

ORIGINAL ARTICLE

International Journal of Applied Mathematics in Control Engineering

Journal homepage: <http://www.ijamce.com>

Multi-Scale Temporally Enhanced Bearing Fault Diagnosis under Variable-Load Conditions

Xinyu Chen¹ | Maoxiang Jiang¹ | Haihong Wang¹

¹School of Microelectronics, Qingdao University of Science and Technology, Qingdao 266000, China

Correspondence

School of Microelectronics, Qingdao University of Science and Technology, Qingdao 266000, China
Email: wanghaihong@qust.edu.cn

Article Info

Article history:

Received 4 April 2026

Accepted 12 June 2026

Available online 14 June 2026

Abstract

Variable-load operation can reduce bearing fault recognition accuracy when a model is trained on one load and tested on another. This study develops a diagnosis model that combines a multi-scale convolutional neural network (MSCNN) with Receptance Weighted Key Value (RWKV) temporal modeling. The model contains an MSCNN module, an RWKV module, and a classification module. MSCNN extracts local fault-impact features under different receptive fields. RWKV then models temporal correlations in the fused feature sequence. Global average pooling and a fully connected layer produce the final fault-category prediction. A ten-class task is constructed from the Case Western Reserve University (CWRU) bearing dataset. The evaluation covers same-load validation, cross-load recognition, noise perturbation, and computational cost. In the 0 hp → 3 hp large-span single-source task, the model achieves relatively high recognition accuracy and shows small variation across multiple random seeds. In the 0/1/2 hp → 3 hp multi-source task, its mean accuracy does not exceed that of MSCNN. This result indicates that the benefit of the model is limited to some large-span single-source settings rather than all cross-load tasks. Under noise levels of 12, 10, 8, 6, and 4 dB, the model maintains relatively high recognition accuracy overall. The results provide evidence for designing bearing fault diagnosis models under variable-load conditions.

KEYWORDS

Bearing Fault Diagnosis, Variable-Load Conditions, Cross-Load Diagnosis, Multi-Scale Convolution, Temporal Modeling

1 | INTRODUCTION

Bearings support rotating shafts in marine propulsion systems, electric drives, and many other machines. Their condition has a direct influence on operating safety and equipment availability. A local defect usually introduces repeated impacts into the measured vibration signal. Modulation and periodic components may also appear. These signal characteristics make vibration measurements useful for bearing monitoring and diagnosis [1, 2].

The operating load often changes during actual service. Such changes can modify the amplitude, waveform shape, and spectral content of the vibration signal. Data collected under two loads may therefore have different distributions. A classifier trained at one load can perform poorly when it is applied at another. This study focuses on this source-load training and target-load testing problem.

Earlier diagnostic methods commonly describe bearing condition through manually selected features. Time-domain indices, frequency-domain quantities, wavelet coefficients, and envelope spectra are typical choices. Their usefulness depends on both the selected feature and the analyst's experience. CNN-based methods provide another route. They learn representations directly from vibration samples and have been used in rotating-machinery diagnosis [3, 4, 5].

A single convolution scale covers only a limited temporal range. Multi-scale convolution uses several kernel sizes in parallel, so short impacts and wider waveform patterns can be represented in the same model [6, 7]. This improves local feature extraction, but the operation still acts mainly on neighboring samples. Under a load change, similar local impacts may appear with different amplitudes or shapes. Information distributed across a longer vibration segment may then become important.

Domain-transfer methods are often used when the training and test conditions differ [8, 9, 10, 11, 12, 13]. These methods usually transfer information from a source condition to a target condition. The protocol in this paper is different. Target-load samples are not included in model training. Training is performed only with source-load data, and evaluation is carried out directly on another load. The resulting accuracy reflects the model's cross-load recognition ability under this restriction.

Sequence models offer a way to use information beyond a local convolution window. RNNs, LSTMs, GRUs, and Transformers have all been applied to vibration data [6, 7, 14]. Their computational requirements and modeling forms differ. RWKV combines recurrent processing with attention-related design ideas [15, 16]. Its recurrent implementation can process a convolutional feature sequence without applying the sequence model directly to the raw signal. This structure is also compatible with the CPU-based experimental setting used here.

