Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 8825587, 33 pages
https://doi.org/10.1155/2023/8825587

WILEY | Hindawi

Research Article

Application of Deep Neural Network with Frequency Domain Filtering in the Field of Intrusion Detection

Zhendong Wang,[1] Jingfei Li ,[1] Zhenyu Xu,[2] Shuxin Yang,[1] Daojing He,[3] and Sammy Chan[4]

[1]*School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China*
[2]*Faculty of Information Science and Engineering, Ocean University of China, Qingdao 266100, China*
[3]*School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China*
[4]*Department of Electrical Engineering, City University of Hong Kong, Hong Kong 999077, China*

Correspondence should be addressed to Jingfei Li; 6120210180@mail.jxust.edu.cn

Received 27 June 2023; Revised 29 October 2023; Accepted 2 November 2023; Published 16 November 2023

Academic Editor: Alexander Hošovský

In the field of intrusion detection, existing deep learning algorithms have limited capability to effectively represent network data features, making it challenging to model the complex mapping relationship between network data and attack behavior. This limitation, in turn, impacts the detection accuracy of intrusion detection systems. To address this issue and further enhance detection accuracy, this paper proposes an algorithm called the Fourier Neural Network (FNN). The core of FNN consists of a Deep Fourier Neural Network Block (DFNNB), which is composed of a Hadamard Neural Network (HNN) and a Fourier Neural Network Layer (FNNL). In a DFNNB, the HNN is responsible for sampling the network intrusion data samples in different time domain spaces. The FNNL, on the other hand, performs a Fourier transform on the samples outputted by the HNN and maps them to the frequency domain space, followed by a filtering process. Finally, the data processed by filtering are transformed back to the time domain space for subsequent feature extraction work by the DFNNB. Additionally, to enhance the algorithm's detection accuracy and filter out noise signals, this paper also introduces a High-energy Filtering Process (HFP), which eliminates noise signals from the data signal and reduces interference on the final detection result. Due to the ability of FNN to process network data in both the time domain space and the frequency domain space, it possesses a stronger capability in expressing data features. Finally, this paper conducts performance evaluations on the KDD Cup99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets. The results demonstrate that the proposed FNN-based IDS model achieves higher detection rates, lower false alarm rates, and better detection performance than classical deep learning and machine learning methods.

1. Introduction

With the arrival of the information age, the Internet has undergone significant development as an important production tool and has gradually permeated all aspects of the national economy and social functioning. The Internet has long been recognized as one of the most critical infrastructures in every country, highlighting the importance of network security. The primary threat to network security is the intrusion of information systems through the network. The process of identifying and detecting intrusion behavior, whether attempted, ongoing, or completed, is known as intrusion detection [1]. The core concept of intrusion detection is to analyze collected network data to distinguish between normal and intrusive data, and subsequently identify unsafe network behavior. However, with the continuous advancement of network technology, the level of attacker techniques has been improving, making it increasingly difficult to distinguish between abnormal and normal behavioral data, and network attacks are becoming increasingly covert. In the face of the escalating level of network attacks, existing intrusion detection technology is gradually exhibiting shortcomings, including lower accuracy, higher false positive rates, and difficulties in effectively differentiating the characteristic data of normal and abnormal samples.

Commonly used intrusion detection techniques include attack detection techniques based on statistical methods [2], intrusion detection methods based on expert systems [3], and methods based on machine learning and deep learning [4–8]. The main advantage of statistical methods is their ability to "learn" user habits, resulting in high detection rates and usability. However, this "learning" ability also provides intruders with the opportunity to gradually "train" intrusion events to mimic normal statistical patterns, leading to the failure of intrusion detection systems. The effectiveness of expert systems in preventing intrusion behavior relies on the completeness of the knowledge base, which is often impractical to achieve for large network systems. Due to the inherent incompleteness of expert system rules, expert systems alone are no longer suitable for intrusion detection, especially with the continuous development of network intrusion technology. Traditional machine learning methods often require extensive upfront feature engineering work, which relies heavily on expert knowledge. The quality of feature engineering has a significant impact on the effectiveness of the algorithm, making it susceptible to human factors.

The concept of deep learning was first introduced by Professor G.E. Hinton at the University of Toronto in 2006 [9]. The network structure of deep learning consists of a large number of individual components called neurons. Each neuron is connected to other neurons, and the strength of the connections between neurons is determined by weights that can be optimized during the learning process to determine the performance of the neural network. Deep learning algorithms, also known as deep neural networks (DNNs) [10], have gained significant attention in recent years as a new research direction in the field of machine learning. Deep neural networks have made breakthroughs in various applications, including speech recognition and computer vision [11–15]. Network intrusion data differs from speech, text, and image data in that its feature values do not exhibit obvious correlations. Speech and text data are examples of time series data. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models can effectively process this type of data by capturing the interrelationships between feature values over time [16–20]. Image data exhibit a property known as translation invariance, and the use of convolutional neural networks (CNNs) can be highly effective in processing image data by leveraging this property [21–23]. There is no fixed, stable, universal a priori knowledge known to humans that governs the relationships between feature data in network intrusion data. This knowledge evolves with the development of Internet technology and is influenced by the skill level and attack techniques employed by attackers. Since network intrusion data differ from image data and time series data, mainstream neural networks such as CNN, RNN, and LSTM are not well suited for handling network intrusion data. This limitation affects the ability of deep learning algorithms to effectively capture the features of network data. In other words, intrusion detection models based on traditional CNN, RNN, and LSTM have limited feature expression capabilities and struggle to accurately model the complex mapping relationship between network data and attack behaviors. The more complex the mapping relationship that a model can capture between network data and attack behaviors, the more promising it is in distinguishing intricate and covert network intrusion behaviors. In essence, a more expressive model is capable of accurately discerning network intrusion data. To uncover the underlying patterns within the intricate and dynamic network intrusion data, we propose a Fourier Neural Network (FNN) with enhanced data processing capabilities and an expanded data mapping space.

