  | NF     | Physical fan speed            | rpm   | ↑     |
| 9  | Nc     | Physical core speed           | rpm   | ↑     |
| 10 | Epr    | Engine pressure ratio         |       | --    |
| 11 | Ps30   | Static pressure at HPC outlet | psia  | ↑     |
| 12 | Phi    | Ratio of fuel flow to Ps30    | pps/psi | ↓  |
| 13 | NRf    | Corrected fan speed           | rpm   | ↑     |
| 14 | NRe    | Corrected core speed          | rpm   | ↑     |
| 15 | BPR    | Bypass ratio                  |       | --    |
| 16 | farB   | Burner fuel-air ratio         |       | --    |
| 17 | BtBleed| Bleed enthalpy                |       | ~     |
| 18 | NF_dmd | Demanded fan speed            | rpm   | ~     |
| 19 | PCNR_d | Demanded corrected fan speed  | rpm   | ~     |
| 20 | W31    | HPT coolant bleed             | lbm/s | ↓      |
| 21 | W32    | LPT coolant bleed             | lbm/s | ↓      |

### FIGURE 7. Engine #1 in FD001 sub-dataset: (a) normalized 14 signals and (b) piecewise linear RUL function.

When the predicted RUL is greater than the actual RUL, it is called late prediction, and the opposite is called early prediction. In order to evaluate the predictive ability of the model, two commonly used evaluation methods are adopted in this paper, as shown in Equations (40) and (41). Comparing the two evaluation methods, it can be found that $E_{RMSE}$ in Equation (40) has the same penalty for both the late prediction and the early prediction, while $E_{score}$ in Equation (41) has a greater penalty for the late prediction.

$$E_{RMSE} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (y^p_i - y_i)^2}$$  \hspace{1cm} (40)

$$E_{score} = \sum_{i=1}^{m} d_i$$  \hspace{1cm} (41)

where $E_{RMSE}$ is the root mean square error of the prediction result, $E_{score}$ is a prediction result evaluation method, $y^p_i$ is the predicted RUL of $i$-th test engine, $y_i$ is the actual RUL of $i$-th test engine, and $m$ is the number of test engines.
and the factor for dropping the learning rate was represent by ‘learning rate drop factor’. The learning rate drop period and learning rate drop factor were set to 10 and 0.5, respectively.

The results of predicting FD001 sub-dataset of the C-MAPSS using the proposed prediction model are provided in Table 6. The information for the 30 iterations in the table include: evaluation result, running time, $\text{E}_{\text{RMSE}}$, best $\text{E}_{\text{RMSE}}$, and selected 5 hyperparameters. There are three measures for the evaluation results: ‘Best’, ‘Accept’, and ‘Error’. The evaluation results are based on $\text{E}_{\text{RMSE}}$. When the obtained $\text{E}_{\text{RMSE}}$ is the current minimum, the measure of the evaluation results for this iteration is ‘Best’. When the obtained $\text{E}_{\text{RMSE}}$ is not the minimum value and the result of this iteration is accepted by Bayesian optimization, the measure of the evaluation results for this iteration is ‘Accept’. When the result cannot be used for the subsequent Bayesian optimization, the measure of the evaluation results is ‘Error’. $\text{E}_{\text{RMSE}}$ is the evaluation result of the model tested by the test dataset. Smaller test $\text{E}_{\text{RMSE}}$ values indicate that the corresponding model has better prediction capacity. ‘Best $\text{E}_{\text{RMSE}}$’ shows the best $\text{E}_{\text{RMSE}}$ obtained by the existing iterations. The 5 parameters, such as network depth and initial learning rate (LR) etc., were selected in the range by the Bayesian optimization.

The proposed prediction method was used on the 4 sub-datasets of C-MAPSS, and the results obtained are shown in Table 7. It contains the test $\text{E}_{\text{RMSE}}$ result, the running time and the hyperparameters corresponding to the best $\text{E}_{\text{RMSE}}$. It can be seen from Table 4 that the time window length of the 4 datasets are different. The obtained sample size is also different. Therefore, the sliding step of the CNN filter for different sub-dataset is different. The sliding step of the FD001 and FD002 datasets is [2, 1], the sliding step of FD003 is mixed with [2, 1] and [1, 1], and the sliding step of FD004 is [1, 1]. Therefore, the running time of the proposed method on the 4 sub-datasets is also quite different as shown in Table 7.

