INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 21, Issue: 2, Month: April, Year: 2026
Article Number: 7068, https://doi.org/10.15837/ijccc.2026.2.7068

CCC Publications

AGORA UNIVERSITY PRESS

Computer Network Fault Diagnosis Based on improved TLBO and MBGD Optimization

J.H. Tan

Tan Jiang Hui
School of Computer Science and Technology
Geely University, China
Building 5, No. 1504, Vanke Park Metropolitan South Area, Luhu South Road, Tianfu New Area, Chengdu City, Sichuan Province, China, 610213
919939485@qq.com

Abstract

With the increasing complexity and frequency of computer network faults, efficient fault diagnosis is crucial to the reliability and security of the network. To this end, this paper proposes an advanced fault diagnosis method based on the improved Teaching Learning-based Optimization (TLBO) algorithm and the Mini Batch Gradient Descent (MBGD) algorithm under the framework of Convolutional Neural Network (CNN). Different from the traditional CNN-based methods, this method innovatively integrates the TLBO algorithm with the Differential Evolution (DE) strategy, optimizes hyper-parameters and training convergence, and significantly improves the detection accuracy and speed. Meanwhile, MBGD can effectively refine the model parameters and prevent convergence to local minima. The experimental results using the public dataset prove the effectiveness of this method, achieving a high classification accuracy rate of up to 88.3% and significantly reducing the false detection rate to below 0.20%. Compared with the traditional CNN model and the latest methods, this method has a faster convergence speed, model stability and fault diagnosis performance. This research provides a robust solution for real-time fault detection and diagnosis, significantly enhancing the reliability and security of complex network systems.

Keywords: mini batch gradient descent algorithm, network malfunction, diagnosis, convolutional neural network, hybrid teaching and learning optimization algorithm.

1 Introduction

With the continuous expansion of computer networks and the increasing complexity of their structures, the frequency of network failures and the difficulty of diagnosis are also constantly increasing, which has a significant impact on the economy and society [1, 2]. Hence, the swift and precise identification of the location and underlying cause of network failures, followed by their timely repair, is of paramount importance in minimizing losses. In recent years, with the rapid development of artificial intelligence and machine learning technologies, the Teaching Learning-based Optimization (TLBO) algorithm and the Mini Batch Gradient Descent (MBGD) algorithm have demonstrated excellent performance in network fault diagnosis problems. The TLBO algorithm performs well in solving complex optimization problems due to its simplicity, efficiency, and ease of implementation [3]. The MBGD algorithm accelerates convergence speed by constructing batch sizes and reducing iteration times, while effectively avoiding the problem of moving to local minima [4, 5]. The task of network fault detection and diagnosis requires processing massive real-time data and conducting precise analysis in a very short period of time. The fault may originate from multiple points in the network, so it is necessary to conduct comprehensive analysis of different types of fault signals [6]. The occurrence and development process of faults are full of uncertainty, including the randomness of faults and noise interference in fault signals [7]. Although the TLBO algorithm has shown simplicity and efficiency in solving complex optimization problems, its independent application suffers from slow convergence in high-dimensional search spaces and limited adaptability to dynamic network environments. Meanwhile, although MBGD algorithm accelerates training through batch processing, it brings significant computational complexity and memory overhead when processing large-scale datasets, which limits its practicality in resource constrained scenarios. Therefore, the study aims to address the slow convergence of TLBO and the computational burden of MBGD by integrating the TLBO algorithm with adaptive population dynamics and the MBGD strategy with dynamic batch sizes. This method innovatively improves the TLBO algorithm to enhance the global search capability and convergence speed of the network fault diagnosis model, thereby more effectively identifying and locating network faults. Additionally, by employing the MBGD algorithm, it swiftly and accurately diagnoses fault points within large-scale network data, minimizing instances of misdiagnosis and missed diagnosis.

2 Literature review

Abou El Houda Z et al. proposed a network intrusion system based on quantum federated learning. It utilizes the decentralized nature of federated learning, enabling multiple consumer devices to collaboratively train the global intrusion detection model while protecting the privacy of individual user data. The results show that the detection accuracy and computational efficiency of this method have been significantly improved [8]. Thiruvenkatasamy S et al. proposed a blockchain-assisted fireworks optimization and machine learning-based intrusion detection system. The system incorporated blockchain technology to enable secure data transmission and broadcasting, and optimized intrusion detection performance through three phases. Experiments showed that the system was superior to other latest algorithms in detection performance and could effectively deal with complex network attacks, providing a strong guarantee for the security of Internet of Things network [9]. Lin Y et al. proposed an adaptive system level fault self diagnosis strategy for folded hypercube networks. This strategy performed fault diagnosis on the folded hypercube network under the Maeng-Malek model, and verified the correctness of the algorithm through the fault tolerance of the folded hypercube network. Even in situations beyond the diagnosable range, it could maintain good performance [10]. To enhance the reasoning accuracy of the edge computing environment, Ahmed S T et al. proposed a one-time test method, which accelerated the neural network by testing a single test vector to achieve 100% fault coverage in large topology and challenging tasks. The fault coverage rate was 24% higher than the existing methods, but the memory overhead was increased [11]. To solve the problem that the mobile edge computing network has limited computing power and is prone to failure, Zhang H et al. proposed a fault-tolerant distributed task unloading scheme, which used multi-agent near end strategy optimization algorithm to minimize task execution time and system energy consumption. The numerical experimental results verified that the algorithm could reduce system energy consumption and processing delay while meeting task reliability [12].

Sasirekha et al. improved the color normalization method and constructed an ensemble model to achieve rapid analysis of histopathological images. They also introduced MBGD algorithm and Convolutional Neural Network (CNN) to improve the precision and effectiveness of diagnosing cancer using histopathological images [13]. In response to the problems of slow convergence speed and susceptibility to local minima in the training process of multi-layer and multi-spike neural networks, Ramesh M et al. used the MBGD algorithm to adjust batch size and reduce iteration times to accelerate convergence. The results indicated that the accuracy and learning performance of this method surpassed existing metaheuristic algorithms [14]. Sun M et al. proposed an innovative method for updating deep reinforcement learning target networks, which adjusted the update rate dynamically, utilizing the MBGD algorithm, based on the discrepancy between the main network and the target network, thereby reducing learning latency while maintaining stability. The simulation results showed that the new method had an 18.05% improvement in overall performance compared to traditional methods [15]. Kaushik A et al. proposed a network intrusion detection system based on TLBO algorithm to address the limitations of traditional network intrusion detection systems in terms of detection capability and resource overhead. After extensive testing, this method could effectively detect various network attacks and significantly outperformed existing advanced algorithms in terms of performance [16]. In response to the limitations of traditional classification algorithms in parameter tuning and feature selection in intrusion detection systems, Al Janabi M introduced the TLBO algorithm and then utilized machine learning features to identify and assess potential intrusion patterns through learning and testing. The simulation outcomes showed that the algorithm achieved extremely high accuracy in detecting malicious attacks [17].

