applied sciences

MDPI

Article

A Novel Deep Learning Approach for Machinery Prognostics Based on Time Windows

Hanbo Yang [1], Fei Zhao [1,2], Gedong Jiang [1,2,*], Zheng Sun [1] and Xuesong Mei [1,2]

[1] State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China; hanboyang@stu.xjtu.edu.cn (H.Y.); ztzhao@xjtu.edu.cn (F.Z.); zheng.sun@xjtu.edu.cn (Z.S.); xsmei@mail.xjtu.edu.cn (X.M.)

[2] Shaanxi Key Laboratory of Intelligent Robots, Xi'an Jiaotong University, Xi'an 710049, China

* Correspondence: gdjiang@mail.xjtu.edu.cn; Tel.: +86-1860-295-1980

Received: 10 October 2019; Accepted: 7 November 2019; Published: 11 November 2019

Abstract: Remaining useful life (RUL) prediction is a challenging research task in prognostics and receives extensive attention from academia to industry. This paper proposes a novel deep convolutional neural network (CNN) for RUL prediction. Unlike health indicator-based methods which require the long-term tracking of sensor data from the initial stage, the proposed network aims to utilize data from consecutive time samples at any time interval for RUL prediction. Additionally, a new kernel module for prognostics is designed where the kernels are selected automatically, which can further enhance the feature extraction ability of the network. The effectiveness of the proposed network is validated using the C-MAPSS dataset for aircraft engines provided by NASA. Compared with the state-of-the-art results on the same dataset, the prediction results demonstrate the superiority of the proposed network.

Keywords: remaining useful life; convolutional neural network; kernel

1. Introduction

As a major task of condition-based maintenance (CBM), health prognostics can predictively maintain equipment, improve the reliability of the machinery and prevent catastrophic consequences [1–5]. The main purpose of health prognostics is to explore the on-going degradation law for the device by feature extraction of the device-related data, and then predict the remaining useful life (RUL) [6]. Consequently, RUL prediction has received more and more attention from academia and industry.

RUL prediction methods can be classified into two categories: model-driven methods and data-driven methods. The model-driven methods aim to derive the degradation process of a machine by setting up mathematical or physical models [7]. With the commonly used models such as the Markov process model, the Winner process model and the Gaussian mixture model, the RUL can be predicted [8]. However, such methods rely on robust physical or mathematical models, which are difficult to obtain owing to the complex environment [9]. Data-driven methods establish the relationship between the acquired data and RUL through data mining, which can be more accurate with the real-time monitoring data. In recent years, a number of effective data-driven algorithms have been proposed and achieved good results, mainly including: neural networks [10], support vector regression (SVR) [11] and gaussian process regression (GPR) [12].

Generally, data-driven methods consist of three stages: data acquisition, health indicator (HI) construction and RUL prediction [13–16]. By learning the deterioration curve of HIs, the time it takes for the current HI to reach the pre-set failure threshold (i.e., RUL) is estimated. Louen et al. [17] proposed a new health indicator creation approach using binary support vector machine classifier

and calculated the RUL through an identified Weibull function. Guo et al. [6,18] applied neural networks to the construction of health indicators and the RUL was obtained by the particle filter. Malhotra et al. [19] constructed a long short-term memory-based encoder-decoder (LSTM-ED) scheme to obtain an unsupervised health indicator (HI).

Generally, the prediction accuracy of RUL is seriously affected by the quality of the constructed HIs [20,21]. The current HI construction methods are perplexed by different amplitude ranges of statistical features, which leads to unifying the failure threshold difficultly under different working conditions [9,18]. In addition, as shown in Figure 1a, prediction methods based on HIs depend on long-term tracking of equipment from the initial to the current time, which is impractical in some cases.

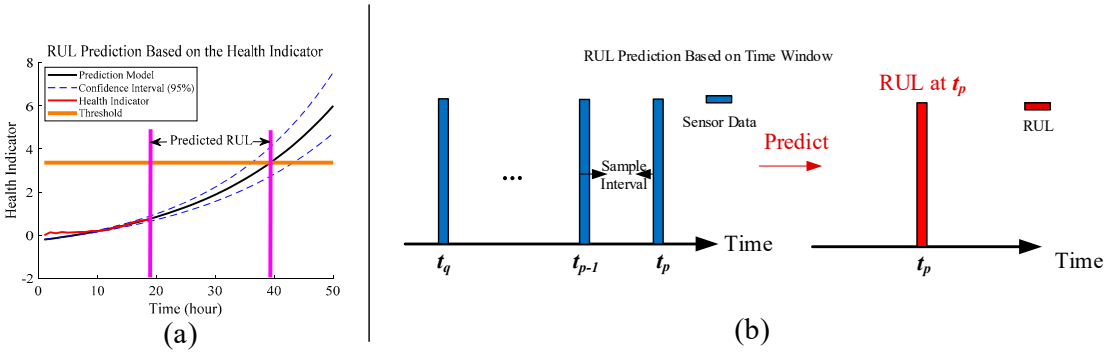

Figure 1. Remaining useful life (RUL) prediction: (**a**) Based on health indicator (HI); (**b**) based on time window.

Hence, some scholars have paid attention to the time window (TW)-based RUL prediction method and have begun to directly establish the correlation between the sensor signals and RUL. Lim et al. [22] developed a machine learning framework with a moving time window to determine the RUL of aircraft engines. Zhang et al. [23] used a fixed time window to predict the RUL of aircraft engines. As depicted in Figure 1b, this method can start sampling at any time when the equipment is running, obtain the data from t_q to t_p, and the RUL at t_p can be predicted directly according to these data. The key issue of these methods is to establish the mapping between the original signal and the RUL.

