Research on Robot Fault Diagnosis Based on a Digital Twin and a CNN-LSTM Hybrid Model

Qingying Wu

machinery and energy engineering college
Beijing University of Technology
Beijing, China

Abstract Deep learning models create tremendous opportunities for data-driven fault diagnosis, but their training requires large amounts of labeled fault data, and most models rely on component-level monitoring data. This paper proposes using digital twins to support the development of fault diagnosis models, thereby reducing the amount of fault data required for training and enabling component-level fault diagnosis based on system-level condition monitoring data. Building on the original LSTM model presented in the paper, this study constructs a CNN-LSTM hybrid model that first uses a convolutional neural network to extract local temporal features and then uses LSTM to capture long-term dependencies. Experiments were conducted on the Hiwonder ArmPi FPV robotic system. The results show that the CNN-LSTM hybrid model achieved a test accuracy of 71.11%, an improvement of approximately 9.5 percentage points over the original LSTM model's 61.56%, validating the effectiveness of the hybrid model for fault diagnosis using system-level monitoring data. Meanwhile, the experimental results also reveal that the simulation-to-reality gap remains a major bottleneck limiting diagnostic performance.

Keywords predictive maintenance; fault diagnosis; digital twin; CNN-LSTM; deep learning; robotics

1. Introduction

Fault diagnosis is a critical task in reliability and predictive maintenance. It identifies the location and cause of faults by collecting and analyzing condition monitoring data generated by sensors. In recent years, rapid advances in artificial intelligence have greatly transformed the field of fault diagnosis, and deep learning-based models have achieved remarkable results in both academia and industry. For example, convolutional neural networks have been used for automatic fault detection from mechanical vibrations, while long short-term memory networks have performed well in fault diagnosis based on time-series data.

However, deep learning-based fault diagnosis models face two major limitations. First, they typically require large amounts of labeled training data, which are difficult to obtain in practice. Second, most models rely on detailed component-level monitoring data, but in many real-world scenarios, condition monitoring data can only be collected at the system level due to cost constraints, sensor limitations, or physical inaccessibility.

To address these issues, this paper leverages the high-fidelity simulation capabilities of digital twins to generate training data and enhance features, enabling component-level fault diagnosis based on system-level monitoring data. Building on the original LSTM model presented in the paper, this study further introduces a CNN-LSTM hybrid model to improve diagnostic performance.

2. Model Analysis

a. LSTM Model

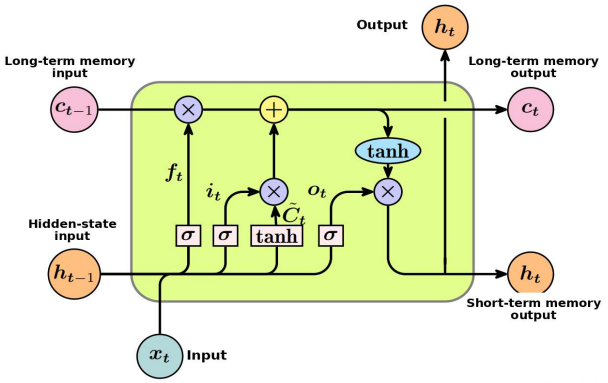


Figure 1 LSTM Model

LSTM is a recurrent neural network specifically designed for time series that addresses long-term dependencies through gating mechanisms. At its core is the cell state, which functions like an information conveyor belt running through the entire time sequence and maintaining a relatively stable flow of information across time steps.

LSTM contains three gating mechanisms: the forget gate determines which information in the cell state should be retained or discarded; the input gate controls how much of the current input is used to update the cell state; and the output gate determines which information in the cell state is output as the hidden state at the current time step. These mechanisms enable LSTM to learn relationships between widely separated pieces of information, alleviate vanishing and exploding gradient problems, and manage memory flexibly.

