Received 22 August 2022, accepted 6 September 2022, date of publication 15 September 2022, date of current version 23 September 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3206954*

RESEARCH ARTICLE

# An Intelligent IoT Sensing System for Rail Vehicle Running States Based on TinyML

**SHAOZE ZHOU, YONGKANG DU, BINGZHI CHEN, YONGHUA LI, AND XINGSEN LUAN**
School of Rolling Stock Engineering, Dalian Jiaotong University, Dalian 116028, China

Corresponding author: Shaoze Zhou (zhoushaoze2003@qq.com)

This work was supported in part by the China National Natural Science Foundation under Contract 51875073, and in part by the Dalian Science and Technology Innovation Fund Plan Project under Contract 2019J11CY017.

**ABSTRACT** Real-time identification of the running state is one of the key technologies for a smart rail vehicle. However, it is a challenge to accurately real-time sense the complex running states of the rail vehicle on an Internet-of-Things (IoT) edge device. Traditional systems usually upload a large amount of real-time data from the vehicle to the cloud for identification, which is laborious and inefficient. In this paper, an intelligent identification method for rail vehicle running state is proposed based on Tiny Machine Learning (TinyML) technology, and an IoT system is developed with small size and low energy consumption. The system uses a Micro-Electro-Mechanical System (MEMS) sensor to collect acceleration data for machine learning training. A neural network model for recognizing the running state of rail vehicles is built and trained by defining a machine learning running state classification model. The trained recognition model is deployed to the IoT edge device at the vehicle side, and an offset time window method is utilized for real-time state sensing. In addition, the sensing results are uploaded to the IoT server for visualization. The experiments on the subway vehicle showed that the system could identify six complex running states in real-time with over 99% accuracy using only one IoT microcontroller. The model with three axes converges faster than the model with one. The model recognition accuracy remained above 98% and 95%, under different installation positions on the rail vehicle and the zero-drift phenomenon of the MEMS acceleration sensor, respectively. The presented method and system can also be extended to edge-aware applications of equipment such as automobiles and ships.

**INDEX TERMS** TinyML, IoT, running state, smart rail vehicle, artificial neural network.

## I. INTRODUCTION

Enabling rail vehicles to have self-awareness through sensors with low energy consumption is a challenge as the rail vehicle industry is rapidly developing towards intelligence and low carbonization. Real-time identification of the running state is the key technology for realizing the self-awareness of smart rail vehicles. However, identifying various complex running states with low energy consumption is a difficult task because the computing power of edge devices at the vehicle side is low and the state data monitored by sensors is complex.

Currently, state monitoring is mostly focused on the automotive domain in the existing studies [1], [2], [3], [4], [5], [6]. Most state monitoring systems of these researches are achieved by the real-time acquisition of vehicle-side ECU sensors or GPS data which is sent to the cloud when the vehicle is moving. States results are fed back to the vehicle after using complex algorithms in the cloud to identify the various states of the vehicle. This type of system is not suitable for smart rail vehicles because of its disadvantages and limitations such as high cost, high power consumption, large size, poor real-time performance, and complicated structure. In contrast, the edge state sensing system with small size, low cost, and low power consumption has high practical and economic value for intelligent rail vehicles. For example, the system can record the running states of each vehicle, and provide accurate and quantitative data for structural health monitoring and condition-based maintenance of rail vehicles. In addition, this type of system can provide long-term tracking and monitoring services for rail vehicles (such as

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun Steven Li.

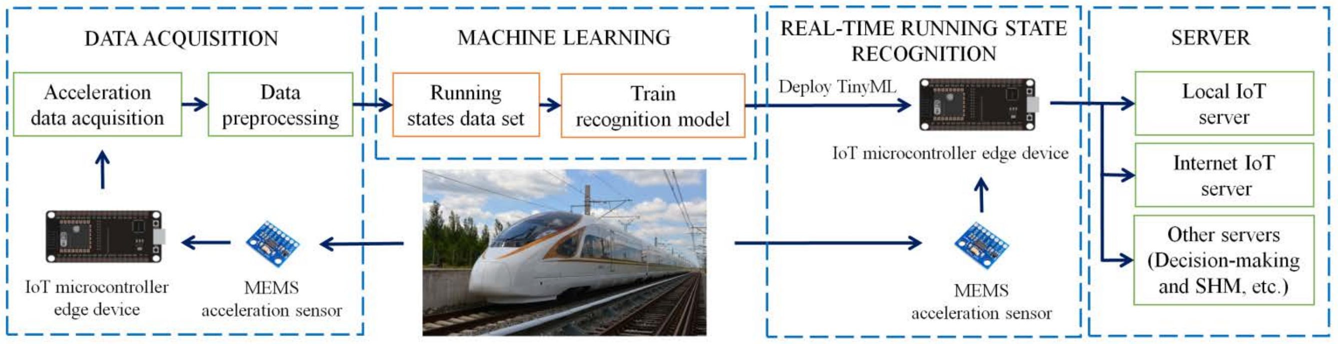


**FIGURE 1.** Schematic diagram of the Intelligent IoT sensing system for rail vehicle running states based on TinyML.

rail wagons) only using solar power. Therefore, carrying out the research on edge identification technology will have important theoretical research and engineering application significance to the design, manufacture, and maintenance of smart rail vehicles.

With the development of artificial intelligence, some researchers have used machine learning in automobile vehicle intelligence [7], [8]. These studies include autonomous vehicle driving [9], [10], intelligent vehicle classification [11], and intelligent vehicle monitoring [12]. Some researchers [13], [14] considered vehicle running safety on vehicle intelligent classification, which classified and identified the vehicle running states. Some researchers have studied the state recognition methods for various equipment. Lan *et al.* [15] represented a diagnosis strategy based on operating conditions and pressure pulsation of the turbine in order to effectively monitor the operating state of hydraulic turbines. Lu *et al.* [16] proposed a GA-CNN model to achieve automatic recognition of the rolling bearing running state.

A number of studies already had been conducted to apply machine learning in combination with microcontrollers. Adhitya *et al.* [17] used microcontrollers that read two inputs from temperature and humidity sensor data and outputted two neurons to control two actuators by a machine learning model algorithm. Chand *et al.* [18] designed a system based on monitoring data of ultrasonic sensors and used machine learning algorithms to analyze the data and analyze personnel behavior. Zhang *et al.* [19] investigated the deployment of convolutional neural networks on an embedded platform to implement a target detector. In recent years, edge machine learning has developed rapidly and Tiny Machine Learning (TinyML) technology becomes an increasing area [20], [21], [22]. Prado *et al.* [23] proposed a TinyML-based method to apply a machine vision algorithm to mini-car automatic driving.

So far, the above research literature has studied the creation or comparative optimization of automotive and device-specific machine learning models. However, in the trail vehicle field, research on Internet-of-Things (IoT) edge machine learning models is still lacking. In particular, research on intelligent state recognition based on TinyML technology for rail vehicles has not been reported. In addition, the research on which sensor data (such as speed, acceleration, or gyroscope data) can accurately determine the running state of rail vehicles should be conducted. Since the state data recorded by rail vehicle operation has its uniqueness, it is essential to analyze its characteristics, and study state recognition methods and machine learning models.