Recently, deep learning has made remarkable achievements in the fields of image recognition, speech recognition and unmanned driving [24,25]. Deep learning is characterized by the deep network architecture where multiple layers are stacked in the network to fully capture the representative information from the raw input data [26]. Therefore, as a powerful tool, deep learning shows its great potential in establishing the correlation between the original signals and the RUL. Lim et al. [22] created a multilayer neural network as the deep learning algorithm to build the correlation between signal characteristics and RUL. Zhang et al. [23] proposed a multi-objective deep belief networks ensemble (MODBNE) method for RUL prediction, which employs a multi-objective evolutionary algorithm integrated with the traditional deep belief networks (DBN) training technique to evolve multiple DBNs simultaneously subject to accuracy and diversity as two conflicting objectives. Within the deep learning architecture, convolutional neural network (CNN) is further applied in this study because of its excellent feature extraction ability in the face of variable and complex signals [27]. Babu et al. [28] firstly applied CNNs to the field of RUL prediction and incorporated automated feature learning from the raw sensor signals in a systematic way. Li et al. [27] established a deep CNN model and conducted a preliminary exploration of the depth of the networks on RUL prediction. While CNNs have shown great potential in prognostics, there are still few applications of CNN in RUL prediction.

In this paper, a novel CNN is presented. The kernel module composed of multiple convolutional kernels was proposed for feature extraction. This module was derived from the inception network proposed by Szegedy et al. [25]. Compared with the module used in the inception network, the feature extraction of the convolution kernel was in the time dimension, which reduced the parameters of the

module and transfers the application in the image field to prognostics. The relationship between the raw signals and the RUL could be established through the learning process. The RUL was calculated according to the new vector of input signals.

In this study, we used the C-MAPSS dataset provided by NASA for the RUL prediction of aircraft engines [29]. The engine is the key component of the aircraft and there is always a pressing need to develop new approaches to better evaluate the engine performance degradation and estimate the remaining useful life [30]. The proposed network showed its superiority compared with the state-of-art methods on the same dataset. Our attempt was a useful exploration for implementing the RUL prediction on the unexpected events and provides the possibility to transfer the proposed method in a simpler scenario.

This remainder of the paper is organized as follows. The proposed CNN is given in Section 2, which illustrates the proposed architecture and the main scheme of the kernel module used in the convolution layers. The experimental results of the proposed model compared with different depths, time windows, optimization algorithms and other state-of-the-art methods are given in Section 3. Section 4 summarizes the conclusions of this work.

2. The Proposed Deep CNN Architecture

As presented in Figure 2, the deep learning architecture proposed consisted of the convolution layers, pooling layers and the fully-connected layer. Firstly, the normalized sensor data with the time window was directly used as input to the CNN, and then the feature maps were increased. Secondly, the convolution layer which was characterized by the kernel module was utilized for feature extraction and the down sampling scheme was operated by the pooling layer. Thirdly, the dimension reduction was performed before the fully connected layer, and then the feature mapping was implemented at the fully connected layer.

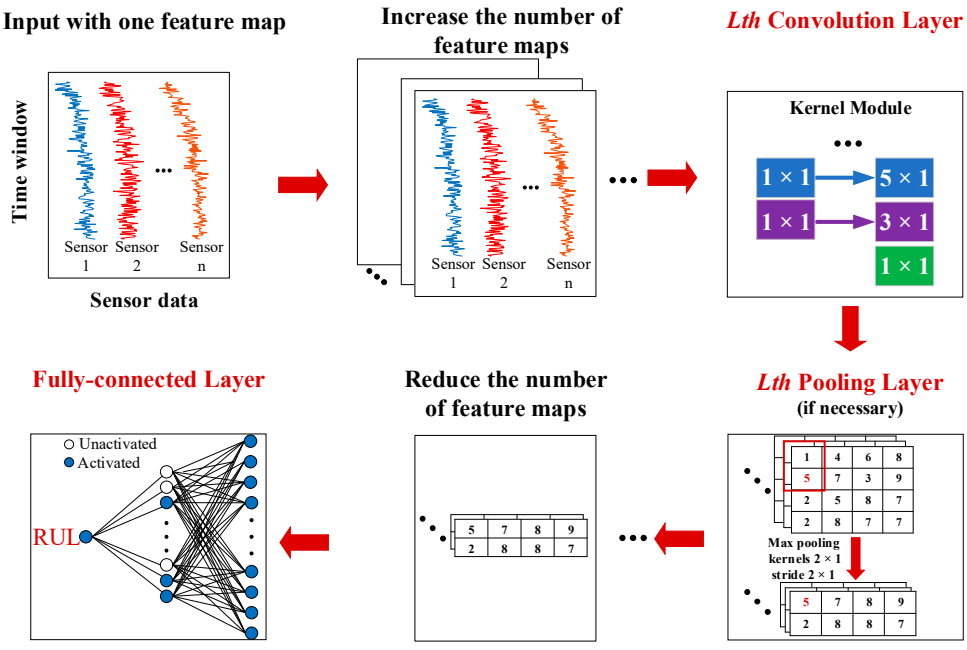

Figure 2. The flowchart of the proposed architecture.

2.1. The Kernel Module

The CNN previously used in prognostics contains only convolution kernels of the same size. However, the field of vision extracted by a small-sized kernel could be smaller, and a larger-sized kernel may ignore the details. Hence, different sizes of kernels can be used in each convolution layer to let the CNN automatically learn to select the size of the kernel. The selected kernels are

one-dimensional because feature extraction is mainly performed in the time dimension. The size of the kernel in the module should be adjusted according to the size of the input data, which means the size of the kernel cannot be larger than the size of the input data for the module in the time dimension. Each convolution layer is characterized by the kernel module which is composed of kernels with different sizes. The padding method was "SAME". The kernel module is illustrated in Figure 3.

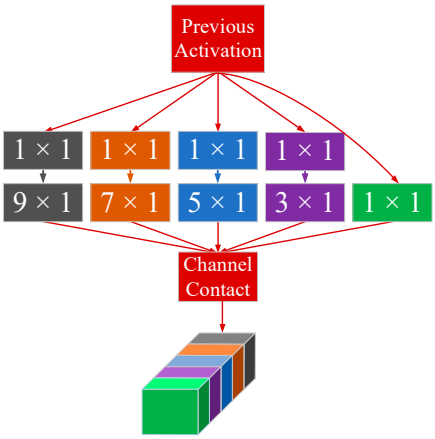

Figure 3. The kernel module.