Fig. 8 shows the prediction results of the prediction method in the FD001 dataset. Fig. 8(a) shows the prediction RUL curve and the actual RUL curve for 100 test engine units in FD001. The prediction capability of the network can be roughly seen from Fig. 8(a). In order to display the prediction results more intuitively, the 100 prediction engine units were sorted by the actual RUL as shown in Fig. 8(b). The red line is the sorted actual RUL, the black triangle label is the distribution of predicted RUL, and the colored area is the error band. The prediction error of the engines can be directly observed by the distribution of prediction RUL in the error band. The prediction error distribution histogram of the FD001 test sub-dataset is shown in Fig. 8(c). It can be seen from Fig. 8(c) that the number of engines with a prediction error between \([-10, 10]\) is 69%, and between \([-20, 20]\) is 89%. Similarly, Fig. 9 - Fig.11 show the test results of the FD002, FD003, and FD004 test datasets.

The two measures shown in Equations (40) and (41) were used for evaluating the prediction results. In order to show the prediction capability of the proposed method, the results...
TABLE 7. The best parameters selected by the Bayesian optimization for the 4 sub-datasets.

| Dataset | Running time /s | $E_{RMSE}$ | Network Depth | Initial LR | Momentum | Batchsize | L2 Regularization |
|---------|-----------------|------------|---------------|------------|----------|-----------|------------------|
| FD001   | 121.52          | 11.94      | 2             | 0.00890    | 0.3450   | 642       | 0.001895096      |
| FD002   | 299.91          | 19.29      | 2             | 0.001397   | 0.9380   | 783       | 9.45E-05         |
| FD003   | 320.42          | 12.31      | 3             | 0.000517   | 0.5239   | 799       | 0.003426757      |
| FD004   | 1470.90         | 22.14      | 1             | 0.001778   | 0.8297   | 800       | 0.003670858      |

FIGURE 8. Test results of FD001 sub-dataset: (a) comparison of actual RUL curve and prediction RUL curve, (b) prediction results error band, and (c) distribution histogram of prediction error.

FIGURE 9. Test results of FD002 sub-dataset: (a) comparison of actual RUL curve and prediction RUL curve, (b) prediction results error band, and (c) distribution histogram of prediction error.

FIGURE 10. Test results of FD003 sub-dataset: (a) comparison of actual RUL curve and prediction RUL curve, (b) prediction results error band, and (c) distribution histogram of prediction error.

of the proposed method along with the prediction methods using C-MAPSS for validation in the past 4 years reported in the literature are presented in Table 8. Although the $E_{RMSE}$ and $E_{Source}$ evaluation measures are slightly different, both represent a good prediction result when their value is small. In order to highlight the best prediction method, the best results for each comparison are bolded in the table. As shown in the table, the predicted results of the proposed method are the best except for the $E_{Source}$ result of the FD004 sub-dataset.

2) INFLUENCE OF THE HYPERPARAMETERS ON PREDICTION RESULTS

The Bayesian optimization method searches for the best combination of parameters by continuously adding observation
FIGURE 11. Test results of FD004 sub-dataset: (a) comparison of actual RUL curve and prediction RUL curve, (b) prediction results error band, and (c) distribution histogram of prediction error.

TABLE 8. Comparison of the prediction results with other methods.