In summary, significant progress has been made in fault detection in specific scenarios through existing research. However, three key issues remain unresolved. (1) Scalability limitations: The methods proposed by Abou El Houda Z et al. [8] and Zhang H et al. [12] rely on static network assumptions (such as fixed VM container mappings) and are difficult to adapt to dynamically expanding IoT/IIoT networks. (2) Computational overhead: Technologies such as blockchain assisted optimization by Thiruvenkatasamy S et al. [9] and multi-agent task offloading by Zhang H et al. [12] result in high memory costs, making it difficult to perform real-time diagnostics in TB level data streams. (3) Global local trade-off: Although the methods of Kaushik A et al. [16] and Ramesh M et al. [14] have improved optimization performance, their independent implementations either stagnate in local optima or require too many iterations. To address these challenges, a computer network fault diagnosis model that integrates improved TLBO and MBGD optimization is proposed. This model effectively processes large-scale static datasets through TLBO and MBGD, converts them into data stream blocks, and inputs these data features into CNNs for classification to achieve accurate fault diagnosis.

3 Research methodology

The study first establishes a basic model for computer network fault diagnosis through CNN. Then, the improved TLBO algorithm is used to optimize the hyperparameters of CNN, such as convolution kernel size, stride, and network depth, to optimize the capability of the model. Finally, the MBGD algorithm is employed to refine the model parameters, thereby enhancing convergence speed and mitigating the likelihood of getting trapped in local minima.

3.1 Network fault detection model based on CNN

Neural networks, due to their complex connections of multiple layers of neurons, exhibit excellent learning and adaptability when processing large amounts of data, and can effectively cope with diverse network fault detection tasks [18, 19]. However, the diversity of network fault data, including non-numerical attributes and multidimensional differences, makes it complex to directly apply CNN for classification [20]. The proposed network fault detection architecture is shown in Figure 1.

Before inputting data into a CNN, appropriate preprocessing is required, including feature digitization, dimensionality reduction, and identification and processing of outliers, to ensure the quality of the data and the effectiveness of model training [21]. Through these preprocessing steps, the performance of CNNs in network fault detection can be improved, enabling them to more accurately identify and classify different types of faults. When confronted with the existing features of network faults, not every feature possesses a notable capacity for discrimination. Moreover, it is no simple task to directly discern and eliminate the unimportant features from the original distribution of network fault data. To address this challenge, it is necessary to first perform dimensionality reduction on the original features.

The study used Principal Component Analysis (PCA) as a dimensionality reduction method. For a given feature set $X = \{\vec{x}_1, \vec{x}_2, \vec{x}_3, ..., \vec{x}_n\}$, where $X \in R^{m\times n}$, m represents the original dimension

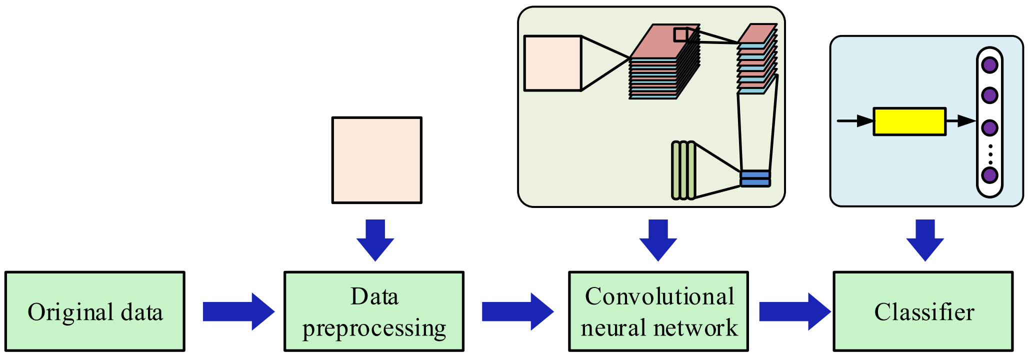

Figure 1: Design of network fault detection architecture based on CNN

of the feature, and n is the number of data points. In the initial stage of dimensionality reduction, each feature is centralized by subtracting its average value from each feature value to obtain a new dataset Y, as shown in equation (1).

$$Y = \{\vec{y}_1, \vec{y}_2, \vec{y}_3, ..., \vec{y}_n\} = \{\vec{x}_1 - \vec{\mu}, \vec{x}_2 - \vec{\mu}, \vec{x}_3 - \vec{\mu}, ..., \vec{x}_n - \vec{\mu}\} \tag{1}$$

In equation (1), $\vec{\mu} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i$ is the average value of the features. Next, the covariance matrix C of the centralized feature set Y is calculated and singular value decomposition is performed on it to determine the eigenvalues λ_i and their corresponding eigenvectors f_i, as shown in equation (2).

$$\begin{cases} C = \dfrac{1}{n} YY^T, C \in R^{m\times m} \\ C \times f_i = \lambda_i \times f_i, i \le m \end{cases} \tag{2}$$

The eigenvalues are sorted based on their size, and the largest k eigenvalues and their corresponding eigenvectors are selected to construct a new eigenvector matrix P, as shown in equation (3).

$$P = \begin{bmatrix} f_1^T \\ f_2^T \\ \cdots \\ f_k^T \end{bmatrix}, P \in R^{k\times m} \tag{3}$$

Finally, the original data X is linearly transformed using the eigenvector matrix P to obtain the reduced dimensional dataset Z, as shown in equation (4).

$$Z = PX, Z \in R^{k\times n} \tag{4}$$

Through this process, PCA not only reduces the dimensionality of the data, but also maintains the maximum separability of the data in the new feature space. To effectively distinguish and binary encode the categories of computer network faults, various encoding strategies are adopted in the study [22]. The entire process flow is shown in Figure 2. To facilitate coding implementation, the data are converted into a form suitable for network processing.

