International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

Research on Deep Learning-based Anomaly Event Detection Using Fiber Optic Vibration Data

Ziyang Wang
China Jiliang University

Date of Submission: 15-07-2025 Date of Acceptance: 25-07-2025

ABSTRACT
This study aims to develop and evaluate machine learning and deep learning models for classifying electrical signals collected by fiber optic sensors under three conditions: calm, shaking, and pressing. First, time-domain and frequency-domain features were extracted, and key features were selected using the recursive feature elimination method to reduce dimensionality. Subsequently, support vector machine (SVM), linear discriminant analysis (LDA), and long short-term memory (LSTM) models were constructed based on the time-frequency domain features of the signals. Additionally, gated recurrent unit (GRU) and convolutional neural network (CNN) models were constructed and improved based on the complete signal data. The results show that the CNN model performed best in terms of classification accuracy and generalization ability, with a validation accuracy of 96.81%, low risk of overfitting, and good classification balance. Furthermore, by optimizing the GRU model, the recognition ability of minority classes was significantly improved, providing valuable experience for the application of the GRU model.

Keywords: Fiber Optic Sensor、Electrical Signal Classification、Machine Learning、Deep Learning、Feature Extraction、Support Vector Machine (SVM)、Linear Discriminant Analysis (LDA)、Long Short-Term Memory (LSTM) Network、Gated Recurrent Unit (GRU)、Convolutional Neural Network (CNN)

I INTRODUCTION

I.I Background

With the rapid development of industrial automation and intelligent manufacturing, equipment condition monitoring and fault diagnosis have become key aspects in ensuring production safety and improving operational efficiency. Among various sensing technologies, fiber optic sensors are widely used in mechanical equipment, building structures, energy systems, and other fields due to their high sensitivity, resistance to electromagnetic interference, long-distance transmission, and multi-point monitoring capabilities. Fiber optic sensors can collect data such as vibrations and electrical signals generated during equipment operation in real-time, providing important reference information for the health status of the equipment.

In recent years, machine learning and deep learning have made significant progress in signal processing and classification tasks. Traditional machine learning methods, such as support vector machines (SVM) and linear discriminant analysis (LDA), are widely used in signal classification due to their advantages in handling high-dimensional data. However, these methods often face limitations in performance and efficiency when dealing with complex nonlinear relationships and large-scale data. With the development of deep learning technology, deep neural network models such as convolutional neural networks (CNN), long short-term memory networks (LSTM), and gated recurrent units (GRU) have demonstrated powerful feature extraction and classification capabilities, achieving breakthrough results especially in fields like image recognition and speech recognition.

In the field of fiber optic sensor signal classification, existing studies have attempted to apply the aforementioned machine learning and deep learning models to classify signals under different operating conditions. For example, using SVM to perform binary or multi-class classification on vibration signals collected by fiber optic sensors has achieved certain classification accuracy; employing LDA for signal feature dimensionality reduction and classification has also shown good results. However, most of these studies focus on the application of a single model, lacking systematic comparison and optimization of multiple models. Additionally, there is relatively little research on signal feature selection and class imbalance issues, which limits further improvement in classification performance.

I.II Significance of the Study

Accurately classifying vibration and electrical signals collected by fiber optic sensors not only enables real-time monitoring of equipment status but also provides early warnings of potential faults, reducing downtime, lowering maintenance costs, and improving production efficiency. Therefore, developing efficient and accurate signal classification methods is of great practical significance for the intelligent and automated industrial production. At the same time, with the rapid development of machine learning and deep learning technologies, applying these advanced algorithms to the field of signal classification is expected to significantly enhance classification performance and promote the development of intelligent monitoring systems.

I.III Research Objectives and Content

Given the above background, this study aims to systematically develop and evaluate various machine learning and deep learning models for classifying vibration and electrical signals collected by fiber optic sensors under three different conditions: calm, shaking, and pressing. The specific objectives include:

(1) Signal Processing and Feature Extraction: Extract time-domain and frequency-domain features from the raw electrical signals collected by fiber optic sensors to comprehensively capture the dynamic characteristics of the signals.

(2) Feature Selection: Use recursive feature elimination (RFE) combined with a random forest model to select the most critical features for the classification task, reduce feature dimensionality, and improve model performance.

(3) Model Construction and Evaluation: Construct and compare various classification models, including traditional support vector machines (SVM), linear discriminant analysis (LDA), and deep learning models such as long short-term memory networks (LSTM), gated recurrent units (GRU), and convolutional neural networks (CNN). Evaluate the classification accuracy, recall, F1 score, and other performance metrics of the models through cross-validation.

(4) Model Optimization and Balancing Strategies: Explore methods to improve the classification performance and generalization ability of the models, addressing the issue of class imbalance.

II LITERATURE REVIEW

Many domestic researchers have already conducted substantial and meaningful work in the field relevant to this study design.

In the study by Sun Yihao (2023), the selected time-domain features include pulse index, mean amplitude, and mean energy, which are primarily used to describe the amplitude, impact intensity, and energy variation of signals. The frequency-domain features are extracted through Fourier transform, including the mean, variance, skewness, kurtosis, and power spectral density (PSD) of the signal, which are used to reveal the energy distribution and spectral characteristics across different frequency components. These features reflect the signal's time-domain and frequency-domain properties from different perspectives, thereby enhancing the accuracy of signal classification and identification[1].

In Li Hua's study (2024), the limitations of the classical Linear Discriminant Analysis (LDA) in high-dimensional data classification were explored. Traditional LDA models depend on the accurate estimation of sample covariance matrices, which can become non-invertible in high-dimensional spaces. To overcome this, the study proposed several improved models, including Regularized LDA (R-LDA), Improved LDA (I-LDA), and Spectrum Corrected LDA (SC-LDA). These improved models address covariance matrix singularity and enhance classification performance in high-dimensional scenarios[2]. Additionally, integrating PCA for dimensionality reduction further proved the effectiveness of the modified LDA models in such contexts, offering new directions for high-dimensional data analysis.

Zhang Fei et al. (2024) proposed an anomaly detection model for power time-series data based on the ConvLSTM network. By integrating Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, the model effectively extracts spatiotemporal features from time-series data, significantly improving the accuracy and robustness of multidimensional data anomaly detection[3]. Experimental results indicate that the model outperforms traditional ARMA algorithms and LSTM-ED models in detecting anomalies under both steady-state and dynamic conditions of power

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

systems, achieving higher accuracy and recall rates. Additionally, the unsupervised learning method adopted in the study enables anomaly detection without labeled data, offering a new modeling approach and implementation pathway for the analysis of multi-state, multi-sensor time-series data. This method effectively addresses the issue where traditional approaches often fail to capture complex temporal dependencies and spatial correlations among multiple sensors in multidimensional time-series anomaly detection.

Several international teams have also conducted related research.

Reza et al. (2008) proposed a dynamic voltage collapse prediction model for power systems based on Support Vector Machines (SVM). Through comparison with a Multi-Layer Perceptron Neural Network (MLPNN), the study demonstrated the efficiency and accuracy of SVM in handling complex feature data[4]. The research suggests that using the Radial Basis Function (RBF) as the kernel function, along with cross-validation to optimize parameters, can further enhance model performance. In vibration signal monitoring, SVM, due to its excellent nonlinear discrimination capability and generalization performance, has been widely used for equipment condition monitoring and fault diagnosis, exhibiting high application value.

Lopes-Lima Ribeiro et al. (2002) proposed an Incremental Construction LSTM (GLSTM) model to address the challenge. The GLSTM model uses a cascaded and fully connected strategy to dynamically increase memory blocks and neurons based on data complexity, enhancing the model's ability to capture varying characteristics of vibration signals[5]. In the study of vibration signals by Zhongdian Haikang, this model addresses the issues of sample imbalance and all-zero data by layer-wise optimization and freezing of existing weights, effectively reducing training difficulty and computational costs while significantly improving the discrimination accuracy for sparse samples. This dynamic construction strategy provides a theoretical foundation and practical guidance for the efficient classification and discrimination of fiber optic vibration alarm systems.

III METHODOLOGY

III.I Fourier Transform

The core idea of the Fourier Transform is that a complex signal can be decomposed into the sum of multiple simple sine or cosine functions. Through this decomposition, the signal's representation in the time domain (or spatial domain) is transformed into a representation in the frequency domain. This frequency-domain representation reveals the strength and phase of the different frequency components in the signal, thus providing a deeper understanding of the signal's nature[6].

Mathematically, the Fourier transform is a mapping from the function space to the frequency space. It projects the signal onto a set of orthogonal basis functions (complex exponential functions) using an inner product. These basis functions are of the form:

$$(1)\ e^{-j\omega t} = cos(\omega t) - j\, sin(\omega t)$$

Here, ωrepresents the angular frequency, and t is the time variable.

The Fourier transform $F(\omega)$ of a signal $f(t)$ defined in the time domain is given by:

$$(2)\ F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt$$

(1) $f(t)$is the signal in the time domain;
(2) $F(\omega)$ is the corresponding signal in the frequency domain, representing the strength of the component with frequency ω;
(3) $e^{-j\omega t}$ is the complex exponential basis function, used to decompose the frequency components of the signal;
(4) $\omega$is the angular frequency (measured in radians per second);
(5) j is the imaginary unit (satisfying $j^2 = -1$);

The Fourier transform obtains the projection strength of the signal at each frequency by multiplying the time-domain signal by the complex exponential basis function $e^{-j\omega t}$ and integrating over the entire time range.

III.II Feature Selection

Random Forest, as an ensemble learning model consisting of multiple decision trees, is often considered an optimal solution for feature selection due to its simplicity in algorithm design and high efficiency[7]. Moreover, Random Forest evaluates the importance of each feature variable by using the out-of-bag (OOB) error, thereby eliminating redundant features in large-scale, high-dimensional datasets for feature selection. When bootstrap samples $k = 2, 3, \dots, K$ (where K represents the number of training samples) are used, and each sample has N feature attributes, the following steps are involved in calculating the ranking of feature importance:

(1) Set the initial value to $k=1$, and create the decision tree T_k.
(2) Train T_k on the k-th bootstrap sample, and calculate the classification accuracy for each out-of-bag (OOB) data as L_k^{oob}.
(3) Randomly perturb the features X_j of the OOB dataset where $j = 1,2,\ldots,N$, and calculate the classification accuracy as $L_{k,j}^{\mathrm{oob}}$
(4) For $k = 2,\ 3,\ldots$, return to step (1)
(5) The importance measure P_j of the feature X_j is calculated using the formula (3).
(6) After sorting P_j in descending order, the feature importance ranking is obtained.

$$(3)\ P_j = \frac{1}{K}\sum_{k=1}^{K}\left(L_k^{\mathrm{oob}} - L_{k,j}^{\mathrm{oob}}\right)$$

In the formula, P_j represents the importance measure of the feature X_j, L_k^{oob} represents the classification accuracy of the OOB data without perturbation, and $L_{k,j}^{\mathrm{oob}}$ represents the classification accuracy of the OOB data after perturbing the j-th feature.