The Lth layer is assumed to be the convolution layer which can be denoted as:

$$x_j^L = \mathrm{f}\left(\sum_{i \in M_{L-1}} x_i^{L-1} \times f_j^L + b_j^L\right), \quad (1)$$

where x_j^L is the jth feature map of the output on the Lth layer, x_i^{L-1} is the ith feature map of the output on the L-1th layer, M_{L-1} is all the feature maps of the output on the L-1th layer, f_j^L is the jth kernel on the Lth layer, b_j^L is the jth bias on the Lth layer.

The 1×1 convolution kernel can be used in front of the other kernels as the bottleneck to directly reduce the computational burden of the convolution layer, which is explained in the next subsection.

2.2. The 1×1 Kernels

The 1×1 convolution kernel was proposed firstly by Szegedy et al. [25], which has two functions. Firstly, by adding the 1×1 kernels in front of other kernels in the kernel module, the computational cost can be reduced. The total number of multipliers is the number of multipliers that need to be calculated for each of output values times the number of output values. As illustrated in Figure 4, two techniques are given for inputting $n_1 \times n_2 \times f_1$ and outputting $n_1 \times n_2 \times m_1$, the first method is to use only $s \times 1$ convolution kernels, and an alternative architecture is to use the 1×1 convolution kernels and then the $s \times 1$ convolution kernels.

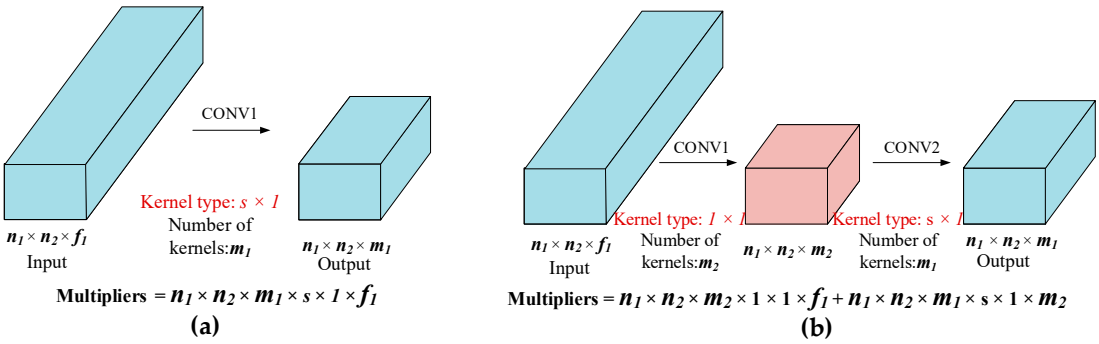

Figure 4. Convolution operation: (**a**) Based on HI; (**b**) using 1×1 convolution.

The total number of multipliers for the first technique tm_1 and the second technique tm_2 can be calculated by

$$\begin{aligned} tm_1 &= n_1 \times n_2 \times m_1 \times s \times f_1, \\ tm_2 &= n_1 \times n_2 \times m_2 \times f_1 + n_1 \times n_2 \times m_1 \times s \times m_2, \end{aligned} \tag{2}$$

where m_2 represents the number of feature maps after the 1×1 kernels.

If $tm_1 > tm_2$, then

$$(m_1/m_2 - m_1/f_1) > 1/s, \tag{3}$$

hence, the computational load can be reduced.

The second function of the 1×1 kernel is to change the number of feature maps and implement linear combination of multiple feature maps. The input data is processed by the 1×1 kernel, and then the number of the feature maps is increased, providing the convolution layer with the data for feature extraction. In addition, the 1×1 convolution kernel is used for dimension reduction before the fully-connected layer, which can further enhance the correlation between the extracted features and RUL.

2.3. The Proposed Network

The proposed deep convolution neural network is shown in Figure 5. The depth of the network is determined by the convolution-pooling layer or convolution layer. The pooling layer is used to reduce the dimensions and does not need to perform complex matrix operations or learn any parameters. The size of the kernel in the pooling layer is 2×1, and the stride is 2×1. The pooling method is 'VALID' max pooling and the size of input data will be halved in the time dimension through the pooling layer. Finally, the fully-connected layer maps the features learned by the CNN to the output through converting three-dimensional data into one-dimensional.

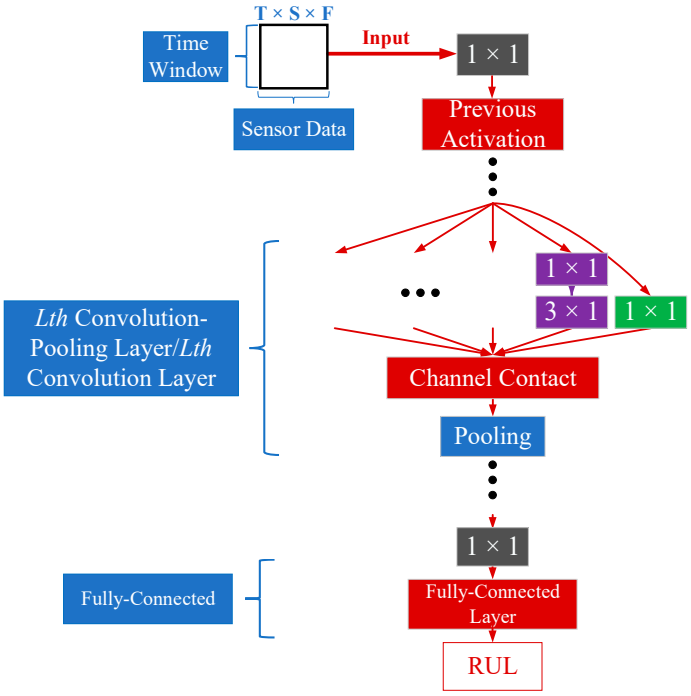

Figure 5. The proposed network.