The detailed procedure of the convolution operation in a CNN unfolds as follows: The convolution kernel traverses the input matrix in accordance with a predetermined step size. At each position it occupies during this traversal, it executes a dot-product operation with the pixel values at that specific location. This process can be described using the mathematical definition of discrete convolution, as shown in equation (5).

$$(f * g) \overset{def}{=} \sum_{m=-\infty}^{\infty} f[m]g[n-m] = \sum_{m=-\infty}^{\infty} f[n-m]g[m] \tag{5}$$

In equation (5), f is the input feature map, that is, the preprocessed network traffic matrix. g is the convolution kernel function, which is used for extracting features. m and n respectively represent

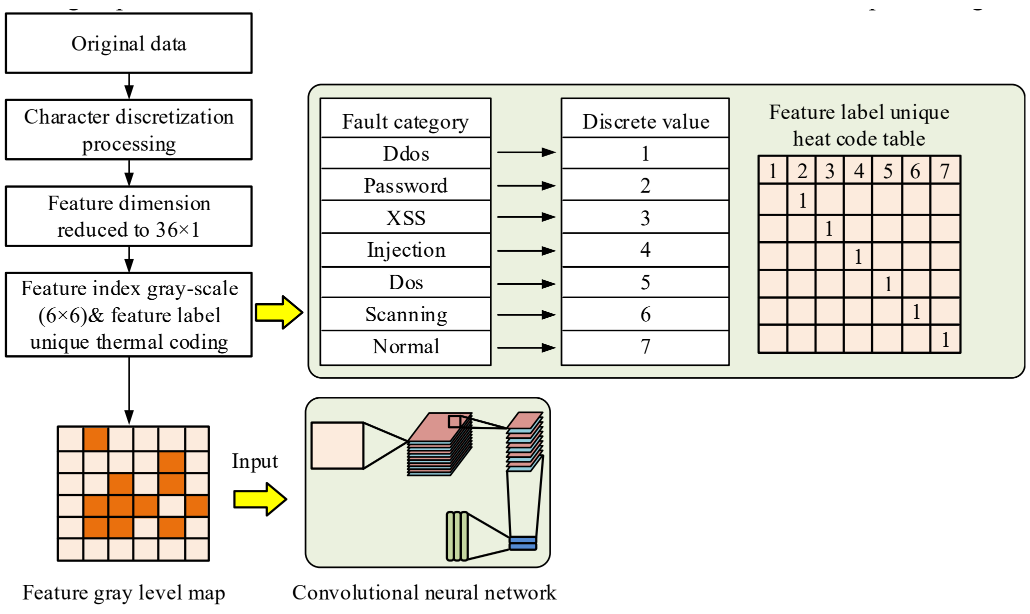
Original data
Character discretization processing
Feature dimension reduced to 36×1
Feature index gray-scale (6×6)& feature label unique thermal coding
Fault category
Ddos
Password
XSS
Injection
Dos
Scanning
Normal
Discrete value
1
2
3
4
5
6
7
Feature label unique heat code table
Input
Feature gray level map
Convolutional neural network

Figure 2: Data preprocessing process

the sliding positions of the convolution kernels on the input matrix. This equation represents the calculation of the weighted sum of local regions through sliding convolution kernels to extract spatial features [23]. In CNN, the implementation of discrete convolution can be expressed by equation (6).

$$(f * g_k) = \sum_{i=0}^{H} \sum_{J=0}^{W} f_{i,j} \cdot (g_k)_{i,j} \tag{6}$$

In equation (6), H and W mean the height and width of the convolution kernel, g_k is the area covered by the convolution kernel when sliding on the input matrix, and $f_{i,j}$ is the value of the convolution kernel at position (i, j). When performing convolution, the convolution kernel window covers every element of the input matrix and extracts local information through dot product operation, which not only compresses the features but also highlights the important features of the input data. Throughout the network's training phase, the convolution kernel and bias parameters undergo continuous updates to enhance the model's performance. Once the training is completed, these parameters constitute the weights of the model and remain unchanged in the model until the next training. The design of each convolutional layer requires careful parameter determination to ensure that the network can effectively learn and extract features. The convolutional layers and required parameters for the research design are in Table 1.

The main reason for choosing the 5×5 convolution kernel in the research is that it achieves an effective balance between local feature extraction and computational efficiency. Larger convolutional kernels can cover a wider input area, capture long-range dependencies (such as burst traffic patterns) in network failure signals, and at the same time avoid the problem of missing key features due to the limited receptive field of too small convolutional kernels [24, 25]. In contrast, overly large convolutional kernels (such as 7×7), although they can further expand the receptive field, will significantly increase the number of parameters and computational complexity, which is not conducive to the deployment of the model in a resource-constrained environment. The pooling layer adopts a 2×2 maximum pooling strategy, aiming to reduce the spatial size of the feature map through dimensionality reduction, thereby reducing the model complexity and suppressing overfitting. Max pooling enhances the model's robustness against noise interference by retaining significant features of local regions, such as abnormal traffic peaks. This design, while ensuring the feature expression ability, optimizes the utilization efficiency of computing resources and is suitable for the real-time processing requirements of large-scale network data.

To iteratively refine the convolution kernel and bias parameters during training, a loss function is essential to quantify the discrepancy between the model's predictions and actual labels. Given that

Table 1: CNN design parameters

Convolutional layer design specific parameters	Input parameter	Original input size	32×32
	Convolution parameter	Convolution kernel size (convolution slide window size)	5×5
		Convolution kernel type (number of channels)	1(there is a bias)
		Convolution step size	28×28 (32-5+1=28)
		Bias parameter	28×28×6
	Output parameter	Feature map output size	(5×5+1)×6 (25 convolution kernel elements plus 1 bias)
	Training parameter	Number of neurons	(5×5+1)×6×28×28=122304
		Trainable parameter	32×32
		Linking number	5×5
Design parameters of pooling layer	Input parameter	The output of the front-end convolution layer	28×28
	Pooling parameter	Pool window size	2×2
		Sampling type	14×14
		Sampling mode	14×14×6
	Output parameter	The output size of the feature map after pooling	(2×2+1)×6×14×14=5880
	Training parameter	Number of neurons	28×28
		Linking number	2×2

Softmax transforms a set of variables into probability distributions, and cross entropy evaluates the distance between two probability distributions, Softmax and cross entropy are frequently paired to ensure the model's predictions closely align with the true values, as illustrated in equation (7).

$$\begin{cases} loss(x,y) = \sum_i y_i \log(p_i) \\ p_i = \dfrac{e^{x_i}}{\sum\limits_i^C e^{x_i}}, i = 123....C \\ p(y^{(i)} = j \mid x^{(i)}; \theta) = \dfrac{e^{\theta_j} x^{(i)}}{\sum\limits_{l=1}^{k} e^{\theta_l} x^{(i)}} \end{cases} \tag{7}$$

In equation (7), C represents the total number of categories classified, θ_j is the weight vector related to the jth category, $x^{(i)}$ is the feature vector of the ith data sample, and k is the total number of possible classifications. p_i represents the probability that the model predicts the sample belongs to Class i (Softmax output). y_i represents the indicator function of the true label (if i is the true category, then $y_i = 1$; otherwise, $y_i = 1$). By minimizing this loss function, the research can guide the parameter updates of the model to improve its accuracy in identifying fault categories. After training, the model will be able to accurately classify network faults on new data samples. The constructed CNN fault diagnosis model is shown in Figure 3.