III.III Machine Learning
III.III.I SVM Model

Support Vector Machine (SVM) is a model for binary classification[8], which works by finding an appropriate hyperplane in the space that maximizes the margin between two samples, thereby achieving classification. The basic optimization problem for SVM is given by the following formula:

$$(4)\ \min_{\alpha}\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{j=1}^{l}\alpha_j$$

$$(5)\ \text{s.t.} \sum_{i=1}^{l} y_i\alpha_i = 0, 0 \le \alpha_i \le C, i = 1,\ldots,l$$

In the formula: $K(x,x')$ is the kernel function; C is the penalty factor; x_i is the feature vector; and $y_i \in Y = \{1,0\}$

For SVM, the choice of the kernel function is crucial for building the model, as the kernel function determines the performance of the SVM. In most cases, it is not possible to construct a kernel function that perfectly fits the sample space, so common kernel functions are typically used, such as the linear kernel, polynomial kernel, Gaussian (RBF) kernel, and sigmoid kernel. Since the RBF kernel has advantages in handling non-linear mappings and requires fewer parameters, the RBF function is chosen as the kernel function for the SVM in this paper.

When using the RBF kernel, it is necessary to consider the values of the penalty factor C and the kernel parameter g, as the selection of C and g affects the accuracy, convergence speed, and predictions of the SVM. Common methods for optimizing SVM parameters include cross-validation (CV) and grid search. There are several variants of CV, with K-fold cross-validation (K-CV) being a common method. The basic idea of K-CV is to divide the original dataset into K groups, and at each iteration, one subset is used as the test set (without repetition), and the remaining $K-1$ groups are used as the training set. This generates K models, and the average classification accuracy of the corresponding test sets is taken as the performance metric of the model under K-CV. K-CV can effectively prevent both overfitting and underfitting of the SVM model.

Grid search is a hyperparameter search algorithm that is simple to use and widely applied. However, because it requires fixed step sizes, it is prone to skipping the optimal parameter points, making it impossible to achieve global optimization. On the other hand, Particle Swarm Optimization (PSO) is a swarm intelligence optimization algorithm that simulates the foraging behavior of birds. PSO can converge to a global optimum by finding the optimal population. Therefore, in this paper, PSO is used to optimize the penalty factor C and kernel parameter g of the traditional SVM model, and the accuracy under K-CV is used as the performance metric of the SVM model. This approach effectively improves the diagnostic accuracy of fault arcs using the SVM model.

III.III.II LDA Model

Linear Discriminant Analysis (LDA) was originally proposed by Fisher in 1936[9]. Let C_0 and C_1 represent the sample sets of the first and second classes, respectively. The two classes are represented as: $\{x_0^0, x_1^0, \ldots, x_{n_0}^0\} \in C_0$, $\{x_0^1, x_1^1, \ldots, x_{n_1}^1\} \in C_1$, where $x \in R^p$. Lnear Discriminant Analysis finds an optimal projection axis $\boldsymbol{w}^*$ based on Fisher's criterion function.For a given test sample x\mathbf$\boldsymbol{x}$, its projection on $\boldsymbol{w}^*$ is calculated as:$y = (w^*)^T x$along with the threshold

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

$$(6)\ y_0 = \frac{n_0 \widetilde{m_0} + n_1 \widetilde{m_1}}{n_0 + n_1}$$

Wherem_0 and $\widetilde{m_1}$ correspond to the class means of the projected samples, and n_i represents the sample size of the i -th class. Finally, classification is performed according to the decision rule.

$$(7)\ \begin{cases} y > y_0 \Rightarrow \boldsymbol{x} \in C_0 \\ y < y_0 \Rightarrow \boldsymbol{x} \in C_1 \end{cases}$$

The LDA proposed by Fisher has always been one of the most classic techniques. Under the assumption that the data is Gaussian, with completely known statistics and a common covariance matrix across classes, it has been proven to be the optimal classifier with the lowest misclassification rate. Given the sample labels, Fisher's discriminant rule leads to the following discriminant function:

$$(8)\ W^{\mathrm{LDA}}(\overline{x_0}, \overline{x_1}, S, \boldsymbol{x}) = \left(\boldsymbol{x} - \frac{\overline{x_0} + \overline{x_1}}{2}\right)^{\mathrm{T}} S^{-1}(\overline{x_0} - \overline{x_1}) - log\frac{\pi_1}{\pi_0}$$

In practice, class statistics, such as the population mean vector and covariance matrix, are unknown. They are typically estimated using the empirical mean vector $\overline{x_i}$ and the sample covariance matrix S_i from the training data, defined as:

$$(9)\ \overline{x_i} = \frac{1}{n_i}\sum_{j=1}^{n_i} x_j^i$$

$$(10)\ S_i = \frac{1}{n_i - 1}\sum_{j_i \in C_i}\left(\boldsymbol{x}_{j_i}^{i} - \overline{x_i}\right)\left(\boldsymbol{x}_{j_i}^{i} - \overline{x_i}\right)^{\mathrm{T}}$$

$$(11)\ S = \frac{(n_0 - 1)S_0 + (n_1 - 1)S_1}{n - 2}$$

Where$\overline{x_0}$, $\overline{x_1}$ are the centroids of the two classes, S is the pooled sample covariance matrix, and π_i is the prior probability corresponding to each class. When W^{LDA} , the new sample is classified into C_0; when$W^{\mathrm{LDA}} < 0$, the new sample is classified into C_1.

Under the assumption of finite low-dimensional data, the traditional sample covariance matrix serves as a consistent estimate of the population covariance matrix and demonstrates goodclassification performance. This model has been widely applied across various fields, including speech recognition, cancer genomics, and facial recognition. In classical statistics, the dimension p of the sample vector is typically much smaller than the sample size n, and it is often treated as a constant, meaning the vector dimension does not change with the sample size n. As a result, asymptotic theory in classical statistics is based on this assumption. However, with the continuous growth of modern information and the increasing complexity of problems, most research fields now face the issue of high-dimensional data, where the dimension is no longer a fixed constant but increases proportionally with the sample size. In this case, the sample covariance matrix is no longer a consistent estimate of the population covariance matrix, and the classification performance of classical LDA shows significant deviation.

III.IV Deep Learning

III.IV.I LSTM Model

The Long Short-Term Memory (LSTM) neural network was proposed by Hochreiter and Schmidhuber in 1997. By improving the recurrent units in traditional Recurrent Neural Networks (RNNs), LSTM effectively addresses the issues of vanishing and exploding gradients present in RNN models, allowing for efficient analysis and learning of longer time sequences. As a result, the predictive performance of LSTM is significantly enhanced[10]. Today, LSTM networks are widely applied in various fields, including classification tasks, machine translation, and more, showing excellent performance across these domains.

The LSTM neural network is an improvement over the RNN, with a similar basic structure but also significant differences. In the network topology of an RNN, there is only a connection between the hidden layers at the previous time step and the current time step. In contrast, in the LSTM neural network, there is a connection not only between the hidden layers but also between the hidden layers and the cell state at each time step[11]. other words, the neurons in an LSTM network can accept three input variables. The difference in topology between RNN and LSTM networks is illustrated in Figure 1.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

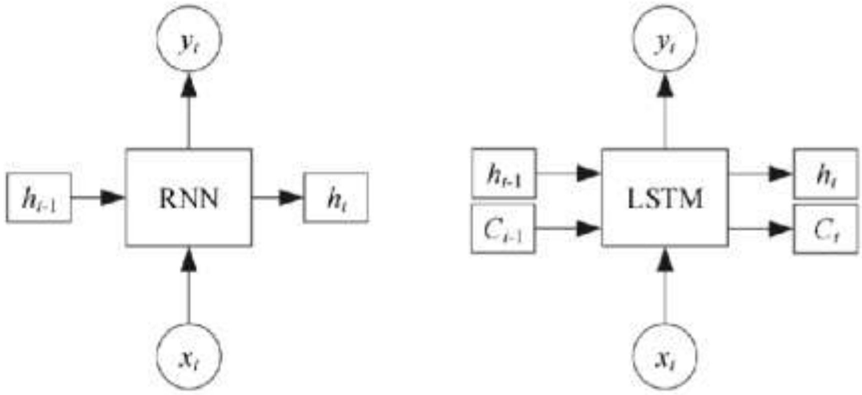


Figure 1 The Topology Diagram of RNN and LSTM Neural Networks

In the LSTM neural network, C_t represents the cell state at the current time step, while C_{t-1} represents the cell state at the previous time step. LSTM networks mainly avoid the gradient issues in RNNs by improving the recurrent units. The simple Tanh recurrent unit in RNNs is replaced by the LSTM recurrent unit, which features a more complex internal structure. Instead of being a simple Tanh activation function, the LSTM unit involves a series of intricate computation processes. The connection structure of the LSTM neural network's recurrent unit is shown in Figure 2.

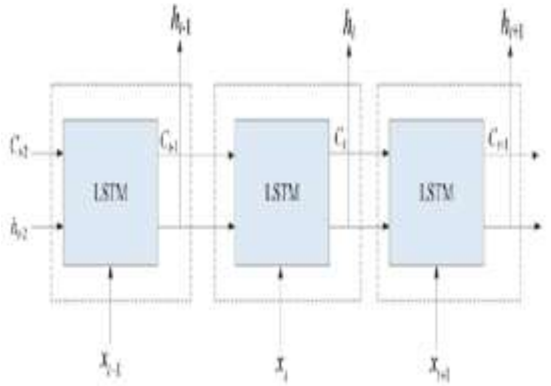
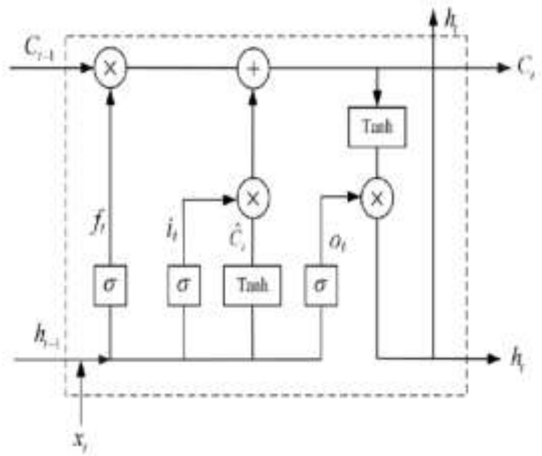


Figure 2The Connection Structure of the LSTM Recurrent Unit

In the LSTM neural network's recurrent unit, there are three input vectors: the hidden state from the previous time step h_{t-1}, the cell state from the previous time step C_{t-1}, and the input at the current time step x_t. The computation process involves three "gates" that analyze and process the input information. These gates are the forget gate f_t, the input gate i_t, and the output gate o_t . The internal structure of the LSTM recurrent unit is shown in Figure 3.



In the internal structure of the LSTM neural network's recurrent unit, the information processing of the hidden state from the previous time step and the current input state is mainly achieved through the forget gate, input gate, and output gate. "Gates" are a type of structure that controls the flow of information. The basic components of a gate include a sigmoid function and a multiplication operation. The fundamental principle is to use the sigmoid function to transform the input data into a value between 0 and 1, and then combine it with the multiplication operation to determine the degree to which the input information should be retained or discarded. A value of 0 means discarding all information, and a value of 1 means retaining all information.