The input data is three-dimensional and is expressed by $T \times S \times F$, where T is the length of the TW, S is the sensor data and F is the number of feature map (the input data contains only one feature map). The normalized data for T consecutive time cycles is directly used as input to the CNN. As the depth of the model increases, the number of convolution kernels after the 1×1 kernels in each layer is

doubled until the input of the *L*th layer is less than six in the time dimension, and the pooling layer will no longer be included in the layer and subsequent layers. If the pooling layer for the input data less than six is used again, only the 1×1 convolution kernel can be used in the next layer, which is only an increase in the number of feature maps rather than feature extraction. The number and size of the convolution kernels in the subsequent layers will be the same as the *L*th layer. Before other kernels, the number of 1×1 kernels is half that of other kernels to satisfy Equation (3), while the first layer is different. This is because the number of the initial 1×1 kernels before the first convolution-pooling layer is half of other kernels in the first layer, so the number of the 1×1 kernels before other kernels is set to one quarter of other kernels in the first layer as the bottleneck to reduce the parameters. Note that, the maximum size of the kernel in each layer should be less than or equal to the time dimension of the input data in that layer.

Here is a concrete example, when the TW size was set to 30, the number of different kernels is presented in Table 1. Due to the dimensionality reduction of the pooling layer, the input size of the third layer was $7 \times S \times 160$, so the kernel module in this layer removed the 9×1 kernels. Similarly, only 1×1 and 3×1 convolution kernels were used in the fourth and subsequent last layers, and no pooling layers were added.

Table 1. The number of different kernels in each layer (time window 30).

Layers	Input	Output	The Number of the Different Kernels					
			- 1×1	1×1 3×1	1×1 5×1	1×1 7×1	1×1 9×1	Sum
First convolution–pooling layer	$30 \times S \times 8$	$15 \times S \times 80$	- 16	4 16	4 16	4 16	4 16	80
Second convolution–pooling layer	$15 \times S \times 80$	$7 \times S \times 160$	- 32	16 32	16 32	16 32	16 32	160
Third convolution–pooling layer	$7 \times S \times 160$	$3 \times S \times 256$	- 64	32 64	32 64	32 64	- -	256
Fourth convolution layer	$3 \times S \times 256$	$3 \times S \times 256$	- 128	64 128	- -	- -	- -	256
Subsequent layers	$3 \times S \times 256$	$3 \times S \times 256$	- 128	64 128	- -	- -	- -	256

3. Experiment and Analysis

3.1. Experimental System

NASA C-MAPSS dataset generated by the aircraft engine simulation models is often used for comparison of different approaches [29]. The C-MAPSS dataset contains four sub-datasets that correspond to different simulation conditions and engine failure modes. Each sub-dataset has one training set and one testing set. The training set contains temporal data of 21 outputs of the aircraft engine simulation model from the unknown state of the model to the stage of failure. These 21 outputs include diverse types of data such as temperature data, speed data and pressure data at different positions of the engines, etc. The information of the C-MAPSS dataset is given in Table 2. The failure modes of FD001 and FD002 are only high-pressure compressor (HPC) degradation, while the failure modes of FD003 and FD004 include high-pressure compressor degradation and fan degradation.

Table 2. Information of NASA C-MAPSS dataset.

Sub-Dataset	FD001	FD002	FD003	FD004
Engine models for training	100	260	100	249
Engine models for testing	100	259	100	248
Simulation conditions	1	6	1	6
Fault modes	1	1	2	2

As shown in Table 3, only 14 sets among the 21 outputs were selected for the study, the others were constant during the life of the engines from health to failure and did not provide any valuable information for RUL prediction [23]. Therefore, as shown in Table 3, 14 sets of time-varying variables with indices 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20 and 21 were selected in the study. The state of the aircraft engine was healthy during the early operation period. In the initial stage, it was assumed that the RUL of the aircraft engine was a constant value. Based on the degradation mechanism of aircraft engines in this dataset [23], the constant value of RUL in the early stage was set to 125 in this paper.

Table 3. The selected time-varying variables.

The Selected Time-Varying Variables	Units
Total temperature at LPC outlet	°R
Total temperature at HPC outlet	°R
Total temperature at LPT outlet	°R
Total pressure at HPC outlet	psia
Physical fan speed	rpm
Physical core speed	rpm
Static pressure at HPC outlet	psia
Ratio of fuel flow to Ps30	pps/psi
Corrected fan speed	rpm
Corrected core speed	rpm
Bypass ratio	–
Bleed enthalpy	–
HPT coolant bleed	l bm/s
LPT coolant bleed	l bm/s

3.2. Experimental Setup

3.2.1. Data Pre-Processing

In the CNN, the characteristics of relatively smaller values will be weakened, and the characteristics of relatively larger values will be amplified. To eliminate the problems caused by different numerical ranges, the data need to be standardized. The zero-mean normalization method was used in this paper, which is denoted as:

$$TRAIN_j^i = \frac{train_j^i - \mu_{train}^i}{\sigma_{train}^i}, \tag{4}$$

where $TRAIN_{ij}$ represents the normalized value of the jth data point from the ith sensor in a training sample, $train_{ij}$ represents the true value of the jth data point from the ith sensor, μ_{train}^i represents the mean of the ith sensor and σ_{train}^i represents the standard deviation.

At the same time, the standardization of data in the testing set should be based on the mean and standard deviation of the training set, so that the impact of future data on the current model can be avoided.

3.2.2. Time Window Processing

Many multivariate temporal data-based prediction models take as an input a multivariate data point sampled at a single time stamp, which neglects some useful temporal information that may

much improve prediction performance [23]. Therefore, the TW is introduced to solve this problem. Specifically, at any particular timestamp, TW covers the data of the current timestamp and its several preceding timestamps, which is then fed as an input into the prediction model [23]. In this study, the effect of the TW size on the prediction results was considered.

3.2.3. Training Process

The models were trained ten times to reduce the fluctuations in the training process. The constant value of RUL in the early stage was 125, the number of 1×1 kernels after the input layer was 8, the number of 1×1 kernels before the fully-connected layer and the number of neurons in the fully-connected layer was 64 and 800, respectively. The dropout rate was set to 0.5 and the activation function was ReLU. The training parameters are presented in Table 4.