| Methods         | Years | FD001 ERMSE | FD001 ERMS | FD002 ERMSE | FD002 ERMS | FD003 ERMSE | FD003 ERMS | FD004 ERMSE | FD004 ERMS |
|-----------------|-------|-------------|------------|-------------|------------|-------------|------------|-------------|------------|
| CNN [39]        | 2016  | 18.45       | 1.29×10^3  | 30.29       | 1.36×10^2  | 19.82       | 1.60×10^3  | 29.16       | 7.89×10^3  |
| LSTM [40]       | 2017  | 16.14       | 3.38×10^3  | 24.49       | 4.45×10^3  | 16.18       | 8.52×10^3  | 28.17       | 5.55×10^3  |
| MODINE [41]     | 2017  | 15.04       | 3.34×10^2  | 25.05       | 5.39×10^3  | 12.51       | 4.22×10^3  | 28.66       | 6.56×10^3  |
| BLSTM [21]      | 2018  | 14.26       | 2.95×10^4  | 23.18       | 4.13×10^3  | 13.74       | 3.17×10^3  | 24.86       | 5.43×10^3  |
| BLSTM [38]      | 2018  | 13.65       | 2.95×10^2  | 23.18       | 4.13×10^3  | 13.74       | 3.17×10^3  | 24.86       | 5.43×10^3  |
| DCNN [33]       | 2018  | 12.61       | 2.74×10^3  | 22.36       | 1.04×10^3  | 12.64       | 2.84×10^3  | 23.31       | 1.25×10^3  |
| DAG [42]        | 2019  | 11.96       | 2.29×10^2  | 20.34       | 2.73×10^3  | 12.46       | 5.35×10^3  | 22.43       | 3.37×10^3  |
| The proposed method | 2019 | 11.94       | 2.20×10^2  | 19.29       | 2.25×10^3  | 12.31       | 2.60×10^3  | 22.14       | 3.63×10^3  |

points and repeating the Gaussian process. The selection of the next observation points is based on the fitting curve by the Gaussian process. As the number of added observation points increases, the curve fitted by the Gaussian process will approach the actual distribution of the function.

The Gaussian optimization was used to optimize the hyperparameters such as initial learning rate, batch size, momentum, and L2 regularization, respectively. When one hyperparameter is being optimized, the other hyperparameters are set as in Table 7. Fig. 12 shows the Gaussian process for 4 hyperparameters with 30 observation points for the FD001 sub-dataset. The horizontal axis corresponds to the search range of the hyperparameters, and the vertical axis represents the \( E_{RMSE} \) value of using the FD001 test dataset for validating the prediction model. The effects of the 4 hyperparameters value on the prediction capacity of the developed model are shown in Fig. 12. The red line in Fig. 12 is the predictive function mean obtained by Bayesian optimization. The 30 blue dots are the 30 observation points for the Gaussian process. The larger black dot is the position of the next observation point selected according to the acquisition function. And the red star point is the point with the lowest function mean. The prediction results by the CNN-based prediction model performed multiple times under the same hyperparameter setting are different. Therefore, it conforms to the Bayesian optimization under the noisy environment. Fig. 12 increases the noise error bars compared to Fig. 1, and the error at the observation point is not 0.

Fig. 12 shows the effect of a single hyperparameter on the prediction model. Next, the effect of the two hyperparameters on the predictive model will be discussed. There are 5 hyperparameters discussed including network depth, learning rate, momentum, batch size, and L2 regularization. Two of the five hyperparameters were randomly selected as a group, and there were 10 groups. Fig. 13 shows the effect of 10 different groups of parameters on the prediction model. The red surface shown in Fig.13 is model mean representing the effect of two hyperparameters on the predictive model. The blue dots, black dots, and red star points have the same meaning as in Fig. 12. The z-coordinate of Fig. 13 represents the \( E_{RMSE} \) value of the FD001 sub-dataset test results, and RMSE is used in the figure instead.

The network depth in the 5 hyperparameters is a discrete integer containing only 1, 2, and 3. The other hyperparameters are consecutive numbers in the range. Therefore, the surface of Fig. 13(a), Fig. 13(b), Fig. 13(c), and Fig. 13(d) containing the network depth has a fracture phenomenon. And the surface of other graphs is continuous. The contoured lines on the bottom of the graphs correspond to changes of the red surface, whereas the blue color represents a smaller value and the yellow color represents a larger value.

Next, the results obtained by the Bayesian optimization is compared with the grid search and the random search. The results of finding the best hyperparameters using the Bayesian optimization method have been provided in Table 6. Tables 9 and 10 are the results of using grid search and random search methods, respectively. The tables contain 5 hyperparameters: learning rate (LR), network depth (Depth), momentum (Mo), batch size (Bs), and L2 regularization (L2). The number of iterations is denoted by #. The test \( E_{RMSE} \) of the FD001 test dataset is also shown in the table.
The grid search method selected several values within the range of each hyperparameter and then selected a value for each hyperparameter to form a parameter combination. Finally, all the parameter combinations were used for training and testing and the parameter combination with the best prediction result was selected. In Table 9, the network depth has three values of 1, 2, and 3; the learning rate has two values 1E-3 and 5E-3; the momentum has two values 0.4 and 0.8; the batch size has two values 400 and 600; the L2 regularization has two values 1E-3 and 1E-6. Therefore, there are a total of $48 (3 \times 2 \times 2 \times 2 \times 2 = 48)$ parameter combinations. The 48 parameter combinations and their corresponding test results are shown in Table 9.