The core of FNN is the Deep Fourier Neural Network Block (DFNNB), which consists of the Hadamard Neural Network (HNN) and the Fourier Neural Network Layer (FNNL). To apply FNN to the field of intrusion detection, this paper first designs a Hadamard Neural Network (HNN) that combines the dot product operation of matrices with the Hadamard product operation. By using HNN, the algorithm can effectively fit the network intrusion data in the multi-temporal space of the sample X^t, thereby enhancing the ability to represent data features. Once X^t is obtained, the frequency spectrum X^f in the frequency domain space is computed by applying a fast Fourier transform to X^t using FNNL. To perform effective filtering operations on the data signal, this paper introduces the High-energy Filtering Process (HFP) to filter X^f and obtain the high-energy spectrum X^{hf}. Subsequently, X^{hf} is inverse transformed using the Fast Fourier Transform to obtain high-energy time domain feature data X^{ht}. Finally, X^{ht} is summed, compressed, and input into a fully connected neural network for classification. By stacking multiple layers of DFNNB, the FNN can achieve a more powerful mapping ability, thereby enhancing its performance on complex data.

Overall, we contribute to the intrusion detection field as follows:

(1) The Hadamard Neural Network (HNN) is proposed, which can effectively enhance the data dimensions and provide a new method for future applications that require data dimension enhancement. HNN assigns different weights to the network intrusion data samples to obtain the sample matrix X under different weights. Then, it performs the Hadamard product operation between X and a weight matrix W with the same dimension as X. This process enables the sampling of network intrusion data samples in different time domain spaces X^t.

(2) The Fourier Neural Network Layer (FNNL) is proposed to integrate the Fourier transform with the neural network algorithm. FNNL transforms the feature data of network intrusion data into the frequency domain for processing, and then applies inverse Fourier transform to convert the processed frequency domain data back to the time domain. This process effectively enhances the feature extraction capability of the neural network algorithm for complex data, thereby improving its ability to handle network intrusion data.

(3) The High-energy Filtering Process (HFP) is designed to effectively process frequency domain data. It has the capability to automatically filter out weak noise

signals, thereby reducing their impact on the final performance of the neural network.

(4) In order to validate the effectiveness of our proposed method, we conduct experimental tests on the FNN using network intrusion datasets such as KDD Cup99, NSL-KDD, UNSW-NB15, and CICIDS2017. We evaluate the performance of the FNN using multiple evaluation metrics and compare its performance with various machine learning and deep learning algorithms.

The paper is organized as follows: Section 2 introduces the related work; Section 3 provides a general introduction to FNN; Section 4 offers a detailed description of the components of FNN and analyzes the back propagation of gradient information in the DFNNB; Section 5 elaborates on the steps of intrusion detection using FNN; Section 6 explains the intrusion detection datasets; Section 7 presents the evaluation criteria for the experiment; Section 8 showcases the experimental results and analyzes the model performance; finally, Section 9 summarizes the entire paper and provides an outlook on future research directions for FNN.

2. Related Work

In recent years, machine learning (ML) and deep learning (DL) algorithms have emerged as the predominant and efficacious models for numerous data processing applications. Notably, ML algorithms have found widespread utilization in the realm of intrusion detection. This section presents an overview of ML and DL algorithms employed in intrusion detection, with a specific focus on the KDDCUP99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets. The common ML algorithms utilized in intrusion detection encompass support vector machine (SVM), logistic regression (LR), decision tree (DT), Plain Bayes, random forest (RF), K nearest neighbors (KNN), and artificial neural network (NN). By optimizing these classical ML algorithms at various levels, the performance of intrusion detection systems can be significantly enhanced.

Jan et al. [24] used SVM based IDS on CICIDS2017 dataset and achieved 98% accuracy. Safaldin et al. [25] proposed an intrusion detection method using Improved Binary Grey Wolf Optimizer (GWOSVM-IDS) and got 96% accuracy on NSL-KDD dataset but the detection time of this method is very long. Ponmalar and Dhanakotid [26] combined ensemble support vector machine (SVM) with Chaos Game Optimization (CGO) algorithm. The proposed ESVM algorithm was employed for classification prediction on the UNSW-NB15 dataset, while the CGO algorithm was utilized to fine-tune the parameters of ESVM, thereby enhancing the accuracy and reducing the occurrence of false positives. Boahen et al. [27] proposed a diversity enhancement strategy based on Improved Particle Swarm Optimization (PSO) algorithm, Gravitational Search Algorithm (GSA) and used to optimize a Random Forest classifier with 98.92% detection accuracy. Ding et al. [28] designed a KNN-based undersampling mechanism and a generative adversarial network model for oversampling attack samples. The model undersamples normal traffic samples and oversamples attack traffic, thus balancing the dataset, but its detection rate is not significantly improved. Yousefnezhad et al. [29] employed KNN and SVM for multiclassification and used Dempster–Shafer method to combine multiple outputs. Gu and Lu [7] applied Naive Bayes feature embedding to the original data to obtain new high-quality training data and used a support vector machine to construct an intrusion detection classifier. However, the method is sensitive to data noise and may affect the effectiveness of the feature transformation if noise is present in the data. In the context of intrusion detection, a substantial number of features and data are typically involved. By employing feature selection techniques [30], the quantity of features can be reduced, thereby diminishing the computational costs associated with training and testing deep learning models. From the wide array of techniques and algorithms that have been developed, intelligent optimization algorithms [31] have proven successful in identifying the most representative and significant features, consequently reducing the dimensionality of the feature space. This reduction in dimensionality serves to enhance the performance and efficiency of intrusion detection systems. Alazab et al. [32] used Moth Flame Optimization (MFO) method as a search algorithm and decision tree (DT) as an evaluation algorithm to generate an effective subset of features for intrusion detection systems. Halim et al. [33] used a modified genetic algorithm (GA) to search for the best features and performed classification experiments on three machine models, namely SVM, kNN, and XgBoost. They designed a novel objective function for the GA which assigns fitness values to individuals in the GA population to be able to select the chromosomes that represent the best set of features, but the algorithm runs slowly due to multiple iterations and reproduction operations.