The model studied in this paper contains a multi-scale convolutional front end and one RWKV temporal-modeling layer. Four convolution branches extract local impact features at different receptive fields. The branch outputs are fused before entering RWKV. The evaluation uses a ten-class task built from the CWRU bearing dataset. Same-load recognition is used as a pipeline check. Cross-load tests, repeated runs with multiple seeds, noise perturbation, and computational-cost measurements are used for the main evaluation.

2 | MULTI-SCALE TEMPORALLY ENHANCED BEARING-FAULT DIAGNOSIS METHOD

2.1 | Overall Architecture

This section presents the proposed multi-scale temporally enhanced diagnostic model. Variable-load conditions can alter vibration amplitude, local waveform morphology, and impact patterns. Local features extracted only by multi-scale convolution may therefore miss temporal dependencies within continuous vibration segments. The raw vibration signal is first segmented and standardized. Parallel multi-scale convolutional branches then extract local fault-impact features under different receptive fields. The fused feature sequence is passed to the RWKV module for temporal modeling. Global average pooling and a fully connected layer produce the final fault-category prediction. The overall architecture is shown in Figure 1.

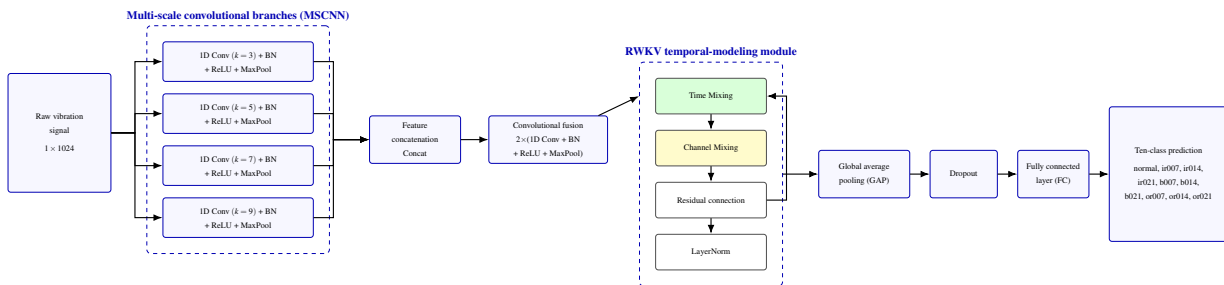


FIGURE 1 Overall architecture of the proposed multi-scale temporally enhanced bearing-fault diagnosis model.

Let the standardized input sample be

$$\mathbf{X} = [x_1, x_2, \dots, x_L] \in \mathbb{R}^{1 \times L}, \quad (1)$$

where L is the input-signal length and is set to 1024. This length retains local fault impacts and periodic waveform information within each sample while controlling computational cost.

2.2 | Data Preprocessing and Sample Construction

The vibration data used in this study come from the CWRU bearing dataset. The data source, load conditions, and fault classes are described in Section 3.1. Each continuous signal is divided into segments with a window length of 1024 and a stride of 512. Adjacent segments therefore have an overlap of 50%. This setting increases the number of samples and retains continuity between neighboring vibration segments.

Each segment keeps the fault label of its original signal. Z-score standardization is performed separately for every segment. This operation reduces the influence of amplitude differences across load conditions.

For same-load validation, the samples are split into training, validation, and test sets at ratios of 70%, 15%, and 15%. In the cross-load experiments, only source-load samples are used for model training. The trained model is then evaluated on target-load samples. Twenty percent of the source-load training samples are reserved for validation.

2.3 | Multi-Scale Convolutional Feature Extraction

Multi-scale convolution is used to describe local fault patterns over different temporal ranges [6, 7]. The convolutional front end contains four parallel one-dimensional branches. Their kernel sizes are 3, 5, 7, and 9. All four branches receive the same input signal $\mathbf{X}$. The feature produced by the branch with kernel size s is

$$\mathbf{F}_s = \text{Conv}_s(\mathbf{X}), \quad s \in \{3, 5, 7, 9\}, \quad (2)$$

where $\text{Conv}_s(\cdot)$ denotes a one-dimensional convolution with kernel size s .