This paper proposes a novel intelligent monitoring and identification method for rail vehicles' running states based on TinyML technology. An IoT intelligent sensing system was developed with a small size, low cost, and low energy consumption. Only a miniature IoT edge device was used to identify multiple complex running states. Taking subway vehicle monitoring as an example, the system realized six kinds of real-time running state identification effectively and transmitted the results to the IoT server for visualization. In order to evaluate the system's reliability and effectiveness, experiments were also conducted with different numbers of acceleration axes, different deployment positions, and zero-drift effects.

## II. SYSTEM ARCHITECTURE

The Intelligent IoT sensing system for rail vehicle running states based on TinyML consists of a data acquisition module, a machine learning module, a real-time state recognition module, and a server module. The system architecture diagram is shown in Figure 1. In this paper, acceleration signals are tried to identify and classify the running state of vehicles.

### A. THE DATA ACQUISITION MODULE

The data acquisition module mainly collects acceleration data for machine learning training. Figure 2 illustrates the module hardware consists of an ESP32 microcontroller, a low-cost Micro-Electro-Mechanical System (MEMS) acceleration ADXL345 sensor, a clock module, an SD card memory module, a battery, and a circuit board. The module size is only 90mm $\times$ 28 mm $\times$ 38 mm and the power consumption is only 7.26E-3W. When the system is initialized, the ESP32 system time is updated from the Internet or local area network NTP server. The acceleration value and time stamp are recorded on the SD card in the module.

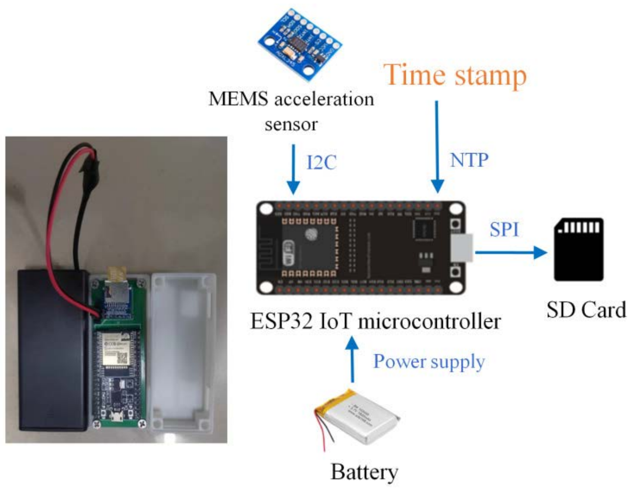


**FIGURE 2.** **The hardware of the data acquisition or the real-time running state recognition module.**

### B. THE MACHINE LEARNING MODULE

The learning module is constructed by an open-source framework such as Tensorflow. The machine learning framework is built with the artificial neural network, and the collected acceleration data is used for training. After training, a model file to identify the running state features is obtained. The model is converted into TinyML model format and deployed to the IoT microcontroller.

### C. THE REAL-TIME RUNNING STATE RECOGNITION MODULE

The real-time state monitoring module has the same hardware configuration as the data acquisition module. But the internal software and functions are different. After deploying the TinyML model file to the IoT microcontroller on the vehicle side, the acceleration values are read from the sensor and the vehicle's running state is identified by the model in real-time.

### D. THE SERVER MODULE

Using the programmability of the edge device ESP32 microcontroller chip, recognition results are sent to the local or remote cloud IoT servers, and other data servers through the WebSocket protocol. The real-time results are accessed by browsers for terminal users.

## III. METHODOLOGICAL STEPS

The overall workflow diagram of the Intelligent IoT sensing system for rail vehicle running states based on TinyML is displayed in Figure 3. The specific steps of the proposed method are as follows.

Step 1: Collect the acceleration data of the complete start-stop cycles. The data acquisition module collects sufficient X, Y, and Z-axis acceleration data of complete start-stop cycles from the rail vehicle.

Step 2: Preprocess the data that has been collected. To generate various running state sample sets for machine training, the collected data is segmented, normalized, dimensionally transformed, and labeled with defined values.

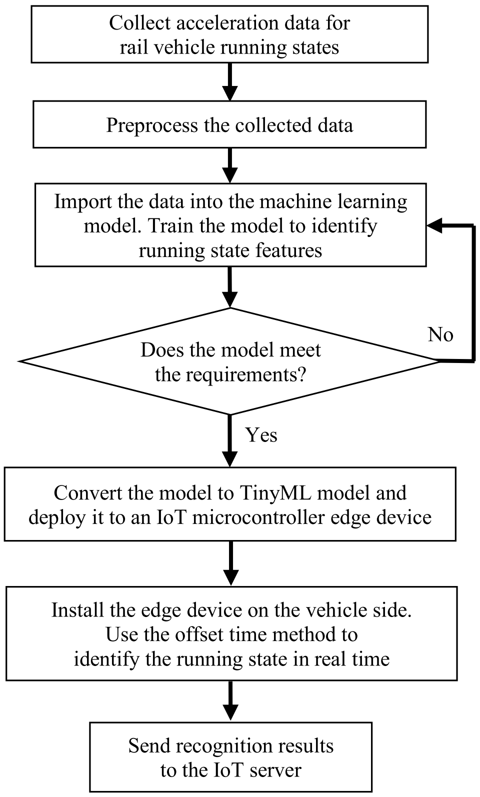

Collect acceleration data for rail vehicle running states
Preprocess the collected data
Import the data into the machine learning model. Train the model to identify running state features
Does the model meet the requirements?
No
Yes
Convert the model to TinyML model and deploy it to an IoT microcontroller edge device
Install the edge device on the vehicle side. Use the offset time method to identify the running state in real time
Send recognition results to the IoT server

**FIGURE 3.** **The workflow chart of the Intelligent IoT sensing system for rail vehicle running states based on TinyML.**

Step 3: Build a machine learning artificial neural network, and generate a model for identifying the running state features. The running state sample set is expanded into one-dimensional data and imported into the input layer. Furthermore, the artificial neural network is constructed by setting proper parameters such as loss function, activation function, and training rounds. After training, a model is generated to identify the running state features. The model prediction accuracy is used as the basis for assessing the model quality.

Step 4: Identify the running state in real time. the identification of the running state feature model is converted and deployed to the IoT microcontroller based on TinyML technology. The microcontroller monitors the acceleration in real time as inputs to the TinyML model for inference recognition. The recognition results are sent by the microcontroller to the IoT server.

The key technologies of the above steps are addressed in detail below.

### A. PREPROCESSING OF RUNNING STATE ACCELERATION DATA

Step 2 is the preprocessing of acceleration data, including data segmentation, normalization, dimensional transformation, and label definition.