Although many literature works prove that intrusion detection methods based on traditional machine learning algorithms are indeed effective, these methods still suffer from the following drawbacks: (1) Traditional machine learning methods usually require human intervention and expertise when performing feature extraction. In intrusion detection, determining an effective feature set is challenging because intrusion behaviors can be dynamic and diverse, making it difficult to capture all intrusion patterns. (2) Traditional machine learning methods may take longer to complete training and prediction. In addition, some complex machine learning algorithms, such as SVM and DT, may be more expensive in terms of computational cost. (3) Intrusion detection data usually have high-dimensional features, and traditional machine learning methods may encounter dimensionality catastrophe problems when dealing with high-dimensional data, and it is difficult to extract information about the nonlinear features in the data, which results in a degradation of the model's performance. Given the limitations of ML algorithms and their variants, as well as the emergence of deep learning, recent advancements in DL algorithms have been applied to the field of intrusion detection. These include deep neural networks (DNNs), recurrent neural networks (RNNs), convolutional neural

networks (CNNs), and deep belief networks (DBNs). Unlike traditional machine learning methods, deep learning methods can effectively extract the underlying patterns in sample feature data by constructing multilayer nonlinear network structures. Consequently, deep learning exhibits superior capability in learning and predicting high-dimensional feature data compared to traditional machine learning methods.

Thakkar and Lohiya [34] proposed a novel feature selection technique that combines statistical significance based on standard deviation and the difference between mean and median. They also employed deep neural networks (DNNs) to learn and derive patterns in simplified subsets of features. However, it should be noted that the presence of noise in the data may significantly impact the computational results and lead to instability in feature selection. Riyaz and Ganapathy [35] achieved an accuracy of 98.8% on the KDDCUP99 dataset using CNN. Fu and Zhang [36] introduced a feature fusion technique based on gradient importance enhancement. They employed ResNet-18 as the detection model and incorporated feature fusion at each layer during training. Additionally, they applied feature enhancement at the last layer of the classification network before forwarding the data to the fully connected layer for classification. It is worth mentioning that the training and inference process of CNNs typically demands substantial computational resources, particularly when dealing with large-scale datasets and complex model structures. This limitation may restrict the application of CNN in resource-constrained intrusion detection environments. Ravi et al. [37] conducted a detailed study on recurrent deep learning models. They employed a sequential feature fusion technique to combine the functionalities of different layers in the network, specifically on the RNN, LSTM, and GRU hidden layer features. Subsequently, the fused features from the recurrent hidden layer were forwarded to an integrated meta-classifier for classification. However, recurrent deep learning models exhibit slower training and testing times compared to CNNs, particularly when processing larger datasets. Moreover, they encounter limitations when addressing the issue of attack class imbalance. Wang et al. [38] introduced an intrusion detection model that leverages Improved deep belief networks (DBNs) employ a kernel-based extreme learning machine (KELM) with supervised learning capability, as an alternative to the BP algorithm in DBNs. Experimental evaluations were conducted on the KDDCUP99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets, demonstrating the robustness of the proposed approach.

By applying optimization algorithms to the engineering design problem of intrusion detection systems [39], it is possible to identify globally optimal intrusion detection model structures or parameters that can adapt to various network environments and intrusion behaviors [40]. Kanna and Santhi [41] combined Hierarchical Multiscale LSTM (HMLSTM) and CNN to effectively extract and learn spatiotemporal features. They also employed a novel metaheuristic method called Lion Swarm Optimization to fine-tune the hyperparameters of the model, thereby enhancing the learning rate of spatial features. In their other proposed deep network model, BWO-CONV-LSTM, Kanna and Santhi [42] utilized the Black Widow Optimization (BWO) algorithm to optimize the hyperparameters and achieve the desired architecture. However, their experiments were limited to binary classification and did not consider the detection of specific attack types. Balasubramaniam et al. [43] proposed the Gradient Hybrid Leader Optimization (GHLBO) algorithm to train Deep Stacked Autoencoders (DSAs) for effective DDoS attack detection. Yang et al. [44] introduced a hybrid partitioning strategy in the Negative Selection Algorithms (NSAs), which divides the feature space into grids based on the density of sample distributions. This strategy generates specific candidate detectors in the boundary grids to effectively mitigate vulnerabilities caused by boundary diversity. Finally, the NSA is enhanced through self-clustering and a novel gray wolf optimizer, enabling adaptive adjustment of detector radius and position.

The use of deep learning methods in solving the intrusion detection problem has been shown in current research to compensate for the limitations of shallow machine learning techniques in detecting high-dimensional data and extracting nonlinear feature information. Table 1 provides a chronological summary of the approaches discussed in the related literature in this section. However, deep learning-based intrusion detection techniques still have the following limitations: (1) they require a large number of parameters to be trained, resulting in high time and space costs for running the models. Currently, parallel processing with multiple GPUs is often needed to handle large-scale data; (2) when the model becomes too deep, it can lead to the vanishing or exploding gradient problems due to the long back propagation path during gradient descent; (3) existing deep learning algorithms are primarily designed for solving problems in other domains, while network traffic exhibits characteristics of large scale and high dimensionality, and network intrusion traffic is characterized by hidden diversity. As a result, many existing deep learning models are not fully suitable for the field of intrusion detection. With the development of information technology, network intrusion techniques have also advanced significantly. Network intrusion behaviors are becoming increasingly covert, making it more difficult to detect differences between network intrusion data and normal data. This paper fully considers the complex and variable characteristics of existing network intrusion data. Starting from improving the algorithm's ability to analyze data, a Fourier Neural Network (FNN) is designed based on deep learning, which has stronger feature extraction and representation capabilities for complex data. The intrusion detection model designed with FNN as the core is end-to-end and does not require manual feature selection. It can learn features directly from the raw data, thereby improving classification performance and demonstrating stronger generalization ability. Additionally, the high-energy filtering process in the model can be used to handle data noise, reducing the impact of weak noise signals on the neural network's final performance and improving the model's performance and robustness.

Table 1: Comparison table of intrusion detection algorithms.