TABLE 9. Grid search for multiple iterations to search for the best parameters in FD001.

| #  | Depth | LR    | Mo  | Bs  | L2     |
|----|-------|-------|-----|-----|--------|
| 1  | 1     | 1E-3  | 0.4 | 400 | 1E-3   |
| 2  | 1     | 1E-3  | 0.4 | 400 | 5E-3   |
| 3  | 1     | 1E-3  | 0.4 | 600 | 1E-3   |
| 4  | 1     | 1E-3  | 0.4 | 600 | 5E-3   |
| 5  | 1     | 1E-3  | 0.8 | 400 | 1E-3   |
| 6  | 1     | 1E-3  | 0.8 | 400 | 5E-3   |
| 7  | 1     | 1E-3  | 0.8 | 600 | 1E-3   |
| 8  | 1     | 1E-3  | 0.8 | 600 | 5E-3   |
| 9  | 1     | 1E-3  | 0.4 | 400 | 1E-3   |
| 10 | 1     | 1E-3  | 0.4 | 400 | 5E-3   |
| 11 | 1     | 1E-3  | 0.4 | 600 | 1E-3   |
| 12 | 1     | 1E-3  | 0.4 | 600 | 5E-3   |
| 13 | 1     | 1E-3  | 0.8 | 400 | 1E-3   |
| 14 | 1     | 1E-3  | 0.8 | 400 | 5E-3   |
| 15 | 1     | 1E-3  | 0.8 | 600 | 1E-3   |
| 16 | 1     | 1E-3  | 0.8 | 600 | 5E-3   |
| 17 | 1     | 1E-3  | 0.4 | 400 | 1E-3   |
| 18 | 1     | 1E-3  | 0.4 | 400 | 5E-3   |
| 19 | 1     | 1E-3  | 0.4 | 600 | 1E-3   |
| 20 | 1     | 1E-3  | 0.4 | 600 | 5E-3   |
| 21 | 1     | 1E-3  | 0.8 | 400 | 1E-3   |
| 22 | 1     | 1E-3  | 0.8 | 400 | 5E-3   |
| 23 | 1     | 1E-3  | 0.8 | 600 | 1E-3   |
| 24 | 1     | 1E-3  | 0.8 | 600 | 5E-3   |

VOLUME 8, 2020

FIGURE 12. The influence of single hyperparameters on prediction model: (a) Initial learning rate, (b) Batchsize, (c) Momentum, and (d) L2 regularization.
$E_{\text{RMSE}}$ values are shown in Table 9. The minimum prediction $E_{\text{RMSE}}$ of 12.56 was obtained by #33 parameter combination. When the random search method was used, each hyperparameter value was randomly generated within the range for training. The random parameters training and results of 30 times are shown in Table 10. The network
TABLE 10. Random search for multiple iterations to search for the best parameters in FD001.

| # | E\_RMSE | Depth | LR | Mo | Bs | L2 |
|---|---|---|---|---|---|---|
| 1 | 12.92 | 1 | 6.74E-03 | 0.4133 | 677 | 2.00E-08 |
| 2 | 13.88 | 1 | 2.79E-03 | 0.9327 | 881 | 5.76E-10 |
| 3 | 12.65 | 1 | 1.85E-03 | 0.4739 | 215 | 8.23E-08 |
| 4 | 12.84 | 1 | 1.29E-03 | 0.8010 | 862 | 2.00E-07 |
| 5 | 13.81 | 1 | 8.22E-03 | 0.6778 | 870 | 2.08E-06 |
| 6 | 15.83 | 1 | 1.52E-03 | 0.6064 | 816 | 8.84E-10 |
| 7 | 12.61 | 1 | 5.51E-03 | 0.3626 | 905 | 1.90E-09 |
| 8 | 12.87 | 1 | 5.55E-04 | 0.5962 | 707 | 8.49E-05 |
| 9 | 12.51 | 1 | 6.03E-03 | 0.4526 | 483 | 8.85E-04 |
| 10 | 13.13 | 1 | 7.08E-04 | 0.4905 | 554 | 1.82E-03 |
| 11 | 13.37 | 2 | 6.10E-04 | 0.4585 | 619 | 8.86E-03 |
| 12 | 13.04 | 2 | 3.12E-03 | 0.8516 | 277 | 3.31E-03 |
| 13 | 14.39 | 2 | 4.50E-03 | 0.7835 | 335 | 3.82E-04 |
| 14 | 13.04 | 2 | 8.02E-03 | 0.5978 | 501 | 8.94E-03 |
| 15 | 12.53 | 2 | 2.28E-03 | 0.4621 | 588 | 2.83E-04 |