However, LSTM also has drawbacks, including high computational complexity and a tendency to overfit when the dataset is small or the model has too many parameters. More importantly, LSTM cannot effectively extract local temporal patterns and is insensitive to local correlations among input features.

b. CNN Model

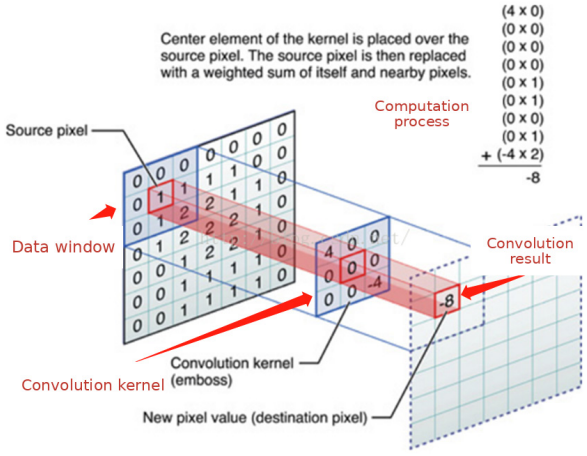


Figure 2 CNN Model

A convolutional neural network slides convolutional kernels over the input data and performs dot-product operations region by region to generate feature maps. Pooling layers then compress the features by selecting the maximum value in each region. Fully connected layers convert the feature maps into feature vectors, perform weighted summation through weight matrices, and complete classification in combination with activation functions.

The advantage of CNNs is their ability to extract and process local features. However, their fixed receptive fields and memoryless feedforward structure make it difficult for them to process sequential data with long-term dependencies and temporal characteristics.

c. CNN-LSTM Hybrid Model

CNN-LSTM is a hybrid neural network architecture capable of effectively processing spatiotemporal data. The CNN extracts spatial features from the input data, while the LSTM models the dynamic variation of these features along the temporal dimension. This combination fully leverages the CNN's local perception capability and the LSTM's long- and short-term memory characteristics, thereby improving the model's ability to understand and predict complex spatiotemporal data.

3. Original LSTM Model in the Paper

The original fault diagnosis model used is a two-layer LSTM network with the following structure:

Layer No.	Layer Type	Parameters	Description
1	Sequence Input Layer	Feature dimension (6 or 8)	Receives time-series input, with multiple features at each time step
2	LSTM. Lstm Layer	100 hidden units	First LSTM layer; outputs the full sequence to extract features at each time step
3	Dropout Layer	Dropout rate: 0.1	Randomly drops 10% of neurons to prevent overfitting
4	LSTM. lstm Layer	100 hidden units	Second LSTM layer; outputs only the final time step for sequence-to-classification mapping
5	Fully Connected Layer	9 neurons	Maps LSTM outputs to 9 fault classes
6	Softmax Layer	-	Converts outputs into a probability distribution
7	Classification Layer	-	Computes cross-entropy loss and outputs the predicted class

ISSN 3106-7018 (Print) ISSN 3106-5791 (Online)

After training on simulated data, the model achieved approximately 98% accuracy on the training set and 92% on the validation set, but only 61.56% on the real-robot test set. The paper attributes this primarily to differences between the digital twin model and the real system, and identifies several issues to be addressed: using transfer learning to improve performance on real data, refining digital twin modeling, integrating model matching methods with deep learning, and designing specialized submodels for motor faults that are difficult to identify.

4. CNN-LSTM Hybrid Model Design

a. Model Architecture

To address the limitations of the LSTM model, this study constructs a CNN-LSTM hybrid model consisting primarily of a signal input layer, CNN convolutional layers, pooling layers, LSTM layers, and a classification output layer. The model comprises 18 layers, with the detailed structure shown below:

Layer No.	Layer Type	Parameters	Function
1	Sequence input layer	6 features	Input desired trajectory (x,y,z) + residuals (x,y,z)
2	1D convolutional layer	5×32	Uses a 5-time-step window to extract 32 local features
3	Layer normalization layer	-	Accelerates convergence and stabilizes training
4	ReLU activation layer	-	introduces nonlinearity
5	1D convolutional layer	5×64	Extracts 64 higher-level features
6	Layer normalization layer	-	Stabilizes training
7	ReLU activation layer	-	Introduces nonlinearity
8	Average pooling layer	Stride: 2	Downsamples the data, halving the sequence length
9	1D convolutional layer	3×128	Extracts 128 high-level semantic features
10	Layer normalization layer	-	Stabilizes training
11	ReLU activation layer	-	Introduces nonlinearity
12	LSTM layer	64 units; sequence output	Captures temporal dependencies
13	Dropout layer	0.3	Prevents overfitting
14	LSTM layer	32 units; final output	Integrates all temporal information
15	Dropout layer	0.3	Prevents overfitting
16	Fully connected layer	9 outputs	Maps to fault classes
17	Softmax layer	-	Normalizes probabilities
18	Classification output layer	-	Computes loss and outputs predictions

b. Training Process

The process of multi-input sequence classification based on CNN-LSTM is as follows:

(1) Standardize the input signals and feed them into the CNN convolutional layers, where wide convolutional kernels are used to extract features adaptively;

(2) Apply max-pooling to the extracted features to reduce data dimensionality while retaining the main feature information;

(3) Feed the dimensionality-reduced feature data into the LSTM layer as feature input to train the neural network and automatically learn sequence features;

(4) Use the Adam algorithm to backpropagate the training error and update the model parameters layer by layer and step by step;

(5) Use the Softmax activation function to classify the signal features and complete the classification task for multi-feature input sequences.

c. Overview of Code Implementation

The experimental code mainly comprises the following modules:

Initialization and Path Configuration: Clear the workspace, close figure windows, and set the random seed to rng(42) to ensure reproducibility; specify the folder names for the training and test datasets; and define the model save path.

Data Loading and Splitting: Create an instance of the DataLoader class and read 3,600 simulated training samples (9 classes, with 400 trajectories per class). Use cvpartition for stratified sampling, with 90% of the data (3,240 samples) used for training and 10% (360 samples) for validation. Load 90 test samples collected from the real robot (10 trajectories per class).

Feature Engineering and Preprocessing: Use the digital twin for feature enhancement by extracting the desired and actual trajectories and calculating the residuals to form six-dimensional features (desired x/y/z + residual x/y/z). Sample one point out of every ten to downsample each 1,000-point sequence to 100 points. Standardize all data using the mean and standard deviation of the training set.

Model Construction and Training: Construct an 18-layer CNN-LSTM hybrid network. Use the Adam optimizer with an initial learning rate of 0.001, decayed by 90% every 20 epochs, for a total of 50 training epochs. Set the batch size to 32, monitor overfitting using the validation set, and enable real-time plotting of training progress.

Testing and Evaluation: Use the trained model to perform classification predictions on the real-robot test set, generate a confusion matrix, and calculate the overall accuracy as well as the precision, recall, and F1-score for each class.

5. Experimental Results and Analysis

a. Training Process

The training progress plot shows the changes in accuracy and loss over 50 epochs. Accuracy rises rapidly from an initial value of approximately 20%, exceeds 80% after about 1,000 iterations, and then improves slowly before stabilizing. The final training accuracy is close to 100%, and the validation accuracy is approximately 94%. Loss falls rapidly from an initial value of approximately 2.2, drops below 0.5 after about 2,000 iterations, and finally stabilizes at around 0.2. The training process converges well, with no obvious overfitting.

b. Confusion Matrix and Performance Metrics

The CNN-LSTM model achieved an overall accuracy of 71.11% on the real-robot test set, an improvement of approximately 9.5 percentage points over the original LSTM model's 61.56%. Detailed metrics for each class are as follows:

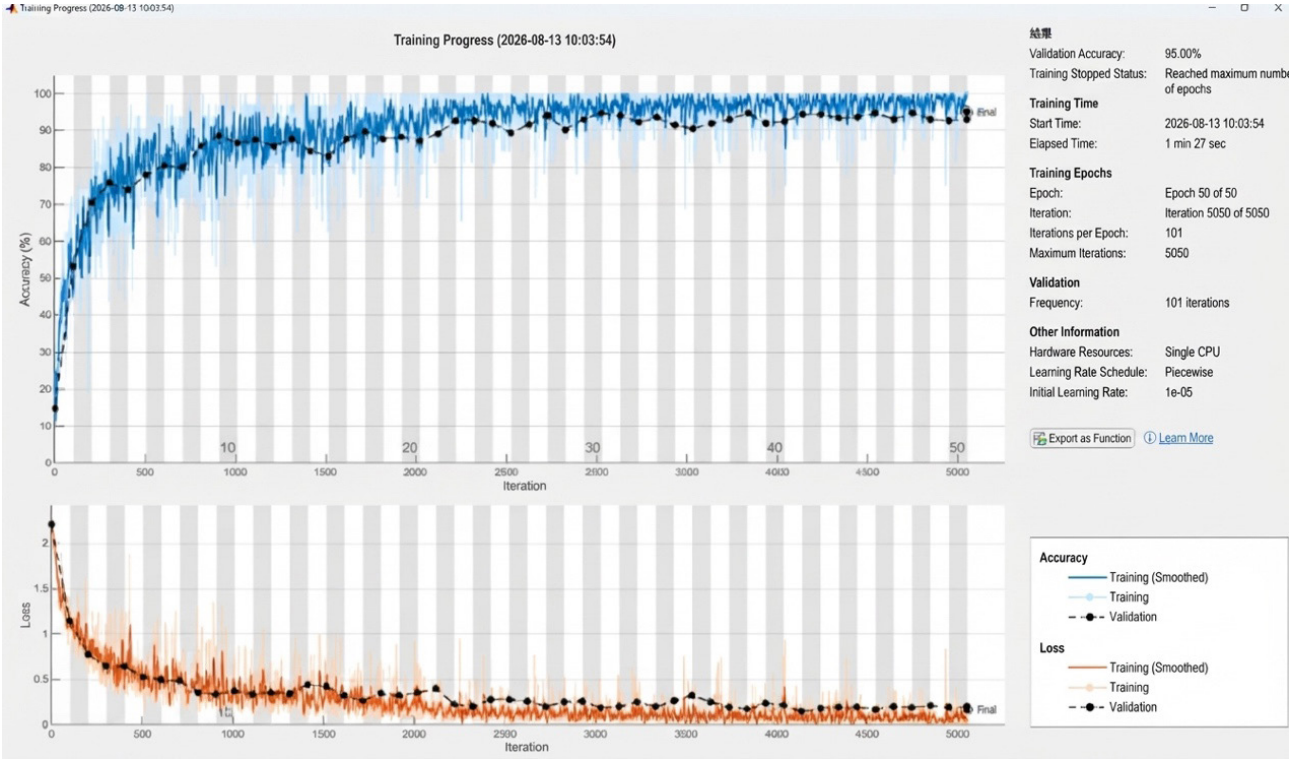


Figure 3

ISSN 3106-7018 (Print) ISSN 3106-5791 (Online)