References	Year	Methods	Datasets	Advantages	Limitation	Evaluation metrics			
						Accuracy (%)	Precision (%)	Recall (%)	*F*1 (%)
Thakkar and Lohiya [34]	2023	DNN	NSL-KDD, UNSW-NB15, CICIDS2017	Learning and inferring patterns in simplified feature subsets using DNNs	The effectiveness of the technique depends on the representativeness of the dataset. If the dataset is noisy, feature selection may be inaccurate	99.84, 89.03, 99.80	99.94, 95.00, 99.85	98.81, 98.95, 99.94	99.37, 96.93, 99.89
Yang et al. [44]	2023	NSA	NSL-KDD, UNSW-NB15, CICIDS2017	Solves the problem of low boundary detection rate due to the diversity of sample boundaries	The problem of detecting adaptive evolution remains unresolved	97.61, 95.76, 98.60	96.68, 94.57, 99.24	99.23, 98.46, 95.73	97.94, 96.48, 97.45
Balasubramaniam et al. [43]	2023	GHLBO + DSA	NSL-KDD	Provides a method for accurately detecting DDoS attacks	No overhead analysis is considered, only DDoS attacks are detected	0.914	—	0.909	—
Ponmalar and Dhanakoti [26]	2022	CGO + SVM	UNSW-NB15	Efficient handling of large-scale data instances	Single data set; it is slightly affected with halting function revisions	96.29	—	—	—
Boahen et al. [27]	2022	PCO + GSA + RF	NSL-KDD, UNSW-NB15	Combines the strengths of PSO, GSA and RF to fully utilize global and local search capabilities	Higher complexity, requiring some computational resources and time	98.56, 98.96	98.82, 98.82	98.78, 96.89	98.63, 98.83
Ding et al. [28]	2022	KNN + GAN	KDDCUP99, UNSW-NB15, CICIDS2017	Deep generative modeling replaces traditional oversampling methods and solves the problem of not being able to generate real samples efficiently	Not designing a more efficient classification model	93.53, 92.39, 95.86	—	91.38, 94.03, 94.79	95.22, 94.39, 95.81
Alazab et al. [32]	2022	MFO + DT	KDDCUP99, NSL-KDD, UNSW-NB15	Increased feature space utilization by using cosine similarity metric to binarize continuous MFO into binary problems	Multiple iterations and evaluations, requiring some computational resources and time	97.8, 89.7, 92.4	—	99.6, 89.1, 92.1	—
Ravi et al. [37]	2022	RNN + LSTM + GRU	KDDCUP99, UNSW-NB15, CICIDS2017	Ability to extract sequential and temporal features from network traffic	RNN models require step-by-step iterative computation when dealing with long sequences, which is less computationally efficient	99.0, 99.0, 99.0	97.0, 99.0, 99.0	99.0, 99.0, 99.0	98.0, 99.0, 99.0
Kanna and Santhi [42]	2022	CNN + LSTM	NSL-KDD, UNSW-NB15	Learning the spatial and temporal characteristics of network traffic	High model complexity, requiring significant computational resources	98.67, 98.66	97.48, 100	100, 98.77	98.73, 98.77
Fu and Zhang [36]	2022	ResNet-18	NSL-KDD, CICIDS2017	Proposed feature fusion technique and feature enhancement technique	The proposed model does not run as fast as traditional machine learning models	99.84, 99.78	99.84, 99.82	99.84, 99.79	99.84, 99.80

Table 1: Continued.

References	Year	Methods	Datasets	Advantages	Limitation	Evaluation metrics			
						Accuracy (%)	Precision (%)	Recall (%)	*F*1 (%)
Safaldin et al. [25]	2021	GWO + SVM	NSL-KDD	Reduced false alarm rates and number of features generated by IDS in WSN environments	Single dataset used, high computational time and model complexity	96.00	—	—	—
Yousefnezhad et al. [29]	2021	KNN + SVM	NSL-KDD, CICIDS2017	Correct decisions can be strengthened and incorrect decisions weakened by integrating decisions from multiple experts	It takes a longer time to compute the output probabilities of SVM and KNN	99.80, 98.97	99.83, 99.90	99.84, 94.42	99.83, 97.08
Gu and Lu [7]	2021	SVM + naïve Bayes	UNSW-NB15, CICIDS2017	The proposed naïve Bayes feature embedding, which improves the detection capability	Only binary classification is considered; data noise affects the effect of feature transformation	93.75, 98.92	—	94.73, 99.46	—
Halim et al. [33]	2021	GA + SVM/KNN/ XgBoost	UNSW-NB15	8–10 features selected from the data can solve the curse of dimensionality in massive datasets	Multiple repetitions of iteration and replication operations, slower computations	96.48	—	—	—
Wang et al. [38]	2021	DBN	KDDCUP99, NSL-KDD, UNSW-NB-15, CICIDS2017	Stable classification performance, insensitive to specific datasets	Long training time and high computational complexity	98.6, 98.6, 93.42, 97.15	94.0, 93.64, 82.30, 96.80	98.73, 98.40, 96.4, 98.19	96.31, 96.06, 88.79, 97.49
Kanna and Santhi [41]	2021	CNN + LSTM	NSL-KDD, UNSW-NB15	Learning hierarchical relationships between different features and extracting temporal features	The method may not perform well on unsuitable training datasets	96.33, 90.67	100, 86.71	95.80, 95.19	98.13, 91.46
Riyaz and Ganapathy [35]	2020	CNN	KDDCUP99	False alarm rate of the proposed model is less than 1%	No validation of effects on newer datasets, single evaluation metrics	98.8	—	—	—
Jan et al. [24]	2019	SVM	CICIDS2017	A lightweight model is designed to effectively solve the problems of resource constraints and limited node storage capacity	The dataset is single, and only binary classification is considered; the detection range is too small, and features with insignificant flow variations are ignored	98.03	—	—	—

3. Intrusion Detection Model

The FNN-based intrusion detection model proposed in this paper is divided into 3 main modules, and the general framework diagram of the model is shown in Figure 1.

Data preprocessing module: (1) Conducting preprocessing operations on the data to complete data conversion tasks, transforming discrete data into continuous data to meet the requirements of the input data; (2) Performing data normalization operations to scale the feature values between 0 and 1, preventing the negative impact of significant differences in feature values on the effectiveness of deep learning; (3) Dividing the dataset into training and testing sets.

Intrusion detection module: Construct an intrusion detection model based on FNN. The framework of FNN is shown in the right half of Figure 1. The core component of FNN is the Deep Fourier Neural Network Block (DFNNB), which consists of Hadamard Neural Network (HNN) and Fourier Neural Network Layer (FNNL). FNN is composed of n ($n \geqq 1$) DFNNBs combined with DNNs. DFNNB is responsible for effective feature extraction of network intrusion data, while DNN maps the feature representation learned by DFNNB to the sample labeling space, achieving the goal of training classifiers and learning global features of the target. The constructed model is trained using the training set and saved for testing after the training is completed.