TABLE 11. The best parameters selected by the Bayesian optimization with the TW length 18.

| # | E\_RMSE | Network Depth | Initial LR | Momentum | Batchsize | L2 Regularization |
|---|---|---|---|---|---|---|
| FD001 | 108.62 | 15.35 | 2 | 0.007139 | 0.30421 | 858 | 1.0557×10<sup>-10</sup> |
| FD002 | 309.38 | 20.13 | 2 | 0.000650 | 0.91703 | 720 | 0.008462 |
| FD003 | 320.42 | 15.90 | 2 | 0.002096 | 0.32194 | 898 | 0.00055706 |
| FD004 | 322.34 | 23.93 | 2 | 0.001778 | 0.45121 | 950 | 3.6048×10<sup>-10</sup> |

TABLE 12. The Cross Prediction results of the 4 sub-datasets.

| Source domain | FD001 | FD002 | FD003 | FD004 |
|---|---|---|---|---|
| Without AdaBN | 15.35 | 7.64×10<sup>-1</sup> | 129.50 | 1.07×10<sup>-8</sup> | 20.27 | 2.16×10<sup>-1</sup> | 197.64 | 6.84×10<sup>-2</sup> |
| AdaBN | 17.31 | 6.78×10<sup>-1</sup> | 17.69 | 8.14×10<sup>-2</sup> | 18.20 | 9.46×10<sup>-2</sup> |
| Without AdaBN | 56.07 | 1.33×10<sup>-1</sup> | 20.13 | 2.51×10<sup>-1</sup> | 59.90 | 1.88×10<sup>-6</sup> | 25.34 | 7.80×10<sup>-2</sup> |
| AdaBN | 26.08 | 6.76×10<sup>-1</sup> | 27.14 | 1.08×10<sup>-4</sup> | 24.85 | 6.76×10<sup>-1</sup> |
| Without AdaBN | 20.19 | 1.26×10<sup>-1</sup> | 177.97 | 1.02×10<sup>-1</sup> | 15.90 | 1.40×10<sup>-1</sup> | 209.69 | 1.03×10<sup>-21</sup> |
| AdaBN | 17.87 | 1.05×10<sup>-1</sup> | 18.23 | 1.54×10<sup>-3</sup> | 18.08 | 1.49×10<sup>-1</sup> |
| Without AdaBN | 52.28 | 6.59×10<sup>-1</sup> | 34.46 | 1.71×10<sup>-4</sup> | 53.31 | 9.47×10<sup>-7</sup> | 23.93 | 3.87×10<sup>-3</sup> |
| AdaBN | 32.26 | 2.86×10<sup>-1</sup> | 29.95 | 9.53×10<sup>-3</sup> | 31.05 | 1.56×10<sup>-4</sup> |

depth is an integer and contains only 3 values. Therefore, the grid search was adopted when selecting the network depth. And 10 trials were performed in each network depth. The minimum prediction \( E_{RMSE} \) of 12.32 was obtained by #28 parameter combination in the random search. Comparing the three parameter selection methods in Table 6, Table 9, and Table 10, the Bayesian optimization method is the best, and the random search method is slightly better than the grid search method.

There are \( m \) hyperparameters, and each hyperparameter selects \( n \) values, then there are a total of \( n^m \) kinds of parameter combinations for the grid search. The disadvantage of the grid search method is that all the parameter combinations are tried, and each attempt takes a lot of time and effort. Therefore, in order to avoid excessive computational times, it is necessary to reduce the number of values of each hyperparameter. However, too few values for each hyperparameter will affect the optimization results. The experimental results show that the random search method is better than grid search. However, there is no correlation between any two trials in the random search, so the stability of the search is poor.

