Precision Mechanics & Digital Fabrication
https://www.acadlore.com/journals/PMDF

A Remaining Useful Life Prediction Method for Rolling Bearings Based on Broad Learning System - Multi-Scale Temporal Convolutional Network

Tichun Wang*, Qiji Teng, Guanghu Jin

College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, 210000 Nanjing, China

* Correspondence: Tichun Wang (wangtichun2010@nuaa.edu.cn)

Received: 08-03-2024 **Revised:** 09-19-2024 **Accepted:** 09-25-2024

Citation: T. C. Wang, Q. J. Teng, and G. H. Jin, "A remaining useful life prediction method for rolling bearings based on broad learning system - multi-scale temporal convolutional network," *Precis. Mech. Digit. Fabr.*, vol. 1, no. 3, pp. 145–157, 2024. https://doi.org/10.56578/pmdf010303.

Abstract: Rolling bearings play a critical role in various industrial applications. However, the complexity and diversity of data, along with the challenge of selecting the most representative features from a large set and reducing dimensionality to lower computational costs, pose significant challenges for accurately predicting the remaining useful life (RUL) of rolling bearings. To address this, a hybrid model combining the broad learning system (BLS) and multi-scale temporal convolutional network (MsTCN) is proposed for RUL prediction of rolling bearings. The BLS is employed to capture a broad range of features from the full-life signals of rolling bearings, while the MsTCN adaptively extracts multi-scale temporal features, effectively capturing both short-term and long-term dependencies in the bearing's operational process. Additionally, the fusion and optimization of features extracted by BLS and MsTCN enhance the representational power of the prediction model. Experiments conducted on the PHM2012 bearing dataset demonstrate that the proposed method significantly improves model performance and prediction accuracy.

Keywords: Remaining useful life (RUL) prediction; Multi-scale temporal convolutional network (MsTCN); Broad learning system (BLS); Rolling bearings

1 Introduction

With the continuous development of industrial equipment, rolling bearings, as key rotating components in mechanical systems, play an irreplaceable role in manufacturing, transportation, and aerospace industries [1]. However, during long-term operation, rolling bearings may be affected by overload, contamination, and impact, which can lead to more severe failure modes and accelerate bearing degradation [2]. Therefore, accurately extracting the features of rolling bearings and predicting their RUL [3] is of profound significance for ensuring the safety performance of equipment [4–6].

RUL prediction methods can be categorized into four types based on their fundamental techniques and approaches: physics-based models, statistical models [7], deep learning models, and hybrid methods [8, 9]. Firstly, the core idea of physics-based models is to predict RUL by establishing a physical model of the equipment's operation, but this approach requires extensive expert knowledge and is difficult to model. Secondly, statistical models often exhibit strong nonlinear behavior, which may lead to information loss and require relatively ideal assumptions [10]. In recent years, with the rapid development of deep learning and neural networks, deep learning and hybrid methods have gradually become research hotspots for RUL prediction of rolling bearings [11]. Schwendemann et al. [12] proposed a combination of convolutional neural networks (CNN) and long short-term memory (LSTM) networks for classification and estimation to predict the RUL of bearings. Qi et al. [13] proposed a bearing RUL prediction method based on a broad learning system-temporal convolutional network (BLS-TCN), which enhances the identification of multiple failure features through the feature extraction capability of the temporal convolutional network while reducing the complexity of the network model through the flattened structure of broad learning, thereby reducing computation time. However, ordinary temporal convolutions can only capture single-scale temporal features. Gao and Lu [14] proposed a method based on bidirectional temporal convolutional networks and long short-term memory (Bi-TCN-LSTM), where data is normalized and fused, followed by feature extraction and deep learning using Bi-TCN-LSTM for RUL prediction of rolling bearings. However, the convolution kernel size of TCN [15] is limited, making it difficult to capture long-term dependencies, which limits the model's ability to predict the RUL of bearings. Deng et al. [16] proposed a MsTCN-transformer network model for prediction, which can adaptively extract inherent temporal feature information from the full-life signals of rolling bearings. They then constructed a transformer network model based on the self-attention mechanism. However, the transformer model typically requires more computational resources, especially when dealing with long sequences, where the computational and memory demands can increase significantly. Additionally, the model's complexity is relatively high, making it less effective at capturing local features of time series data.

To solve the above problems, this paper proposes a hybrid model (BLS-MsTCN) combining BLS and MsTCN for RUL prediction of rolling bearings. BLS [17], proposed by Junlong Chen and his students, is a novel neural network structure that increases the model's representation ability by expanding its breadth (width), and it can learn and update quickly, with significant advantages in computational efficiency. MsTCN has made several improvements based on TCN, widely using dilated convolutions with different dilation factors to expand the receptive field without increasing the number of parameters, enabling the network to capture long-term dependencies and multi-scale patterns in temporal data. Furthermore, residual connections [18] are introduced at each stage to alleviate the vanishing gradient problem.

The main innovations of BLS-MsTCN are as follows:

(1) The operational data of rolling bearings is usually high-dimensional and complex. BLS-MsTCN, through its broad structure, can process large amounts of data in a short time, improving the efficiency of data processing and the scalability of the model.

(2) The operational state and failure modes of rolling bearings often exhibit long-term dependencies. BLS-MsTCN, with its multi-stage dilated convolution structure, can capture features at different temporal scales, and the stepwise refinement between stages allows for better modeling of long-term dependencies.

(3) The operational environment of rolling bearings is complex and variable, and the data characteristics under different operating conditions may vary significantly. The efficient feature extraction and multi-scale temporal modeling capabilities of BLS-MsTCN enable the model to adapt to the data characteristics under different operating conditions, enhancing prediction performance in various complex environments.

2 Construction of the BLS-MsTCN Model

2.1 Basic Modules of BLS-MsTCN

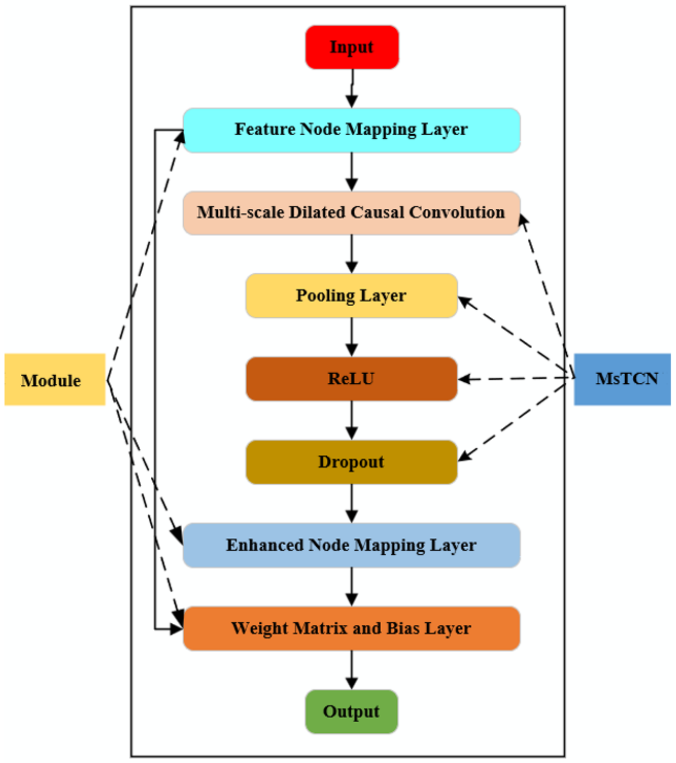
Input
Feature Node Mapping Layer
Multi-scale Dilated Causal Convolution
Pooling Layer
ReLU
Dropout
Enhanced Node Mapping Layer
Weight Matrix and Bias Layer
Output
Module
MsTCN

Figure 1. BLS-MsTCN module

The composition of the BLS-MsTCN modules is shown in Figure 1. As can be seen from the figure, BLS-MsTCN mainly consists of a multi-scale temporal convolution module and a broad learning module.