Table 4. Training parameters.

Sub-Dataset	FD001	FD002	FD003	FD004
Length of the TW	10/20/30	20	30	15
Training samples	19731/18731/17731	48,819	21,820	57,763
Batch size	760/721/682	2220	809	1992
Epoch number	40	40	30	50

The relationship between batch size, iteration and epoch can be expressed as

$$\begin{aligned} \text{one epoch} &= \text{number of iterations} \\ &= \lceil \text{total samples/batch size} \rceil \end{aligned} \tag{5}$$

The loss function is defined as follows

$$Loss = \sqrt{\frac{1}{N_b}\sum_{i=1}^{N_b} h_i^2}, \tag{6}$$

where N_b is the number of the batch size and $h_i = RUL_{predicted} - RUL_{true}$.

The loss function was minimized during the training process using the Adam optimization algorithm and the parameters of the model were saved. Experiments were run on a personal computer with Intel Core i7-8700 CPU, 16GB memory and GTX 1080 Ti GPU. The programming language was Python 3.6 with scientific computing library numpy and deep learning library tensorflow-gpu.

3.2.4. Evaluation Functions

There are mainly two kinds of evaluation functions for this dataset. One evaluation function gives different weights to the late predictions and early predictions, and the other is that the weights for both conditions are the same.

The first type of evaluation function [29] is shown in Equation (7), where N represents the number of aircraft engines in the testing set.

$$S = \begin{cases} \sum_{i=1}^{N} (e^{-\frac{h_i}{13}} - 1) & h_i < 0 \\ \sum_{i=1}^{N} (e^{\frac{h_i}{10}} - 1) & h_i \geq 0 \end{cases}. \tag{7}$$

The scoring function is asymmetric, which penalizes late prediction (i.e., the estimated RUL value is larger than the actual RUL value) more heavily than early prediction (i.e., the estimated RUL value

is smaller than the actual RUL value) since late prediction may result in more severe consequences [23]. The second type of evaluation function can be expressed as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} h_i^2}. \quad (8)$$

Root mean square error (RMSE) has the same weight for early predictions and lag predictions.

3.3. Performance Analysis

In this section, the performance of the proposed deep learning architecture for prognostics is presented. In addition, extensive experiments were performed to investigate the impact of key factors on the results, including the depth of the model and the size of the time window, as well as different optimization algorithms. Compared with the state-of-the-art results on the C-MAPSS dataset, the prediction results demonstrated the superiority of the proposed network.

3.3.1. Performance of the Results

The trained model was used to predict the RUL of different aeroengines in the testing set. The testing set FD001, FD002, FD003 and FD004 contain 100, 259, 100 and 248 sets of aircraft engine simulation data from the unknown state of t_1 to the unknown state of t_2 respectively, and each set of simulation data is truncated to obtain T consecutive cycles data of the aeroengine. The testing set also gives the actual RUL of aeroengines at t_2 to verify the effectiveness of the algorithm. Then, the testing data of T consecutive time cycles is used to predict the RUL of the aeroengine in the unknown state of t_2.

Overall, the models achieved accurate results for the prediction of the RUL of aircraft engines in the testing set. Based on the two evaluation functions, Table 5 presents the performance of the trained models on the testing set. The model was trained ten times. Using RMSE as the evaluation function, the model performed best in the FD001 testing set, with a mean score of 12.18, while the model performed the worst in FD004 with a mean score of 22.12. It can be seen from Figure 6 that the prediction results of the FD001 and FD003 were better than those of the FD002 and FD004 due to the multiple simulation conditions of FD002 and FD004. At the same time, the prediction result of the algorithm for FD001 was better than that of FD003, because the failure mode of FD003 included high-pressure compressor degradation and fan degradation, while the failure mode of FD001 was only high-pressure compressor degradation.

Table 5. The performance of the model (std: standard deviation).

Sub-Dataset	FD001	FD002	FD003	FD004
RMSE (mean)	12.18	19.58	15.67	22.12
RMSE (std)	0.31	0.27	0.46	0.47
RMSE (best)	11.77	19.24	14.97	21.32
RMSE (worst)	12.67	20.29	16.16	22.74
S (mean)	224.16	2494.35	1279.85	4523.32
S (std)	23.08	264.54	245.31	809.26
S (best)	200.55	2118.57	865.06	3268.99
S (worst)	261.69	3012.73	1620.35	6163.57

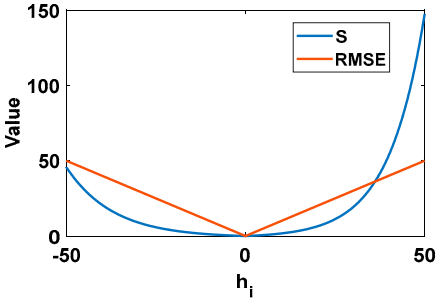

Figure 6. Comparison between the score of the first evaluation function (S) and root mean square error (RMSE) when the number of aircraft engines (N) is set to 1.

Figure 7 shows the prediction results of the model. For FD001, the model used had seven layers and the size of the time window was selected as 30. For FD002, the model used had three layers and the size of the time window was selected as 20. For FD003, the model used had three layers and the size of the time window was chosen to be 30. For FD004, the model used had two layers and the size of the time window was selected as 15.