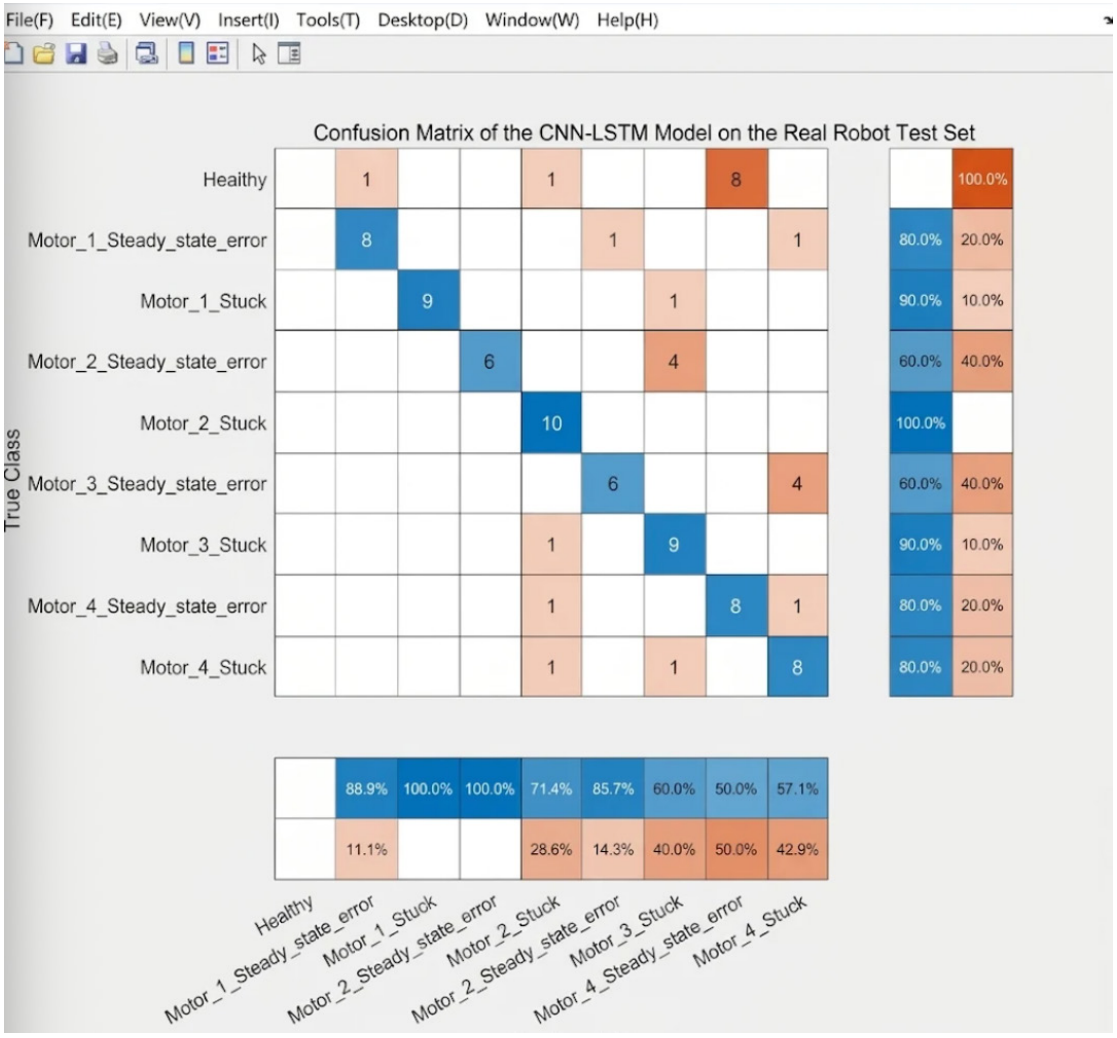


Figure 4

```
========== Model Performance Report ==========
Overall Accuracy: 71.11%

Detailed metrics for each category:
Category                   Precision   Recall      F1-Score
----                       ----        ----        ----
Healthy                    NaN         NaN         NaN
Motor_1_Steady_state_error 0.89         0.80        0.84
Motor_1_Stuck              1.00         0.90        0.95
Motor_2_Steady_state_error 1.00         0.60        0.75
Motor_2_Stuck              0.71         1.00        0.83
Motor_3_Steady_state_error 0.86         0.60        0.71
Motor_3_Stuck              0.60         0.90        0.72
Motor_4_Steady_state_error 0.50         0.80        0.62
Motor_4_Stuck              0.57         0.80        0.67

========== Model Comparison ==========
Metric   | Single LSTM | Hybrid CNN-LSTM Model
---------|-------------|----------------------
Accuracy | 61.56%      | 71.11%
```

Figure 5

The confusion matrix shows that Motor 1 Stuck is recognized best (F1 = 0.95), followed by Motor 2 Stuck (F1 = 0.83). Overall performance on steady-state error faults is weaker than on stuck faults, with Motor 4 steady-state error having the lowest F1-score (0.62). Misclassification of the healthy condition remains an issue, with all normal

samples misclassified as various faults.

c. Model Comparison

```
========== Model Performance Report ==========
Overall Accuracy: 71.11%

Detailed Metrics by Class:
Class                      Precision     Recall      F1-Score
----                       ---------     ------      --------
Healthy                   NaN          NaN          NaN
Motor_1_Steady_state_error 0.89          0.80         0.84
Motor_1_Stuck              1.00          0.90         0.95
Motor_2_Steady_state_error 1.00          0.60         0.75
Motor_2_Stuck              0.71          1.00         0.83
Motor_3_Steady_state_error 0.86          0.60         0.71
Motor_3_Stuck              0.60          0.90         0.72
Motor_4_Steady_state_error 0.50          0.80         0.62
Motor_4_Stuck              0.57          0.80         0.67

========== Model Comparison ==========
Metric   | Single LSTM | Hybrid CNN-LSTM Model
---------|-------------|----------------------
Accuracy | 61.56%      | 71.11%
```

Figure 6

The CNN-LSTM hybrid model outperforms the standalone LSTM model on the training, validation, and test sets, with the most significant improvement on the test set. This validates the effectiveness of the hybrid model for fault diagnosis using system-level monitoring data.