2.1.1 Multi-scale temporal convolution module

The MsTCN aims to enhance the modeling capability for time series data and better handle changes at different temporal scales. The basic module structure of MsTCN is shown in Figure 2.

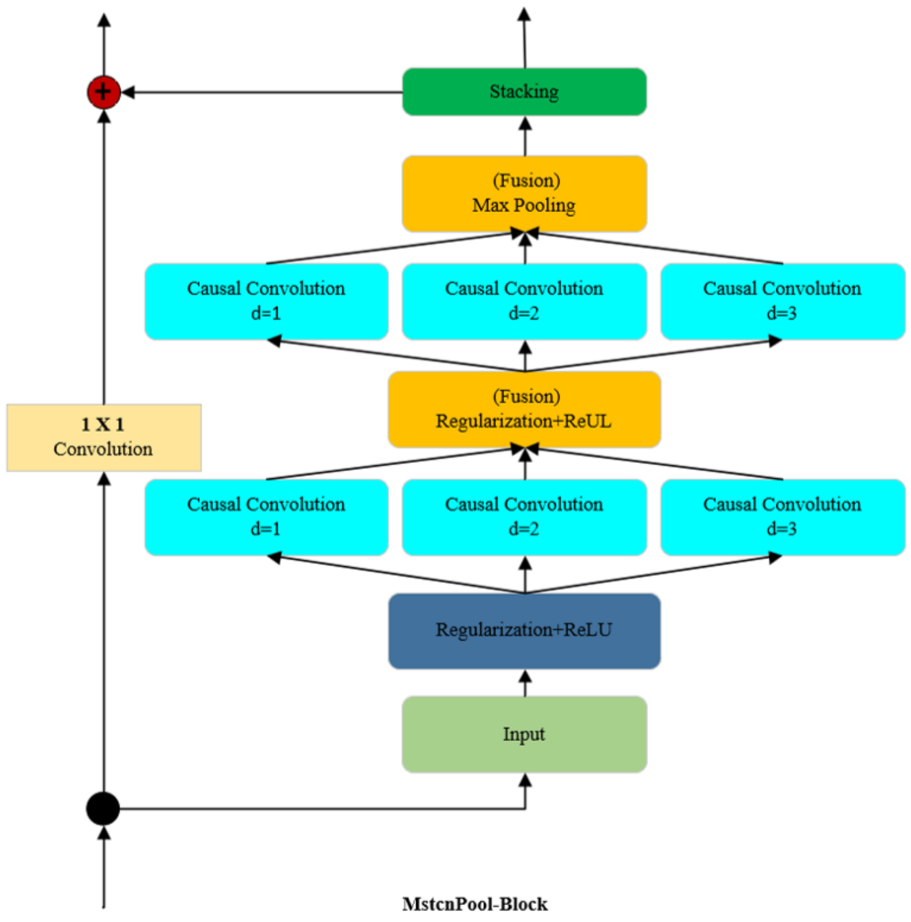

Figure 2. Basic structure of the MsTCN module

As shown in Figure 2, compared with ordinary temporal convolutional networks, the biggest difference in MsTCN lies in the feature extraction method, specifically in its multi-scale feature extraction and fusion strategy.

Although ordinary temporal convolutional networks can also use different dilation factors to expand the receptive field, the convolution at each layer can only use a fixed dilation factor, so the receptive field can only be expanded by stacking layers. As the number of stacked convolutional layers increases, the network depth also increases, resulting in higher training complexity.

In contrast, MsTCN introduces multi-scale convolution at each layer, that is, applying convolutional kernels with different dilation factors in parallel within the same layer (such as d=1, d=2, d=3), and each layer adopts full pre-activation [19], thereby capturing short-term, medium-term, and long-term dependency features simultaneously within the same layer. This structure significantly reduces the network depth and computational complexity while enhancing the effectiveness of feature extraction.

In ordinary temporal convolutional networks, feature fusion is mainly achieved through simple connections between layers, and the integration of features at different temporal scales primarily relies on the stacking of deep networks. This approach may result in suboptimal performance when capturing complex dependencies and can lead to information loss.

In MsTCN, feature fusion after multi-scale convolution at each layer is more flexible: it extracts features at different temporal scales through parallel convolution and then integrates these features into a unified representation through a fusion layer. This fusion process typically incorporates activation functions (such as ReLU), thereby enhancing the model's nonlinear representation capability.

Pooling operations are added to feature fusion in the MsTCN module, further improving the model's performance in complex scenarios. Pooling operations aggregate these multi-scale features, making the information more concentrated, which helps to integrate key information across different temporal scales and improves the overall performance of the model. Moreover, by using pooling operations, the model can avoid overfitting detailed information to a certain extent, thereby improving its generalization ability. The overall structure of MsTCN is shown in Figure 3.

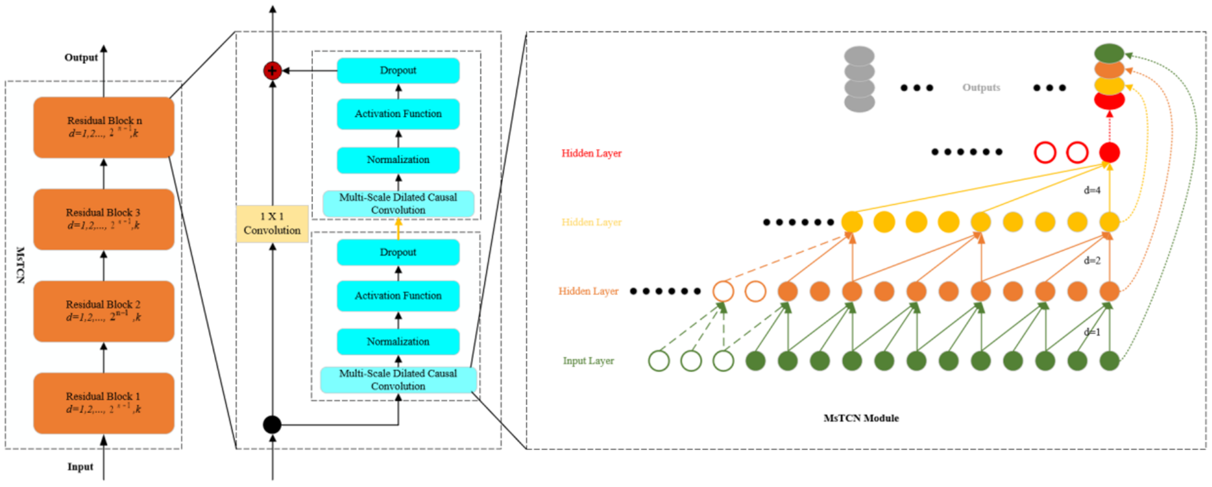

Figure 3. Overall structure of the MsTCN

As shown in Figure 3, the network structure of MsTCN effectively captures both short-term and long-term dependencies in sequences by introducing dilated causal convolution, residual connections, and multi-scale feature fusion. The residual block used in the residual connections is an improved structure by Bai et al. [20], which addresses the overfitting issue in residual structures.