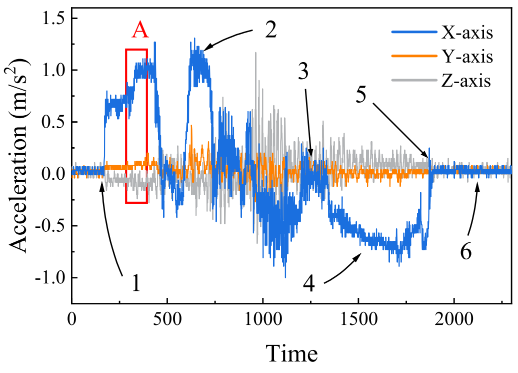


**FIGURE 4.** **The typical one start-stop cycle of the rail vehicle.**

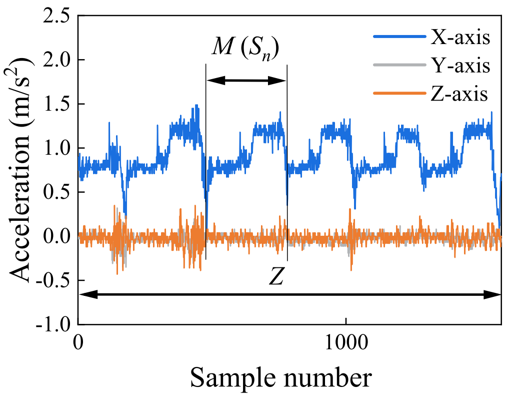


**FIGURE 5.** **The example of the training set.**

### 1) DATA SEGMENTATION

Figure 4 shows the acceleration values of a typical rail vehicle in one complete start-stop cycle. The X-axis is the longitudinal direction of the vehicle, while the Y-axis and Z-axis are the lateral direction and vertical direction, respectively. The vehicle moved along the X-axis direction. The figure presents that the X-axis acceleration values of the rail vehicle are positive when accelerating, negative when decelerating, and steady and close to zero when stationary. According to the acceleration values, the running states of some key positions in the figure are presented as follows: 1) starting, 2) accelerating, 3) moving, 4) decelerating, 5) stopping, and 6) stationary. These running state accelerations have distinct characteristics. Taking the X-axis acceleration as an example, the acceleration values of the starting state is from small to large. The acceleration values of the moving state have a wide range of fluctuation around the zero axes. In the decelerating state, the acceleration values are negative.

The acceleration data is divided into sample sections of different running state types, and each sample section has the same time interval $T_g$, such as 2 seconds. Each sample contains three columns and $M$ rows of data for X, Y, and Z-axis accelerations. As shown in Figure 4, section A is the accelerating type sample, which has obvious acceleration characteristics. The same types of samples are segmented, extracted, and combined to form a sample set of $Z$ rows, as in Figure 5. Then $Z$ equals $M \times N$, where $N$ is the number of samples. $S_n(n = 1,2,3\ldots)$ is defined as the name of each sample. Add the $Z$ rows sample set with the state label name $T_n(n = 1,2,3\ldots)$. For example, $T_1$ is the accelerating sample set and $T_2$ is the decelerating sample set. To ensure the reliability of the machine learning model and to determine whether the training degree is overfitting or underfitting, the neural network dataset is divided into a training set and a verification set. The ratio of the training set to the validation set is 4:1.

### 2) NORMALIZATION

In the classification algorithm, the classification performance of the learning algorithm will be reduced when the attributes and different feature information of multiple description objects are fused [24], [25], [26]. To avoid this situation, Eq. (1) normalization is used in machine learning tasks to ensure that each feature magnitude has an equal contribution when fed to the classifier and to eliminate the effect of odd data. After normalization, the data is distributed between 0 and 1, and the optimization of the data is faster and the accuracy is higher during training.

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

where $x_i'$ is normalized data, while $x_i$ is collected data, and $x_{max}$ and $x_{min}$ represent the maximum value and the minimum value of the $M$ rows of data, respectively.

### 3) DIMENSIONAL TRANSFORMATION

To simplify the model training process, the three axes data of the $S_n$ group data are connected end to end to form a one-dimensional neural array $L$ of $3 \times M$ values. This array is used as the input of the neural network.

### 4) LABEL VALUE DEFINITION

Since the machine learning model is based on a multi-classification learning algorithm, one-hot code [27] is adopted to identify classification information for different running state data. For example, to identify the three running states of rail vehicles in accelerating, decelerating, and moving, the three labels are defined as 0, 1, and 2. The one-hot encoding forms are [1,0,0], [0,1,0], and [0,0,1]. The reason for using one-hot coding is that it extends the value of discrete features to Euclidean space.

## *B. TINYML RUNNING STATE RECOGNITION MODEL TRAINING*

In Step 3, running state recognition is implemented using a classification machine algorithm, which is modeled and trained using the sequence model provided by the Keras library of artificial intelligence framework, as displayed in Figure 6. Relevant parameters such as loss function, activation function, and optimization algorithm are adjusted according to the model training condition. The sample set

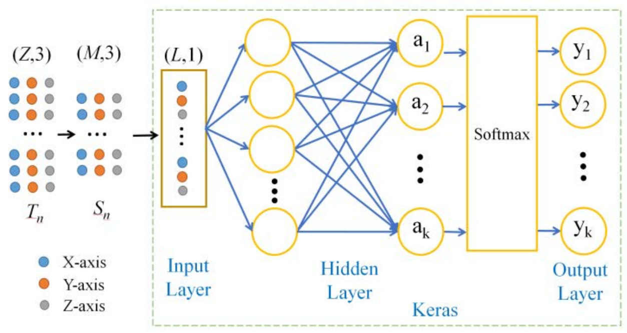


**FIGURE 6.** The neural network structure of the running state recognition model.

processed in subsection A is utilized as the input of the neural network. If the model needs to recognize $K$ types of running states, the number of output layer units is set to $K$.

The vehicle running state model belongs to the multi-classification machine learning algorithm. The Softmax function is chosen for the activation function from the hidden layer to the output layer of this model. Its expression is presented in Eq. (2).

$$softmax\ (x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2}$$

The numerator in equation (4) is the exponential function of the signal $x_i$ received by the neurons in the output layer, and the denominator sums the exponential functions of all the output results $x_j$. Each output result of the neural network is presented in the form of a probability distribution.

In a multi-classification problem based on the Softmax classifier, the loss function is used to define the effect of the neural network model and the goal of optimization. The cross-entropy loss function is adopted in this study. The formula for calculating the cross entropy from the sample output of the machine learning model with the actual label values is shown in Eq. (3).

$$H_{y'}(y) = -\sum_i y_i' \log(y_i) \tag{3}$$

where $y_i$ is the probability of each result output by the model, and $y_i'$ represents the category corresponding to the one-hot code.

The Adam optimization algorithm, which combines the features of the Adagrad and RMSprop algorithms, is adopted in the machine learning of the system.

The labeled samples from subsection A are used for machine learning to determine the ideal values of all weights and deviations. When constructing the model, the parameters of machine learning need to be determined. Among them, the numbers of neurons in the input layer and the output layer are determined according to the number of identified running states. Other parameters to be determined are the number of hidden layers, the number of neurons in the hidden layer, the number of training rounds, etc. When the model evaluation

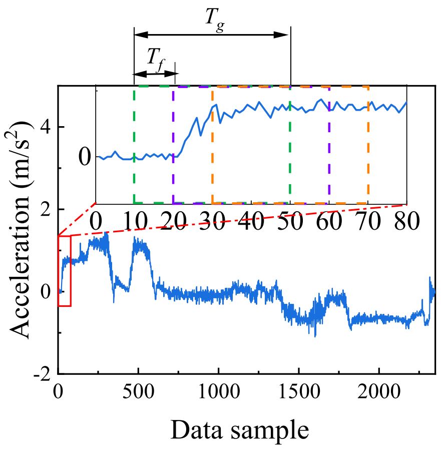


**FIGURE 7.** Schematic diagram of vehicle running state data time offset window.

is completed, the optimal values are selected as the final parameter values. The number of neurons in the hidden layer is decided by Eq. (4).

$$N = \sqrt{N_i(N_o + 1) + 1} \tag{4}$$

where $N$ is the number of neurons in the hidden layer, and $N_i$ and $N_o$ represent the number of neurons in the input and output layers, respectively. If the training result is not satisfactory, adjust the model parameters and retrain.