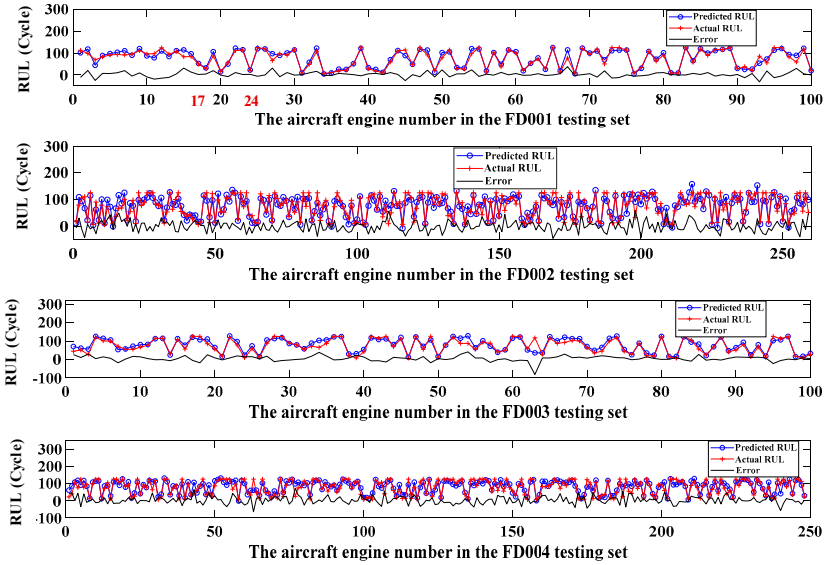

Figure 7. Prediction results on the testing dataset.

Since the testing set gives the actual RUL of the aeroengines in the unknown state of t_2, the RUL of aeroengines from t_1 to t_2 can be obtained from this value, e.g., the RUL at time t_1 is the actual RUL at time t_2 plus $t_2 - t_1$. The function to calculate the RUL from t_2 to t_1 can be expressed as

$$actualRUL\ at\ t = (actualRUL\ at\ t2) + (\ t2 - t), \tag{9}$$

where $t_1 + N \leq t < t_2$.

When the time window is selected as N, the RUL of aircraft engines from $t_1 + N$ to t_2 can be predicted. Figure 8 shows the predicted results from t_1 + 30 to t_2 of two randomly selected aeroengines in the FD001 testing set. The RUL prediction values for the No. 17 and No. 24 aeroengines at time t_2 corresponding to the predicted results in Figure 7. In Figure 8, the yellow dashed box represents that the RUL of the aeroengine was a constant value of 125 at the initial stage of aeroengine operation. The predicted result of the No. 17 aeroengine was slightly lower than the actual value at the beginning, but the error was small. With the increasement of the time cycle, the result was more accurate. The predicted result of RUL of the No. 24 aeroengine had slight fluctuations at the initial stage. As the time

cycle increased, the predicted and actual values became very close and eventually the more accurate result was obtained.

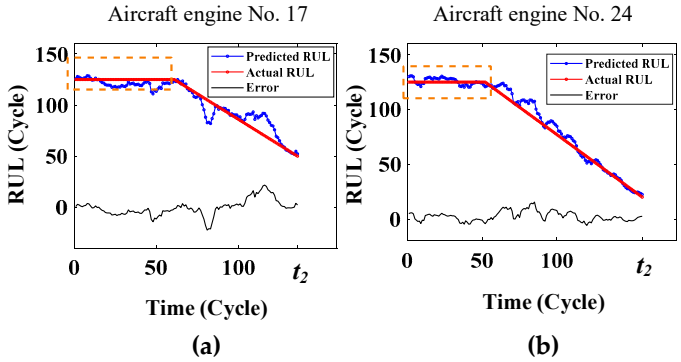

Figure 8. Part of prediction results on FD001 testing set: (**a**) Aircraft engine No. 17; (**b**) aircraft engine No. 24.

3.3.2. Effects of the Depth and Time Window

As for the models applied to the FD001 dataset, the depth of the network would be gradually increased from one to eight while other parameters remained unchanged to explore the influence of the depth of the CNN on the results. At the same time, the effect of the size of the TW on the results is also discussed.

With the evaluation function RMSE as a criterion, Figure 9a depicts the prediction results of models with different time windows and depths. When the size of the TW was selected as 10, the value of RMSE grew gradually to 17.64 at the five layers, after which it fluctuated between 17.55 and 17.92. For the algorithm with the TW of 20, the value of RMSE fluctuated between 15.61 and 16.48 from one layer to eight layers, ending at 16.32. Regardless of the TW of 10 or 20, the prediction accuracy of the model did not get better as the depth of the neural network increased. However, when the TW was set to 30, the performance of the model became better as the depth of the layer increased. The increase in the depth of the model after seven layers could not lead to the improvement of the result and would have increased the burden on the network. In this experiment, the seven-layer model with the TW 30 achieved the best result in the FD001 testing set.

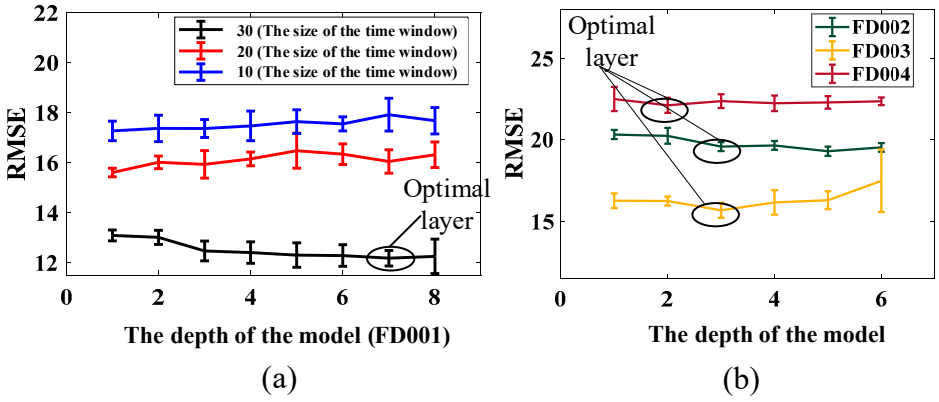

Figure 9. Prediction results: (**a**) FD001;(**b**) FD002, FD003 and FD004.

Figure 9a shows that the increase of the TW could improve the prediction result because the larger time window meant that more raw information could be covered, which could better reflect the degradation of the aircraft engines [27]. However, the TW could not be increased indefinitely, the maximum allowable TW was determined by the smallest operating cycles present in the testing set. For example, for an engine with an operating history of 20 cycles, it was impossible to create a time

window of 30 cycles [22]. Therefore, the TWs selected for FD001, FD002, FD003, and FD004 were 30, 20, 30 and 15, respectively.