Through the combined actions of the forget gate, input gate, and output gate, important information can be preserved while unnecessary information can be forgotten, enabling the learning of longer sequences. The gate computations in the internal structure of the LSTM neural network's recurrent unit are analyzed as follows.

(1) Forget gate

The forget gate mainly serves to forget redundant information in the input data. The closer the value of the forget gate control is to 0, the higher the degree of forgetting; the closer it is to 1, the lower the degree of forgetting. Its specific calculation formula is:

$$(12)\ f_t = \sigma\left(w_f x_t + u_f h_{t-1} + b_f\right)$$

The forget gate's output is denoted by f_t, where w_f and u_f are the weights of the LSTM neural network, and b_f is the bias.

(2) Input gate

The input gate primarily selects the input information, consisting of two computational processes. The first process involves taking the current input and the previous hidden state, multiplying them by weights, and then processing them through the sigmoid function. This results in the input gate control. The input gate control's function is to decide how much input information should be retained. The closer its value is to 0, the less information is retained, and the closer its value is to 1, the more information is retained. The specific calculation formula for the input gate control is:

$$(13)\ i_t = \sigma(w_i x_t + u_i h_{t-1} + b_i)$$

The second computation process of the input gate mainly aims to obtain the activation state C_t of the cell state t_C, which will be used in subsequent calculations to update the cell state. This process involves multiplying the current input and the previous hidden state by their respective weights and then processing them through the Tanh function. The formula is expressed as:

$$(14)\ \widetilde{C_t} = \mathrm{Tanh}(w_c\, x_t + u_c\, h_{t-1} + b_c)$$

Where $\widetilde{C_t}$ the cell activation state, w_c and u_c are the weights, and b_c is the bias term.

(3) Output gate

The output gate primarily controls the output at the current time step and the flow into the state of the neuron at the next time step. The closer the output gate value is to 1, the greater the weight of the state at this time step in the output result. The calculation formula for the output gate is:

$$(15)\ o_t = \sigma(w_o x_t + u_o h_{t-1} + b_o)$$

In the LSTM neural network, each recurrent unit outputs two variables to the next time step: the cell state C_t and the hidden state h_t. After the input information passes through the forget gate, input gate, and output gate, further processing is performed to update the cell state and hidden state. The mathematical expression for updating the cell state is:

$$(16)\ C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C_t}$$

Where C_{t-1} is the cell state at the previous time step, and $\odot$ denotes the Hadamard product (element-wise multiplication of matrices).

From equation (16), we can see that the cell state update formula consists of two main parts: the first part multiplies C_{t-1} by the forget gate, which helps forget redundant information in C_{t-1}; the second part multiplies the cell activation state by the input gate, which preserves and filters important information in the cell activation state. Finally, the two parts are added together, completing the update process of the cell state in the recurrent unit.

The update of the hidden state is mainly achieved by applying a Tanh function to the cell state, followed by multiplication with the output gate. The mathematical expression is:

$$(17)\ h_t = o_t \odot Tanh(C_t)$$

Equations (12) to (17) represent the computation process of a single neuron in the LSTM neural network. By arranging and combining multiple neurons, the neural network can perform prediction functions. Using the traditional BPTT (Backpropagation Through Time) algorithm to update the weights of the LSTM neural network allows the model to be trained and results in a prediction model that provides ideal performance, meeting the requirements of actual production.

The LSTM neural network is a well-designed improvement over the RNN, utilizing a gating mechanism to control input information.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

This allows for the retention of important information and the forgetting of redundant information, effectively addressing the gradient issues in the RNN model. With better prediction performance, it has been widely applied in various fields.

The LSTM neural network achieves high accuracy and stability in predictions, but the computations in each neuron are complex. The existence of the gating mechanism significantly increases the number of activation functions and training parameters, which raises the difficulty of training the model and decreases the computation speed of the prediction model. To improve the computational efficiency and speed of the LSTM neural network, researchers have begun simplifying its structure, leading to the development of the GRU neural network.

III.IV.IIGRU model

The structure of the GRU neural network is similar to that of the LSTM neural network, but the GRU neural network's recurrent unit has only two input variables: the current input and the previous hidden state[12]. By replacing the LSTM recurrent unit with the GRU recurrent unit, we obtain the connection diagram of the GRU neural network's recurrent unit, as shown in Figure 4.

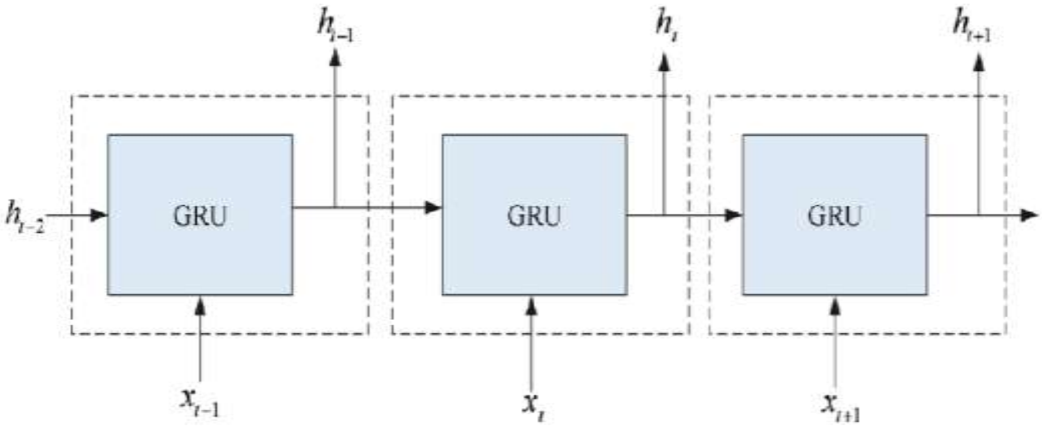


Figure 4 Connection Structure of the GRU Neural Network's Recurrent Unit

Both the LSTM and GRU neural networks solve the gradient problem of RNNs and improve the prediction accuracy for long time sequences by introducing gating mechanisms. In an RNN, the state of the recurrent unit is calculated based on the current input and the hidden state from the previous time step. Each calculation overwrites the information from the previous recurrent unit, making it difficult to maintain long-term information flow. As a result, RNNs can only learn internal information from shorter sequences. The introduction of gating mechanisms in both LSTM and GRU neural networks allows the recurrent unit to retain and forget information[13]. This means that instead of overwriting the entire state of the unit, they can adjust the information by adding or removing it. Thus, this enables them to analyze and learn from longer sequences.

The internal structure of the recurrent unit in the GRU neural network is the main reason for its distinction from the LSTM neural network. LSTM neural networks use three gating mechanisms—forget gate, input gate, and output gate—to manage the retention and forgetting of input information and the hidden state from the previous time step. In contrast, the GRU neural network only introduces two gating mechanisms: the update gate and the reset gate. These two gates are sufficient to achieve the same functionality as the LSTM, thus improving computational speed, saving system memory, and reducing training difficulty. The internal structure of the GRU recurrent unit is shown in Figure 5.

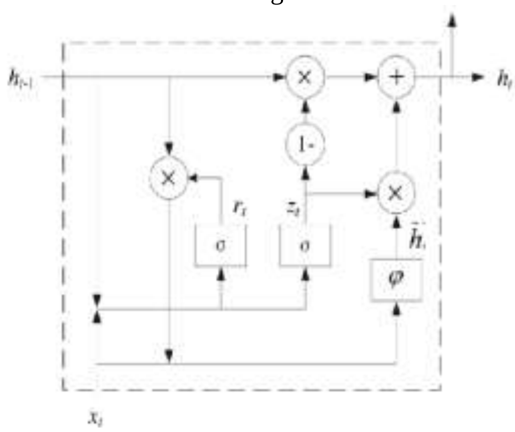


Figure 5 The Internal Structure of the GRU Cell

Where $\widetilde{h_t}$ is the activation state of the hidden layer at the current time step, ϕ is the Tanh activation function, h_t is the hidden state at the current time step, h_{t-1} is the hidden state from the previous time step, r_t is the reset gate, and z_t is the update gate.

From Figure 5, it can be seen that the GRU neural network's recurrent unit contains only two gate mechanisms: the reset gate r_t and the update gate z_t. The current input and the previous time step's hidden state are used as input information, which are multiplied by weight matrices and then processed by the sigmoid function to obtain the reset and update gate controls. The update gate represents the degree to which the previous time step's information flows into the current time step. The closer the value is to 1, the greater the retention of information from the previous time step. The mathematical expression for the update gate calculation is:

$$(18)\ net_z = w_z x_t + u_z h_{t-1}$$
$$(19)\ z_t = \sigma(net_z)$$

The weights w_z and u_z represent the parameters for the reset gate, with x_t being the input at the current time step, σ the sigmoid activation function, and net_z the input to the activation function.

The reset gate represents the degree of forgetting of the information from the previous time step. The closer the value of the reset gate is to 0, the greater the forgetting of the information from the previous time step. The mathematical expression for the reset gate calculation is:

$$(20)\ net_r = w_r x_t + u_r h_{t-1}$$
$$(21)\ r_t = \sigma(net_r)$$

The weights wrw_rwr and uru_rur represent the parameters for the reset gate, with netr\text{net}_rnetr being the input to the activation function.

After obtaining the update gate and the output gate, the reset gate is applied to the hidden state of the previous time step. The resulting value is then combined with the input information at the current time step. This combined input is processed through a weight matrix, summed up, and passed through a Tanh activation function to obtain the activation state of the hidden layer at the current time step. The specific mathematical expression for this calculation is:

$$(22)\ net_{\tilde{h}} = wx_t + u(r_t \odot h_{t-1})$$

$$(23)\ \widetilde{h_t} = \phi(net_{\tilde{h}})$$

The weights w and u represent the parameters, and ⊙epsilonϵ denotes the Hadamard product of the matrices.

By using the update gate to process both the hidden state from the previous time step and the activation state of the hidden layer at the current time step, we sum the results from both processes to obtain the current hidden layer state. The mathematical expression for this can be written as:

$$(24)\ h_t = (1 - z_t)h_{t-1} + z_t\widetilde{h_t}$$

In the process of calculating the current hidden state using the above formula, the GRU unit cleverly utilizes z_t and $1 - z_t$ to process the current hidden state activation and the previous hidden state. This allows it to both forget and retain information from the input and the history. By relying on a single gating unit, it can selectively remember information, saving memory and improving computational efficiency, which effectively speeds up the model's performance.

n both LSTM and GRU neural networks, the recurrent unit analyzes and processes the input information to compute the current hidden state output, h_t. Once the hidden state output is obtained, the neural network's output layer performs additional calculations to derive the overall output of the network. The output layer computation can be expressed as:

$$(25)\ net_y = w_y h_t$$
$$(26)\ y = \sigma(net_y)$$

In this case, w_y represents the weight, y is the output of the neural network, and netynet_ynety is the input to the activation function.