The core part of MsTCN consists of multiple multi-scale dilated convolution residual blocks. These residual blocks use different dilation rates (such as d=1, d=2, d=4), thus extracting features in parallel across different time spans. The introduction of dilated convolution enables the network to have a larger receptive field while maintaining fewer parameters, allowing it to capture both short-term details and long-term trends simultaneously. Moreover, the use of residual connections ensures that the network can still be trained effectively as its depth increases while preserving the original input information.

For a one-dimensional input sequence $x = [x_1, x_2, \ldots, x_t]$, the output $F(s)$ of the dilated causal convolution at time step t can be expressed as:

$$F(s) = \sum_{i=0}^{k-1} f(i) \times x_{s-d\times i} \tag{1}$$

where $F(\cdot)$ represents the convolution operation function, k represents the size of the convolution kernel ($k = 3$ in the figure), $f(i)$ is the i-th data point in the convolution kernel, $s - d \cdot i$ represents the direction of historical time, and d represents the dilation factor

In the residual block, by setting different dilation rates, the convolution operation is performed in parallel at different time scales, enabling the capture of both short-term and long-term dependencies. After the convolution operation, a normalization layer (such as Batch Normalization) is applied to help stabilize training, followed by activation functions such as ReLU to introduce non-linearity and enhance the model's expressiveness.

To prevent overfitting, a Dropout layer is introduced in the residual block, randomly deactivating some neurons and improving the model's generalization ability.

MsTCN captures information at different temporal scales through convolutions with different dilation rates and fuses these multi-scale features at each layer. By transmitting information across hidden layers, the model can gradually aggregate features from different time windows, thus capturing local details while maintaining an understanding of the overall trend. This multi-scale fusion design enables MsTCN to handle diverse time series data.

In the MsTCN structure, assuming the output of the n-th layer is denoted as $T(n)$, the output of an MsTCN with m layers can be expressed as:

$$Outputs(x) = Concate(T(1), T(2), \cdots, T(M), x) \tag{2}$$

where, $Outputs(\cdot)$ represents the final output of the MsTCN, and $Concate(\cdot)$ represents the concatenation operation, i.e., concatenating multiple feature matrices along the last channel dimension.

2.1.2 Broad learning

BLS is derived from a network structure called the Random Vector Functional-Link Neural Network (RVFLNN) [21]. It is an efficient neural network learning framework designed to improve learning capacity and efficiency by expanding the width rather than the depth of the network. Compared with traditional deep learning, broad learning builds the network by stacking multiple layers of feature nodes and enhancement nodes, thus avoiding the common issues of gradient vanishing and complex training processes in deep learning.

Broad learning consists of an input layer, a hidden layer, and an output layer, with its structure diagram shown in Figure 4.

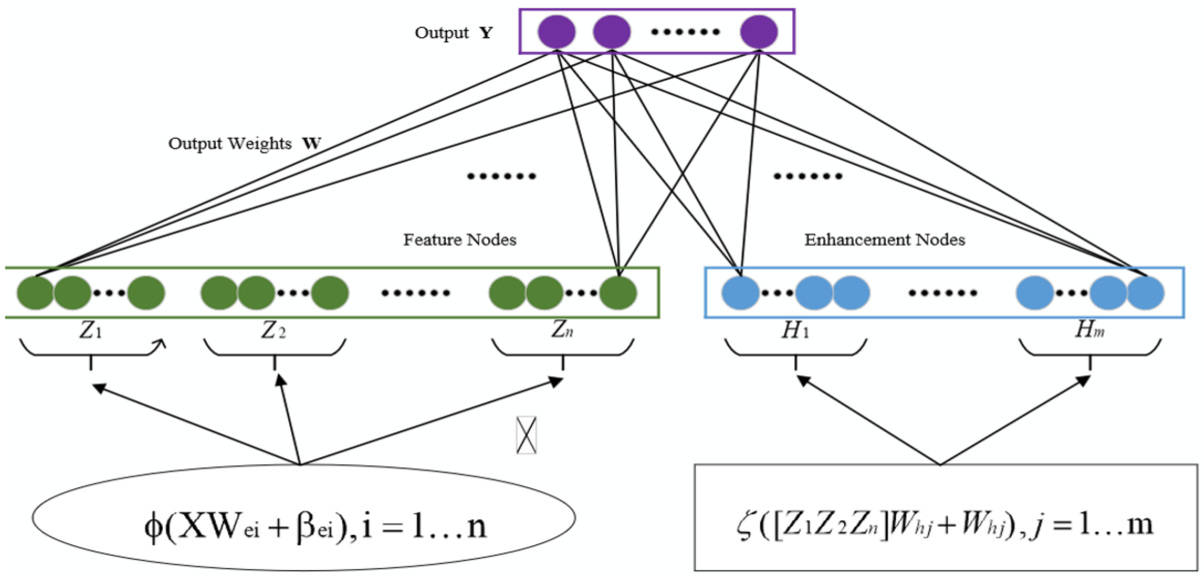

Figure 4. BLS network structure

As shown in Figure 4, the hidden layer of BLS is composed of two parts: one part consists of nodes obtained through neural network learning of the original features, called feature nodes, while the other part consists of new nodes obtained through incremental learning of the feature nodes, called enhancement nodes.

In BLS, after processing through the feature node layer, the input undergoes a linear transformation using specific weight matrices and biases and is processed non-linearly through activation functions, generating an initial feature representation. This process can be expressed as:

For a given training set $\left\{(X, Y) \mid X \in R^{N\times p}, Y \in R^{N\times q}\right\}$, where N is the number of training samples, p is the sample dimension, and q is the label dimension, the output of the mapping layer is:

$$Z_{\mathrm{i}} = \varphi\left(XW_{ei} = \beta_{ei}\right), i = 1, \ldots, n \tag{3}$$

where, $\varphi(\cdot)$ represents the activation function from the input layer to the mapping layer, and W_{ei} and β_{ei} are the weights and biases of the i-th feature node. The final feature layer Z is:

$$Z = [Z_1, Z_2, \ldots, Z_n] \tag{4}$$

The feature nodes extract features from the input data and map the input data into the feature space using different weight matrices and nonlinear activation functions. In BLS, the feature node layer plays a role similar to the hidden layer in traditional neural networks but with a simpler structure and lower computational overhead.