## C. PERFORMANCE EVALUATION METHOD FOR RUNNING STATE RECOGNITION MODELS

In the training task in Step 3, the machine learning models obtained differ due to the selected parameters. Ideally, the generalization error of all candidate models should be compared to find the optimal model. However, since the generalization error cannot be obtained directly, this paper evaluates the model by its accuracy.

For the classification task, the sample error rate and accuracy rate are defined as (5) and (6).

$$E(f;D) = \frac{1}{m}\sum_{i=1}^{m}\prod(f(x_i) \neq y_i) \tag{5}$$

$$acc(f;D) = \frac{1}{m}\sum_{i=1}^{m}\prod(f(x_i) = y_i) = 1 - E(f;D) \tag{6}$$

where $D$ is the sample set and $m$ represents the total number of samples. In addition to the accuracy, the loss function also is used to evaluate the recognition model.

## D. DEPLOYMENT AND REAL-TIME RECOGNITION OF TINYML MODEL

After completing the recognition model training in Step 4, the machine learning model file is compressed as a TinyML model and deployed to the MCU of the IoT edge device in the vehicle. The edge device sensor monitors the running accelerations and transmits them to the TinyML model for real-time discrimination.

To prevent the state information from being lost because the intercepted signal misses the characteristic signal,

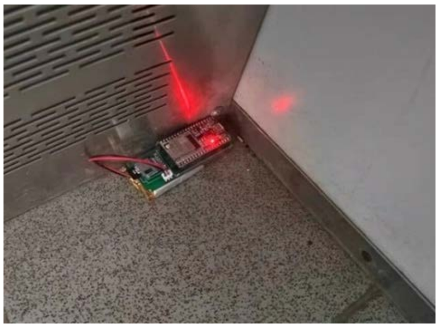


**FIGURE 8. The subway vehicle experiment for testing the Intelligent IoT sensing system.**

this paper presented an offset time window method. Figure 7 shows a start-stop cycle of the rail vehicle. When in real-time monitoring, the data is segmented and moved forward with $T_f$ as the time offset window and $T_g$ as the recognition time interval for each acquisition. For example, $T_f$ takes 0.5 seconds and $T_g$ takes 2 seconds, which means that every 0.5 seconds, 2 seconds of acceleration data is intercepted for identification. As a result, the offset time window method will repeatedly recognize the running state. For statistical convenience, the results of these consecutive repetitions are combined into one. The process described above is an online real-time identification method, which is also applicable to offline recognition of recorded acceleration data. Through the offline method, the accuracy of the recognition model can be tested to evaluate the quality of the trained model. The recognition result and time stamp will be recorded.

## IV. EXPERIMENTS

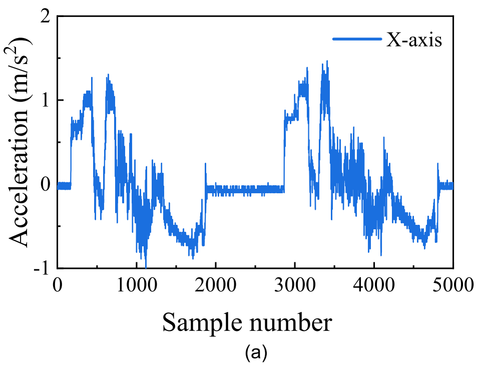
The effectiveness of the constructed model and system was verified by a real-time recognition experiment for subway running states. The machine learning framework adopted the open-source framework TensorFlow. As presented in Figure 8, an acquisition module was placed in the front part of a subway carriage to acquire acceleration values for training. After training, a state recognition module was used for real-time recognition.

### A. ACCELERATION DATA ACQUISITION

A data acquisition module with an acquisition frequency of 20Hz was used to collect acceleration data from the subway vehicle. To ensure sufficient data, the test collected more than 100,000 rows of three-axis acceleration data. In Figure 9, the acceleration data of two start-stop cycles at the head of the subway carriage. The subway traveled along the longitudinal X-axis direction of the accelerometer. Figure 9a) shows a significant change in acceleration in the direction. The vehicle in Figure 9b) also reveals vibration in the Y and Z axes with a certain pattern during running.

### B. TRAINING DATA PREPROCESSING

To identify three running states, the data sets of three running states of rail vehicles in accelerating, moving, and decelerating were extracted from the collected data. The data sets were set as $T_1$, $T_2$, and $T_3$ ($K$ equals 3) as training data. Figure 10 is the schematic diagram of the X-axis acceleration of three states for one start-stop cycle without the stationary state. Using $M$ equals 40 and $Z$ equals 100, the collected three-axis acceleration data were preprocessed according to the proposed method. Figure 11 illustrates some collected data.



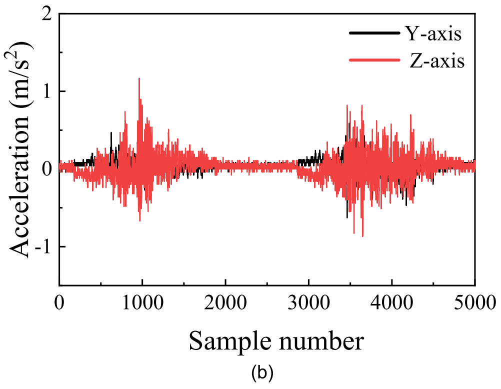


**FIGURE 9. Acceleration sample data of two start-stop cycles (a) X-axis acceleration data (b) Y and Z-axis acceleration data.**

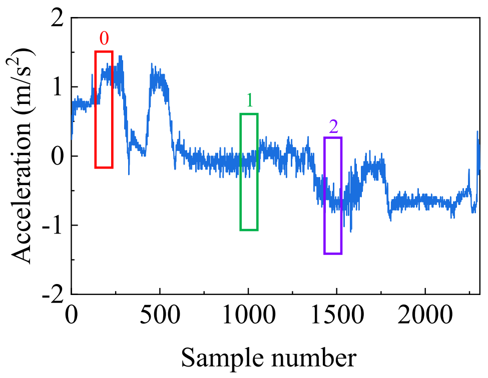
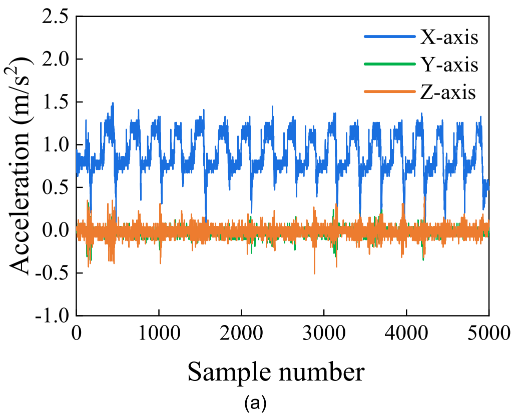


**FIGURE 10. Schematic diagram of X-axis acceleration of three running states.**

Because the stationary state has a single characteristic and is easily judged programmatically, the three running states do not contain the stationary state. The specific judgment method of the stationary state is that the three-axis acceleration values are always less than a small threshold range in the $T_g$ section.