III.IV.III CNN model

CNN, or Convolutional Neural Network, is a type of neural network model proposed by LeCun et al., originally designed for document image recognition[14]. It has since been widely used in fields such as speech analysis and image recognition. CNNs process two-dimensional images directly through weight sharing and convolution operations, eliminating the need for the complex feature extraction and data reconstruction processes that are typical in traditional pattern recognition algorithms[15].

The three structural characteristics of the CNN model (local receptive fields, weight sharing, and subsampling) enable it to adapt to transformations such as translation, scaling, and rotation of input samples[16]. The local receptive field means that each neuron in a layer is only connected to neurons in a corresponding neighborhood of the input layer (usually a 5×55 \times 55×5 rectangular region of neurons), allowing each neuron to extract structural features of the input image, such as direction, lines, endpoints, and corners. Weight sharing greatly reduces the training parameters of the network model and requires relatively fewer training samples. Subsampling, by reducing the resolution of feature maps and sampling to select important features, enhances the model's ability to resist noise while also reducing the dimensionality of the features.

CNN consists of an input layer, hidden layers, and an output layer. The hidden layers are divided into three types: 1) Convolutional layer, used for feature extraction; 2) Subsampling layer, used for feature optimization and selection; 3) Traditional hidden layers from multilayer perceptron networks.

The input to the convolutional layer is connected to the neurons of the previous layer through a K×K convolution kernel (which serves as the connection weights). Each convolutional layer is composed of multiple feature maps, and each feature map can be connected to multiple feature maps from the previous layer. The convolution kernels between feature maps are shared among the neurons within the same feature map, and they can be determined through learning. Neurons within the same feature map share a common bias. The input to the convolutional layer is:

$$(28)\ x_j^l = \sum_{i \in M_j} y_i^{l-1} k_{ij}^l + b_j^l$$

The output of the convolutional layer is:

$$(29)\ y_j^l = \mathrm{sigmoid}\left(x_j^l\right)$$

In the equation, k_{ij}^l represents the convolution kernel that connects the i-th feature map of layer l—1 to the j-th feature map of layer l, M_jj is a selected input feature map, b_j^l is the bias of the jjj-th feature map at layer$l$, and $sigmoid()$ is the activation function of the neuron. Typically, the size of the convolution kernel K is set to 5. If K is too small, it cannot extract the structural features between the data; if K is too large, it will increase the computational cost.

The pooling layer improves noise resistance by reducing the resolution of the feature maps, i.e., by performing feature sampling on the convolutional layer's feature maps to extract the important classification features while ignoring irrelevant details. Currently, the pooling layer in CNN models mainly uses two methods: average pooling (which uses the mean value for sampling) and max pooling (which uses the maximum value for sampling). As shown in Figure 1, each neuron in the pooling layer is connected to the neighboring neurons within a 2×2 region of the convolutional layer. The input to the average pooling layer is:

$$(30)\ x_j^l = \beta \sum_{i \in N \times N} y_i^{l-1} + b$$

The input to the max pooling layer is:

$$(31)\ x_j^l = \max_{i \in N \times N} y_i^{l-1}\, u(n, n)$$

In the formula, β represents the trainable weight; bbb is the threshold parameter for the neuron, which is the learnable bias; $u(n,\ n)$ is the input window function. The sampling layer parameter is N^{-2}. Since the sampling layer does not involve overlapping sampling, it can reduce the feature dimension at a rate of N^{-2}. Studies have proven that the max pooling layer offers better recognition performance, which is why this paper adopts the max pooling method.

Through the convolutional and pooling layers, the dimensions of the feature maps gradually decrease, while the number of feature maps increases. The input to the fully connected layer can be considered as the feature vector extracted by the CNN model, which can then be processed in the traditional manner of artificial neural networks. In the CNN model, the fully connected layer functions as the hidden layer of a traditional neural network. The output layer is related to the classification task of the network model. The number of units in the output layer generally matches the number of categories in the classification task, particularly in the case of handwritten Chinese character recognition using CNN and random elastic deformation. In this study, the simplest approach is used to construct the output layer: for the iii-th category, only the iii-th unit in the output layer outputs 1, while the others output 0.

The CNN learning process typically uses the random diagonal LM (Levenberg-Marquardt) algorithm. This second-order gradient descent learning algorithm enables the model to converge quickly during training. When the number of learning iterations is less than 20, the error function can converge to a satisfactory result. The random diagonal LM algorithm applies different learning rates to each connection weight, adjusting the learning rate based on the second derivative of the parameters to be learned, i.e.,

$$(32)\ \eta_{ki} = \frac{\varepsilon}{\frac{\partial^2 E}{\partial w_{ki}^2} + \mu}$$

In the formula, ε represents the global learning rate, which decreases sequentially with each learning iteration. μ is used to prevent the learning rate from becoming too large due to a small denominator in equation (32). $\frac{\partial^2 E}{\partial w_{ki}^2}$ is the estimated second derivative of the error function E with respect to the weight parameter w_{ki} , which is estimated during training using a certain number of samples. In the experiments described in the paper, after each iteration, the model learns from all the samples once, then re-estimates $\frac{\partial^2 E}{\partial w_{ki}^2}$ and updates the learning rate for each weight. The mean squared error of the output vector is used as the error function.

IV DATA PROCESSING

IV.I Data Loading and Reading

The data for this project comes from experiments. The electrical signals from the same optical fiber sensor are recorded under three conditions: calm, shaking, and pressing. The signals are saved in binary format. The sampling frequency of the signals is 18,680,000 Hz, and the duration of each sample is 0.01 seconds, containing 186,800 data points. Each sample includes 100 cycles. The samples for each condition are stored in three separate folders.

Therefore, any sample data in this project can be viewed as a sequence with 186,800 data points. Since it is a vibration signal read by the sensor, we hypothesize that it may exhibit periodicity. Thus, we will attempt to read one sample.

```
First signal data: [ 0.          0.          -0.02661052 ...  0.          0.
  0.        ]
```

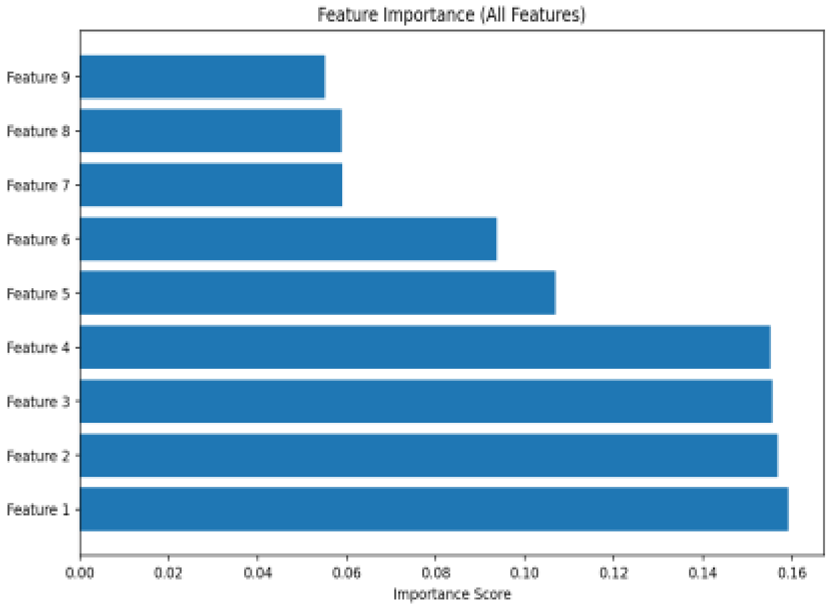
Figure 6 A Sample of Data

Upon doing so, the result is:

It can be seen that this sample data is a one-dimensional sequence. Now, all the samples will be stored in the same dataset, which consists of 467 rows (corresponding to the number of samples) and 186,800 columns (corresponding to the data points). The labels "0", "1", and "2" will be assigned based on the storage paths, corresponding to the three states: crushing, shaking, and calm. These labels will be stored in another dataset. This dataset will have 467 rows (corresponding to the number of samples) and 1 column (for storing the labels). In this way, the two datasets can be matched row by row, linking each sequence with its corresponding label.

IV.II Feature Extraction and Selection

In the field of vibration signal analysis, time-domain and frequency-domain features are the two most commonly used signal processing methods. In existing research, time-domain features are primarily used to capture overall signal patterns, fluctuations, and instantaneous changes, while frequency-domain features help identify the periodic components and frequency distributions hidden within the signal. Time-domain features such as mean, variance, kurtosis, skewness, and RMS (Root Mean Square) provide an intuitive description of the vibration signal's amplitude and fluctuations, and are often used to identify the basic characteristics and trends of the signal. Frequency-domain features, on the other hand, convert the time-domain signal into the frequency domain through methods such as Fourier Transform (FFT), revealing the hidden frequency components in the signal. This helps in identifying different vibration modes or the natural frequencies of the system. By combining both time-domain and frequency-domain features, a more comprehensive analysis and understanding of the dynamic behavior of the vibration signal can be achieved, providing richer input features for pattern recognition and machine learning model training.



Therefore, this project chooses to process the raw signal data by extracting sixteen time-domain features (Mean value 、 Mean of the absolute value, Standard deviation, variance, square root, amplitude, Root mean square, peak value, Maximum value, Minimum value, Shape factor, Peak, factor , Impulse factor, Margin factor , Skewness, Kurtosis, Zero-crossing count) and five frequency-domain features (Power Spectral Density, Spectral Centroid, Bandwidth, Instantaneous, Frequency, Spectral Roll-off) for subsequent research. The project obtain the time-domain features by calculation, and then convert the signal into the frequency domain using Fourier transform.

In order to reduce their complexity without affecting the classification performance of the model, this project performs feature selection on the 21 time-domain features and frequency-domain features mentioned above.

This project uses Recursive Feature Elimination (RFE) to calculate the importance of all features[17]. In this method, the random forest model is employed to determine the top 9 most important features.

The features ranked by importance are as follows: Zero Crossings, Spectral Bandwidth, Spectral Roll-off, Standard Deviation, Energy, Kurtosis, Skewness, Spectral Centroid, and Variance. Among them, Zero Crossings, Spectral Bandwidth, Spectral Roll-off, and Standard Deviation have an importance score exceeding 0.15. Therefore, these four features are selected as the input for the subsequent model.

V MODELINGAND RESULTS

V.I Machine learning model

V.I.I LDA Model

In this experiment, a Linear Discriminant Analysis (LDA) model was used to classify data into different categories. The time-domain and frequency-domain features selected above were used as inputs to the model for signal classification.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

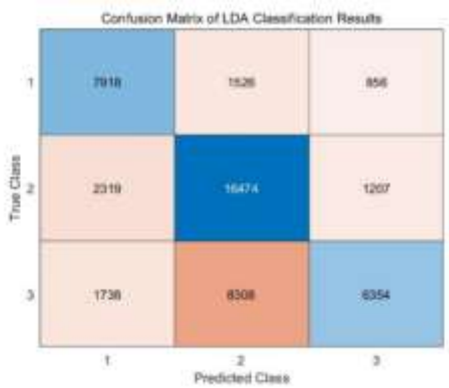


Figure 8 Confusion Matrix of LDA Model

To evaluate the performance of the model, classification accuracy, precision, recall, and F1 score for each category were calculated. Additionally, a confusion matrix for the model was plotted.