The enhancement node layer combines the output of the feature nodes with a set of new weight matrices in a linear combination and applies a nonlinear function, producing the output:

$$H_{\mathrm{j}} = \zeta\left(ZW_{hj} + \beta_{hj}\right), j = 1, \ldots, m \tag{5}$$

$$H = [H_1, H_2, \ldots, H_{\mathrm{m}}] \tag{6}$$

By concatenating the feature nodes and enhancement nodes, the final output Y is obtained.

The output from both the feature nodes and enhancement nodes is connected to the final output layer through output weights. The output layer combines the information from both feature and enhancement nodes and generates the final prediction result Y by weighted summation:

$$Y = [Z \mid H]W \tag{7}$$

where, W is the output weight between the hidden layer and the output layer, then $W = X^{+}Y$ can be obtained through pseudo-inverse calculation.

BLS expands the feature space through random feature mapping and enhancement nodes, analyzing the health state of bearings from multiple dimensions and enhancing the model's expressive capacity to improve the performance of RUL prediction.

2.2 BLS-MsTCN Model Structure and Hyperparameters

2.2.1 BLS-MsTCN model structure

The multi-scale temporal convolution module and the broad learning module mentioned earlier form the basic structure of BLS-MsTCN. The model structure of BLS-MsTCN is shown in Figure 5.

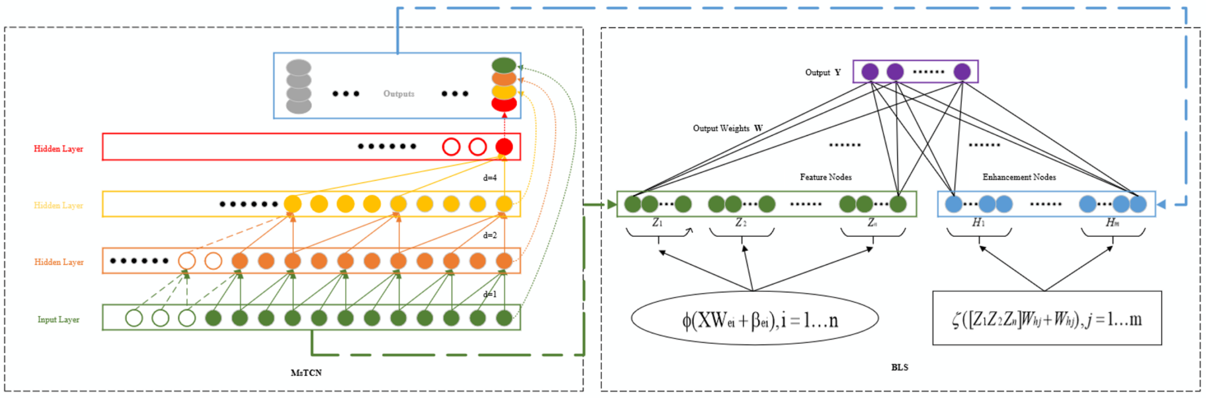

Figure 5. BLS-MsTCN model structure

As shown in Figure 5, MsTCN replaces the mapping relationship between feature nodes and enhancement nodes in BLS, enhancing the learning capacity of the network without increasing its complexity.

Algorithm 1

```
Input:
    $X, X, \varphi, \zeta, Y$
Start Training:
    Initialize the number of feature node groups $n$, the number of enhancement node groups $m$, the activation function
weights and biases from the input layer to the mapping layer $W_{\mathrm{ei}}$ , $\beta_{ei}$, and the activation function weights and biases from the
mapping layer to the enhancement layer $W_{MSTCN}, \beta_{MSTCN}$.
Loop:
1. First, pass the input data through the activation function $\varphi$ to obtain the output of the feature nodes $Z_i$, and group them
into the MsTCN module. After applying the activation function $\zeta$, obtain the output of the enhancement nodes $H_j$.
2. Concatenate the feature nodes and enhancement nodes to obtain a new feature matrix A.
3. Add additional enhancement nodes $H_{m+i} = [\zeta\,(A_i w_{MSTCN} + \beta_{MSTCN})]$.
4. Update the feature matrix and calculate the pseudo-inverse.
5. Update the new weights $W_{M\mathrm{s}\mathrm{TCN}}{}'$ and biases $\beta_{MSTCN}{}'$ from the mapping layer to the enhancement layer.
6. Obtain the updated weight matrix $W^{m+1}$.
7. Check if there are additional enhancement nodes. If none, end the loop.
Output: Trained weights $W_{MsTcN}{}'$ and biases $\beta_{MSTCN}{}'$ from the feature layer to the enhancement layer, and weight matrix
$W^{m+1}$ from the feature layer and enhancement layer to the output layer.
```

Furthermore, BLS-MsTCN combines broad learning and multi-scale temporal convolution to capture multiple signal features simultaneously, enabling the model to integrate diverse information when processing large datasets, thereby improving prediction accuracy.

The multi-layer convolution structure in BLS-MsTCN can capture both short-term and long-term features by using convolution kernels of different sizes. Small convolution kernels can identify rapidly changing signal patterns, while large convolution kernels can extract slowly changing trends, enhancing the model's prediction precision.

Assume X is the input data for rolling bearings, $x\phi, \zeta.$ are activation functions from the input layer to the feature layer and from the feature layer to the enhancement layer, respectively, and Y is the actual remaining life. The BLS-MsTCN network algorithm flow is shown in Algorithm 1.

2.2.2 BLS-MsTCN hyperparameter settings

Hyperparameters play a crucial role in the performance of the BLS-MsTCN model. Different configurations of hyperparameters not only affect the training speed but also directly impact the model's generalization ability. For bearing RUL prediction, appropriate hyperparameter settings enhance the model's capability to capture complex temporal dependencies, thus improving prediction accuracy.

As shown in Table 1, the hyperparameter settings have been experimentally validated to achieve optimal prediction performance on the specific dataset used in this study.

Table 1. Hyperparameter settings

Parameter Name	Specific Parameter Value
Optimizer	Adam
Loss Function	Mean Squared Error (MSE), Mean Absolute Error (MAE)
Dilation Factors (d)	$1/2/4$
Number of Convolution Kernels	32
Kernel Size (k)	3×3
Learning Rate	0.001
Feature Multiplier	5
Number of Feature Node Windows	5
Number of Enhancement Nodes	5
Total Number of Enhancement Nodes	64
Batch Size	512
Epochs	200

The model uses the Adam optimizer to effectively handle training on large datasets, with a global learning rate set at 0.001 [22, 23]. Adam is a popular and widely-used optimizer, known for its adaptive learning rate adjustment mechanism, which performs well across many tasks. The combination of MSE and MAE loss functions is applied to comprehensively evaluate model performance. By setting dilation factors to 1/2/4, the model can capture features across different temporal scales, enabling the extraction of multi-scale temporal information. The use of 32 convolution kernels and a kernel size of 3×3 ensures effective feature extraction. Setting the learning rate to 0.001 enables the model to converge more stably, reduces fluctuations, and ensures stable updates of the model.

To enhance the model's nonlinear representation capabilities, the feature multiplier and number of enhancement nodes are both set to 5, with a total of 64 enhancement nodes. During training, a batch size of 512 and 200 epochs ensures that the model is sufficiently trained. These hyperparameter choices provide a solid foundation for the model's accuracy, stability, and generalization ability.