Compared to the above two parameter-finding method, Bayesian optimization can determine the next trial parameters based on the existing results. This allows for better predictions with fewer trials.

3) PREDICTION RESULTS IN DIFFERENT DATA DOMAINS

The application of data-driven prediction methods does not require a deep understanding of the mechanical equipment, relying solely on data to develop a prediction model. However, when the test data differs greatly from the data used for the training model, the accuracy of the prediction is low. Retraining the model is time consuming and laborious, so the ability of the model to adapt to different DDs is very important. In this paper, the BN unit is added to the model. When the test DD changes, adjusting the offset and scale factor in the BN unit can quickly adapt the model to the new DD. The 4 sub-datasets in CMAPSS were used to test the ability of the proposed prediction model to adapt to different DDs. In order to make the input data size of the 4 sub-datasets consistent, the window length during data processing was all set to 18. Table 11 shows the results of hyperparameter
selection for Bayesian optimization of 4 sub-datasets with the WT length of 18. This section is intended to discuss the ability of the prediction model to predict different DDs. The sliding steps of the 4 sub-datasets convolution filters were all set to [2, 1]. Hence, the running time of the 4 datasets was similar. Since the input data time series length was shortened to 18, the diagnostic $E_{\text{RMSE}}$ in Table 11 is slightly larger than that in Table 7. And the 5 hyperparameters were selected in the same way as above.

Table 12 shows the results of cross prediction of 4 sub-datasets. The source domain was the data used to train the prediction model, and the target domain was the data domain used in the test. The prediction results include both $E_{\text{RMSE}}$ and $E_{\text{source}}$. It also compares the results of the fine-tuning
the network with or without the AdaBN method. It can be seen that the Table 12 that the prediction results can be improved after the fine-tuning by the proposed AdaBN method. In particular, the $E_{\text{RMSE}}$ fine-tuned by the AdaBN method in the 4 cases of FD002→FD001, FD002→FD003, FD004→FD001, and FD004→FD003 is reduced by nearly 10 times. In addition, the $E_{\text{RMSE}}$ fine-tuned by the AdaBN method in the 4 cases of FD001→FD002, FD001→FD004, FD003→FD002, and FD003→FD004 is reduced by nearly 2 times. In other cases, the improvement effect of AdaBN is also obvious.

Fig. 14 compare the results of the cross DD prediction with or without the AdaBN fine-tuned network. The three graphs in Fig. 14(a-c) are based on FD001 as the source domain training model, and the other 3 sub-datasets are used as target domains for testing. The prediction results in the three cases in Fig. 14(a) are: (1) prediction of the RUL of the FD002 using the model trained by FD001 without AdaBN; (2) prediction of the RUL of the FD002 using the model trained by FD001 with AdaBN; (3) prediction of the RUL of the FD002 using the model trained by FD002. It can be seen intuitively from Fig. 14 that the prediction results after AdaBN fine-tuning have been significantly improved. And it is close to the prediction results that are trained and tested with the same DD. Fig. 14(a-c) correspond to the FD001 column in Table 11. The following Fig. 14(d-f), Fig. 14(g-i), and Fig. 14(j-l) are cross-predictions with FD002, FD003, and FD004 as the source domains, respectively. Table 4 shows that FD001 and FD003 contain fewer fault modes and operating conditions, and more fault modes and operating conditions are included in FD002 and FD004. Combined with Table 12 and Fig. 14, it can be concluded that AdaBN performs better when predicting the DD with fewer fault modes and operating conditions using models trained in a DD with more fault modes and operating conditions.

V. CONCLUSION

This paper presented a novel RUL prediction model developed using a DCNN combined with Bayesian optimization and AdaBN. The model has a self-optimized structure and hyperparameters of DCNN. The Bayesian optimization was integrated into DCNN to achieve automatic structure and hyperparameter selection. In addition, in order to improve the ability of the prediction model to adapt to multiple DDs, BN units and AdaBN fine-tuning strategy were also integrated. The proposed prediction method was validated by the CMAPSS simulated turbofan engine dataset. The prediction performance of the proposed method was compared with other prediction methods validated by the CMAPSS dataset in the past 4 years. The comparison results show that the performance of the proposed prediction method is better than other methods. The following conclusions can also be obtained:

(1) The 4 sub-datasets in CMAPSS were all used to test the proposed self-optimization prediction method. The prediction results of the FD001 and FD003 sub-datasets with fewer operating conditions and fault modes are better. Both $E_{\text{RMSE}}$ and $E_{\text{source}}$ show that the overall prediction results of the proposed method are better than other prediction methods in the past 4 years.