Firstly, the original features are preprocessed to obtain a 6×6 grayscale matrix. The convolution operation is performed by sliding the convolution kernel on the input matrix to complete the local transformation of features. The pooling layer is utilized to reduce the spatial size of features, decrease the number of parameters and computational complexity, while preserving important features. After convolution and pooling, the feature mapping results are flattened and input into the fully connected layer, which is responsible for integrating the learned features and outputting a confidence matrix about the type of network failure. The output of the fully connected layer is processed by a Softmax classifier, which converts the output into a probability distribution. The confidence level of each fault type represents the probability of the model predicting the occurrence of that fault. Finally, based on the probability distribution output by Softmax, the network fault type with the highest probability is selected as the detection result.

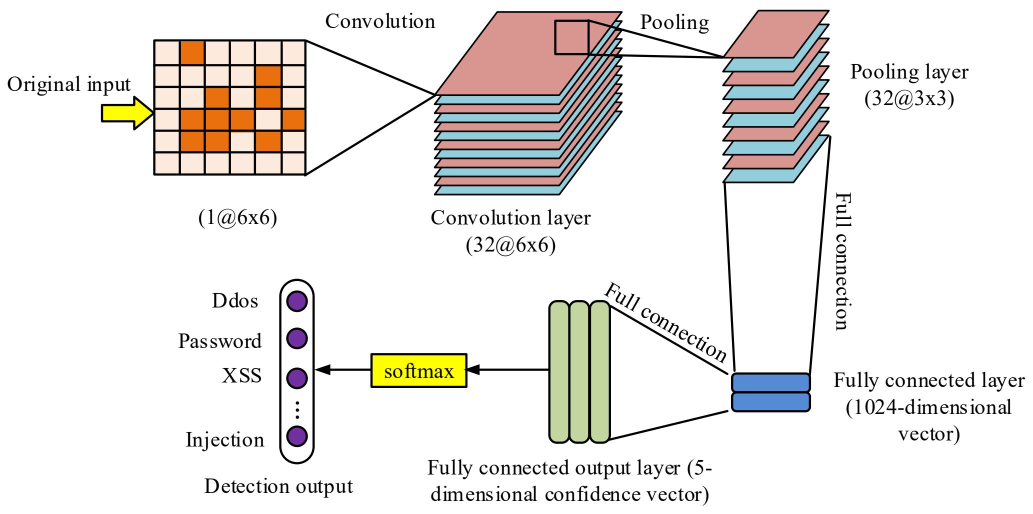

Figure 3: CNN detection model

3.2 Optimization of CNN fault detection model based on improved TLBO and MBGD

The CNN-based computer network fault diagnosis model can automatically learn and extract useful features from raw data, but the CNN model has high complexity and requires significant computing resources and storage space. As the number of network layers increases, CNN is prone to overfitting, which may affect its performance in certain application scenarios. To address these issues, further optimization of CNN model parameters is studied through TLBO and MBGD.

During the model training process, the weights and biases of each neuron are activated during forward propagation, while these parameters are continuously updated during back propagation [26]. This process usually requires multiple iterations, so the computational workload is quite large. In some cases, the model may experience overfitting or difficulty converging. To address these issues, the MBGD algorithm randomly selects a batch of samples from the dataset for training at each iteration, and the size of this batch of samples is determined by $batch\_size$ [27]. The loss function E of the MBGD algorithm is calculated using equation (8).

$$E = \frac{1}{2 \cdot batch\_size} \sum_{i=batch\_start}^{batch\_satr+batch\_size-1} \sum_{k=1}^{n} \left(d_k\left(i\right) - y_k\left(i\right)\right)^2 \tag{8}$$

In equation (8), d_k means the predicted output of the model, y_k means the actual output, $batch\_start$ denotes the start index of the current batch. By updating the value of $batch\_size$ in each iteration, the MBGD algorithm ensures that different sample batches are used for training in each iteration. CNN typically consists of multiple residual network blocks or dense network blocks, as shown in Figure 4. Through this approach, the network is able to construct complex feature maps layer by layer, providing powerful feature support for subsequent fault detection tasks. In practical applications, MBGD algorithm accelerates the training process through matrix operations. Although its speed may be slightly slower than stochastic gradient descent algorithm, it can find the optimal parameter model that minimizes the overall error in fewer iterations.

The hyper-parameters and structural configuration of CNN are actually an optimization problem. To further enhance the ability of TLBO algorithm in global search, researchers combined it with Differential Evolution (DE) algorithm to form a hybrid TLBO-DE algorithm, which finely adjusts the hyper-parameters and architecture of CNN. As shown in Figure 5, the hybrid TLBO-DE strategy initially relies mainly on the global search capability of the DE algorithm to broaden the search range and reduce the risk of falling into local optima. As the iteration progresses, the algorithm gradually transitions to the "learning" stage of the TLBO algorithm for more in-depth optimization work. This can accelerate convergence speed and improve efficiency in the later stages of evolution. Through this hybrid strategy, the TLBO-DE algorithm can accelerate convergence speed while maintaining

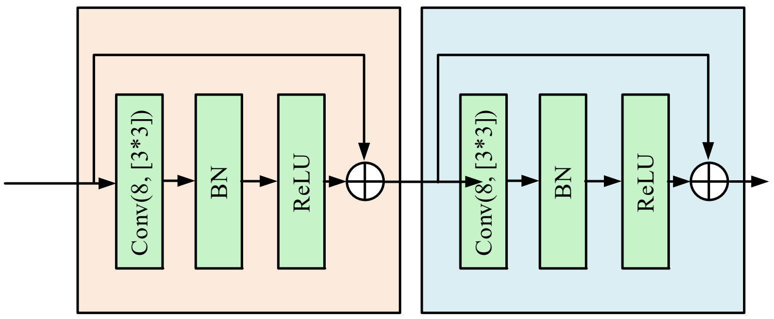

Figure 4: Concrete network diagram represented by instance vector

global search capability, thereby achieving better performance in CNN hyper-parameter and structural optimization problems.