The combination of the Adam optimizer, a small learning rate, MSE and MAE loss functions is well-suited for complex time series prediction tasks. The dilation factors, multi-scale convolution kernels, a large batch size, and sufficient epochs grant the model adequate expressive power and training space.

3 Experiment

The data used in this experiment comes from the bearing run-to-failure dataset provided by the PHM 2012 Data Challenge, collected from the PRONOSTIA accelerated degradation platform [24], as shown in Figure 6.

Two accelerometers are mounted horizontally and vertically on the experimental setup to collect vibration signals from two directions, namely the x and y directions. The data sampling rate is 25.6 kHz, and data is recorded for 0.1 seconds every 10 seconds, totaling 2560 samples. For safety, the experiment stops when the amplitude of the vibration data exceeds 20g (20 times the gravitational acceleration, where $1\ \mathrm{g} \approx 9.8\ \mathrm{m/s^2}$). The specific test conditions are shown in Table 2.

Table 2. Bearing data overview

Condition	Load (N)	Speed (rpm)	Training Set	Testing Set
Condition 1	4000	1800	Bearing1_1-Bearing1_6	Bearing1_7
Condition 2	4200	1650	Bearing2_1-Bearing2_6	Bearing2_7
Condition 3	5000	1500	Bearing3_1-Bearing3_2	Bearing3_3

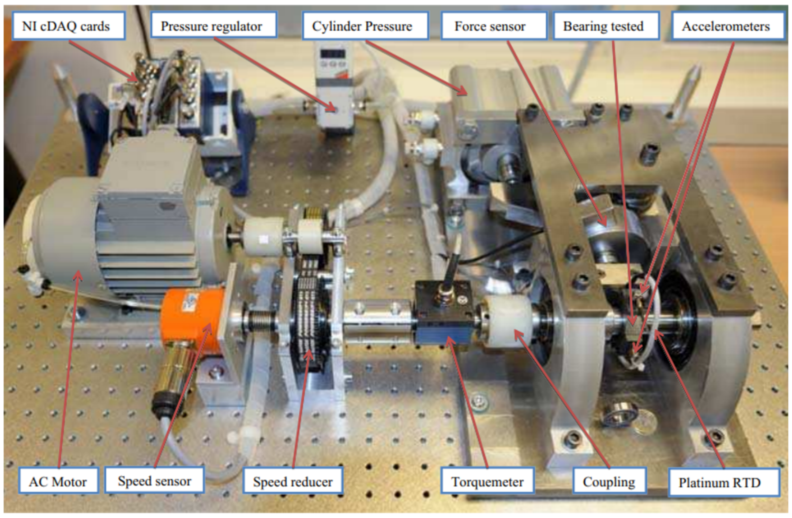

Figure 6. PRONOSTIA accelerated degradation platform

3.1 Experimental Process

In the prediction of RUL for bearings, the RUL prediction is treated as a regression problem. During modeling, the bearing's operating time is transformed into input features, and the remaining life percentage is used as the RUL prediction label. This label design intuitively reflects the degradation level of the bearing throughout its lifecycle, helping improve the performance of the prediction model. The formula is as follows:

$$y_i = 1 - \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{8}$$

where, x_i represents the current operating time of the bearing, $x_{\min}, x_{\max}$ are the bearing's initial and maximum operating times, respectively; y_i represents the severity of bearing degradation, and the smaller its value, the more severe the degradation. When y_i=0, the bearing is fully failed.

In evaluating the experimental results, in addition to the commonly used MAE and Root Mean Squared Error (RMSE), a scoring function specifically adapted for the PHM 2012 dataset, referred to as "score", is used to assess the performance of the developed method. The formulas are as follows:

$$MAE = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} |er_{\mathrm{i}}| \tag{9}$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (er_i)^2} \tag{10}$$

$$score = \frac{1}{n} \sum_{i=1}^{n} S_i \tag{11}$$

$$S_i = \begin{cases} \exp\left(-lin(0.5) \cdot (E_i/5)\right), E_i \leq 0 \\ \exp\left(lin(0.5) \cdot (E_i/20)\right), E_i > 0 \end{cases} \tag{12}$$

$$E_i = \frac{y_i - \hat{y}_i}{y_i} \times 100 \tag{13}$$

In practice, underestimating or overestimating RUL leads to different consequences. Underestimating RUL might result in premature maintenance and unnecessary downtime, which increases costs but poses relatively minor risks. Overestimating RUL is more dangerous as it might allow the equipment to continue running without timely maintenance, increasing the risk of failures and accidents.

The score function gives higher penalties for overestimation, reflecting its severity, ensuring that the model is more cautious when predicting the remaining lifespan of the equipment. At the early stages of an equipment lifecycle, prediction errors have relatively little impact because the equipment is far from failure, and inaccurate predictions pose lower risks. In contrast, errors in the later stages of the lifecycle are more critical as they are closer to equipment failure, making the consequences of prediction errors more severe. Therefore, the score function assigns higher weights to prediction errors in the later stages of the lifecycle, ensuring more accurate predictions at critical stages, which helps reduce failure risks.

The score range is between (0,1), and the higher the score value, the better the prediction performance. However, in performance evaluation, it is common to encounter inconsistent results between the three scoring standards (MAE, RMSE, and score). This inconsistency arises because MAE and RMSE do not consider the sign of the error (overestimation or underestimation) or differentiate between the impacts of different errors, while score incorporates a penalty mechanism for overestimations and underestimations, with a heavier penalty for overestimation. This design aligns better with practical applications, making score a more suitable metric when selecting or evaluating models.

For data processing, the x-axis and y-axis signals are stacked to form a new array of dimension (2660, 2), which is used as input to the model.