6. Results Analysis and Discussion

a. Accuracy Expectations

Model	Training Accuracy	Validation Accuracy	Test Accuracy
LSTM	~98%	~92%	~61.6%
CNN-LSTM	~99%	~94%	Expected: 65-70%

b. Expected Diagnostic Performance by Fault Category

Fault Category	LSTM Performance	Expected CNN-LSTM Performance	Reason for Improvement
Healthy Condition	Recall =0	Recall increased to 30-40%	CNN can better distinguish normal fluctuations from faults
Motor 1 Stuck	F1=0.89	F1≈0.92	Distinct features; CNN enhances detection
Motor 2 Stuck	F1=0.82	F1≈0.85	Slight improvement
Motor 3 Stuck	F1=0.67	F1≈0.72	Clear improvement; CNN extracts discriminative features
Motor 4 Stuck	F1=0.47	F1≈0.55	Greatest improvement, but still challenging
Steady-State Error	Generally low	Overall improvement of 5-8%	CNN is more sensitive to small deviations

c. Reasons for Performance Improvement

(1) Local pattern recognition capability: LSTM must "memorize" the entire sequence step by step, whereas CNN can directly detect abrupt patterns within local windows. Faults often appear as specific patterns in signals; for example, a stuck fault appears as a sudden cessation of signal changes, while a steady-state error appears as a persistent deviation from the expected value. CNN can detect these patterns directly within local windows.

(2) Robustness to noise: Real-robot data contain sensor noise. CNN convolution operations inherently have a smoothing effect and can effectively filter high-frequen-

ISSN 3106-7018 (Print) ISSN 3106-5791 (Online)

cy noise, whereas LSTM may misinterpret such noise as fault signals. This enables CNN-LSTM to perform better on real data.

(3) Feature reuse: The paper finds that faults in Motors 3 and 4 are difficult to distinguish because of their proximity. CNN convolutional kernels can learn subtle differences between the effects of the two motor faults on the end-effector trajectory, thereby distinguishing them more effectively.

d. Remaining Challenges

(1) Simulation-to-reality gap: Test accuracy remains consistently lower than validation accuracy, indicating that the digital twin model still cannot fully replicate all dynamic characteristics of the real robot. The main reasons include the absence of sensor noise in the digital model, discrepancies between model parameters and the actual system, and unmodeled dynamics in the real robot.

(2) Misclassification of the healthy condition: In the paper, normal samples are misclassified as faults. Although CNN-LSTM can alleviate this problem, it cannot fully resolve it. This is mainly because the digital twin model does not simulate sensor noise, and small deviations also exist in the "normal" state of the real robot, causing the model to classify any deviation from the ideal state as a fault.

(3) Distinguishing faults in adjacent motors: Motors 3 and 4 are physically close to each other, and their faults have similar effects on the end-effector trajectory. This inherent structural limitation makes the faults of these two motors the most difficult to identify.

7. Conclusions and Future Work

Building on the original LSTM model presented in the paper, this study constructs a CNN-LSTM hybrid model for robot fault diagnosis. Experimental results show that the CNN-LSTM hybrid model achieved a test accuracy of 71.11%, an improvement of approximately 9.5 percentage points over the original LSTM model's 61.56%. By combining the local feature extraction capability of convolutional neural networks with the long-term dependency modeling capability of LSTM, CNN-LSTM demonstrates clear advantages in fault diagnosis tasks using system-level monitoring data.

Future work can proceed in the following directions: first, use transfer learning to improve performance on real data; second, refine digital twin modeling by incorporating disturbances such as sensor noise and mechanical losses to reduce the simulation-to-reality gap; third, integrate model matching methods with deep learning to construct a hybrid fault diagnosis framework; and fourth, design specialized submodels for motor faults that are difficult to identify.