(2) The Bayesian optimization method finds the best combination of hyperparameters through multiple iterations. Comparing the grid search and the random search, the Bayesian method can obtain better prediction results with fewer iterations. The Bayesian optimization method can intuitively reflect the relationship between the hyperparameters and the prediction results through the fitted curve or surface.

(3) Adding BN units to the model and fine-tuning the network through the AdaBN method when predicting different DDs can make the prediction model work better. The proposed AdaBN fine-tuning strategy can avoid the loss of time and effort in retraining the model. And the results obtained are close to the prediction results of training and testing using the same DD. In addition, it can be concluded that AdaBN performs better when predicting simple DD using models trained in complex DD.

REFERENCES

[1] K. L. Tsui, N. Chen, Q. Zhou, Y. Hai, and W. Wang, “Prognostics and health management: A review on data driven approaches,” Math. Problems Eng., vol. 2015, pp. 1–17, May 2015, doi: 10.1155/2015/793161.

[2] Y. Cao, W. Wei, C. Wang, S. Mei, S. Huang, and X. Zhang, “Probabilistic estimation of wind power ramp events: A data-driven optimization approach,” IEEE Access, vol. 7, pp. 23261–23269, 2019.

[3] L. Liu, Q. Guo, D. Liu, and Y. Peng, “Data-driven remaining useful life prediction considering sensor anomaly detection and data recovery,” IEEE Access, vol. 7, pp. 58336–58345, 2019.

[4] D. N. T. How, M. A. Hannam, M. S. H. Lipu, and P. J. Ker, “State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review,” IEEE Access, vol. 7, pp. 136116–136136, 2019.

[5] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep learning and its applications to machine health monitoring,” Mech. Syst. Signal Process., vol. 115, pp. 213–237, Jan. 2019, doi: 10.1016/j.ymssp.2018.05.050.

[6] L. Ren, Y. Sun, H. Wang, and L. Zhang, “Prediction of bearing remaining useful life with deep convolution neural network,” IEEE Access, vol. 6, pp. 13041–13049, 2018.

[7] W. Yu, I. Y. Kim, and C. Mechefske, “Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme,” Mech. Syst. Signal Process., vol. 129, pp. 764–780, Aug. 2019, doi: 10.1016/j.ymssp.2019.05.005.

[8] S. Zhao, Y. Zhang, S. Wang, B. Zhou, and C. Cheng, “A recurrent neural network approach for remaining useful life prediction utilizing a novel trend features construction method,” Measurement, vol. 146, pp. 279–288, Nov. 2019, doi: 10.1016/j.measurement.2019.06.004.

[9] A. Elsheikh, S. Yacout, and M.-S. Ouali, “Bidirectional handshaking LSTM for remaining useful life prediction,” Neurocomputing, vol. 325, pp. 148–156, Jan. 2019, doi: 10.1016/j.neucom.2018.09.076.

[10] Y. Wu, M. Yuan, T. Dou, L. Lin, and Y. Liu, “Remaining useful life estimation of engineered systems using vanilla LSTM neural networks,” Neurocomputing, vol. 275, pp. 167–179, Jan. 2018, doi: 10.1016/j.neucom.2017.05.063.

[11] S. Rigamonti, P. Baraldi, E. Zio, I. Roychoudhury, K. Goebel, and S. Poll, “Ensemble of optimized echo state networks for remaining useful life prediction,” Neurocomputing, vol. 281, pp. 121–138, Mar. 2018, doi: 10.1016/j.neucom.2017.11.062.

[12] L. Yang, F. Wang, J. Zhang, and W. Ren, “Remaining useful life prediction of ultrasonic motor based on elman neural network with improved particle swarm optimization,” Measurement, vol. 143, pp. 27–38, Sep. 2019, doi: 10.1016/j.measurement.2019.05.013.

[13] X. Li, W. Zhang, and Q. Ding, “Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction,” Rel. Eng. Syst. Saf., vol. 182, pp. 208–218, Feb. 2019, doi: 10.1016/j.ress.2018.11.011.