Detection and classification module: use the test set to test the trained FNN, and use the detection and classification results to analyze and evaluate the model.

4. Fourier Neural Network Model

The detailed structure of the FNN is shown in Figure 2. The FNN can be composed of multiple Deep Fourier Neural Network Blocks (DFNNBs), and the detailed structure of the DFNNB is shown in the upper part of Figure 2. The DFNNB processes the data as follows:

(1) Before entering the jth DFNNB, the data are transformed by the DNN to a different feature space by mapping the learned feature representation of the $j-1$th DFNNB, achieving a change in data dimension;

(2) The one-dimensional data $X_{j-1} = \{X_{j-1}[0], X_{j-1}[1], \ldots, X_{j-1}[k], \ldots, X_{j-1}[K-1]\}$ (the discrete sequence X_{j-1} consists of K elements), obtained after processing by the DNN, is input into the jth DFNNB;

(3) In the jth DFNNB, X_{j-1} is first dimensionally expanded by HNN, fitting the sampled network intrusion data X_j^t in the multitemporal space;

(4) After obtaining X_j^t, split X_j^t into m 1-dimensional tensor data X_{ij}^t. X_{ij}^t denotes the ith time domain spatial sampling of network intrusion data within the jth DFNNB, and $X_{ij}^t = \{X_{ij}^t[0], X_{ij}^t[1], \ldots, X_{ij}^t[n], \ldots, X_{ij}^t[N-1]\}$ (the discrete sequence X_{ij}^t consists of N elements), where, $1 \leq i \leq m$, $1 \leq j \leq M$, m and M are manually set hyperparameters and are taken as integers;

(5) The Fast Fourier Transform (FFT) is performed on this X_{ij}^t, respectively, to obtain m representations of the time domain signal in the frequency domain space, X_{ij}^f. $X_{ij}^f = \{X_{ij}^f[0], X_{ij}^f[1], \ldots, X_{ij}^f[n], \ldots, X_{ij}^f[N-1]\}$ (discrete sequence X_{ij}^f consists of N elements);

(6) X_{ij}^f is filtered using a high-energy filtering process that removes the noisy mass signal to obtain the high-energy spectrum X_{ij}^{hf};

(7) Using the inverse Fourier transform of the fast Fourier transform, obtain the time domain representation X_{ij}^{ht} of X_{ij}^{hf};

(8) Finally, each X_{ij}^{ht} is summed and compressed to obtain a feature signal X_j that integrates the spatial samples in each time domain, X_j is a vector where the number of elements is the same as X_{j-1}.

Similar to traditional DNN, VGG19, and VGG16, FNN allows for a deeper exploration of data features by stacking multiple DFNNBs. In FNN, a DNN is added at the end of multiple DFNNB structures, enabling the mapping of feature representations learned by the DFNNBs to the sample labeling space, thereby achieving the goal of training classifiers and learning global features of the target.

4.1. Hadamard Neural Network. The top left part of Figure 2 provides a detailed illustration of the Hadamard Neural Network structure. Based on Figure 2, it can be observed that in the jth DFNNB, the HNN assigns different weights W_{je} to the input data X_{j-1} to obtain the signal expansion matrix X_j^e. Then, a weight matrix W_{jh} with the same shape as X_j^e is used in the Hadamard product operation with X_j^e to fit the network intrusion data X_j^t sampled in the multitemporal space. The above process can be represented by (1) and (2) (the weights W_e and W_h are optimized using the back propagation algorithm combined with the gradient descent algorithm, and the specific optimization process will be elaborated in detail in the subsequent sections).

$$X_j^e = X_{j-1} \cdot W_{je}, \tag{1}$$

$$X_j^t = X_j^e \times W_{jh}. \tag{2}$$

In (1), W_{je} is a vector and the number of elements in W_{je} is m. m is a manually set hyperparameter. From the above, it is clear that m determines how many time domain spaces the fitted network intrusion data are sampled in.

4.2. Forward Propagation Process of Information in Fourier Neural Network Layer. It can be seen from Figure 2 that the forward propagation of information in FNNL mainly accomplishes four operations: (1) Fast Fourier Transform; (2) High-energy Filtering Process; (3) Inverse Fast Fourier Transform (IFFT); and (4) summation and compression process of X_{ij}^{ht}. The following paper will introduce the above four processes in detail, of which the process 4 operation is relatively simple, and this paper will be introduced together with the process 3.

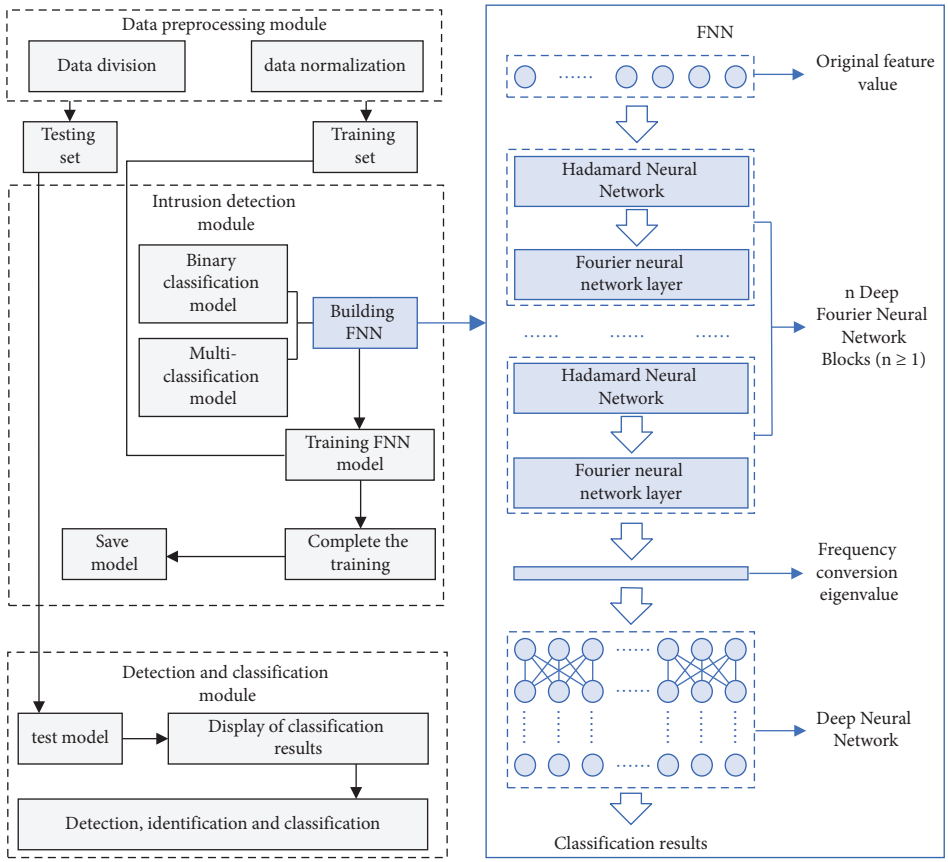

Figure 1: Intrusion detection model structure diagram.