The four branch outputs are combined along the channel dimension:

$$\mathbf{F}_{\text{cat}} = \text{Concat}(\mathbf{F}_3, \mathbf{F}_5, \mathbf{F}_7, \mathbf{F}_9). \quad (3)$$

A convolutional fusion operation is then applied to the concatenated features:

$$\mathbf{F}_{\text{fusion}} = \text{Conv}_f(\mathbf{F}_{\text{cat}}), \quad (4)$$

where $\text{Conv}_f(\cdot)$ denotes the convolutional fusion operation.

Max pooling with a kernel size and stride of 2 is applied after each multi-scale branch and after each of the two fusion layers. The temporal length is reduced from 1024 to 512, then to 256, and finally to 128. The resulting 128-step feature sequence is passed to the RWKV module.

2.4 | RWKV Temporal Modeling

RWKV combines recurrent computation with attention-inspired sequence modeling [15, 16]. In this work, it is applied after multi-scale feature fusion to model temporal dependencies within the convolutional feature sequence.

A single RWKV layer is adopted as a lightweight configuration to control model size and computational cost. This design is used to evaluate the effect of adding RWKV temporal modeling and is not claimed to be the optimal depth. The principal architecture parameters are listed in Table 1.

The RWKV temporal-modeling process is represented in simplified form as

$$\mathbf{H}_t = \phi(\mathbf{f}_t, \mathbf{S}_{t-1}), \quad (5)$$

TABLE 1 Architecture and parameter settings of MSCNN-RWKV.

Module	Parameter	Setting
MSCNN	Convolution kernel sizes	3, 5, 7, 9
MSCNN	Channel configuration	$4 \times 16 \rightarrow 64 \rightarrow 128$
RWKV	Number of layers	1
RWKV	Input feature dimension	128
RWKV	Channel-mixing hidden dimension	256
RWKV	Sequence/effective context length	128 time steps
RWKV	Internal dropout	0.2
Classifier	Dropout	0.2
RWKV	Time-decay parameters	Trainable
RWKV	Mixing mechanisms	Time mixing and channel mixing
RWKV	Positional encoding	No explicit positional encoding
Classifier	Feature aggregation	Global average pooling

where $\mathbf{H}_t$ denotes the temporally enhanced feature at time step t , $\mathbf{f}_t$ is the current input feature, $\mathbf{S}_{t-1}$ is the recurrent state from the previous time step, and $\phi(\cdot)$ represents the time-mixing and channel-mixing operations. Time mixing first combines the current-time-step feature $\mathbf{f}_t$ with the previous recurrent state $\mathbf{S}_{t-1}$ through trainable weights, recursively updates the historical state using trainable time-decay parameters, and regulates the influence of historical information on the current output through a gating mechanism. Channel mixing expands the feature dimension, applies a nonlinear transformation, reduces the dimension, and uses a gating mechanism to enhance information interaction across channels. Both components employ layer normalization and residual connections to improve training stability.

2.5 | Classification Output and Training Objective

The RWKV output sequence is denoted by

$$\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_T], \quad (6)$$

where T is the sequence length. To obtain a fixed-dimensional sample-level representation and reduce sensitivity to the exact location of an impact, global average pooling is applied:

$$\mathbf{g} = \frac{1}{T} \sum_{t=1}^T \mathbf{H}_t. \quad (7)$$

The classification scores and predicted class are

$$\mathbf{z} = \mathbf{W}\mathbf{g} + \mathbf{b}, \quad \hat{y} = \arg \max_c z_c, \quad (8)$$

where $\mathbf{W}$ and $\mathbf{b}$ are the classifier weights and bias, respectively, and $\mathbf{z}$ contains the class scores. Here, p_{ic} denotes the probability that the model assigns sample i to class c . This probability is obtained by applying Softmax to the class scores of the corresponding sample. During training, the classification scores are passed directly to the cross-entropy loss implementation.

All models are trained using the cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{ic} \log p_{ic}, \quad (9)$$

where N is the number of samples, $C = 10$ is the number of classes, y_{ic} is the ground-truth indicator for sample i and class c , and

p_{ic} is the predicted probability that sample i belongs to class c .