Class	Precision	Recall	F1-score
Class 1	0.66	0.77	0.71
Class 2	0.63	0.82	0.71
Class 3	0.75	0.39	0.51
Overall	0.68	0.66	0.64

Table 1 LDA Model Performance Table

The classification accuracy is 65.84%, indicating that the model's ability to predict the sample categories is acceptable. However, this accuracy is only a rough indicator of the overall classification performance and does not fully reflect the model's performance on each category. Specifically, the performance of Class 1 and Class 2 is relatively good, with high precision and recall. Class 1 has a precision of 0.66, a recall of 0.77, and an F1-score of 0.71; Class 2 has a precision of 0.63, a recall of 0.82, and an F1-score of 0.71. This indicates that the model is relatively good at identifying these two categories, especially with high recall, capturing more samples that truly belong to these categories.

However, the classification performance for Class 3 is relatively poor. Although its precision is 0.75, the recall is only 0.39, and the F1-score is 0.51. This indicates that the model has significant issues in identifying Class 3 samples. The low recall means that many true Class 3 samples are misclassified into other categories, leading to a high rate of missed detections for Class 3.

Through further analysis of the confusion matrix, 1,738 samples from Class 3 were misclassified as Class 1, and 8,308 samples were misclassified as Class 2. Since Class 3 has the largest amount of data in the dataset, the possibility that the poor classification performance of Class 3 is due to the relatively small number of samples can be ruled out. Additionally, because most of the misclassified samples from Class 3 were classified as Class 2, it is initially suspected that the model cannot effectively distinguish the features of Class 2 and Class 3.

In summary, the LDA model can perform effective classification to a certain extent, but there are significant differences in the model's performance across different categories. In particular, the classification performance for Class 3 samples is relatively poor.

V.I.II SVM Model

In this project, we explored multi-class classification tasks using a Support Vector Machine (SVM) model. Specifically, we developed a classification system based on time-domain and frequency-domain feature extraction to identify three different categories of data. These three categories of data originate from different experimental scenarios (vehicle crushing, calm, and shaking) and are stored as binary files.

First, we read the experimental data from the folders corresponding to each category and divided it into fixed time periods. Then, we extracted time-domain features and frequency-domain features from each period of data. These features were used as input features for the classification model.

To ensure the generalization ability of the model, we divided the complete dataset into a training set and a test set, accounting for 80% and 20%, respectively.

In constructing the classification model, we selected the Support Vector Machine (SVM) as the base classifier and used a linear kernel function for multi-class classification modeling and training. After completing the model training, we evaluated the classification performance on the test set using a confusion matrix and a series of performance metrics, including precision, recall, and F1 score.

The results are following:
OverallPrecision=0.71,OverallRecall=0.67,Overall F1-score=0.66

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

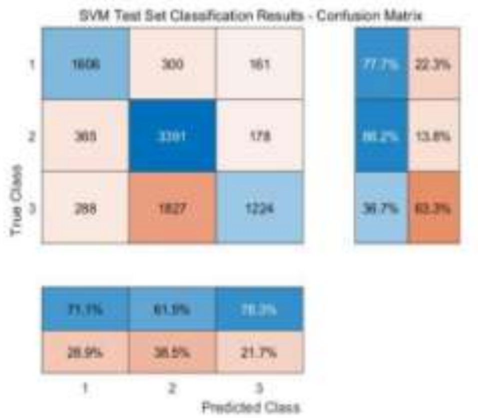


Figure 8 The Confusion Matrix for the SVM Model

The overall accuracy of the SVM model is 66.61%. The results indicate that the classification ability of the SVM model on the test data varies, with moderate overall performance and an F1 score of 0.65. Specifically, the model performs best in classifying category 2, with a recall rate of 86.2%, indicating that most samples belonging to category 2 are correctly identified. However, its precision is only 63%, suggesting that although the recognition ability for category 2 is strong, many samples predicted as category 2 actually belong to other categories. This may be related to the larger number of category 2 samples or differences in feature distribution.

In contrast, the performance issues for category 3 are more pronounced. Although the precision for category 3 is relatively high (78.3%), indicating that most predictions for category 3 are correct, the recall rate is only 36.7%, suggesting that many samples actually belonging to category 3 are misclassified into other categories. This low recall rate directly results in an F1 score of only 0.5 for category 3, the lowest among the three categories. The performance for category 1 falls in between, with a recall rate of 77.7% and a precision of 70%, indicating a more balanced overall classification.

From the overall metrics, the model exhibits imbalanced classification among categories, with a significantly insufficient recognition ability for category 3. This may be related to the uneven distribution of training data, limitations in feature extraction methods, or the parameter settings of the model itself.

The SVM and LDA models each have distinct characteristics in their performance on the classification task. The SVM model outperforms LDA in terms of recall (87%) and precision for Class 2, but it also performs poorly in recall for Class 3 (37%), with significant differences in performance across categories. On the other hand, the LDA model shows more balanced overall performance, with classification performance for Class 1 and Class 2 close to that of the SVM, and similar precision and recall for Class 3, though its overall accuracy is slightly lower than that of the SVM. Both models have similar strengths and weaknesses, and neither performs well in this classification task.

V.IIDeep Learning Machine

We have observed that traditional machine learning models perform poorly on the classification task proposed in this project. In addition to low classification accuracy, there are significant issues with recall for Class 3. Therefore, we sought other models that could address the problems in this project. Here, we chose the LSTM model, GRU model, and CNN model.

V.II.I LSTM Model

For the implementation of the LSTM model, the program loads signal data files from specified folders and extracts time-domain features from each signal sample, including mean, standard deviation, kurtosis, skewness, zero-crossing rate, and signal energy. Through Fourier transform, it extracts three frequency-domain features: spectral centroid, central frequency position, spectral roll-off, and spectral bandwidth. Additionally, the program calculates the time-frequency features of the signal using Short-Time Fourier Transform (STFT) to further capture the variations in time and frequency. These features are used to convert the raw signals into numerical feature vectors that can be processed by the LSTM model as input.

After feature extraction, the program standardizes the data to scale all features to a range with a mean of 0 and a standard deviation of 1, reducing the impact of feature value range differences on model training. Subsequently, the standardized data is reshaped into a three-dimensional format (number of samples, time steps, number of features) to fit the input requirements of the LSTM model. The data is divided into training, validation, and test sets for model training and evaluation, ensuring the reliability and generalization ability of the model's performance.

DOI: 10.35629/5252-0707801826 |Impact Factorvalue 6.18| ISO 9001: 2008 Certified Journal

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

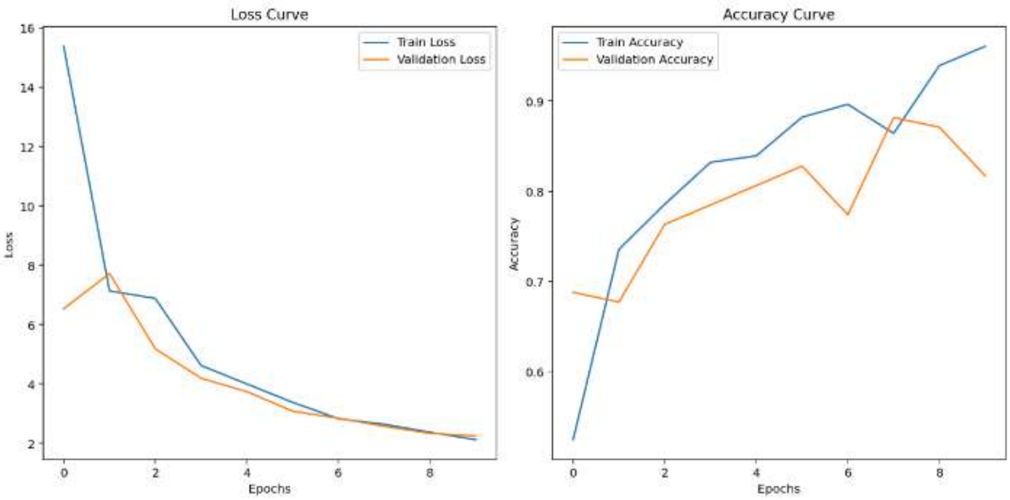


Figure 10 Loss and Accuracy Curves of the LSTM Model

The program uses a Long Short-Term Memory (LSTM) network as the classification model, designing a deep learning architecture with three LSTM layers. The first LSTM layer contains 256 neurons to capture the temporal dependencies of the input sequence; the second and third layers have 128 and 64 neurons, respectively, to progressively extract higher-order temporal features. The model is followed by a fully connected (Dense) layer to further map the extracted features, and the Softmax activation function in the output layer predicts the three-class classification results. The model employs L2 regularization to prevent overfitting and uses the Adam optimizer for efficient training.

The program trains the model on the training set while using the validation set to monitor model performance and plot loss and accuracy curves to observe the model's convergence and generalization ability. The test set is used to evaluate the final performance. The program analyzes the classification results through a confusion matrix, showing the model's classification effectiveness across different categories, and calculates the overall accuracy to measure the model's practical performance. Additionally, the results are visually represented.
The calculated test accuracy is approximately 71.28%

The results indicate that the overall performance of the model is relatively good. The loss curves show that both training and validation losses exhibit a downward trend and eventually converge, suggesting that the model does not have significant overfitting issues on the training and validation sets. However, the accuracy curve on the right shows considerable fluctuations in validation accuracy, especially in the later stages of training, indicating that the model still has shortcomings in generalization ability. These fluctuations may be due to a high learning rate, insufficient feature discrimination, or slight differences in the distribution between the validation and training sets.

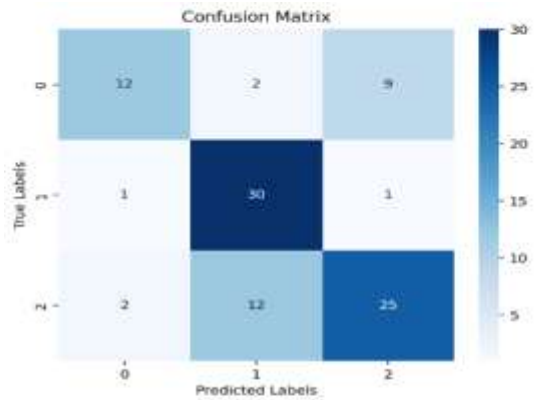


Figure 11 The Confusion Matrix for the LSTM Model

The confusion matrix further reveals that the classification performance for category 1 is excellent, with almost no confusion, while the classification performance for categories 0 and 2 is relatively poor. Particularly for category 2, a large number of samples are misclassified as category 1. This suggests that there may be feature overlap between categories 2 and 1, making it difficult for the model to distinguish between them. Additionally, a significant proportion of category 0 samples are misclassified as category 2, further indicating insufficient feature discrimination between categories. This could be due to imbalanced category distribution or the current features not adequately capturing the differences between categories.

V.II.II GRU Model

The previously introduced LDA, SVM, and LSTM models all used time-domain and frequency-domain features as inputs. It was found that the classification abilities of these models were not satisfactory. Considering the limitations of using these features for classification tasks, in the following introduction of the GRU and CNN models, we will treat the signal data as a time series and input it into the models for classification.