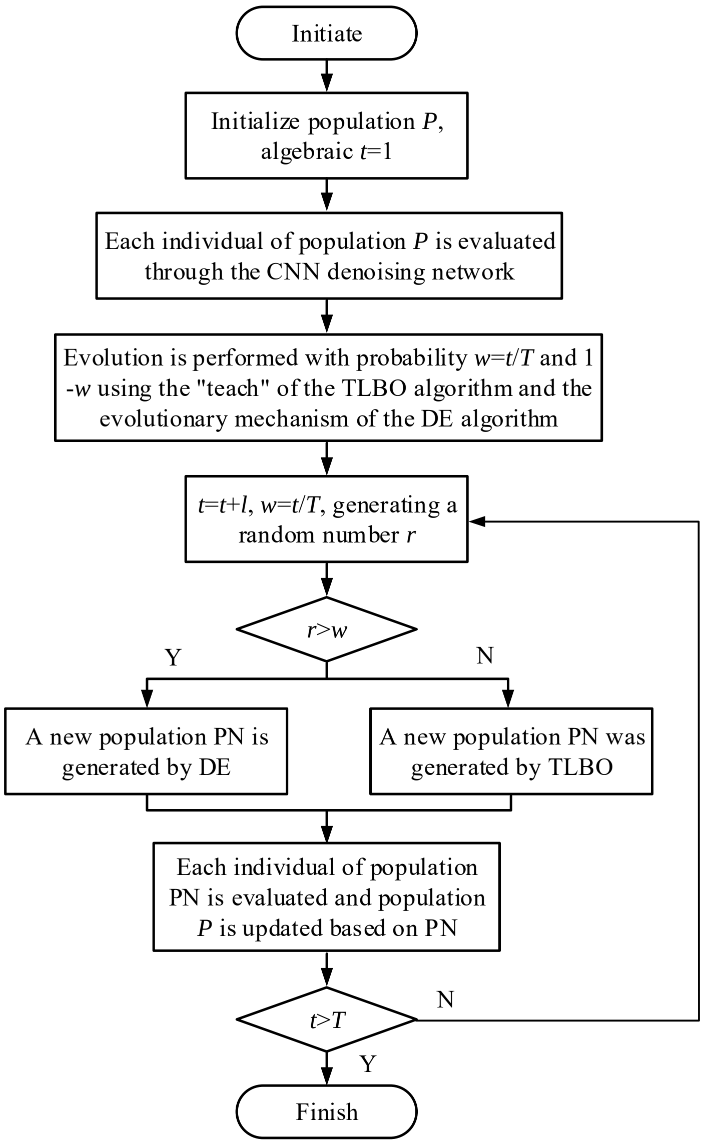
Initiate
Initialize population P, algebraic t=1
Each individual of population P is evaluated through the CNN denoising network
Evolution is performed with probability w=t/T and 1 -w using the "teach" of the TLBO algorithm and the evolutionary mechanism of the DE algorithm
t=t+l, w=t/T, generating a random number r
r>w
Y
N
A new population PN is generated by DE
A new population PN was generated by TLBO
Each individual of population PN is evaluated and population P is updated based on PN
t>T
N
Y
Finish

Figure 5: Flowchart of the hybrid TLBO-DE algorithm

In the TLBO-DE algorithm, the generation and updating of individuals are the core processes. Firstly, the algorithm randomly generates individual $P_i = (P_{i,1}, P_{i,2}, \cdots, P_{i,d})(i = 1, 2, \cdots, N)$ from the fine parameter set θ_d, where d is the dimension of P_i and N is the total number of individuals in the population. The fitness of an individual is determined by calculating the average weight of all network fault categories trained on the corresponding CNN on the training set. To achieve a balance between global and local search, the population will adjust the "teaching" mechanism of the TLBO algorithm and the probabilities $\omega = t/T$ and $1-\omega$ of the DE algorithm based on the current evolution algebra $1-\omega$ and maximum evolution algebra T during the evolution process [28]. When the DE algorithm is adopted, for each individual P_i in the population, three different individuals $P_{i,1}$, $P_{i,2}$, and $P_{i,3}$ are randomly selected from the current population. The selection process is achieved through uniform random sampling to ensure the independence among individuals. A new individual V_i is generated through the differential mutation operation, as shown in equation (9).

$$V_i = P_{i,1} + F \times (P_{i,2} - P_{i,3}) \tag{9}$$

In equation (9), the scaling factor F controls the amplification degree of the difference vector. Binomial crossover between the mutant individual V_i and the original individual P_i is performed to generate the experimental individual $U_{i,j}$. The crossover rule is shown in equation (10).

$$U_{i,j} = \begin{cases} P_{i,j}, & Rv \leq Cr \quad \text{or} \quad j = \bar{j}(\bar{j} = 1, 2, ..., d) \\ \quad V_{i,j}, & otherwise \end{cases} \tag{10}$$

In equation (10), Rv is a random number between 0 and 1, Cr is the crossover probability, and $\bar{j}$ is a random index. If the fitness of $U_{i,j}$ is better than P_i, P_i will be replaced by $U_{i,j}$; If not, then P_i remains unchanged. Through this method, every individual in the population is updated. When using the "teaching" mechanism of the TLBO algorithm, the individual P_t with the highest fitness is first selected from the population, and then the mean vector M of the population is calculated to obtain the new individual P_{new}.

$$\begin{cases} M = \dfrac{1}{N}\sum\limits_{i=1}^{N} P_i \\ P_{new} = P_i + TF \times (P_t - round(1+TF) \times Mean) \end{cases} \tag{11}$$

If the fitness of P_{new} is better than P_i, P_i is updated with P_{new}; Otherwise, P_i shall not be updated. According to this method, the entire population is updated. In this way, individuals in the population constantly update and find the optimal solution during the evolution process.

4 Results and discussion

Firstly, the proposed algorithm was tested on publicly available datasets, and then an experimental platform was constructed to simulate the actual network environment, further verifying the usability of the proposed algorithm, including its ability to identify different types of network faults, diagnostic speed, and accuracy.

4.1 Simulation analysis

In this study, the experimental environment was set up. It operated on a Windows 10 64-bit operating system. This particular system was outfitted with a 3.2GHz Intel Pentium Processor CPU and featured 8GB of memory. The TON-IoT dataset employed in the study was collaboratively collected by the University of New South Wales Canberra IoT Laboratory and the Australian Defence Force Academy Network Range. It serves as an IoT dataset that has been tailored specifically for Industry 4.0. The dataset spans a wide variety of data sources. It encompasses diverse types of telemetry data gathered from IoT devices. Logging information for both Linux and Windows operating systems is also part of the dataset. Moreover, it includes network traffic data from industrial IoT systems. These data were used to evaluate the effectiveness of machine learning and deep learning algorithms in the field of network security. The TON-IoT dataset contains seven different types of network activity, namely "dos", "ddos", "injection", "normal", "xss", "password", and "scanning".