3 | EXPERIMENTAL SETUP

3.1 | Dataset and Class Construction

The CWRU bearing dataset is used for evaluation [17, 18, 19]. The 12 kHz drive-end vibration signals are selected under four loads: 0, 1, 2, and 3 hp. Here, 0 hp denotes the no-load condition and 3 hp the maximum load used in this study. Cross-load diagnosis means that the model is trained on source-load data and tested on a different target load. For example, 0 hp $\rightarrow$ 3 hp denotes training on 0 hp data and testing on 3 hp data.

A ten-class recognition task is constructed from the normal condition and three fault locations—inner race, rolling element, and outer race at the 6 o'clock position—with fault diameters of 0.007, 0.014, and 0.021 inch. The class definitions are listed in Table 2. The CWRU outer-race fault data include fault positions at 3, 6, and 12 o'clock. To ensure a consistent outer-race class definition, only data at the 6 o'clock position are selected and denoted as or@6; samples from the other angular positions are not mixed into the same class. Faults with a diameter of 0.028 inch are not used.

TABLE 2 Class definitions and sample counts for the ten-class bearing-state recognition task.

Label	Class	Description	Samples
0	normal	Normal condition	3310
1	ir007	Inner-race fault, 0.007 inch	948
2	ir014	Inner-race fault, 0.014 inch	944
3	ir021	Inner-race fault, 0.021 inch	946
4	b007	Rolling-element fault, 0.007 inch	946
5	b014	Rolling-element fault, 0.014 inch	947
6	b021	Rolling-element fault, 0.021 inch	947
7	or007	Outer-race fault at 6 o'clock, 0.007 inch	949
8	or014	Outer-race fault at 6 o'clock, 0.014 inch	946
9	or021	Outer-race fault at 6 o'clock, 0.021 inch	949

Except for the normal condition, the numbers of samples in the fault classes are similar. All compared models use the same data configuration.

3.2 | Baseline Models and Evaluation Metrics

Three models are compared: CNN, MSCNN, and MSCNN-RWKV. CNN serves as the single-scale convolutional baseline, MSCNN evaluates the effect of multi-scale convolution, and MSCNN-RWKV evaluates the additional effect of RWKV temporal modeling and its computational overhead.

Recognition accuracy is the principal evaluation metric. Test loss, macro-F1, and weighted-F1 are also reported for the same-load random-split experiment. Cross-load performance is measured by accuracy on the target-load test set. Noise robustness is evaluated by test accuracy under different signal-to-noise ratios (SNRs). For repeated experiments, the mean and standard deviation of accuracy are reported. Computational cost is characterized by total and trainable parameters, FP32 parameter-storage requirement, and average CPU inference time per sample.

3.3 | Runtime Environment and Training Parameters

The model training and testing are performed on a personal computer in a CPU environment using Python and PyTorch. The runtime environment and principal training parameters are listed in Tables 3 and 4.

TABLE 3 Runtime environment.

Item	Configuration
Operating system	Windows 11
Development tool	PyCharm
Python environment	Anaconda virtual environment
Deep-learning framework	PyTorch
Running device	CPU
Data processing/evaluation	NumPy, SciPy, scikit-learn
Visualization	Matplotlib

TABLE 4 Training parameters.

Parameter	Setting
Input length	1024
Sliding-window stride	512
Batch size	32
Epochs	10
Optimizer	Adam
Learning rate (CNN/MSCNN)	1×10^{-3}
Learning rate (MSCNN-RWKV)	5×10^{-4}
Weight decay	1×10^{-4}
Default random seed	42
Standard/cross-load sampling	Shuffled training loader
Noise-experiment training sampling	Class-balanced weighted sampling

Seed 42 is used for single-run experiments. The core cross-load tasks are additionally repeated using seeds 42, 2024, and 3407, and the mean and standard deviation of test accuracy are reported.

4 | RESULTS AND ANALYSIS

4.1 | Same-Load Random-Split Results

The same-load experiment is first conducted to verify the data-processing and training pipelines. Table 5 presents the results on the ten-class test set.