The GRU model constructed in this project is a deep learning classifier based on Gated Recurrent Units (GRU), consisting of two GRU layers, each with 32 hidden units, to extract temporal features from the input signals. The input data shape is (batch_size, time_steps, input_features). The model processes the time series and extracts features from the hidden state of the last time step. These features are then mapped to a probability distribution over three categories (multi-class classification task) through a fully connected (Dense) layer. The model supports Dropout regularization to prevent overfitting and uses the cross-entropy loss function and Adam optimizer for training, efficiently learning the temporal dependencies in the data to complete the classification task.

The steps for implementing the classification task with this model are as follows: The model takes time series signal data as input, uses GRU layers to extract temporal dependency features from the sequence, and obtains a fixed-dimensional feature representation from the hidden state of the last time step. This representation is passed to a fully connected (Dense) layer, which maps it to the number of categories (3 classes). Finally, the Softmax activation function generates a probability distribution for each sample belonging to each category, and the category with the highest probability is chosen as the classification result. The model optimizes the predicted probability distribution against the true class labels using the cross-entropy loss function, enabling it to accurately fit the multi-class classification task.

Model evaluation involves calculating classification accuracy, confusion matrix, and classification report on the test set to comprehensively analyze model performance. During the testing phase, the predicted results for each sample are compared with the true labels to calculate overall accuracy. The confusion matrix visually displays the classification situation for different categories, including correct and incorrect classifications. Additionally, the classification report provides precision, recall, and F1 score for each category.

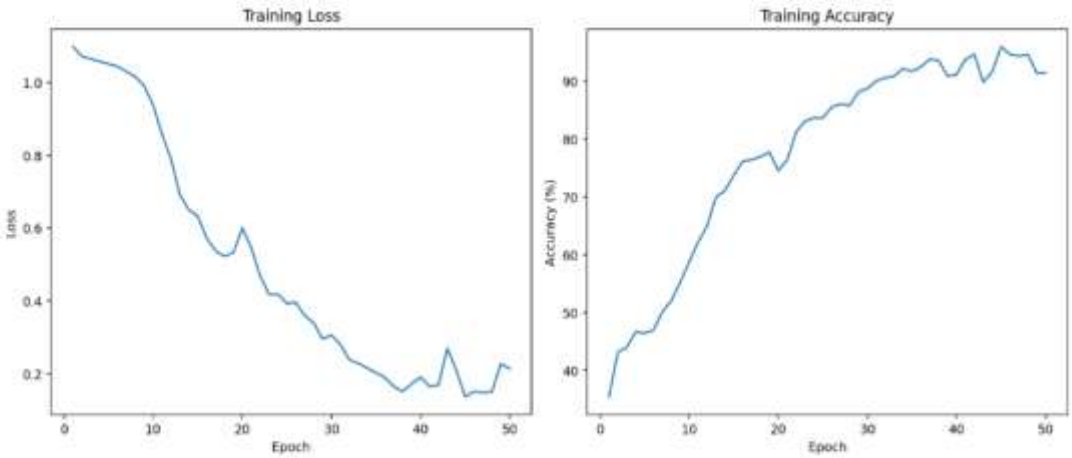


Figure 9 Loss and Accuracy Curves of the GRU Model

Firstly, from the classification report, the model's performance varies significantly across the three tasks. The classification effect of State2 is the best, with precision, recall, and F1 scores all exceeding 0.9, indicating that the model can effectively identify this category. In contrast, the classification effect of State1 is significantly weaker, with a recall rate of only 0.43 and an F1 score of just 0.47, reflecting that the model has not sufficiently learned the features of this category and tends to misclassify its samples as other categories. The performance of State3 is between the two; although the overall accuracy is acceptable,

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

there is still some misclassification, especially with significant confusion with State1.

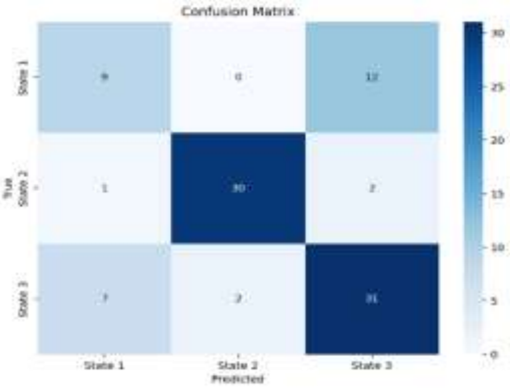


Figure 10 The Confusion Matrix for the GRU Model

The training curves further demonstrate the model's performance during the training process. From the loss curve, we can see that the model's loss value drops rapidly in the early stages of training, indicating that the model can quickly learn features and reduce errors. It then gradually stabilizes, suggesting that the model is approaching convergence and has reached a good training state. The accuracy curve shows that the training set's accuracy eventually approaches 90%, indicating that the model has strong classification ability on the training data. However, due to possible differences in the distribution between the training set and the test set, this high accuracy is not fully reflected in the test results, leading to an overall classification accuracy of 74%.

	precision	recall	f1-score	support
State1	0.53	0.43	0.47	21
State2		0.91	0.92	33
State3		0.78	0.73	40
accuracy			0.74	94
macroavg	0.72	0.7	0.71	94
weightedavg	0.74	0.7	0.74	94

Table 2 Evaluation Table for GRU Model

From the confusion matrix, the model's recognition of State2 is very stable, with most samples correctly classified and only a few misclassified into other categories. However, for State1 and State3, there is significant confusion in classification. For example, many samples of State1 are incorrectly classified as State3, and some samples of State3 are misclassified as State1. This confusion may be due to a large overlap in the feature space of these two categories, making it difficult for the model to extract distinctly differentiating features.

Overall, the model shows good convergence during the training process and demonstrates excellent classification ability on State2. However, the distinction between State1 and State3 is still insufficient, especially with the low recall rate of State1, indicating that the model's ability to recognize this category needs improvement.

This phenomenon is believed to be caused by the imbalance in sample quantities. Therefore, this project proposes two solutions to improve the model: undersampling and adding a weighted loss function.

V.II.II.I GRU Model with Weighted Loss Function

In the new program, a weighted cross-entropy loss function is introduced to balance the impact of sample imbalance on model training. The new program calculates weights based on the number of samples in each category in the training set. The category weights are inversely proportional to the number of samples, meaning categories with fewer samples are given higher weights, making them more significant in the loss calculation. Compared to the previously used standard cross-entropy loss function (which treats all samples equally), the weighted cross-entropy amplifies the penalty for misclassifying minority categories while reducing the weight of the majority categories. This improvement allows the model to focus more on learning features of the minority categories, thereby enhancing the recall

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

and F1 scores of the minority categories and improving classification balance.

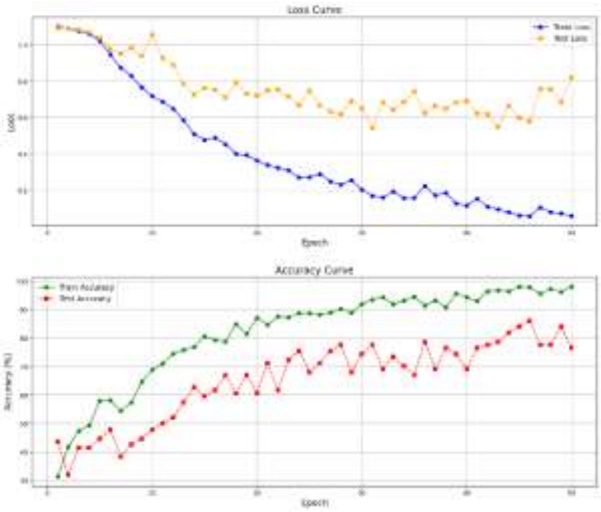


Figure 11 Loss and Accuracy Curves of the GRU Model with Weighted Loss Function

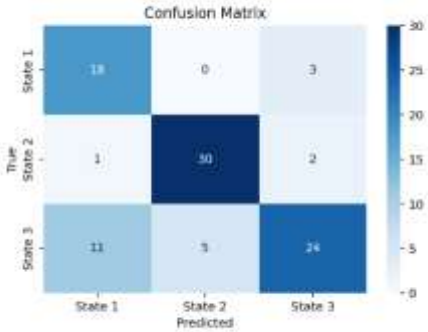


Figure 12 The Confusion Matrix for the GRU Model with Weighted Loss Function

Class	Precision	Recall	F1-Score	Support
State1	0.6	0.86	0.71	21
State2	0.86	0.91	0.88	33
State3	0.83	0.6	0.7	40
Accuracy			0.77	94
Macro Avg	0.76	0.79	0.76	94
Weighted Avg	0.79	0.77	0.76	94

Table 3 Evaluation Table for GRU Model with Weighted Loss Function

From the results:

The classification report shows that the overall accuracy of the model is 77%, which is a slight improvement compared to previous results. Notably, the recall rate for State1 has increased from a previously low level to 0.86, with a precision of 0.60. This indicates that the model can identify more samples of State1, though some misclassification still occurs. The classification effect for State2 remains the best, with precision and recall rates of 0.86 and 0.91, respectively, indicating strong differentiation ability for this category. The performance for State3 is relatively unstable, with a recall rate of only 0.60, suggesting that there are still many misclassifications for this category. The macro-average and weighted-average F1 scores are 0.76 and 0.77, respectively, indicating that the model's classification is relatively balanced overall, but further optimization is needed for minority categories.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

From the loss curve and accuracy curve, the training loss continues to decrease and eventually approaches convergence, with training accuracy stabilizing around 95%, indicating good performance on the training set. However, the test loss fluctuates significantly throughout the process and does not show a clear convergence trend, with test accuracy oscillating between 75% and 80%. This phenomenon suggests that the model's generalization ability on the test set still needs improvement.

The confusion matrix further illustrates the classification effects. The performance of State2 is the most stable, with most samples correctly classified and only a few misclassified into other categories. The confusion between State1 and State3 remains significant; for example, 11 samples of State3 were misclassified as State1. This indicates that although the introduction of the weighted loss function has improved the performance of minority categories, the distinction between State1 and State3 is still unsatisfactory.

Overall, after introducing the weighted loss function, the recall rate for minority categories has significantly improved, especially the classification performance of State1. However, the fluctuations in test loss and accuracy indicate that the model's generalization ability still needs optimization.

V.II.II.II GRU Model with Under sampling Method

We use the undersampling method for groups with a larger number of samples to address the issue of classification imbalance. The new program determines the number of samples in each category in the training set, identifies the category with the fewest samples as the baseline, and randomly samples the same number of samples from other categories to create a balanced training set. Specifically, we use PyTorch's Subset to construct a training set containing the undersampled samples and use DataLoader to load the balanced data for model training, while keeping the original distribution of the test set unchanged to evaluate model performance.