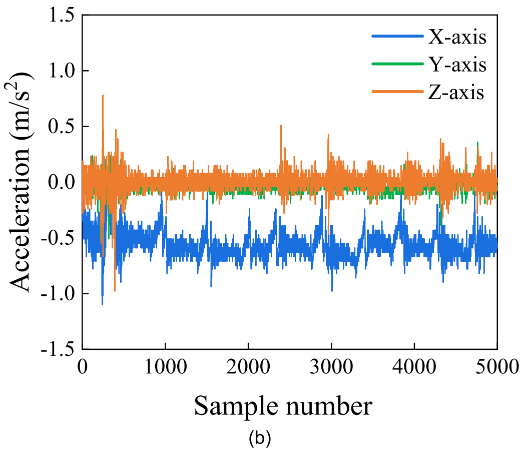


**FIGURE 11.** Part of the sample set of subway (a) sample set of the accelerating state (b)sample set of the decelerating state.

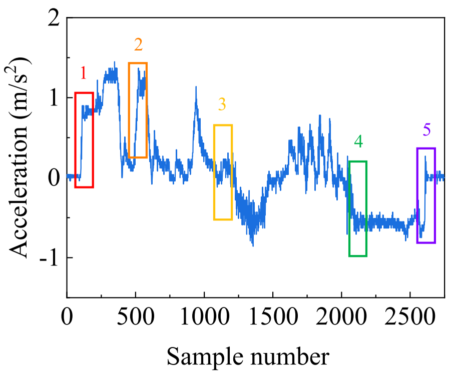


**FIGURE 12.** Schematic diagram of five different vehicle running states recognition.

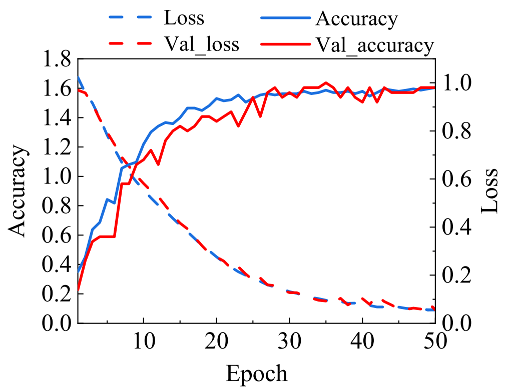


**FIGURE 13.** Training and recognition results of five vehicle running states by the model.

**TABLE 1.** The machine learning model's parameters for recognizing three running states.

| Parameter | Value |
|---|---|
| Z (Training set of single running state) | 10000 |
| M | 40 |
| Neurons of the input layer | 120 |
| Neurons of the output layer | 3 |
| Number of hidden layers | 2 |
| Neurons of the hidden layer | 16 |
| Total number of parameters to be trained | 2259 |

Based on the method in subsection III, the segmentation data was fed into the neural network as input neurons. The three-dimensional array of triaxial acceleration $S_n$ with $M$ equals 40 was flattened into a $120 \times 1$ one-dimensional array as the input to the neural network. After defining the three running states as 0, 1, and 2 labels, the data was normalized.

### *C. COMPLEX RUNNING STATE RECOGNITION AND TRAINING*

The preprocessed sample data set was trained for machine learning. The parameters of the machine learning model for three running states are listed in Table 1. After training, the evaluation accuracy of the training set was 100%. The machine learning model was used for offline prediction with 4640 rows of acceleration data of the vehicle running for two complete cycles, and the prediction accuracy of the offset time window method was 99.8%.

Furthermore, the proposed method was verified with five complex types: (1) starting, (2) accelerating, (3) moving, (4) decelerating, and (5) stopping. Figure 12 presents the X-axis acceleration data for the five running states. The parameters of the machine learning model for five running states are presented in Table 2. After the model training, the accuracy of the training set evaluation is 100%, and the accuracy of the validation set is 99.67%, as shown in Figure 13.

The model was used to predict the offline data of three complete cycles. A total of 6520 acceleration data points were predicted 652 times using the offset time window method. The results had seven errors, and the model accuracy was 98.9%. Most of the errors were identifying acceleration state features as starting states. The reason for this is that some acceleration data had very similar characteristics of accelerating and starting in the $T_g$ time interval. This problem can be eliminated by determining whether the previous state of the current state is stationary or not. If the previous $T_g$ state is stationary, the current running state is starting, and if it is not, it is accelerating.

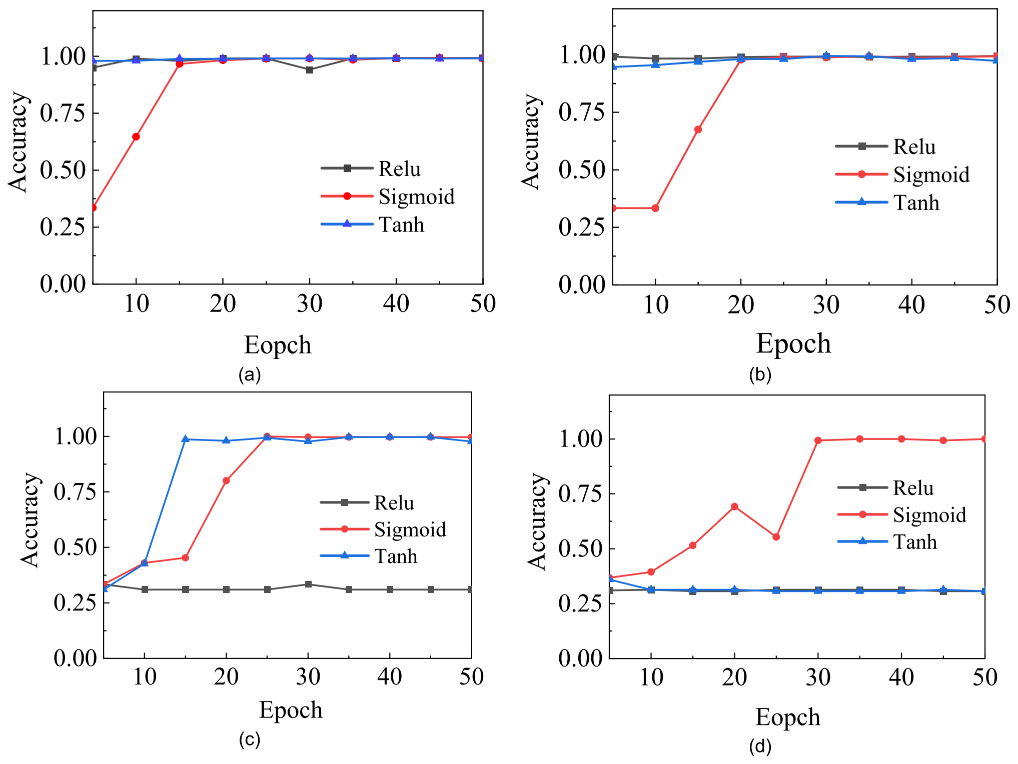


**FIGURE 14.** Comparison of accuracy of different activation functions (a)M=5 (b)M=10 (c)M=20 (d)M=40.

**TABLE 2.** The machine learning model's parameters for recognizing five running states.

| Parameter | Value |
|---|---|
| Z (Training set of single running state) | 4000 |
| M | 40 |
| Neurons of the input layer | 200 |
| Neurons of the output layer | 5 |
| Number of hidden layers | 2 |
| Neurons of the hidden layer | 16 |
| Total number of parameters to be trained | 2565 |