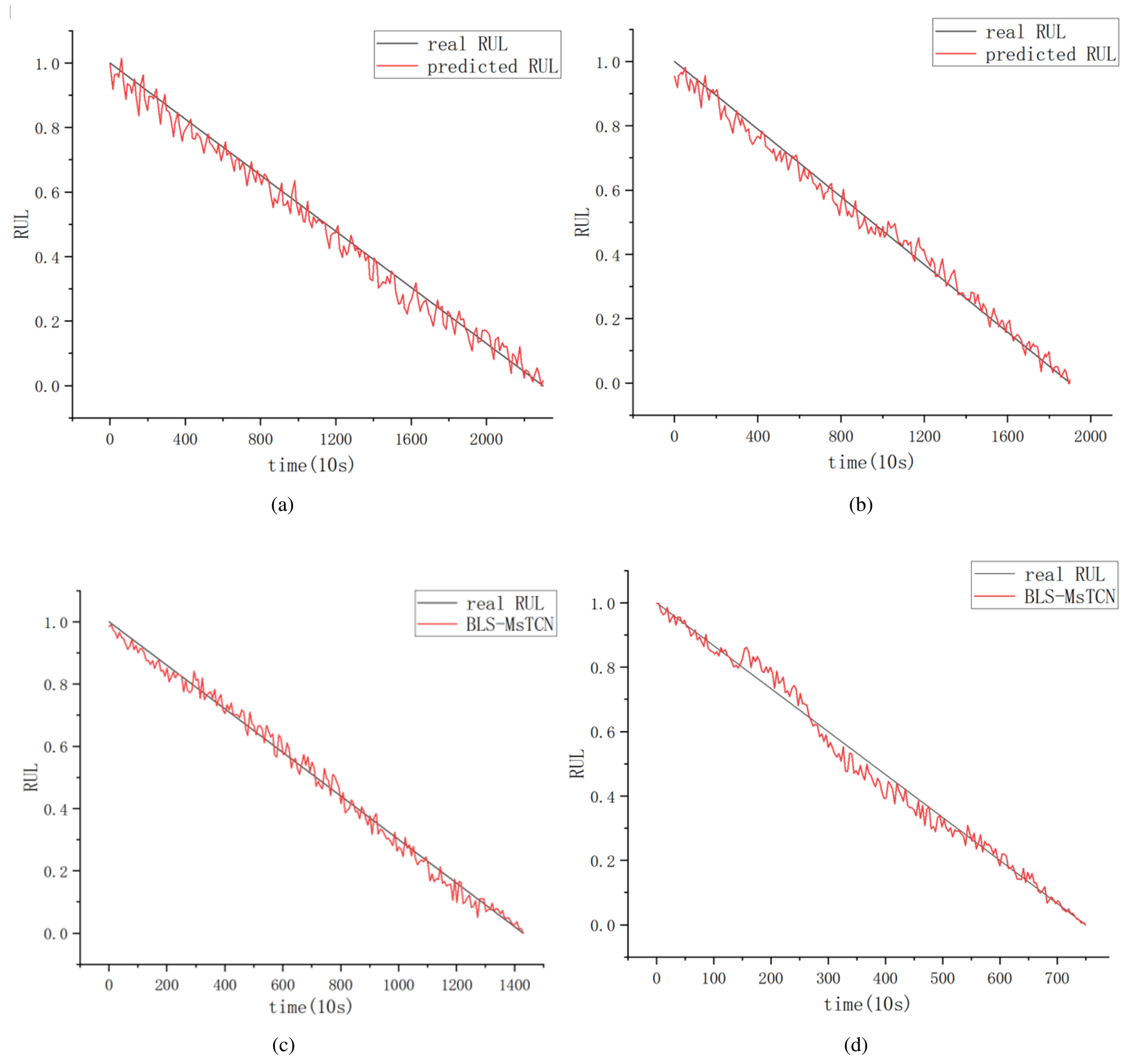

Figure 7. Prediction curves of bearings 1_3, 2_3, 1_4, and 2_4: (a) RUL Prediction curve of bearing 1_3; (b) RUL Prediction curve of bearing 2_3; (c) RUL Prediction curve of bearing 1_4; (d) RUL Prediction curve of bearing 2_4

The data is first divided into training and test sets, followed by preprocessing, and then fed into the BLS-MsTCN model for prediction. The hyperparameters used are shown in Table 1.

The preprocessing step primarily involves normalizing the vibration data and RUL. The raw RUL values can vary widely, making it difficult for the model to effectively process during training. Normalization compresses the data into the (0,1) range, ensuring smoother weight updates and faster convergence, improving training efficiency. The normalized RUL is defined by Eq. (10).

This experiment employs a comparative validation method using the data from Condition 1 and Condition 2. For both conditions, the data from the first six bearings is used to train the model, and the last set of data is used to test the results. Predictions are then compared across the selected bearings. Figure 7 shows the RUL prediction results for bearings 1_3, 2_3, 1_4, and 2_5.

As shown in Figure 7, the RUL prediction results of bearings using the BLS-MsTCN model can effectively track the actual RUL of the bearings.

The experiment selected two bearings from each of Condition 1 and Condition 2, and predictions were made using four different models. Figure 8 provides a comparison of the RUL prediction results for bearings 1_6, 2_6, 1_7, and 2_7 across the four models.

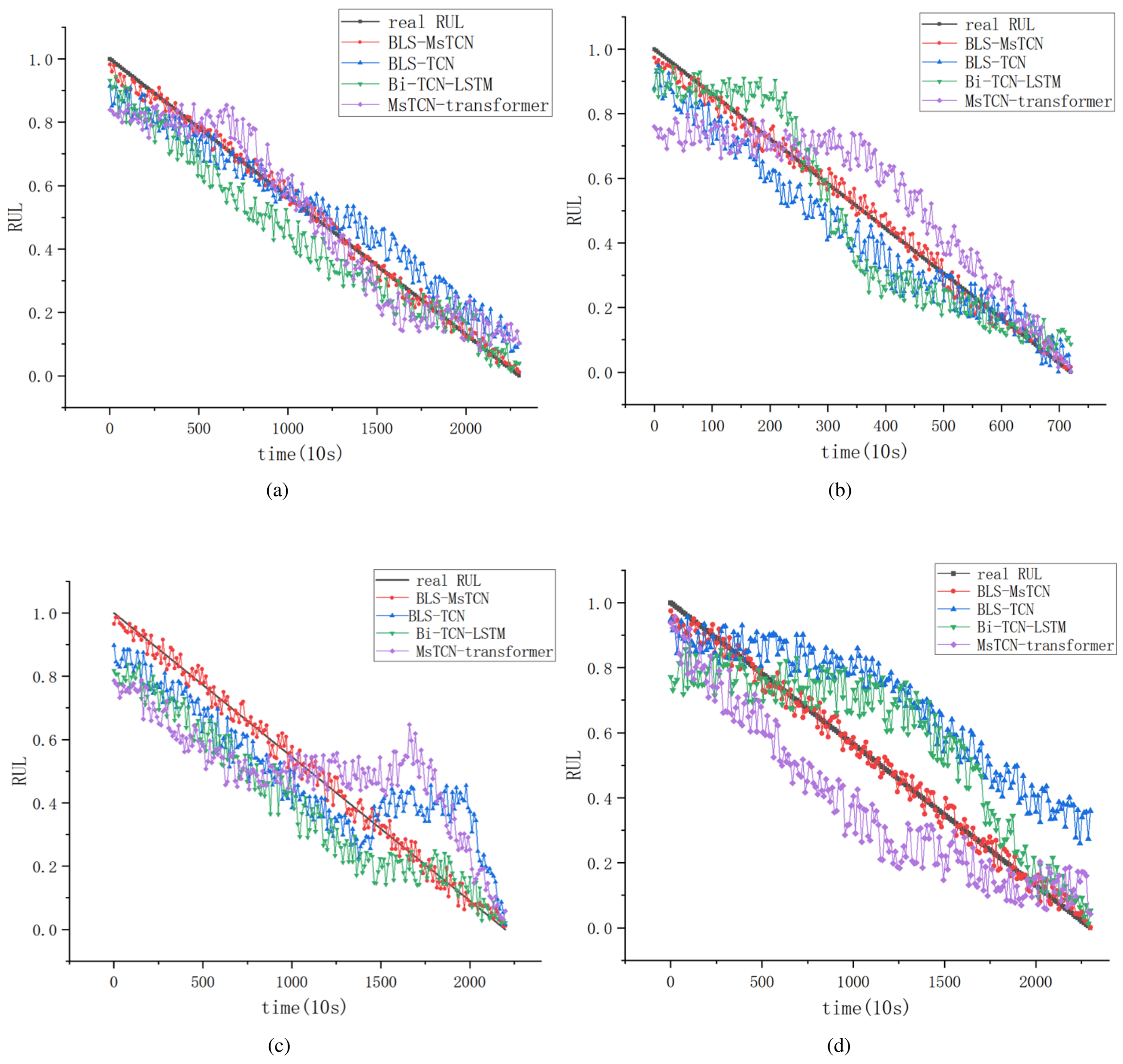

Figure 8. Prediction curve comparison of bearings 1_6, 2_6, 1_7, and 2_7: (a) Bearing 1_6 RUL prediction comparison; (b) Bearing 2_6 RUL prediction comparison; (c) Bearing 1_7 RUL prediction comparison; (d) Bearing 2_7 RUL prediction comparison