The study used a 6:2:2 ratio to divide the training set, validation set, and testing set. The main reason was that a training set of 60% can ensure that the model has sufficient data to learn complex

Table 2: Distribution of TON-IoT data sets

Serial number	Category	Training set	Test set	Validation set
1	Ddos	2723	907	907
2	Password	2156	718	718
3	XSS	744	248	248
4	Injection	363	121	121
5	Dos	303	101	101
6	Scanning	260	86	86
7	Normal	5844	1948	1948
/	Total	12393	4129	4129

network failure patterns, avoiding overfitting caused by small samples. The validation set and test set each accounted for 20%, and their absolute quantity was sufficient to cover the diversity of seven types of faults, meeting the statistical significance requirements. This ratio was consistent with similar studies, making it easier to compare algorithm performance horizontally. Table 2 provides a detailed breakdown of the data categories and quantity distribution. To ensure the quality of the data and the effectiveness of model training, the study removed rows containing missing values from the dataset and removed columns with all eigenvalues being zero. Then, the data was labeled and encoded to convert categorical data into numerical data for model processing.

The algorithm proposed in the study was continuously executed for 116 minutes in a specific computer server environment. The experimental outcomes are in Figure 7. In Figure 6 (a), the accuracy increased with the number of network iterations. After 3785 iterations, the algorithm achieved an accuracy of approximately 88.3% on the training dataset. From Figure 6 (b), the learning rate gradually decreased as the iteration progresses. Figure 6 (c) depicts the test results of the loss function, where the cross entropy loss value gradually decreased as the number of iterations increased. These results indicated that the proposed CNN-based fault detection algorithm had advantages in classification performance and could quickly reach the convergence state of training.

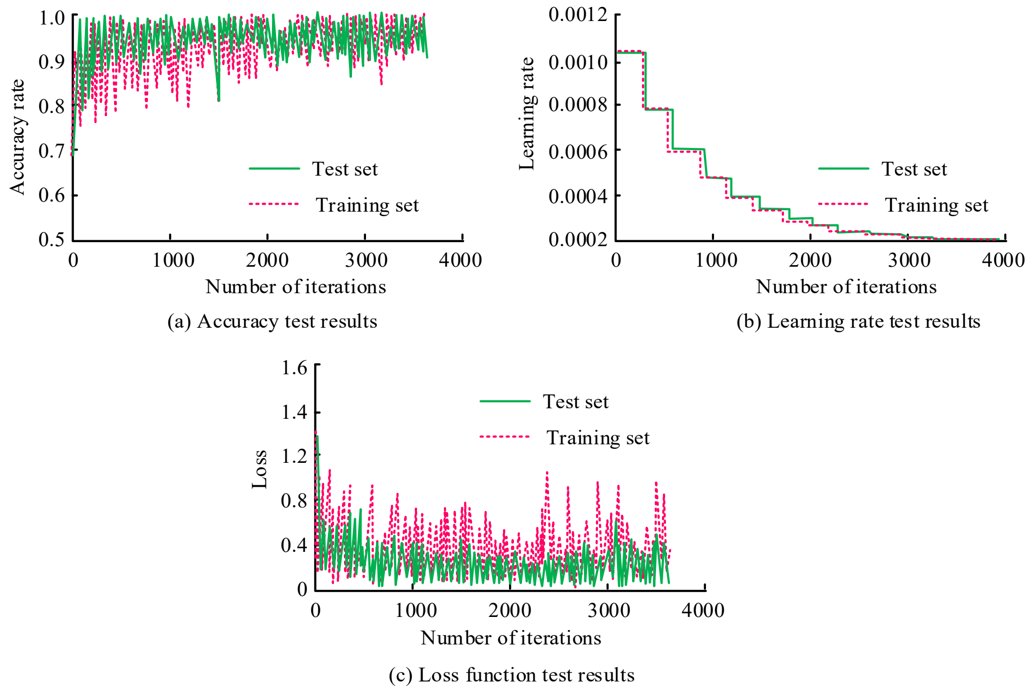

Figure 6: Algorithm performance test results

To confirm the practical application effect of the proposed method in network intrusion anomaly data detection, the original CNN, the method in reference [29], and the method in reference [30] were chosen to confirm the efficacy of the raised approach, as shown in Figure 7. The study classified and tested 7 types of intrusion data, conducting a total of 10 experiments. The final result was obtained

Table 3: Overall performance of improved KNN algorithm

Data set	Method	Accuracy (%)	Precision (%)	F1 score (%)
WUSTL-EHMS-2020	CNN	99.63	98.54	98.52
	Reference [29]	96.85	91.92	96.10
	Reference [30]	98.75	92.83	98.23
	This study	99.35	99.82	99.11
IoTID20	CNN	99.93	98.12	99.40
	Reference [29]	93.22	92.72	95.72
	Reference [30]	93.82	93.42	95.34
	This study	99.17	99.24	99.52
WUSTL-IIOT-2021	CNN	99.63	96.70	98.37
	Reference [29]	96.41	95.12	97.67
	Reference [30]	97.78	97.09	97.93
	This study	99.52	97.36	99.44

by taking the average of the final results. In Figure 7(a), among the seven types of attack detection, the average accuracy rate of this method (99.35%) was significantly better than that of Reference [29] (96.85%, p=0.001) and reference [30] (98.75%, p=0.023). For example, the difference in the detection accuracy rate of the 'Password' attack reached 5.2% (p=0.008). The outlier recognition rates of the four algorithms are shown in Figure 7 (b). From the figure, the outlier recognition rate of the algorithm suggested in this research was substantially higher than the methods in references [29] and [30].