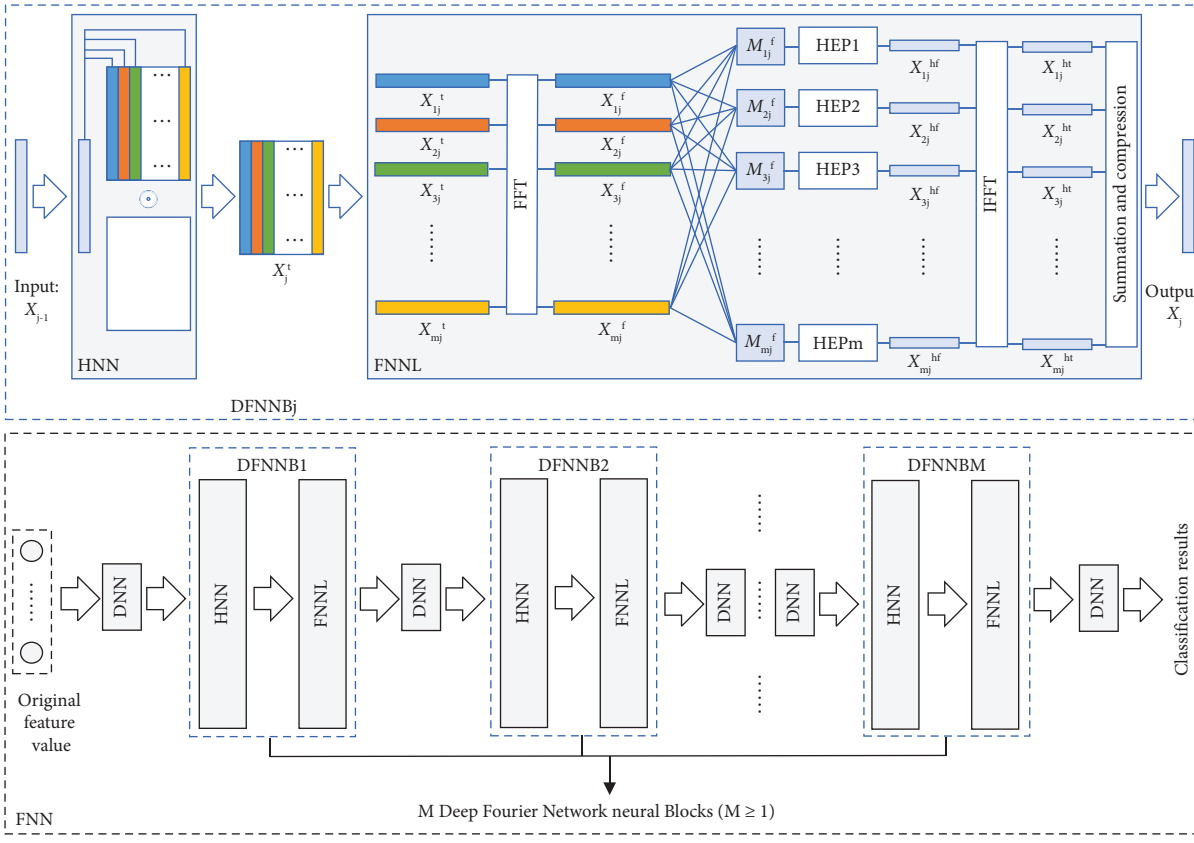

Figure 2: Structural details of Fourier Neural Network.

4.2.1. Fast Fourier Transform inside FNNL. The theory and methods of the Fourier transform have a wide range of applications in many disciplines such as mathematical equations, linear system analysis, and signal processing. Since computers can only handle discrete sequences of finite length, it is the discrete Fourier transform (DFT) that really operates on computers [45].

From the above analysis, it is evident that FNNL performs a discrete Fourier transform on $X_{ij}^t = \{X_{ij}^t[0], X_{ij}^t[1], \ldots, X_{ij}^t[n], \ldots, X_{ij}^t[N-1]\}$. The discrete sequence X_{ij}^t is composed of N elements, and the operation process of obtaining $X_{ij}^f = \{X_{ij}^f[0], X_{ij}^f[1], \ldots, X_{ij}^f[k], \ldots, X_{ij}^f[N-1]\}$ through the discrete Fourier transform of X_{ij}^t is represented by (3).

$$X_{ij}^f[k] = \sum_{n=0}^{N-1} X_{ij}^t[n] W_N^{nk}. \tag{3}$$

In (3), $W_N^{nk} = e^{-j\cdot 2\cdot \pi/N\cdot nk}$, e is a natural constant, j is an imaginary unit, $X_{ij}^f[k]$ is the discrete Fourier transform amplitude, $0 \leqq k \leqq N-1$, and k is an integer. From (3), it can be seen that calculating one $X_{ij}^f[k]$ needs to complete N times of complex multiplication and $N-1$ times of complex addition, and calculating all the values of $X_{ij}^f[k]$ needs to complete N^2 times of complex multiplication and $N \times (N-1)$ times of complex addition, and the time complexity of the algorithm is $O(N^2)$.

In order to reduce the algorithm time complexity and the running cost of the FNN, this paper uses the Fast Fourier Transform (FFT) with a running time complexity of $O(N/2\log_2^N)$ within the FNNL [46]. FFT is a fast computational method for DFT. When using FFT, control $N = 2^L$ and L is a positive integer, which can be realized in FNNL by controlling the number of neurons in the DNN part of DFNNB.