**TABLE 3.** Statistics on the training effect of different activation function models.

| M | Number of activation function training rounds | | |
|---|---|---|---|
| | Relu | Sigmoid | Tanh |
| 5 | 10 | 15 | 5 |
| 10 | 5 | 20 | 15 |
| 20 | - | 30 | 15 |
| 40 | - | 30 | - |

## D. THE TRAINING EFFECT OF VARIOUS ACTIVATION FUNCTION MODELS

The activation function in a neural network can improve the nonlinear fitting ability of the model, which is crucial to the quality of the trained model. To choose a suitable activation function, three common activation functions were compared by giving different $M$ rows of data. Three common activation functions were applied to the model for 50 rounds of training, and the accuracy of the training results is presented in Figure 14.

Figure 14 reveals that the training results using different activation functions are different when $M$ is used as the variable. When $M$ equals 5 and 10, all three activation functions achieve good learning results. The model training speed is faster when using the Relu function as the activation function. When the number of input rows gradually increases ($M$ equals 40), the model training effect of the Relu function and the Tanh function has been greatly reduced. When using the Sigmoid activation function, the accuracy of the model does not decrease with the number of input $M$. The model is robust when using the Sigmoid function as the activation function, and the system has a more stable recognition capability. The number of training rounds changes as the number of input rows increases, as shown in Table 3.

## E. TRAINING EFFECT COMPARISON ON DIFFERENT AXIS NUMBERS

Since the vehicle running state features in Figure 9a) are already visible on the X-axis, the effect of comparing one X-axis and three-axis acceleration data as input to the model recognition must be investigated.

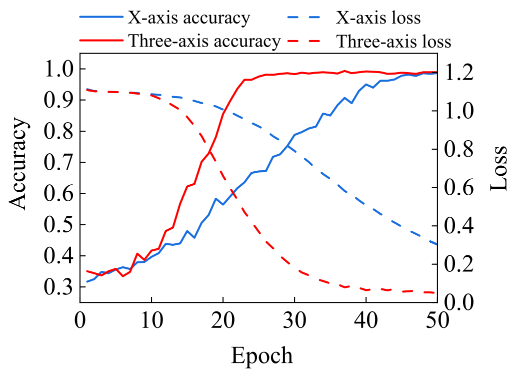


**FIGURE 15.** Comparison of the training effects of different axes with M=20.

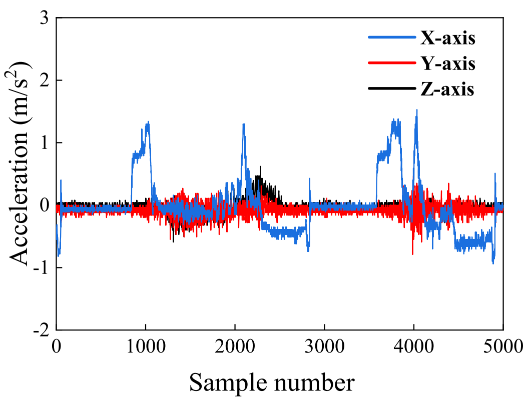


**FIGURE 16.** Comparison of the training effects of different axes with M=20.

Acceleration data with two different numbers of axes were used separately as inputs for model training. When $M$ equals 20, the resultant accuracy and cross-entropy loss values are described in Figure 15.

The same activation function, optimization function, and other parameters were chosen for the construction of the two machine learning models. The model in figure 15 has been trained after 20 iterations when the three-axis acceleration data is taken as the model input. In contrast, using the X-axis acceleration data as input takes more than 50 training rounds. Consequently, the model with three axes converged faster with higher accuracy and lower loss value than the model with one axis. Therefore, the system chose three-axis acceleration as the input to train the model of the running state.

### F. COMPARISON OF RECOGNITION ACCURACY FOR DIFFERENT INSTALLATION POSITION

The length of a subway carriage is generally long. A typical subway carriage is 19 to 23 meters. To test the reliability, robustness, and generalization ability of the system, the accuracy influence of different installation positions of the system in the vehicle needs to be studied. In the experiments, the recognition model was trained with acceleration data from the head of the carriage only and without training by the data from the rear of the carriage. A sample of the collected data from the rear of the subway carriage is shown in Figure 16. The figure reveals that the rear acceleration data have the same characteristics as the head one in Figure 9.

Experiments were conducted to identify the three running states of accelerating, moving, and decelerating as an example. After splitting and defining the label value, the dataset with $Z$ equals 1000 was imported into the model for recognition evaluation. The accuracy rate of the evaluation results is demonstrated in Figure 17.

After model identification, the three states were identified with 99.8%, 98.2%, and 100% accuracy. The high recognition accuracy of the model demonstrated that the installation position of the state recognition module in the carriage had little effect on the accuracy. Therefore, the state recognition module can be deployed in different positions in the same carriage.

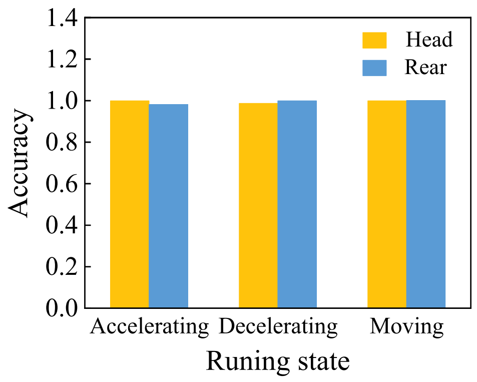


**FIGURE 17.** Comparison of the accuracy of different monitoring positions.

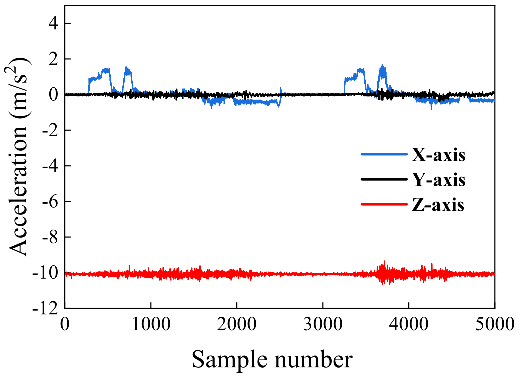


**FIGURE 18.** The acceleration data with zero drift.

### G. COMPARISON OF THE ACCURACY OF ZERO-DRIFT RECOGNITION

Since zero drift may occur in low-cost MEMS sensors, it is necessary to analyze the effect of this phenomenon on recognition results if zero drift is not removed.

As shown in Figure 18, the data with z-axis zero drift ($Z$ equals 1000) was imported into the model for offline identification. The recognition results are presented in Figure 19.