All three models achieve 100% accuracy, indicating that the data-processing and training pipelines can perform the basic classification task. However, because the experiment uses a window-level random split, adjacent overlapping segments may be assigned to different subsets. The result is therefore used only as a pipeline check and is not treated as the principal evidence of model generalization.

TABLE 5 Recognition results under the same-load random split.

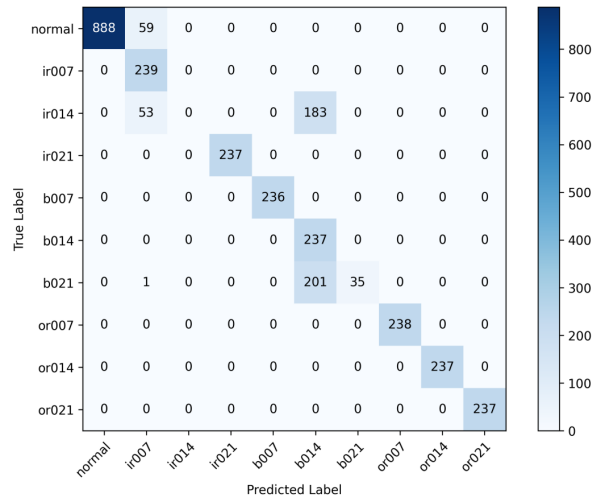
Model	Test loss	Accuracy	Macro-F1	Weighted-F1
CNN	0.0084	1.0000	1.0000	1.0000
MSCNN	0.0010	1.0000	1.0000	1.0000
MSCNN-RWKV	0.0003	1.0000	1.0000	1.0000

4.2 | Cross-Load Recognition Accuracy

Table 6 compares target-load accuracy under the cross-load protocol defined in Section 3.1, and Figure 2 shows the corresponding confusion matrix of MSCNN-RWKV for the 0 hp $\rightarrow$ 3 hp task using the default random seed.

TABLE 6 Cross-load recognition accuracy.

Cross-load task	CNN	MSCNN	MSCNN-RWKV
0 hp $\rightarrow$ 1 hp	0.9961	1.0000	1.0000
0 hp $\rightarrow$ 2 hp	0.9837	0.9661	1.0000
0 hp $\rightarrow$ 3 hp	0.8001	0.5897	0.8387
1 hp $\rightarrow$ 3 hp	0.7838	0.7991	0.8182
2 hp $\rightarrow$ 3 hp	0.8189	0.9215	0.8981
0/1/2 hp $\rightarrow$ 3 hp	0.8601	0.9133	0.9247

**FIGURE 2** Confusion matrix for the 0 hp $\rightarrow$ 3 hp cross-load task.

When the load difference is small, as in 0 hp $\rightarrow$ 1 hp, all models achieve high accuracy. For the larger 0 hp $\rightarrow$ 3 hp task, MSCNN-RWKV reaches 83.87%, exceeding CNN and MSCNN by 3.86 and 24.90 percentage points, respectively.

The classes ir007, ir021, b007, b014, or007, or014, and or021 are recognized reliably, whereas most errors are concentrated in normal, ir014, and b021. Some normal samples are predicted as ir007; ir014 is mainly confused with ir007 and b014; and many b021 samples are predicted as b014. Thus, the accuracy degradation is mainly caused by concentrated confusion among a few categories rather than a uniform degradation across all classes. A plausible explanation is that load variation changes impact amplitudes and local waveform patterns, bringing the target-load feature distributions of some categories closer together.

In the single-run 0/1/2 hp $\rightarrow$ 3 hp task, MSCNN-RWKV obtains 92.47%, compared with 86.01% for CNN and 91.33% for MSCNN. In the single-run 2 hp $\rightarrow$ 3 hp task, MSCNN obtains 92.15%, compared with 89.81% for MSCNN-RWKV. These single-run values describe task-specific outcomes but are insufficient to establish stable performance differences.

To reduce the effects of random initialization and data partitioning, the 0 hp $\rightarrow$ 3 hp and 0/1/2 hp $\rightarrow$ 3 hp tasks are repeated using three random seeds. The results are reported in Table 7.