The core idea of FFT is to continuously divide the sequence X_{ij}^t into two sets of sequences with the number of elements: X_{ij-1}^t and X_{ij-2}^t according to the odd and even nature of the positions of the elements therein, and then perform the DFT operation on X_{ij-1}^t and X_{ij-2}^t, and the above process can be expressed by equations (4)–(6) as follows:

$$\begin{cases} X_{ij-1}^t[z] = X_{ij}^t[2z], \\ X_{ij-2}^t[z] = X_{ij}^t[2z+1], \end{cases} \tag{4}$$

$$\begin{aligned} X_{ij}^f[z] = &\sum_{r=0}^{N/2-1} X_{ij-1}^t[z] \cdot W_{N/2}^{rz} + W_N^z \\ &\cdot \sum_{r=0}^{N/2-1} X_{ij-2}^t[z] \cdot W_{N/2}^{rz} = X_1^t[z] + W_N^z \cdot X_2^t[z], \end{aligned} \tag{5}$$

$$\begin{aligned} X_{ij}^f\left[z+\frac{N}{2}\right] = &\sum_{r=0}^{N/2-1} X_{ij-1}^t\left[z+\frac{N}{2}\right] \cdot W_{N/2}^{r\cdot(z+N/2)} \\ &+ W_N^{(z+N/2)} \cdot \sum_{r=0}^{N/2-1} X_{ij-2}^t\left[z+\frac{N}{2}\right] \cdot W_{N/2}^{r\cdot(z+N/2)} \\ &= X_1^t[z] - W_N^z \cdot X_2^t[z], \end{aligned} \tag{6}$$

where $0 \leqq z \leqq N/2-1$ and z is an integer, and $X_1^t[z]$, $X_2^t[z]$ are the DFT transform results of X_{ij-1}^t and X_{ij-2}^t, respectively. Equations (5) and (6) together form the FFT transform result of X_{ij}^t. Referring to (5) and (6) as a butterfly operation process, the process is represented by Figure 3 as follows.

The number of complex multiplications and additions required to compute X_{ij}^f after one division of the sequence X_{ij}^t is $N^2/2 + N/2$ and $N^2/2$, respectively, so the workload is approximately halved by one decomposition.

According to the FFT, the elements in the sequence X_{ij}^t are continuously divided into two subsequences with an equal number of elements based on the odd and even nature of their positions, following the rule shown in (4), until each subsequence contains only one element. Then, the butterfly operation, as illustrated in Figure 3, is performed on each subsequence until the final result of the FFT transformation is obtained. In order to illustrate the above process clearly, Figure 4 demonstrates the FFT operation with $N = 8$ as an example.

There is a specific correspondence between the elements in X_{ij}^t and the elements in $X_{ij\text{-end}}^t$. The correspondence is as follows: the bth element in $X_{ij\text{-end}}^t$, $X_{ij\text{-end}}^t[b]$, corresponds to the ath element in X_{ij}^t, $X_{ij}^t[a]$. Here, b represents the inverted bit order of the binary encoding of a. Table 2 illustrates the relationship between each element in $X_{ij\text{-end}}^t$ and each element in the original sequence X_{ij}^t, using $N = 8$ as an example.

The specific implementation process can use Rader's algorithm [47–50] to obtain the correspondence between the sequence $X_{ij\text{-end}}^t$ and the elements in the sequence X_{ij}^t, and then obtain the sequence $X_{ij\text{-end}}^t$. Since Rader's algorithm has been introduced in many literature works, it will not be repeated here.

After determining the elements in the sequence $X_{ij\text{-end}}^t$ using Rader's algorithm, X_{ij}^f can be obtained according to the butterfly operation rules shown in Figures 3 and 4. The process can be represented by Algorithm 1.

In summary, in FNNL, the fast Fourier transform of X_{ij}^t, denoted as X_{ij}^f, is obtained by combining the Radix algorithm with butterfly operations. This process can be represented by (7) for simplicity.

$$X_{ij}^f = \mathrm{FFT}\left(X_{ij}^t\right). \tag{7}$$

4.2.2. High-Energy Filtering Process. Figure 5 illustrates the process of obtaining X_{ij}^{hf} from X_{ij}^f through the high-energy filtering process (HFP).

From Figure 5, it can be observed that before entering the s-th high-energy filtering module, X_{ij}^f needs to be assigned a coefficient vector $M_{ij}^s = \{M_{ij}^s[0], M_{ij}^s[1], \ldots, M_{ij}^s[k], \ldots, M_{ij}^s[N-1]\}$ for each $X_{ij}^f = \{X_{ij}^f[0], X_{ij}^f[1], \ldots, X_{ij}^f[k], \ldots, X_{ij}^f[N-1]\}$. These coefficient vectors (M_{ij}^s) are determined during the neural network training process using gradient descent algorithm.

The Hadamard product is performed between X_{ij}^f and M_{ij}^s to obtain the primary energy wave of X_{ij}^f, denoted as $Z_{ij}^s = \{Z_{ij}^s[0], Z_{ij}^s[1], \ldots, Z_{ij}^s[k], \ldots, Z_{ij}^s[N-1]\}$. In the s-th high-energy filtering module, all Z_{ij}^s are combined to form

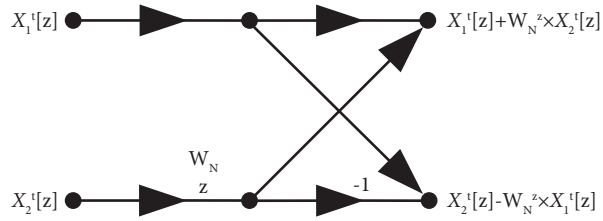

Figure 3: Butterfly operation process.