As shown in Figure 9b, for the FD002 dataset, as the depth of the model increased from one to three, the prediction accuracy was improved, but if the depth of the model further increased, the prediction result would fluctuate. For the FD003 and FD004 datasets, as the depth of the corresponding model increased, the optimal prediction results were obtained when the third layer and the second layer were respectively reached. If the model depth was continuously increased, the prediction result of the model would fluctuate and increase computational costs.

3.3.3. The Effect of the Optimization Algorithms

An appropriate optimization algorithm is crucial for the prediction results, which can further improve the prediction accuracy. The optimization algorithm has the effects of damping out the oscillations in the gradient descent. In the field of deep learning, optimization algorithms such as Adagrad, Adadelta, RMSprop and Adam are often used for gradient descent. This paper studied the effect of these algorithms on the prediction results. In this set of comparative experiments, the size of the TW was set to 30, and the depth of the deep CNN was seven for the FD001 dataset.

Table 6 presents that Adam had the lowest score with the best performance, while the other algorithms spent more computational costs with higher score. The RMSE of Adam and Adagrad were almost the same, and RMSprop had the largest standard deviation. The learning rate of 0.001 was appropriate to Adam, Adagrad and RMSprop. The reduction of the learning rate will cause the training time to increase, and the larger value of the learning rate will cause the result to oscillate. However, the learning rate of 0.001 was too slow for Adadelta algorithm to converge, so the learning rate of the Adadelta was set to 0.01. According to the time spent in the training process, Adam algorithm ran on the GPU for the shortest time, with the average running time of 72.56s. While the learning rate was increased, the Adadelta algorithm took the longest time on the GPU and cost 621.15s. Adam combines the advantages of Adagrad and RMSprop, having the ability to handle sparse gradients and deal with non-stationary targets. Therefore, from the results of the computational cost and the evaluation function, Adam was the optimal gradient descent algorithm.

Table 6. Computational cost and result.

Optimization Algorithms	Learning Rate	Epoch Number	RMSE (Mean)	RMSE (Std)	GPU Time (s)
Adam	0.001	40	12.18	0.31	72.56
Adagrad	0.001	150	12.22	0.24	303.49
Adadelta	0.01	300	12.34	0.12	621.15
RMSpro	0.001	50	13.89	0.69	102.16

3.3.4. Comparing with Related Works

This paper used all the sub-datasets in the NASA C-MAPSS dataset. Some literature has only reported results on the FD001 dataset in terms of RMSE, but in order to fully evaluate the effectiveness of the model, the results of all four sub-datasets were given. In this section, the performance of the proposed method is compared to other state-of-the-art methods reported in the literature.

As shown in Table 7, the proposed models outperformed the existing prediction models in the datasets except for FD003. There are two failure models of FD003, therefore the network proposed in this paper had relatively weak learning ability in this sub-dataset, but it was still better than the results in References [23,28]. FD002 and FD004 correspond to complex simulation conditions, and have more practical significance. The method proposed in this paper achieved better results than the previous methods on these two datasets. The RUL prediction was based on health indicators in References [17,19], and these two methods were only validated on FD001. Compared with these two methods, the proposed network reduced the score on FD001 by 17.64 and 0.63, respectively. A similar

study, which was published in Reference [27] and achieved the best results on the C-MAPSS dataset in published papers, were selected to be compared with our method. However, the shortage of this method was that it selected a specific kernel artificially. Therefore, the performance of this method was worse than the proposed method in terms of FD001, FD002 and FD004 datasets. In addition, the proposed network showed its superiority over the other time window-based methods in References [22,23,28].

Table 7. Performance of various methods on the C-MAPSS dataset (RMSE).

Approach	FD001	FD002	FD003	FD004	Constant Value of RUL in the Early Stage
Support vector machine [17]	29.82	N/A	N/A	N/A	Not applied
First attempt of CNN [28]	18.45	30.29	19.82	29.16	130
Random forest [23]	17.91	29.59	20.27	31.12	Not applied
Gradient boosting [23]	15.67	29.09	16.84	29.01	Not applied
ANN [22]	15.16	N/A	N/A	N/A	Not provided
MODBNE method [23]	15.04	25.05	12.51	28.66	Not applied
LSTM [19]	12.81	N/A	N/A	N/A	125
CNN with time window [27]	12.61	22.36	12.64	23.31	125
Proposed method	12.18	19.58	15.67	22.12	125

4. Conclusions

The traditional health indicator-based methods need the long-term tracking of the signals, which are laborious and time-consuming. A novel CNN based on the time window was proposed to tackle this issue. Additionally, the kernel module was constructed, which can let the network learn to select the kernels automatically. The effectiveness of the proposed network was demonstrated using the C-MAPSS dataset. The main contributions of this paper are as follows:

(1) The raw industrial sensor signals were standardized and transmitted to the network for RUL prediction without long-term tracking and constructing HIs that relied on signal processing expertise.

(2) A novel kernel module composed of one-dimensional convolutional kernels of different sizes was proposed for feature extraction in time series.

(3) The 1×1 convolutional kernel was introduced to handle the multiple channels in the kernel module.

However, some limitations still existed in the presented methods. For example, the networks for each sub-dataset were trained and evaluated separately. Besides, the C-MAPSS dataset was created by simulation models in controlled environments. The real world has many uncontrollable events, which should be considered further.