TABLE 7 Multi-seed results for representative cross-load tasks (mean $\pm$ standard deviation).

Cross-load task	CNN	MSCNN	MSCNN-RWKV
0 hp $\rightarrow$ 3 hp	0.6327 $\pm$ 0.1386	0.5725 $\pm$ 0.0266	0.8420 $\pm$ 0.0088
0/1/2 hp $\rightarrow$ 3 hp	0.8934 $\pm$ 0.0415	0.9173 $\pm$ 0.0060	0.8959 $\pm$ 0.0330

For 0 hp $\rightarrow$ 3 hp, MSCNN-RWKV achieves the highest mean accuracy (84.20%) and the smallest standard deviation (0.0088). For 0/1/2 hp $\rightarrow$ 3 hp, MSCNN achieves the highest mean accuracy (91.73%) and the lowest variability. These results show that the gain of MSCNN-RWKV is task dependent.

4.3 | Recognition Accuracy under Noise Perturbation

Noise experiments are conducted on the random-split dataset. During training, Gaussian white noise is added to a mini-batch with probability 0.5, and the SNR is randomly selected from 20, 16, 12, 10, 8, and 6 dB; the remaining batches are kept clean. Testing is performed under clean, 12, 10, 8, 6, and 4 dB conditions. Class-balanced weighted sampling is used for the training set.

TABLE 8 Recognition accuracy under different noise levels.

Model	Clean	12 dB	10 dB	8 dB	6 dB	4 dB
CNN	1.0000	0.8975	0.7179	0.4352	0.1132	0.0777
MSCNN	0.9994	0.8998	0.7010	0.5676	0.4133	0.2066
MSCNN-RWKV	0.9989	0.9910	0.8812	0.7078	0.5315	0.3333

Accuracy decreases as the SNR is reduced for all three models. MSCNN-RWKV achieves the highest accuracy at 12, 10, 8, 6, and 4 dB, indicating better noise adaptability under this experimental setting. Nevertheless, its accuracy also drops substantially under strong noise, demonstrating a clear performance boundary.

Figures 3 and 4 are plotted from a single noise experiment without multi-seed averaging or curve smoothing; the original values at each SNR are retained.

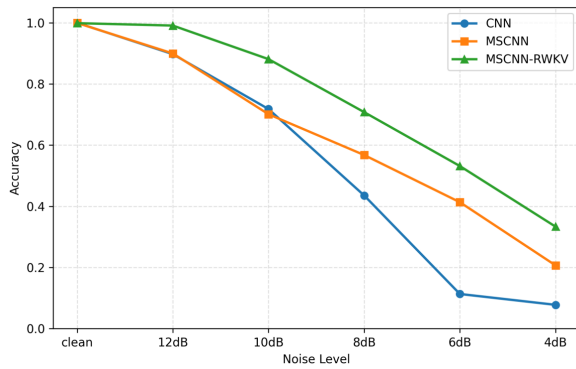


FIGURE 3 Recognition accuracy of the three models under different noise levels.

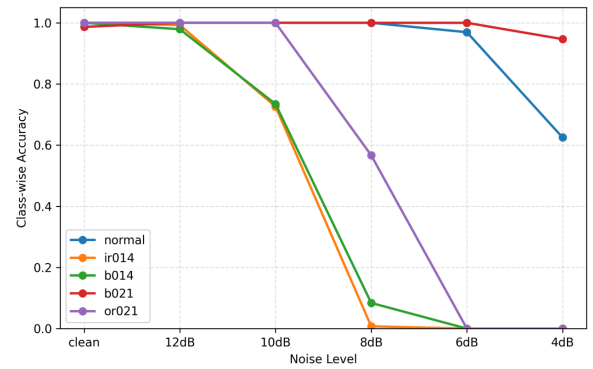


FIGURE 4 Class-wise recognition accuracy of MSCNN-RWKV under different noise levels.

The overall accuracy of each model decreases with SNR. The three models perform similarly under the clean condition, whereas MSCNN-RWKV performs best at every noisy SNR. Category-level behavior is heterogeneous: normal and b021 remain comparatively robust across several noise levels, while ir014, b014, and or021 deteriorate markedly at low SNRs.