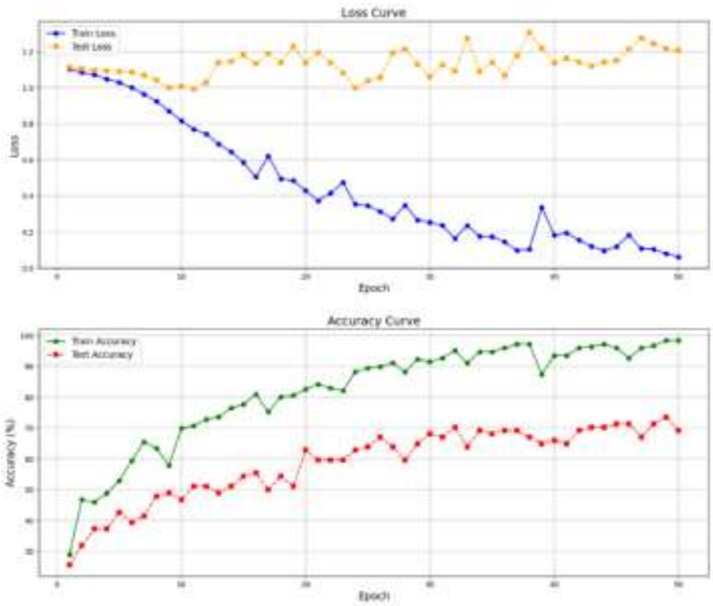


Figure 16 Loss and Accuracy Curves of the GRU Model with Undersampling Method

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

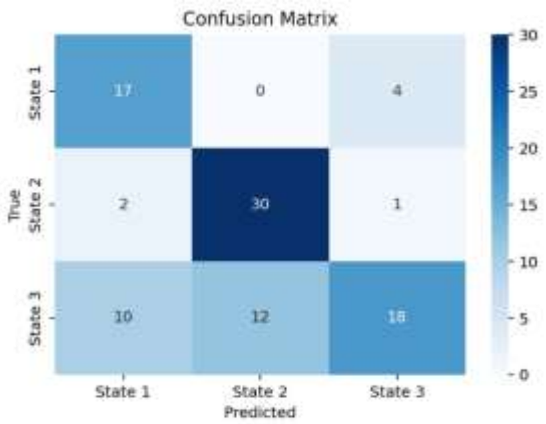


Figure 17 The Confusion Matrix for the GRU Model with Undersampling Method

Class	**Precision**	**Recall**	**F1-Score**	**Support**
State1	0.59	0.81	0.68	21
State2	0.71	0.91	0.8	33
State3	0.78	0.45	0.57	40
Accuracy			0.69	94
MacroAvg	0.69	0.72	0.68	94
WeightedAvg	0.71	0.69	0.68	94

Table 4 Evaluation Table for GRU Model with Undersampling Method

From the results of the new model, the recall rates for State1 and State2 have improved, especially with State1's recall rate reaching 0.81, indicating that undersampling is effective in enhancing the classification performance of minority categories. However, the recall rate for State3 is relatively low at only 0.45, suggesting that undersampling has led to a decline in the performance of the majority category. The overall accuracy is 69%, which is lower than the 77% of the weighted loss model but slightly worse than the original model's 71%. The test curves show significant fluctuations in test loss, and test accuracy did not improve significantly during training, indicating limited generalization performance. The confusion matrix shows reduced misclassification for State1 and State2, but significant misclassification remains for State3. Compared to the previous two models, the undersampling method focuses more on minority categories but at the cost of the majority category's performance, resulting in a less balanced overall classification effect than the weighted loss model.

V.II.III CNN Model

"First, store all the signal data in a two-dimensional matrix. Each signal data has 186,800 points, which can be considered as a sequence of length 186,800. Since there are 467 samples, the two-dimensional matrix has the form (467, 186800). To make this matrix suitable for the input of a CNN model, use the statement np.expand_dims to add a channel dimension to the signal data, adapting it to the input format of the CNN model. Since the original data is single-channel, the number of channels is 1, so the input matrix should be in the form (number of samples, signal length, 1).

This CNN model consists of multiple convolutional layers, pooling layers, fully connected layers, and a classification layer. The first layer of the model is a Conv1D convolutional layer, which uses 32 filters, each with a kernel size of 3, and extracts local features from the signal using the ReLU activation function. The convolution operation can capture local patterns in the signal within a small window (length of 3), such as fluctuations in the signal. Next, a MaxPooling1D layer downsamples the convolutional features with a pooling window size of 2, significantly reducing the feature length while retaining the most important information. The model then adds a second Conv1D convolutional layer, this time using 64 filters with the same kernel size of 3, and extracts higher-level, more abstract signal features using the ReLU activation function. This layer is followed by another MaxPooling1D layer with the same pooling window size of 2, further reducing the dimensionality of the features and ensuring the model focuses on key signal information. After two rounds of convolution and pooling operations, the extracted features are passed to a Flatten layer, which flattens the three-dimensional feature maps into a one-dimensional feature vector to fit the input format of the fully connected layer. The flattened feature vector enters a fully connected layer with 128 neurons, using the ReLU activation function, responsible for non-linearly combining the extracted signal features to generate more complex feature representations. To prevent overfitting, especially with limited training data, the model includes a Dropout layer that drops 50% of the neurons, ensuring the model does not overly rely on specific neurons, thereby enhancing generalization ability. Finally, the model's output layer is a Dense fully connected layer with 3 neurons, corresponding to the three labels of the signal classification task. Using the Softmax activation function, it outputs the probability distribution for each category, representing the model's confidence in each category prediction, and the category with the highest probability is taken as the classification result. The model uses the cross-entropy loss function (sparse_ categorical_ crossentropy) as the optimization target, combined with the Adam optimizer to dynamically adjust the learning rate, improving the model's convergence speed and stability during training.

In summary, this CNN model gradually extracts local and global features by stacking convolutional layers. It combines fully connected layers and a Softmax classification layer to map the signal data to three labels, thereby achieving the one-dimensional signal classification task.

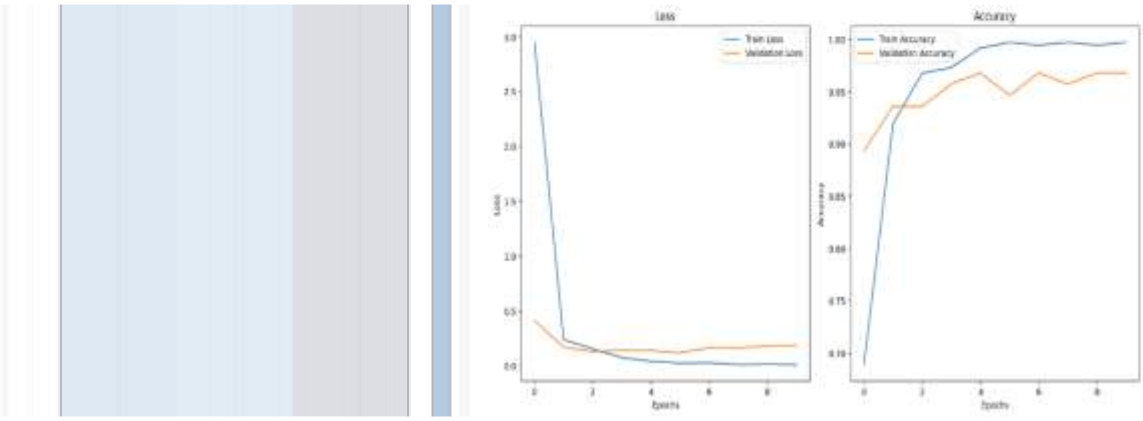


Figure 18 Loss and Accuracy Curves of the CNN Model

The training loss of this model quickly decreased from an initial 2.9491 to 0.0095, and the training accuracy increased from 68.9% to 99.73%, indicating that the model effectively learned the

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 801-826 www.ijaem.net ISSN: 2395-5252

patterns in the training data. The validation loss also continuously decreased and stabilized at 0.1852, with a validation accuracy of 96.81%, demonstrating the model's generalization ability on unseen data. The validation loss curve is stable and lower than the training loss, showing no signs of overfitting. The gap between training and validation results is only 2-3%, which is within the normal range. From the confusion matrix, the model's classification for Class1 and Class2 is almost perfect, with no misclassified samples, indicating that the model has thoroughly learned the features of these two signal classes.

Overall, the model's performance on both the training and validation sets is consistent, with rapid convergence and high accuracy. This suggests that the CNN model is an efficient and reliable tool for signal classification.

To further reduce the gap between training and validation results, attempt some improvements to the model.

V.II.III.I Improved CNN Model

In the convolutional layers, introduce progressively increasing intensity of Dropout to build a gradual feature filtering system. Start with 20% low-intensity filtering in the first convolutional layer and increase to 30% filtering intensity in the second layer.

In the fully connected layers, use a deep architecture design by adding an intermediate layer with 64 neurons after the main layer with 128 neurons, and apply high-intensity Dropout of 50% and 40% to construct a more detailed feature transformation network.

Increase the number of training epochs to 200

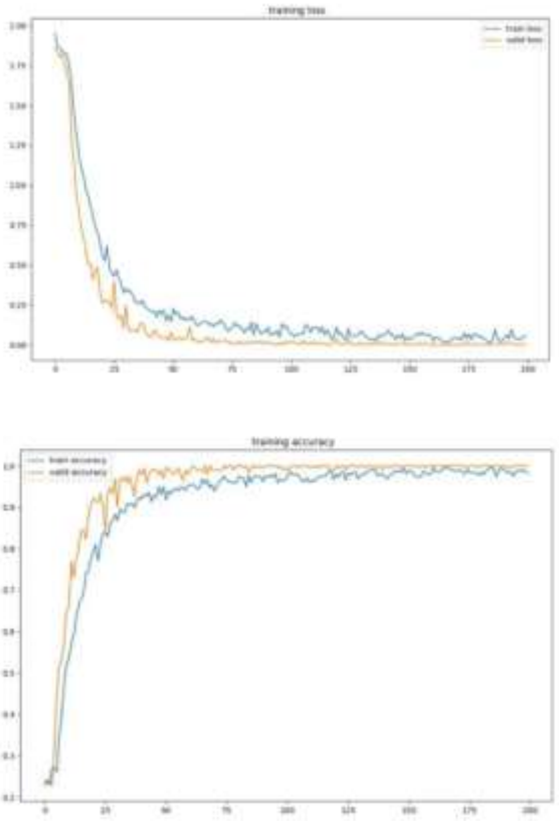


Figure 20 Loss and Accuracy Curves of the Improved CNN Model

The results obtained are as follows:

The improved CNN model performs well, with a test accuracy of 0.9574 and a training accuracy close to 1.0. The training process converges quickly, and the loss and accuracy on the validation set are stable, indicating a significant improvement in the model's generalization ability. From the loss curves, we can see that both training and validation losses decrease rapidly and stabilize, with validation loss consistently lower than training loss, possibly due to optimization strategies or regularization methods (such as Dropout).