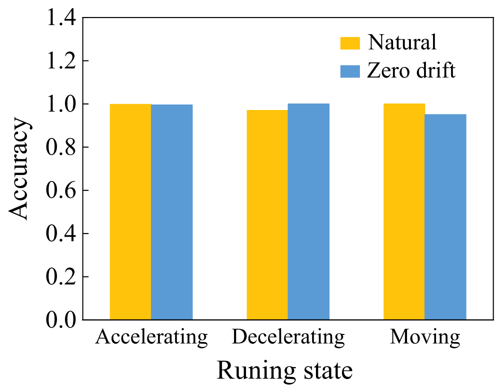
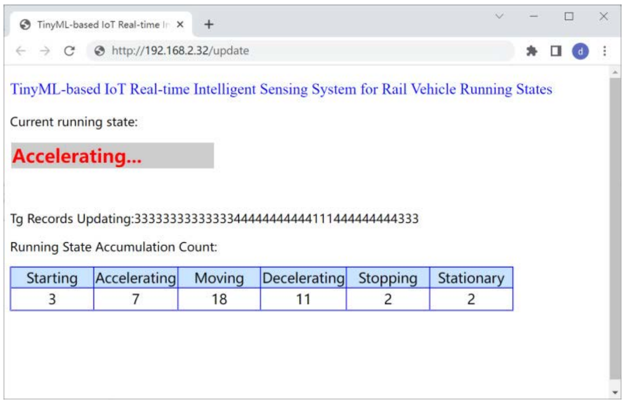


**FIGURE 19.** Comparison of model evaluation with zero-drift acceleration data.



TinyML-based IoT Real-time Intelligent Sensing System for Rail Vehicle Running States

Current running state:

Accelerating...

Tg Records Updating:333333333333334444444444411144444444433 3

Running State Accumulation Count:

| Starting | Accelerating | Moving | Decelerating | Stopping | Stationary |
|---|---|---|---|---|---|
| 3 | 7 | 18 | 11 | 2 | 2 |

**FIGURE 20.** The web interface of real-time recognition results of running states.

The recognition accuracies of accelerating, decelerating, and moving states are 99.5%, 99%, and 95.1%. Therefore, the recognition model has strong generalization ability and reliability because it can identify the acceleration data with zero drift to a certain extent. On the other hand, it also reveals that the zero-drift decreased the identification accuracy and had a certain influence on the recognition results. Thus, the zero-drift needs to be removed for obtaining better recognition results.

### H. REAL-TIME SENSING OF RUNNING STATES FOR THE RAIL VEHICLE ONLINE

The TinyML recognition model was deployed to the IoT edge device on the vehicle to sense the vehicle running states online. Then, the effectiveness of the system was tested by sensing six types of complex motion states: starting, accelerating, decelerating, stopping, moving, and stationary. Among the six states, the stationary state was identified by the method in subsection IV(B), while the rest of the states were identified by the machine learning method. Four start-stop cycles of the subway were monitored in the experiment. The total real-time monitoring time was about 9 minutes. Each start-stop cycle had a deceleration time of about 1 minute, a stop time of about 1.5 minutes, and an acceleration time of about 20 seconds.

The system used the online offset time window method for intelligent identification. To verify the accuracy of the recognition results, besides online intelligent recognition, the whole test process was also recorded on video for offline rechecking. The $T_f$ of the experiment was taken as 0.5 seconds and $T_g$ was taken as 2 seconds, which means that running states were recognized every 0.5 seconds and were judged by 2 seconds of acceleration data. A total of 1108 judgments were made in the whole test. During sensing, the recognition results were uploaded to the local IoT server. Each complex running state of the vehicle was presented on the browser in real time, as shown in Figure 20. After online and offline result comparison, the test results revealed that the real-time recognition results using the proposed method matched with the subway vehicle running status results, and there were three abnormalities in 1108 times of recognition result data, with an accuracy rate of 99.7%.

## V. CONCLUSION

TinyML technology is an effective way to make rail vehicles intelligent and perceptive. The constructed system is low-cost, small-sized, and low energy consumption. With only one micro IoT edge device on the vehicle side, each carriage can recognize complex running states. The following conclusions are obtained from this paper's research:

(1) The recognition model and system proposed in this paper can identify multiple complex running states in real-time, and the monitoring results are accurate. The recognition model is robust when using the Sigmoid function as the activation function. The subway experiments showed that the system was capable of identify six complex running states in real-time and the identification accuracy was higher than 99%. Moreover, the experiment shows that the acceleration data can be used for monitoring data for identifying the running state of smart rail vehicles.

(2) Three-axis acceleration data monitoring is better than one-axis monitoring. Using three-axis acceleration data as model input for model training is faster, more accurate, and has lower loss values than using one-axis acceleration data.

(3) The sensing system can obtain high accuracy in running state recognition regardless of whether it is installed in the front or rear of the carriage. The experiment on the subway revealed that the recognition accuracy of different installation positions was high, reaching more than 98%. The model has good reliability and generalization capability.

(4) The established model retains some high-running state recognition capability under zero-drift acceleration. The experimental result showed that the recognition accuracy of the model was still high, reaching more than 95%, without removing zero drift. Furthermore, it indicated that the zero-drift has a certain influence on the recognition accuracy

and the zero-drift should be removed during monitoring for obtaining better results.

The method proposed in this paper can identify more complex running states such as turning, collision, rapid acceleration, slow acceleration, emergency braking, etc. In addition, the system is suitable for mass installation on rail vehicles. The identification results can provide essential data support for structural health monitoring and condition-based maintenance. The method and system can also be applied to various rail vehicles like high-speed trains, railway wagons, and intercity trains, as well as other equipment with TinyML chips such as cars, ships, aircraft, and spacecraft.

## REFERENCES

[1] S. J. Kamble and M. R. Kounte, "Machine learning approach on traffic congestion monitoring system in internet of vehicles," *Proc. Comput. Sci.*, vol. 171, pp. 2235–2241, Jan. 2020.

[2] J. G. Choi, C. W. Kong, G. Kim, and S. Lim, "Car crash detection using ensemble deep learning and multimodal data from dashboard cameras," *Expert Syst. Appl.*, vol. 183, Nov. 2021, Art. no. 115400.

[3] T. Praveenkumar, B. Sabhrish, M. Saimurugan, and K. I. Ramachandran, "Pattern recognition based on-line vibration monitoring system for fault diagnosis of automobile gearbox," *Measurement*, vol. 114, pp. 233–242, Aug. 2018.

[4] Q. Lu, L. Huang, W. Li, T. Wang, H. Yu, X. Hao, X. Liang, F. Liu, P. Sun, and G. Lu, "Mixed-potential ammonia sensor using Ag decorated $FeVO_4$ sensing electrode for automobile *in-situ* exhaust environment monitoring," *Sens. Actuators B, Chem.*, vol. 348, Dec. 2021, Art. no. 130697.

[5] S. Liu, N. Guan, D. Ji, W. Liu, X. Liu, and W. Yi, "Leaking your engine speed by spectrum analysis of real-Time scheduling sequences," *J. Syst. Archit.*, vol. 97, pp. 455–466, Aug. 2019.

[6] A. O. V. P. Kumar, D. Nandini, M. M. Sairam, and B. P. Madhusudan, "Development of GPS & GSM based advanced system for tracking vehicle speed violations and accidents," *Mater. Today, Proc.*, Jul. 2021.