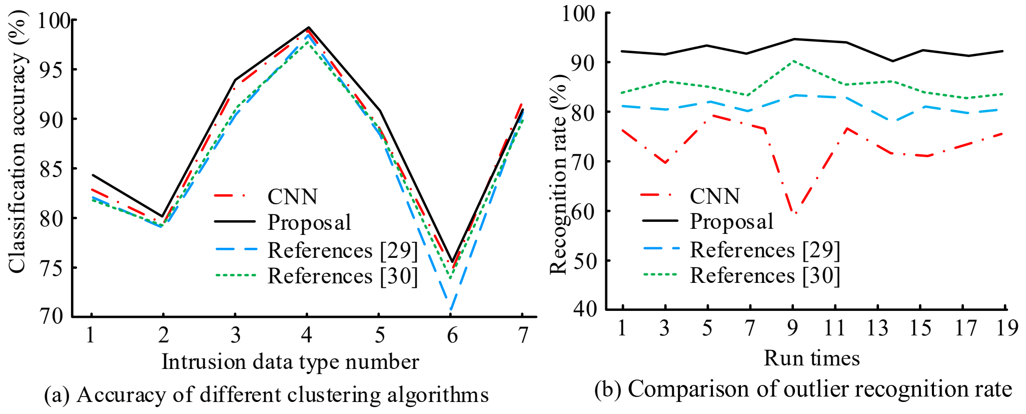

Figure 7: Improved KNN algorithm performance analysis

The performance of the suggested method was validated on the WUSTL-EHMS-2020 dataset, IoTID20 dataset, and WUSTL-IoT-2021 dataset, and compared with other network intrusion detection algorithms. Table 3 compares the proposed method with other excellent intrusion detection methods. The average accuracy and F1 score of the method proposed by the research reached over 99%, but the accuracy on the WUSTL-EHMS-2020 dataset, IoTID20 dataset, and WUSTL-IoT-2021 dataset was slightly lower than CNN, but the accuracy and F1 score were higher than CNN. Compared with the methods in references [29] and [30], the proposed method had higher accuracy, correctness, and F1 score. This was because the CNN temporal feature extraction ability optimized by TLBO in this research method was relatively strong. Therefore, the Accuracy (99.35%) was close to that of CNN (99.63%), but the Precision (99.82%) was higher, indicating that there were fewer false positives for minority classes. Although the research method in this study alleviated the category bias through the dynamic batch sampling of MBGD, the accuracy rate for small sample attacks was slightly lower than that of CNN. Therefore, the fixed batch strategy of CNN might be more suitable for static distribution. The TLBO-DE optimization of this method enhanced the adaptability of hyper-parameters to dynamic environments. Therefore, the F1 score (99.44%) was significantly higher than that of CNN (98.37%).

Table 4: Statistical Analysis of switch mirroring data (1min)

/	Network congestion				Network normal			
Statistical characteristic	S	A	B	C	S	A	B	C
Packet number	759725	584	110	758275	1042	392	230	326
Average number of packets per second	12519.3	9.8	1.9	12495.4	17.4	6.5	3.8	5.4
Average packet size /B	1015	227	112	1017	248	328	207	228
Bytes	771443784	132727	12360	771192685	257842	129784	47740	75402
Average bits per second	101M	17K	1738	101M	34K	17K	6356	9930

4.2 Analysis of actual network fault diagnosis results

In the virtual LAN managed by switch S, there are three hosts A, B, and C. During a certain period of time, there was a large-scale data exchange between host C and host D outside the local area network, which resulted in network link congestion, causing host C to occupy most of the bandwidth of the local area network switch. By conducting statistical analysis on the basic parameters of the data captured by the switch within one minute, some key statistical information can be obtained under congested conditions, as represented in Table 4. The outcomes indicated that there was a significant difference in the average number of groups for host C compared to other hosts. When the network was running normally, the data transmission volume and bandwidth occupancy of each host in the local area network were relatively balanced, and there was no significant difference in the average packet size and quantity, indicating a stable network state.

Figure 8 shows the false detection rates of the research algorithm for DDoS, Password, XSS, Injection, Dos, Scanning, and Normal7 attack types, as well as a comparison with the methods in references [29] and [30]. From the figure, the method proposed in reference [29] had a relatively low accuracy in fault diagnosis due to the limited amount of labeled data. Although the method in reference [30] fully utilized a large quantity of unlabeled data, the high complexity of the model results in a final diagnostic accuracy of only 91.2%. In this study, through the mutation operation of the TLBO-DE algorithm, the solution space was extensively explored in the hyper-parameter optimization stage to avoid the model overfitting the noise data due to the selection deviation of the initial parameters. In the later stage of optimization, the "teaching" mechanism of TLBO dynamically adjusted parameters based on the population mean and the optimal individual, suppressing the risk of overfitting. MBGD dynamically adjusted the batch size. Initially, it used a large number of batches to stabilize the gradient direction, and later introduces more randomness in small batches to avoid the model being overly sensitive to specific noise patterns. Eventually, it achieved a relatively good fault diagnosis accuracy rate, with an average false detection rate not exceeding 0.20%. Compared with other classification algorithms applied in intrusion detection technology, the false detection rate of the algorithm proposed in the study was lower, which could prove the effectiveness and feasibility of this method.

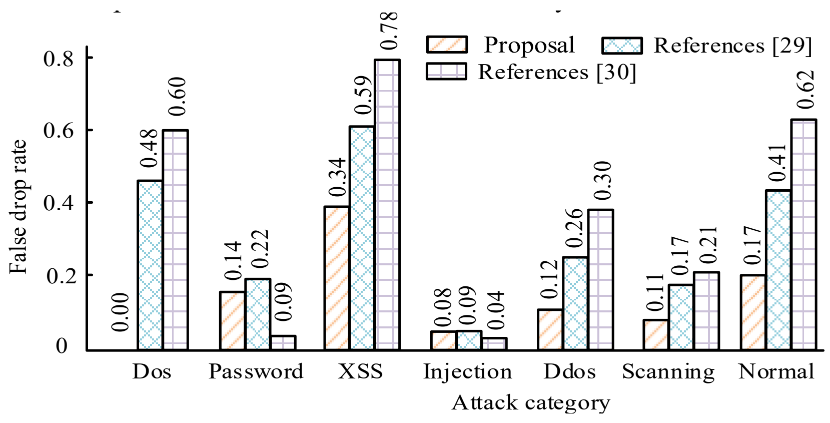

Figure 8: Comparison of false drop rates

5 Conclusion

This study proposed an innovative computer network fault diagnosis framework based on CNN, optimized through an enhanced hybrid Teaching Learning-based Optimization (TLBO) integrated with Mini Batch Gradient Descent (MBGD). By effectively addressing the challenges of low diagnostic accuracy and high computational overhead prevalent in existing methods, the proposed approach significantly accelerated convergence and mitigated issues related to local minima. Experimental validations demonstrated that the optimized model achieved superior diagnostic accuracy (up to 88.3%) and maintained an impressively low false detection rate (below 0.20%). The proposed hybrid optimization approach innovatively integrated the advantages of TLBO and MBGD, significantly outperforming traditional approaches in accuracy and convergence speed, thus offering practical and effective solutions for rapid and reliable network fault diagnosis. Future research could explore the generalization of this methodology across diverse network configurations and further optimize computational efficiency, ultimately enhancing its applicability in complex real-world network environments.

References

[1] Sri vidhya G, Nagarajan R. A novel bidirectional LSTM model for network intrusion detection in SDN-IoT network. Computing, 2024, 106(8): 2613-2642.

[2] Calik Bayazit E, Koray Sahingoz O, Dogan B. Deep learning based malware detection for android systems: A comparative analysis. Tehnicki vjesnik-Technical Gazette, 2023, 30(3): 787-796.