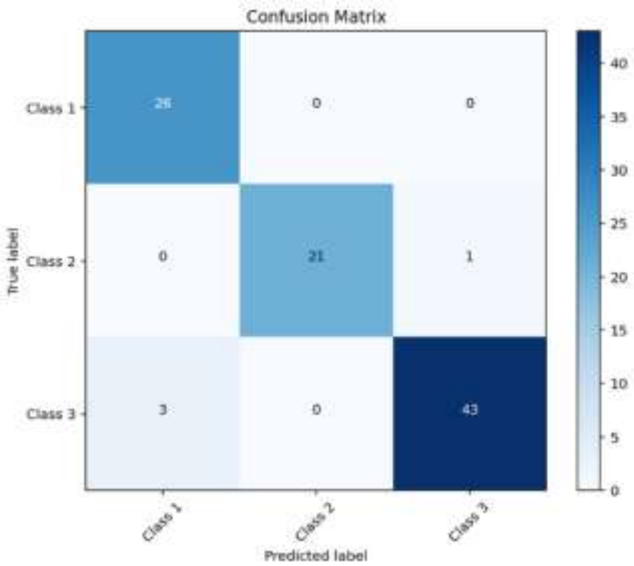


Figure 13 The Confusion Matrix for the Improved CNN Model

However, it is important to note that despite the overall excellent performance of the improved model, there are noticeable oscillations in the loss and accuracy curves for both training and validation. These oscillations may be due to a high learning rate, data characteristic fluctuations, or high model complexity. Although the final curves tend to converge and the performance on validation loss and accuracy does not degrade significantly, this unstable optimization process may indicate that the model's training in local regions is not stable enough, especially in the mid-to-late stages of training (e.g., 150-200 epochs). These fluctuations could affect the model's performance on the test set.

In summary, although the improved model has minimal risk of overfitting, the oscillations in the loss and accuracy curves and the lack of significant improvement in test accuracy lead us to choose the original CNN model.

VI DISCUSSION

The main objective of this study is to develop and evaluate various machine learning and deep learning models for classifying electrical signals collected by fiber optic sensors under three different conditions: calm, shaking, and pressing. Through systematic signal processing, feature extraction and selection, as well as model construction and optimization, this paper thoroughly explores the performance and advantages and disadvantages of different algorithms in this classification task.

In the signal processing stage, this study extracted 16 time-domain features and 5 frequency-domain features from the raw electrical signals. Time-domain features include mean, standard deviation, zero-crossing rate, etc., which reflect the basic statistical characteristics of the signals. Frequency-domain features, extracted through Fourier transform, such as spectral bandwidth and spectral roll-off, reveal the frequency components of the signals. To mitigate the impact of high-dimensional features on model performance and improve computational efficiency, a recursive feature elimination (RFE) method based on random forests was used. This process ultimately selected four key features: zero-crossing rate, spectral bandwidth, spectral roll-off, and standard deviation. This not only reduced the feature dimensionality and computational complexity but also enhanced the interpretability and generalization ability of the model.

First, the support vector machine (SVM) and linear discriminant analysis (LDA) models were evaluated. The SVM model achieved an overall accuracy of approximately 66.61%, but there were significant differences in performance across different categories. Specifically, the SVM performed well in identifying category 2 (pressing) with a recall rate of 86.2%, but had a recall rate of only 36.7% for category 3 (shaking), resulting in a low F1 score for this category. This indicates that the SVM model has certain limitations in handling class imbalance and feature overlap issues, particularly in distinguishing between category 1 (calm) and category 3. Although the LDA model showed more balanced performance across categories, its overall accuracy was slightly lower than that of the SVM. This suggests that traditional machine learning models face performance bottlenecks in high-dimensional signal classification tasks, especially when dealing with complex class distributions.

The LSTM model improved accuracy to approximately 71.28% by capturing the temporal dependencies in the signals. However, despite the advantages of LSTM in handling long sequence data, the model still experienced confusion in distinguishing between category 1 (calm) and category 3 (shaking) in practical applications, with many category 3 samples being misclassified as category 1. This indicates that the LSTM model may have deficiencies in feature representation, or the model parameters have not been sufficiently optimized to differentiate these similar categories.

After introducing a weighted loss function, the accuracy of the GRU model further improved to 77%. The weighted loss function significantly increased the recall rate of the minority class (State 1) to 0.86 by assigning higher weights to this class. However, despite the improved recognition ability for the minority class, the recall rate for category 3 (shaking) remained low at only 0.60. This indicates that although the weighted loss function alleviated the class imbalance issue to some extent, the model still faces challenges in distinguishing between category 1 (calm) and category 3.

The convolutional neural network (CNN) performed the best in this study, achieving a validation accuracy of 96.81%. The CNN effectively captured both local and global features of the signals through its convolutional layers, particularly excelling in the classification of category 1 (calm) and category 2 (pressing). Although higher intensity Dropout and additional intermediate fully connected layers were introduced to optimize the model structure, these improvements did not significantly enhance model performance and instead introduced some instability. Therefore, the original CNN model, which achieved the best balance between complexity and performance, was considered the most effective model.

This study found that deep learning models, especially CNNs, have significant advantages in high-dimensional vibration signal classification tasks by comparing different types of models. While traditional machine learning models perform well in certain tasks, deep learning models can better capture the complex patterns and latent features of signals when dealing with complex feature distributions and class imbalances, thereby improving classification performance.

Although this study has achieved certain results, there are still some limitations. First, the sample size is relatively limited, which may affect the generalization ability of the models. Second, this study only used a few feature extraction methods. In the future, more advanced feature extraction and dimensionality reduction techniques, such as autoencoders or principal component analysis (PCA), can be explored to further optimize feature representation. Additionally, model optimization strategies such as hyperparameter tuning and the application of different regularization methods are also worth further investigation. Future research can be carried out in the following aspects:

(1) Data Augmentation and Expansion: Future research can improve model performance by increasing the amount of data, especially balanced samples for each category. By using synthetic data generation techniques (such as SMOTE) or collecting more actual data, particularly minority class samples, the training data can be enhanced, thereby improving the model's generalization ability.
(2) Improvement of Data Extraction Methods: Additionally, this study only used a few feature extraction methods. In the future, more advanced feature extraction and dimensionality reduction techniques, such as autoencoders or principal component analysis (PCA), can be explored to further optimize feature representation.
(3) Model Architecture Optimization: Explore more complex deep learning models, such as residual networks (ResNet) and attention mechanisms, to further enhance classification performance and model robustness.
(4) Real-time Classification and Deployment: Research how to deploy trained models into actual sensor systems to achieve real-time signal classification and monitoring, enhancing the system's practicality and response speed.

In summary, this study not only validated the application effects of various machine learning and deep learning models in high-dimensional vibration signal classification but also provided valuable experience and direction for future research. By continuously optimizing model structures and data processing methods, it is expected to achieve more efficient and accurate signal classification and monitoring systems in practical applications, further advancing research on deep learning-based fiber optic vibration anomaly detection.

VII CONCLUSION

This study successfully developed and evaluated various machine learning and deep learning models for classifying electrical signals collected by fiber optic sensors. Through feature extraction and selection, model construction, and optimization, it was found that convolutional neural networks (CNN) performed the best in terms of classification accuracy and generalization ability, with a validation accuracy of 96.81%. Although traditional models like support vector machines (SVM) and linear discriminant analysis (LDA) performed well in certain categories, their overall performance was limited in the presence of class imbalance and feature overlap. Deep learning models such as long short-term memory networks (LSTM) and gated recurrent units (GRU) improved classification accuracy by capturing temporal dependencies but still faced difficulties in distinguishing between category 1 (pressing state) and category 3 (calm state). The study indicates that deep learning, especially CNN models, have significant advantages in processing and classifying high-dimensional vibration signal data, providing support for the identification of actual fiber optic cable states. Future research should further optimize model structures, expand dataset sizes, and explore more advanced feature extraction and optimization methods to enhance model performance.

REFERENCES

[1] Sun, H. (2023). Research on Signal Analysis Technology of Fiber Optic Perimeter Security System Based on Deep Neural Networks. Master's Thesis, Beijing University of Posts and Telecommunications. doi:10.26969/d.cnki.gbydu.2023.002764.

[2] Li, H., & Liu, J. (2024). Comparative Analysis of LDA Classification Models and Improved Classification Models. Journal of Jilin Normal University (Natural Science Edition), 45(02), 60-68. doi:10.16862/j.cnki.issn1674-3873.2024.02.010.

[3] Zhang, F., Huang, L., & Lü, J. (2024). Anomaly Detection Model for Power Time-series Data Based on ConvLSTM Network. Computer Knowledge and Technology, 20(24), 40-42. doi:10.14004/j.cnki.ckt.2024.1232.

[4] Nizam, Muhammad, et al. "Using support vector machine for prediction dynamic voltage collapse in an actual power system." International Journal of Energy and Power Engineering 2.5 (2008): 1037-1042.

[5] Ribeiro, Evandsa Sabrine Lopes-Lima, and René Alquézar Mancho. "Incremental construction of LSTM recurrent neural network." (2002).

[6] Wu Haitao & Bian Yafang. (2024). Longitudinal Directional Protection of Transmission Lines in Power Systems Based on Fourier Algorithm. Wireless Internet Technology, (20), 91-94.

[7] Sun Feifei, Cao Zhuo & Xiao Xiaolei. (2014). Application of Random Forest Classifier in Crime Prediction. Journal of Information, (10), 148-152.

[8] Yang Jing & Lin Shiping. (2011). Sentiment Analysis of Text Sentences Based on SVM. Computer Applications and Software, (09), 225-228.

[9] Zhang Xiaoyan, Wang Ting & Liang Xiaobo. (2011). Application of LDA Model in Topic Tracking. Computer Science, (S1), 136-139+152.

[10] Li Bing, Zhang Yan & Liu Shi. (2018). Research on Short-term Wind Speed Prediction Based on LSTM. Computer Simulation, (11), 456-461.

[11] Yang Li, Wu Yuxian, Wang Junli & Liu Yili. (2018). A Review of Research on Recurrent Neural Networks. Computer Applications, (S2), 1-6+26.】

[12] Yao Chengwen, Yang Ping & Liu Zejian. (2020). Load Forecasting Method Based on CNN-GRU Hybrid Neural Network. Power System Technology, (09), 3416-3424. doi:10.13335/j.1000-3673.pst.2019.2058.

[13] Huang Lei & Du Changshun. (2017). Research on Text Classification Based on Recurrent Neural Networks. Journal of Beijing University of Chemical Technology (Natural Science Edition), (01), 98-104. doi:10.13543/j.bhxbzr.2017.01.017.

[14] Zhou Feiyan, Jin Linpeng & Dong Jun. (2017). A Review of Research on Convolutional Neural Networks. Journal of Computer Science, (06), 1229-1251.

[15] Chang Liang, Deng Xiaoming, Zhou Mingquan, Wu Zhongke, Yuan Ye, Yang Shuo & Wang Hong'an. (2016). Convolutional Neural Networks in Image Understanding. Acta Automatica Sinica, (09), 1300-1312. doi:10.16383/j.aas.2016.c150800.

[16] Fang Luping, He Hangjiang & Zhou Guomin. (2018). A Review of Research on

Object Detection Algorithms. Computer Engineering and Applications, (13), 11-18+33.

[17] Wu Chenwen, Liang Jinghan, Wang Wei & Li Changsheng. (2017). Random Forest Algorithm Based on Recursive Feature Elimination Method. Statistics and Decision, (21), 60-63. doi:10.13546/j.cnki.tjyjc.2017.21.014.