Author Contributions: Methodology, H.Y., Z.S., G.J., F.Z. and X.M.; validation: H.Y., G.J. and Z.S.; writing—original draft preparation, H.Y., Z.S., G.J., F.Z. and X.M.; funding acquisition, F.Z.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2018AAA0101800.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Si, X.S.; Wang, W.; Hu, C.-H.; Zhou, D.-H. Remaining useful life estimation—A review on the statistical data driven approaches. *Eur. J. Oper. Res.* **2011**, *213*, 1–14. [CrossRef]
2. Lei, Y.; Lin, J.; He, Z.; Zuo, M.J. A review on empirical mode decomposition in fault diagnosis of rotating machinery. *Mech. Syst. Signal Process.* **2013**, *35*, 108–126. [CrossRef]
3. Yin, S.; Li, X.; Gao, H.; Kaynak, O. Data-Based Techniques Focused on Modern Industry: An Overview. *IEEE Trans. Ind. Electron.* **2015**, *62*, 657–667. [CrossRef]
4. Lee, J.; Wu, F.; Zhao, W.; Ghaffari, M.; Liao, L.; Siegel, D. Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mech. Syst. Signal Process.* **2014**, *42*, 314–334. [CrossRef]

5. Lei, Y.; Li, N.; Guo, L.; Li, N.; Yan, T.; Lin, J. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mech. Syst. Signal Process.* **2018**, *104*, 799–836. [CrossRef]
6. Heng, A.; Zhang, S.; Tan, A.C.C.; Mathew, J. Rotating machinery prognostics: State of the art, challenges and opportunities. *Mech. Syst. Signal Process.* **2009**, *23*, 724–739. [CrossRef]
7. Li, N.; Lei, Y.; Lin, J.; Ding, S. An improved exponential model for predicting remaining useful life of rolling element bearings. *IEEE Trans. Ind. Electron.* **2015**, *62*, 7762–7773. [CrossRef]
8. Lei, Y.; Li, N.; Gontarz, S.; Lin, J.; Radkowski, S.; Dybala, J. A Model-Based Method for Remaining Useful Life Prediction of Machinery. *IEEE Trans. Reliab.* **2016**, *65*, 1314–1326. [CrossRef]
9. Yoo, Y.; Baek, J.-G. A Novel Image Feature for the Remaining Useful Lifetime Prediction of Bearings Based on Continuous Wavelet Transform and CNN. *Appl. Sci.* **2018**, *8*, 1102. [CrossRef]
10. Tian, Z. An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. *J. Intell. Manuf.* **2012**, *23*, 227–237. [CrossRef]
11. Benkedjouh, T.; Medjaher, K.; Zerhouni, N.; Rechak, S. Health Assessment and Life Prediction of cutting tools based on support vector regression. *J. Intell. Manuf.* **2015**, *26*, 213–223. [CrossRef]
12. Hong, S.; Zhou, Z.; Zio, E.; Hong, K. Condition assessment for the performance degradation of bearing based on a combinatorial feature extraction method. *Digit. Signal Process.* **2014**, *27*, 159–166. [CrossRef]
13. Huang, Z.; Xu, Z.; Wang, W.; Sun, Y. Remaining Useful Life Prediction for a Nonlinear Heterogeneous Wiener Process Model with an Adaptive Drift. *IEEE Trans. Reliab.* **2015**, *64*, 687–700. [CrossRef]
14. Gebraeel, N.; Lawley, M.; Liu, R.; Parmeshwaran, V. Residual life predictions from vibration-based degradation signals: A neural network approach. *IEEE Trans. Ind. Electron.* **2004**, *51*, 694–700. [CrossRef]
15. Hanachi, H.; Jie, L.; Banerjee, A.; Ying, C.; Koul, A. A Physics-Based Modeling Approach for Performance Monitoring in Gas Turbine Engines. *IEEE Trans. Reliab.* **2015**, *64*, 197–205. [CrossRef]
16. Lu, C.; Tao, L.; Fan, H. An intelligent approach to machine component health prognostics by utilizing only truncated histories. *Mech. Syst. Signal Process.* **2014**, *42*, 300–313. [CrossRef]
17. Louen, C.; Ding, S.X.; Kandler, C. A new framework for remaining useful life estimation using Support Vector Machine classifier. In Proceedings of the Control & Fault-tolerant Systems, Nice, France, 9–11 October 2013.
18. Guo, L.; Li, N.; Jia, F.; Lei, Y.; Lin, J. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing* **2017**, *240*, 98–109. [CrossRef]
19. Malhotra, P.; Tv, V.; Ramakrishnan, A.; Anand, G.; Vig, L.; Agarwal, P.; Shroff, G. Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder. *arXiv* **2016**, arXiv:1608.06154.
20. Guo, L.; Lei, Y.; Li, N.; Yan, T.; Li, N. Machinery HI construction based on convolutional neural networks considering trend burr. *Neurocomputing* **2018**, *292*, 142–150. [CrossRef]
21. Lei, Y.; Li, N.; Lin, J. A New Method Based on Stochastic Process Models for Machine Remaining Useful Life Prediction. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 2671–2684. [CrossRef]
22. Lim, P.; Chi, K.G.; Tan, K.C. A time window neural network based framework for Remaining Useful Life estimation. In Proceedings of the International Joint Conference on Neural Networks, Vancouver, BC, Canada, 24–29 July 2016.
23. Zhang, C.; Lim, P.; Qin, A.K.; Tan, K.C. Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. *IEEE Trans. Neura. Netw. Learn. Syst.* **2017**, *28*, 2306–2318. [CrossRef] [PubMed]
24. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
25. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
26. Hinton, G.E.; Salakhutdinov, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]
27. Li, X.; Ding, Q.; Sun, J.-Q. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliab. Eng. Syst. Saf.* **2018**, *172*, 1–11. [CrossRef]
28. Babu, G.S.; Zhao, P.; Li, X.L. Deep CNN Based Regression Approach for Estimation of Remaining Useful Life. In Proceedings of the International Conference on Database Systems for Advanced Applications, Dallas, TX, USA, 16–19 April 2016.

29. Saxena, A.; Kai, G.; Simon, D.; Eklund, N. Damage propagation modeling for aircraft engine run-to-failure simulation. In Proceedings of the International Conference on Prognostics & Health Management, Denver, CO, USA, 6–9 October 2008.
30. Xu, J.; Wang, Y.; Xu, L. PHM-Oriented Integrated Fusion Prognostics for Aircraft Engines Based on Sensor Data. *IEEE Sens. J.* **2014**, *14*, 1124–1132. [CrossRef]