4.4 | Computational Cost

Table 9 compares parameter counts, FP32 parameter-storage requirements, and average CPU inference time per sample.

TABLE 9 Computational cost of the compared models.

Model	Total params.	Trainable params.	FP32 storage (MB)	Time (ms)
CNN	9,802	9,802	0.0374	1.1852
MSCNN	39,306	39,306	0.1499	2.5998
MSCNN-RWKV	189,322	189,322	0.7222	14.4029

The additional multi-scale branches and RWKV module increase both parameter count and inference time. MSCNN-RWKV contains 189,322 parameters, requires approximately 0.7222 MB for FP32 parameters, and takes approximately 14.4029 ms per sample on the tested CPU. Its storage requirement remains below 1 MB and is suitable for the offline validation setting considered here. The inference times are primarily intended for relative comparison among the three models under the same software and hardware environment.

5 | DISCUSSION

The differences across cross-load settings indicate that the performance gain of MSCNN-RWKV is task dependent. In the 0 hp $\rightarrow$ 3 hp large-span single-source task, temporal modeling may improve the stability of the fused feature representation when the source and target loads differ substantially. In contrast, when multiple source loads are available, the training set already covers a broader range of operating states, and MSCNN may learn sufficiently adaptable local features; consequently, the additional gain from RWKV is less evident. The present evidence therefore supports the effectiveness of MSCNN-RWKV for some large-span single-source cross-load tasks, but it does not justify claiming superiority over convolutional baselines in all cross-load settings.

Several limitations remain. First, only one RWKV layer is used as a lightweight configuration, and different network depths have not been systematically compared. Second, global average pooling assigns equal weight to all time steps and may weaken information from a small number of critical impact instants. Third, the noise experiment has not yet been repeated under multiple random seeds. Future work should investigate RWKV depth, alternative temporal aggregation mechanisms, and feature enhancement under strong noise.

6 | CONCLUSIONS

This study develops an MSCNN-RWKV bearing-fault diagnosis model and evaluates it on a ten-class task constructed from the CWRU dataset. In the 0 hp $\rightarrow$ 3 hp large-span single-source cross-load task, the model achieves a high mean accuracy with low multi-seed variability. In the 0/1/2 hp $\rightarrow$ 3 hp multi-source task, however, it does not attain the highest mean accuracy, confirming that its advantage is task dependent. Under 12, 10, 8, 6, and 4 dB noise, MSCNN-RWKV outperforms CNN and MSCNN in overall accuracy, although strong noise still causes substantial degradation.

MSCNN-RWKV contains 189,322 parameters, requires approximately 0.72 MB of FP32 parameter storage, and takes approximately 14.40 ms per sample on the tested CPU. These results satisfy the offline validation requirements of this study. Because the evaluation is based on the public CWRU dataset, applicability to real marine propulsion systems should be further assessed using field measurements.

ACKNOWLEDGEMENTS

This research was funded by the National Natural Science Foundation of China No. 61104004, the Joint Fund Project of the Natural Science Foundation of China No. U1806201, and the Natural Science Foundation of Shandong Province No. ZR2021ZD12.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, X.C. and H.W.; methodology, X.C. and M.J.; software, X.C.; validation, X.C. and M.J.; formal analysis, X.C.; investigation, X.C.; data curation, X.C.; writing—original draft preparation, X.C.; writing—review and editing, M.J. and H.W.; visualization, X.C.; supervision, H.W.; project administration, H.W. All authors have read and agreed to the published version of the manuscript.

DATA AVAILABILITY

The bearing data used in this study are publicly available from the Case Western Reserve University Bearing Data Center [17, 18].