[7] S. Lee, Y. Kim, H. Kahng, S.-K. Lee, S. Chung, T. Cheong, K. Shin, J. Park, and S. B. Kim, "Intelligent traffic control for autonomous vehicle systems based on machine learning," *Expert Syst. Appl.*, vol. 144, Apr. 2020, Art. no. 113074.

[8] M. N. Khan and M. M. Ahmed, "Trajectory-level fog detection based on in-vehicle video camera with TensorFlow deep learning utilizing SHRP2 naturalistic driving data," *Accident, Anal. Prevention*, vol. 142, Jul. 2020, Art. no. 105521.

[9] D. A. Alghmgham, G. Latif, J. Alghazo, and L. Alzubaidi, "Autonomous traffic sign (ATSR) detection and recognition using deep CNN," *Proc. Comput. Sci.*, vol. 163, pp. 266–274, Jan. 2019.

[10] Z. Zhu, Z. Hu, W. Dai, H. Chen, and Z. Lv, "Deep learning for autonomous vehicle and pedestrian interaction safety," *Saf. Sci.*, vol. 145, Jan. 2022, Art. no. 105479.

[11] B. R. Vasconcellos, M. Rudek, and M. de Souza, "A machine learning method for vehicle classification by inductive waveform analysis," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 13928–13932, 2020.

[12] S. Jaiswal, D. K. Sharma, T. Jaiswal, B. Basumatary, M. Tiwari, and T. Tiwari, "Real time analysis of intelligent placing system for vehicles using IoT with deep learning," *Materials Today, Proc.*, vol. 51, pp. 339–343, Jan. 2022.

[13] T. Yonezawa, I. Arai, T. Akiyama, and K. Fujikawa, "Random forest based bus operation states classification using vehicle sensor data," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 747–752.

[14] P. Li, M. Abdel-Aty, Q. Cai, and Z. Islam, "A deep learning approach to detect real-time vehicle maneuvers based on smartphone sensors," *IEEE Trans. Intell. Transport. Syst.*, vol. 23, no. 4, pp. 3148–3157, Oct. 2022.

[15] C. Lan, S. Li, H. Chen, W. Zhang, and H. Li, "Research on running state recognition method of hydro-turbine based on FOA-PNN," *Measurement*, vol. 169, Feb. 2021, Art. no. 108498.

[16] W. Lu, H. Mao, F. Lin, Z. Chen, H. Fu, and Y. Xu, "Recognition of rolling bearing running state based on genetic algorithm and convolutional neural network," *Adv. Mech. Eng.*, vol. 14, no. 4, 2022, Art. no. 168781322210956.

[17] R. Y. Adhitya, M. A. Ramadhan, S. Kautsar, N. Rinanto, S. T. Sarena, I. Munadhif, M. Syai'in, R. T. Soelistijono, and A. Soeprijanto, "Comparison methods of fuzzy logic control and feed forward neural network in automatic operating temperature and humidity control system (oyster mushroom farm house) using microcontroller," in *Proc. Int. Symp. Electron. Smart Devices (ISESD)*. Piscataway, NJ, USA: IEEE, Nov. 2016, pp. 168–173.

[18] G. Chand, M. Ali, B. Barmada, V. Liesaputra, and G. Ramirez-Prado, "Tracking a person's behaviour in a smart house," in *Proc. Service-Oriented Comput. (ICSOC)*, in Lecture Notes in Computer Science, X. Liu, Eds. Cham, Switzerland: Springer, 2019, pp. 241–252.

[19] Y. Zhang, S. Bi, M. Dong, and Y. Liu, "The implementation of CNN-based object detector on ARM embedded platforms," in *Proc. IEEE 16th Int. Conf. Dependable, Autonomic Secure Comput., 16th Int. Conf. Pervasive Intell. Comput., 4th Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Athens, Greece, 82018, pp. 379–382.

[20] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, "GAP-8: A RISC-V SoC for AI at the edge of the IoT," in *Proc. IEEE 29th Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Jul. 2018, pp. 1–4.

[21] P. Warden and D. Situnayake, *TinyML*. Sebastopol, CA, USA: O'Reilly Media, 2019.

[22] I. Fedorov, R. P. Adams, M. Mattina, and P. Whatmough, "SpArSe: Sparse architecture search for CNNs on resource-constrained microcontrollers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.

[23] M. de Prado, M. Rusci, A. Capotondi, R. Donze, L. Benini, and N. Pazos, "Robustifying the deployment of tinyML models for autonomous mini-vehicles," *Sensors*, vol. 21, no. 4, p. 1339, 2021.

[24] B. Saha and D. Srivastava, "Data quality: The other face of big data," in *Proc. IEEE 30th Int. Conf. data Eng.*, Mar. 2014, pp. 1294–1297.

[25] P. J. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," *Data Mining Knowl. Discovery*, vol. 1, no. 1, pp. 73–79, 2011.

[26] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 105524.

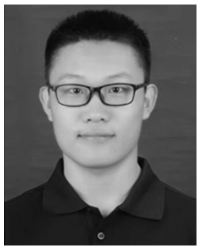
[27] P. Rodríguez, M. A. Bautista, J. Gonzàlez, and S. Escalera, "Beyond one-hot encoding: Lower dimensional target embedding," *Image Vis. Comput.*, vol. 75, pp. 21–31, Jul. 2018.

**SHAOZE ZHOU** received the Ph.D. degree in mechanical engineering from Dalian Jiaotong University, Dalian, China, in 2011. He is currently an Associate Professor with the School of Locomotive and Rolling Stock Engineering, Dalian Jiaotong University. He has published more than 30 journal articles. His main research interests include intelligent technology and application, numerical simulation, and fatigue analysis.

**YONGKANG DU** was born in Heze, Shandong, China, in 2000. He received the bachelor's degree in vehicle engineering from Weifang University, in 2016, and the master's degree in mechanics from Dalian Jiaotong University, in 2022. His research interests include machine learning, embedded device development, and applications of the IoT technologies.

**BINGZHI CHEN** received the B.S. and M.S. degrees from Dalian Jiaotong University, Dalian, China, in 1994 and 1997, respectively, and the Ph.D. degree from the Dalian University of Technology, in 2002. He is currently a Professor with the School of Locomotive and Rolling Stock Engineering, Dalian Jiaotong University. He has published more than 50 journal articles. His main research interests include intelligent computing, railway vehicle engineering, virtual prototype, numerical simulation, structure optimization, and biomechanics.

**YONGHUA LI** received the B.S. and M.S. degrees from Liaoning Technical University, in 1995 and 2002, respectively, and the Ph.D. degree from the Dalian University of Technology, in 2005. She is currently a Professor with the School of Locomotive and Rolling Stock Engineering, Dalian Jiaotong University, Dalian, China. She has published more than 50 journal articles. Her main research interests include digital simulation, optimization design of mechanical products, and fatigue reliability analysis of vehicle structures.



**XINGSEN LUAN** received the B.S. degree in transportation from Ludong University, Yantai, China, in 2021. He is currently pursuing the M.S. degree in vehicle engineering with Dalian Jiaotong University, Dalian, China. His research interests include machine learning and embedded device development.