[3] Chen X, Ye C, Zhang Y, Zhao L, Guo J, Ma K. Strengthened teaching–learning-based optimization algorithm for numerical optimization tasks. Evolutionary Intelligence, 2024, 17(3): 1463-1480.

[4] Garrido Martinez-Llop P, Sanz Bobi J D, Huera Plaza A. Application of neural networks for the prediction of railway bearing failures. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, 2022, 236(10): 1147-1153.

[5] Thangaraj A, Sadayan G. Improving Spam Intrusion Detection with the Machine Learning-Enhanced Chaotic Horse Ride Optimization Algorithm. Tehnicki vjesnik-Technical Gazette, 2024, 31(6): 2072-2078.

[6] Kamal Idrissi H, Kartit A. Network Intrusion Detection using Combined Deep Learning Models: Literature Survey and Future Research Directions. IAENG International Journal of Computer Science, 2024, 51(8): 998-1010.

[7] Alhussien N, Aleroud A, Melhem A, Khamaiseh S Y. Constraining adversarial attacks on network intrusion detection systems: transferability and defense analysis. IEEE Transactions on Network and Service Management, 2024, 21(3): 2751-2772.

[8] Abou El Houda Z, Moudoud H, Brik B, et al. A privacy-preserving framework for efficient network intrusion detection in consumer network using quantum federated learning. IEEE Transactions on Consumer Electronics, 2024, 70(4): 7121-7128.

[9] Thiruvenkatasamy S, Sivaraj R, Vijayakumar M. Blockchain Assisted Fireworks Optimization with Machine Learning based Intrusion Detection System (IDS). Tehnički vjesnik, 2024, 31(2): 596-603.

[10] Lin Y, Lin L, Huang Y, Xu L, Hsieh S Y. Endogenous Security of FQn Networks: Adaptive System-Level Fault Self-Diagnosis. IEEE Transactions on Reliability, 2024, 73(3): 1659-1668.

[11] Ahmed S T, Tahoori M B. One-shot online testing of deep neural networks based on distribution shift detection. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2024, 43(10): 3250-3263.

[12] Zhang H, Liao K, Tai Y, Ma W, Cao G, Sun W, et al. Decentralized and Fault-Tolerant Task Offloading for Enabling Network Edge Intelligence. IEEE Systems Journal, 2024, 18(2): 1459-1470.

[13] Sasirekha N, Karuppaiah J, Shekhar H, Saranya Naga N. Breast cancer detection using histopathology image with Mini-batch stochastic gradient descent and convolutional neural network. Journal of Intelligent & Fuzzy Systems, 2023, 45(3): 4651-4667.

[14] Ramesh M, Revoori S, Edla D R, Kiran K V D. A novel multi-layer multi-spiking neural network for EEG signal classification using Mini Batch SGD. Soft Computing, 2023, 27(14): 9877-9890.

[15] Sun M, Yang Z, Dai X, Nian X, Xiong H, Wang H. An Adaptive Updating Method of Target Network Based on Moment Estimates for Deep Reinforcement Learning. Neural Processing Letters, 2023, 55(5): 5515-5537.

[16] Kaushik A, Al-Raweshidy H. A novel intrusion detection system for internet of things devices and data. Wireless Networks, 2024, 30(1): 285-294.

[17] Al-Janabi M, Ismail M A, Ali A H. Intrusion Detection Systems, Issues, Challenges, and Needs. International Journal of Computational Intelligence Systems, 2021, 14(1): 560-571.

[18] Jin H, Jiang C, Lv S, He H, Liao X. A hybrid teaching-learning-based optimization algorithm for QoS-aware manufacturing cloud service composition. Computing, 2022, 104(11): 2489-2509.

[19] Fan H, Xue L, Song Y, Li M. A repetitive feature selection method based on improved ReliefF for missing data. Applied Intelligence, 2022, 52(14): 16265-16280.

[20] Hourri S. Empowering Speaker Verification with Deep Convolutional Neural Network Vectors. Studies in Informatics and Control, 2024, 33(2): 97-107.

[21] Nawroly S S, Popescu D, Antony M C T. Category-based and Target-based Data Augmentation for Dysarthric Speech Recognition Using Transfer Learning. Studies in Informatics and Control, 2024, 33(4): 83-93.

[22] Unluturk M S, Komesli M, Keceli A. Convolutional Neural Network for Cotton Yield Estimation. Studies in Informatics and Control, 2024, 33(2): 109-117.

[23] Fei F, Liu W, Shu L. A Lightweight Convolutional Neural Network for Salient Object Detection. Tehnicki vjesnik-Technical Gazette, 2024, 31(4): 1402-1410.

[24] Nisha S R, Muthurajkumar S. Semantic Graph Based Convolutional Neural Network for Spam e-mail Classification in Cybercrime Applications. International Journal of Computers Communications & Control, 2023, 18(1).

[25] Babu P A, Rao B V S, Reddy Y V B, Kumar G R, Rao J N, Reddy Koduru S K, et al.Optimized CNN-based Brain Tumor Segmentation and Classification using Artificial Bee Colony and Thresholding. International Journal of Computers Communications & Control, 2023, 18(1).

[26] Ma Y, Ma Z, Li Y, Gao H, & Xue Y. A Project Recommender Based on Customized Graph Neural Networks in Online Labor Markets. International Journal of Computers Communications & Control, 2023, 18(4), 5173.

[27] Swain D, Parmar B, Shah H, Gandhi A, Acharya B, Hu Y C. Enhanced handwritten digit recognition using optimally selected optimizer for an ANN. Multimedia Tools and Applications, 2023, 82(28): 44021-44036.

[28] Sagheer M, Asif Jan M, Shah Z, Mashwani W K, Adeeb Khanum R, Shutaywi M. Enhancing teaching learning based optimization algorithm through group discussion strategy for CEC 2017 benchmark problems. Soft Computing, 2025, 29(2): 895-932.

[29] Patel N D, Mehtre B M, Wankar R. A computationally efficient dimensionality reduction and attack classification approach for network intrusion detection. International Journal of Information Security, 2024, 23(3): 2457-2487.

[30] Li G, Wang J, Qin Y, Bai X, Jiang Y, Deng Y, et al. Enhancing Wind Farm Reliability: A Field of View Enhanced Convolutional Neural Network-Based Model for Fault Diagnosis and Prevention. International Journal of Computers Communications & Control, 2024, 19(3): 6609.

Cite this paper as:

Tan, J.H. (2026). Computer network fault diagnosis based on improved TLBO and MBGD optimization, *International Journal of Computers Communications & Control*, 21(2), 7068, 2026.
https://doi.org/10.15837/ijccc.2026.2.7068