REFERENCES

- [1] Jardine AKS, Lin D, Banjevic D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing* 2006;20(7):1483–1510.
- [2] Randall RB, Antoni J. Rolling element bearing diagnostics—A tutorial. *Mechanical Systems and Signal Processing* 2011;25(2):485–520.
- [3] Zhao R, Yan R, Chen Z, Mao K, Wang P, Gao RX. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing* 2019;115:213–237.
- [4] Song X, Zhao W. Review of rolling bearing fault diagnosis methods based on signal feature analysis of AC motors. *Proceedings of the CSEE* 2022;42(4):1582–1595. In Chinese.
- [5] Li X, Zhang W, Ding Q, Sun JQ. Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation. *Journal of Intelligent Manufacturing* 2020;31(2):433–452.
- [6] Chen X, Zhang B, Gao D. Bearing fault diagnosis base on multi-scale CNN and LSTM model. *Journal of Intelligent Manufacturing* 2021;32(4):971–987.
- [7] Saghi T, Bustan D, Aphale SS. Bearing fault diagnosis based on multi-scale CNN and bidirectional GRU. *Vibration* 2023;6(1):11–28.
- [8] Zhang W, Li C, Peng G, Chen Y, Zhang Z. A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. *Mechanical Systems and Signal Processing* 2018;100:439–453.
- [9] Huo C, Jiang Q, Shen Y, Zhu Q, Zhang Q. Enhanced transfer learning method for rolling bearing fault diagnosis based on linear superposition network. *Engineering Applications of Artificial Intelligence* 2023;121:105970.
- [10] Tang S, Ma J, Yan Z, Zhu Y, Khoo BC. Deep transfer learning strategy in intelligent fault diagnosis of rotating machinery. *Engineering Applications of Artificial Intelligence* 2024;134:108678.
- [11] Sun T, Gao J. New fault diagnosis method for rolling bearings based on improved residual shrinkage network combined with transfer learning. *Sensors* 2024;24(17):5700.
- [12] Jiang Y, Wang H, Pan Y, Wang T, Fu H. Deep transfer fault diagnosis method for extra-large bearings based on GADF-MDSC. *Journal of Vibration and Shock* 2024;43(19):10–18. In Chinese.
- [13] Wang P, Li X, Zhang C, Han L. Transfer diagnosis method for motor rolling bearings based on angular-domain resampling and feature enhancement. *Transactions of China Electrotechnical Society* 2025;40(12):3905–3916. In Chinese.
- [14] Jin Y, Hou L, Chen Y. A time series Transformer based method for the rotating machinery fault diagnosis. *Neurocomputing* 2022;494:379–395.
- [15] Peng B, Alcaide E, Anthony Q, Albalak A, Arcadinho S, Biderman S, et al. RWKV: Reinventing RNNs for the Transformer Era. In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Singapore: Association for Computational Linguistics; 2023. p. 14048–14077.
- [16] Hou H, Yu FR. RWKV-TS: Beyond Traditional Recurrent Neural Network for Time Series Tasks. *arXiv* 2024;2401.09093. <https://arxiv.org/abs/2401.09093>, accessed 22 June 2026.
- [17] Case Western Reserve University Bearing Data Center, 12k Drive End Bearing Fault Data; n.d. <https://engineering.case.edu/bearingdatacenter/12k-drive-end-bearing-fault-data>, database online; accessed 22 June 2026.
- [18] Case Western Reserve University Bearing Data Center, Download a Data File; n.d. <https://engineering.case.edu/bearingdatacenter/download-data-file>, database online; accessed 22 June 2026.
- [19] Smith WA, Randall RB. Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study. *Mechanical Systems and Signal Processing* 2015;64–65:100–131.



Xinyu Chen was born in 2003, Hebei Province, China. She graduated from the Computer Science Department of Tangshan Normal University in 2025 with a bachelor's degree. She is currently studying for a master's degree at Qingdao University of Science and Technology. Her research interests include artificial intelligence, deep learning, and their applications in the field of marine science.



Maoxiang Jiang was born in 2001, Shandong Province, China. He received his B.S. degree in 2025 from the School of Computer Science and Technology, Qingdao University of Technology. He is currently a master's student at Qingdao University of Science and Technology. His research interests include artificial intelligence, deep learning, and their applications in the marine field.



HaiHong Wang was born in 1974, Liaoning Province, China. She received her Ph.D. in 2007 from College of Information Science and Engineering, Ocean University of China. She took her current position as an associate professor at Qingdao University of Science and Technology in 2013. Her research interests include optimal control, control theory and its application, artificial intelligence in the control field.