As shown in Figure 8, compared with the other three models, the BLS-MsTCN model tracks the actual RUL of the bearings more effectively, with relatively smaller fluctuations.

The 14 sets of bearing data from Condition 1 and Condition 2 were tested across the four prediction models, and the final prediction results were compared. The scores of the seven bearings from Condition 1 are displayed as a bar chart. The results are shown in Table 3 and Figure 9.

From Table 3 and Figure 9, it can be observed that the score of the BLS-MsTCN model is significantly better than that of other models. Additionally, the BLS-MsTCN model demonstrates some advantages in terms of MAE and RMSE compared to the other three models. Although the prediction results for certain bearings may not show a significant advantage in score and may even be slightly lower than those of other models, the overall comparison in Figure 9 clearly indicates that the BLS-MsTCN model achieves better predictions.

Table 3. Prediction scores of the PHM 2012 dataset using four methods

	MsTCN-Transformer			Bi-TCN-LSTM			BLS-TCN			BLS-MsTCN		
Bearing	**MAE**	**RMSE**	**Score**	**MAE**	**RMSE**	**Score**	**MAE**	**RMSE**	**Score**	**MAE**	**RMSE**	**Score**
1_1	12.30	9.90	0.79	14.80	11.30	0.84	13.50	14.00	0.90	16.20	8.10	0.95
1_2	29.00	19.50	0.86	21.30	19.20	0.83	11.00	16.20	0.86	11.20	21.10	0.94
1_3	15.20	13.50	0.87	16.50	21.30	0.88	12.60	19.80	0.81	9.90	9.20	0.89
1_4	12.30	20.40	0.86	15.40	18.70	0.85	15.90	17.10	0.85	19.20	9.50	0.88
1_5	16.70	15.60	0.84	10.80	16.70	0.83	15.70	18.20	0.84	16.50	10.80	0.96
1_6	17.50	17.50	0.84	11.30	19.20	0.85	13.90	13.20	0.90	17.80	11.20	0.92
1_7	12.30	12.30	0.88	17.20	13.40	0.90	20.20	16.40	0.81	15.00	15.30	0.89
2_1	14.20	11.00	0.89	21.20	15.50	0.75	14.30	15.80	0.79	11.60	17.50	0.96
2_2	10.90	16.20	0.85	15.10	9.70	0.77	18.00	21.00	0.77	14.30	16.40	0.94
2_3	13.50	9.90	0.81	14.80	20.90	0.78	20.40	20.50	0.77	17.20	9.70	0.91
2_4	18.70	21.10	0.83	16.50	17.90	0.76	18.70	15.40	0.90	16.20	9.90	0.89
2_5	21.60	16.70	0.81	15.40	16.20	0.79	21.30	19.20	0.79	17.30	10.00	0.89
2_6	10.00	19.20	0.88	16.70	20.20	0.80	14.60	17.60	0.81	13.50	12.20	0.96
2_7	15.30	16.30	0.84	15.10	15.60	0.79	15.10	16.50	0.83	13.00	16.90	0.92

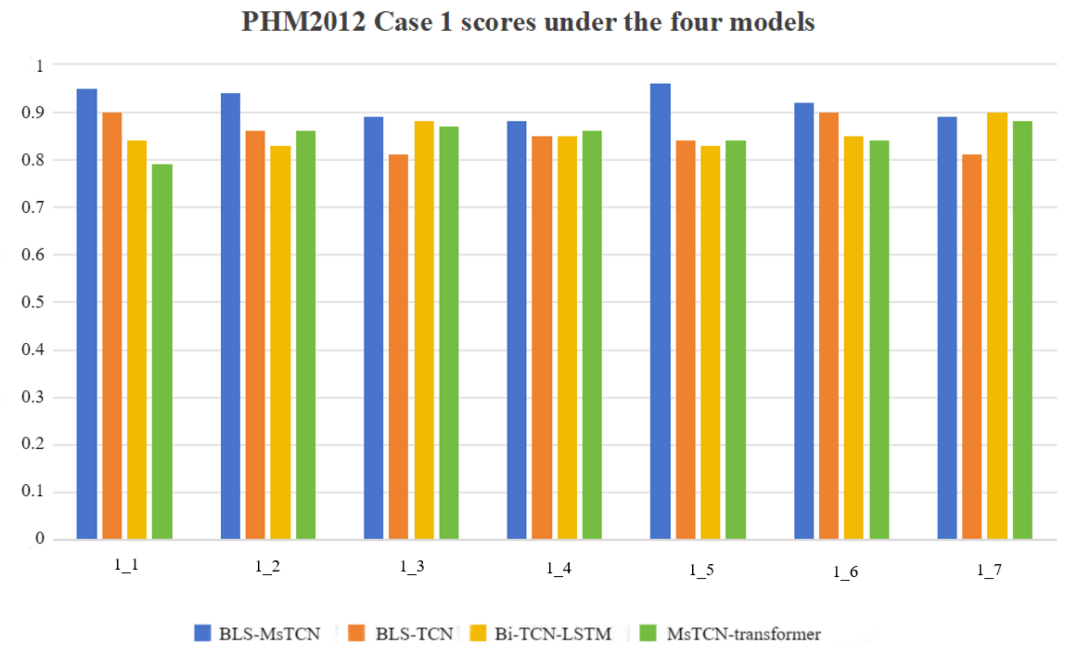

Figure 9. Scores for bearings in Condition 1 under the four models

Table 4. The parameter count and computation time of four models

Model	Parameter Count	Computation Time (s)
BLS-MsTCN	4061	1.01
BLS-TCN	4129	0.89
Bi-TCN-LSTM	3785	1.17
MsTCN-transformer	3947	1.24

Table 4 presents the parameter quantities and computation times for the four models in predicting the RUL of the bearings.

Table 4 shows that the BLS-MsTCN model processes a larger volume of data in a shorter time compared to Bi-TCN-LSTM and MsTCN-transformer. Though its parameter count is slightly lower than BLS-TCN, the BLS-MsTCN model takes 0.12 seconds longer due to the multi-scale operations it performs. However, as indicated in Figure 8, the BLS-MsTCN model provides more accurate RUL predictions, making it a better choice for tracking the actual RUL of bearings.

4 Conclusions

To address the limitations of current RUL prediction models—such as insufficient model expressiveness and difficulties in feature selection and extraction—this paper proposes a novel model combining BLS and MsTCN, named BLS-MsTCN. The BLS component enhances the model's feature representation through random feature mapping and expansion with enhanced nodes, while the MsTCN component improves prediction accuracy by extracting features at different time scales using various convolution kernel sizes. The experimental results show that the BLS-MsTCN model not only increases the model's expressiveness but also effectively captures complex nonlinear relationships within the data. In the future, research will focus on improving the model's performance across different operating conditions to maintain high accuracy for bearing RUL predictions.

Funding

This research was funded by the National Key Laboratory of Helicopter Aeromechanics Foundation, China (Grant No.: 2023-HA–LB-067-07), the Jiangsu Provincial Natural Science Foundation General Project, China (Grant No.: BK20221481).

Data Availability

The data used to support the research findings are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] A. Kumar, C. Parkash, H. Tang, and J. Xiang, "Intelligent framework for degradation monitoring, defect identification and estimation of remaining useful life (RUL) of bearing," *Adv. Eng. Inform.*, vol. 58, p. 102206, 2023. https://doi.org/10.1016/j.aei.2023.102206

[2] P. J. Gao, J. L. Wang, Z. Q. Shi, W. W. Ming, and M. Chen, "Long-term temporal attention neural network with adaptive stage division for remaining useful life prediction of rolling bearings," *Reliab. Eng. Syst. Saf.*, vol. 251, p. 110218, 2024. https://doi.org/10.1016/j.ress.2024.110218

[3] N. Gebraeel, Y. G. Lei, N. P. Li, X. S. Si, and E. Zio, "Prognostics and remaining useful life prediction of machinery: Advances, opportunities and challenges," *J. Dyn. Monit. Diagn.*, vol. 2, no. 1, pp. 1–12, 2023. https://doi.org/10.37965/jdmd.2023.148

[4] M. Hamadache, J. H. Jung, J. Park, and B. D. Youn, "A comprehensive review of artificial intelligence-based approaches for rolling element bearing PHM: Shallow and deep learning," *JMST Adv.*, vol. 1, pp. 125–151, 2019. https://doi.org/10.1007/s42791-019-0016-y

[5] A. Kumar, C. Parkash, H. Tang, and J. Xiang, "Intelligent framework for degradation monitoring, defect identification and estimation of remaining useful life (RUL) of bearing," *Adv. Eng. Inform.*, vol. 58, p. 102206, 2023. https://doi.org/10.1016/j.aei.2023.102206

[6] M. S. Rathore and S. P. Harsha, "An attention-based stacked BiLSTM framework for predicting remaining useful life of rolling bearings," *Appl. Soft Comput.*, vol. 131, p. 109765, 2022. https://doi.org/10.1016/j.asoc.2022.109765

[7] M. Pecht and J. Gu, "Physics-of-failure-based prognostics for electronic products," *Trans. Inst. Meas. Control*, vol. 31, no. 3-4, pp. 309–322, 2009. https://doi.org/10.1177/0142331208092031

[8] C. Ferreira and G. Gonçalves, "Remaining useful life prediction and challenges: A literature review on the use of machine learning methods," *J. Manuf. Syst.*, vol. 63, pp. 550–562, 2022. https://doi.org/10.1016/j.jmsy.2022.05.010

[9] O. Das, D. B. Das, and D. Birant, "Machine learning for fault analysis in rotating machinery: A comprehensive review," *Heliyon*, vol. 9, no. 6, p. e17584, 2023. https://doi.org/10.1016/j.heliyon.2023.e17584

[10] B. X. Tang, J. Q. Shu, Y. H. Xu, Y. Zheng, and D. Wang, "Multiscale similarity ensemble framework for remaining useful life prediction," *Meas.*, vol. 188, p. 110565, 2022. https://doi.org/10.1016/j.measurement.2021.110565

[11] J. H. Zhou, Y. Qin, D. L. Chen, F. Q. Liu, and Q. Qian, "Remaining useful life prediction of bearings by a new reinforced memory GRU network," *Adv. Eng. Inform.*, vol. 53, p. 101682, 2022. https://doi.org/10.1016/j.aei.2022.101682

[12] S. Schwendemann, A. Rausch, and A. Sikora, "A hybrid predictive maintenance solution for fault classification and remaining useful life estimation of bearings using low-cost sensor hardware," *Procedia Comput. Sci.*, vol. 232, pp. 128–138, 2024. https://doi.org/10.1016/j.procs.2024.01.013

[13] C. M. Qi, Z. H. Mao, and G. W. Zhang, "Rolling bearing life prediction method based on width-time convolution network," *Control Eng. China*, 2023. https://doi.org/10.14107/j.cnki.kzgc.JACA2022-98

[14] M. Gao and Y. J. Lu, "Prediction method of remaining useful life of rolling bearings based on Bi-TCN-LSTM," *Light Ind. Mach.*, vol. 2024, no. 3, pp. 66–73+79, 2024.

[15] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 1003–1012. https://doi.org/10.1109/CVPR.2017.113

[16] F. Y. Deng, Z. Chen, R. J. Hao, and S. P. Yang, "Prediction of bearing remaining useful life based on MsTCN-transformer model," *J. Vib. Shock*, vol. 2024, no. 4, pp. 279–287, 2024. https://doi.org/10.13465/j.cnki.jvs.2024.04.032

[17] C. L. P. Chen and Z. L. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, 2018. https://doi.org/10.1109/TNNLS.2017.2716952

[18] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90

[19] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference*, Amsterdam, The Netherlands, 2016, pp. 630–645. https://doi.org/10.1007/978-3-319-46493-0_38

[20] S. J. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv Preprint*, 2018. https://doi.org/10.48550/arXiv.1803.01271

[21] R. Bisoi, D. R. Dash, P. K. Dash, and L. Tripathy, "An efficient robust optimized functional link broad learning system for solar irradiance prediction," *Appl. Energy*, vol. 2022, no. 319, p. 119277, 2022. https://doi.org/10.1016/j.apenergy.2022.119277

[22] D. P. Kingma and J. Ba, "A method for stochastic optimization," *arXiv Preprint*, 2014. https://doi.org/10.48550/arXiv.1412.6980

[23] T. Hu, Y. M. Guo, L. D. Gu, Y. F. Zhou, Z. S. Zhang, and Z. T. Zhou, "Remaining useful life prediction of bearings under different working conditions using a deep feature disentanglement based transfer learning method," *Reliab. Eng. Syst. Saf.*, vol. 2022, no. 219, p. 108265, 2022. https://doi.org/10.1016/j.ress.2021.108265

[24] P. Nectoux, R. Gouriveau, K. Medjaher, E. Ramasso, B. Chebel-Morello, N. Zerhouni, and C. Varnier, "PRONOSTIA: An experimental platform for bearings accelerated degradation tests," in *IEEE International Conference on Prognostics and Health Management, IEEE Catalog Number: CPF12PHM-CDR*, 2012, pp. 1–8.