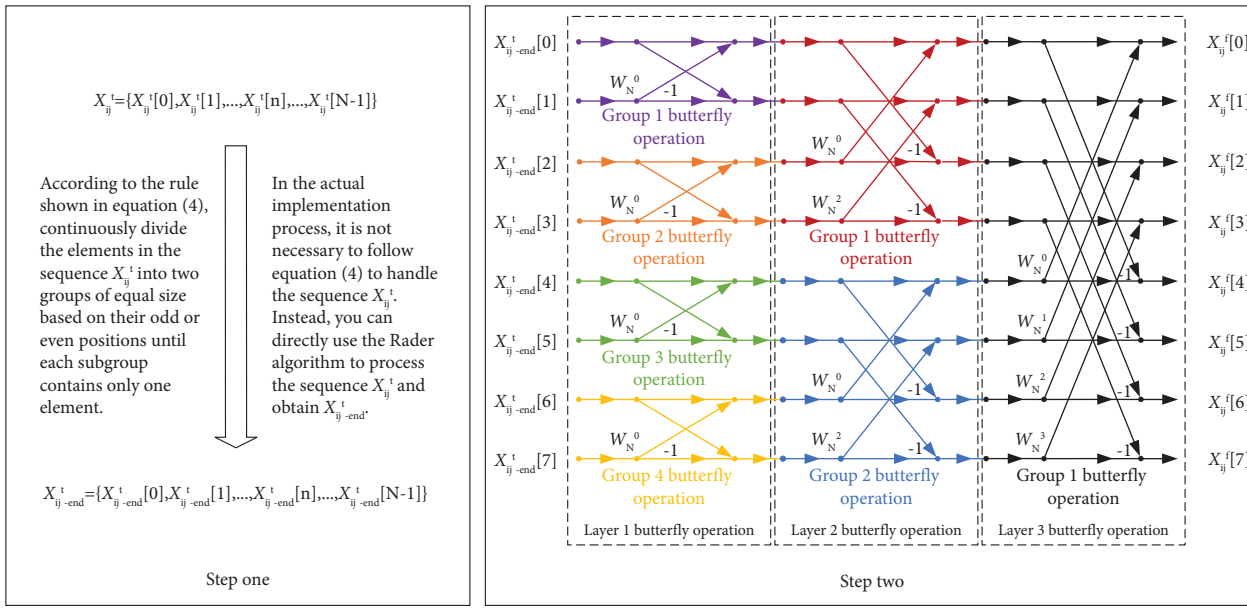

Figure 4: An example of FFT operation process.

Table 2: Example of element relationship between X_{ij}^{t} and $X_{ij\text{-end}}^{t}$.

Element number b in $X_{ij\text{-end}}^{t}$	Binary representation of b	The inverted order of binary coding of b	The decimal representation of binary code reverse order of b	The relationship between $X_{ij\text{-end}}^{t}[b]$ and $X_{ij}^{t}[a]$
0	000	000	0	$X_{ij\text{-end}}^{t}[0] = X_{ij}^{t}[0]$
1	001	100	4	$X_{ij\text{-end}}^{t}[1] = X_{ij}^{t}[4]$
2	010	010	2	$X_{ij\text{-end}}^{t}[2] = X_{ij}^{t}[2]$
3	011	110	6	$X_{ij\text{-end}}^{t}[3] = X_{ij}^{t}[6]$
4	100	001	1	$X_{ij\text{-end}}^{t}[4] = X_{ij}^{t}[1]$
5	101	101	5	$X_{ij\text{-end}}^{t}[5] = X_{ij}^{t}[5]$
6	110	011	3	$X_{ij\text{-end}}^{t}[6] = X_{ij}^{t}[3]$
7	111	111	7	$X_{ij\text{-end}}^{t}[7] = X_{ij}^{t}[7]$

the primary energy wave matrix E_j^s. This process can be represented by (8) and (9).

$$Z_{ij}^{s} = X_{ij}^{f} \times M_{ij}^{s} = \left[X_{ij}^{f}[0] \times M_{ij}^{s}[0], X_{ij}^{f}[1] \times M_{ij}^{s}[1], \ldots, X_{ij}^{f}[k] \times M_{ij}^{s}[k], \ldots, X_{ij}^{f}[N] \times M_{ij}^{s}[N-1]\right], \tag{8}$$

$$E_j^s = \begin{bmatrix} Z_{1j}^{s} \\ Z_{2j}^{s} \\ \vdots \\ Z_{ij}^{s} \\ \vdots \\ Z_{mj}^{s} \end{bmatrix} = \begin{bmatrix} Z_{1j}^{s}[0] & Z_{1j}^{s}[1] & \cdots & Z_{1j}^{s}[k] & \cdots & Z_{1j}^{s}[N-1] \\ Z_{2j}^{s}[0] & Z_{2j}^{s}[1] & \cdots & Z_{2j}^{s}[k] & \cdots & Z_{2j}^{s}[N-1] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Z_{ij}^{s}[0] & Z_{ij}^{s}[1] & \cdots & Z_{ij}^{s}[k] & \cdots & Z_{ij}^{s}[N-1] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Z_{mj}^{s}[0] & Z_{mj}^{s}[1] & \cdots & Z_{mj}^{s}[k] & \cdots & Z_{mj}^{s}[N-1] \end{bmatrix}. \tag{9}$$

Input: $X^t_{ij\text{-end}} = \{X^t_{ij\text{-end}}[0], X^t_{ij\text{-end}}[1], \ldots, X^t_{ij\text{-end}}[n], \ldots, X^t_{ij\text{-end}}[N-1]\}$, and $N = 2^k$, k is a positive integer.
(1) initialize dep to record the number of layers of the butterfly operation in which it is currently operating; m to record the number of elements involved in each set of butterfly operations; w_m is principal mth unit root.
(2) for dep $\longleftarrow$ 1 to $\log_2{}^N$, do /* Determine the current layer of butterfly operation */
$m = 2^{\text{dep}}$ /* The number of elements participating in each group of butterfly operations */
$w_m = e^{i2\pi/m} = \cos(2\pi/m) + i\sin(2\pi/m)$ /* principal mth unit root */
for $k \longleftarrow 0$ to $n-1$ by m /* Perform the butterfly operation as shown in step two of Figure 4 */
$w = 1$
for $j \longleftarrow 0$ to $-1 + m/2$ /* Complete a set of butterfly operations as shown in Figure 3 */
$t = w \times X^t_{ij\text{-end}}[k + j + m/2]$
$u = X^t_{ij\text{-end}}[k + j]$
$X^t_{ij\text{-end}}[k + j] = u + t$
$X^t_{ij\text{-end}}[k + j + m/2] = u - t$
$w = w \times w_m$
$X^f_{ij} = X^t_{ij\text{-end}}$
end

Algorithm 1: FFT algorithm.

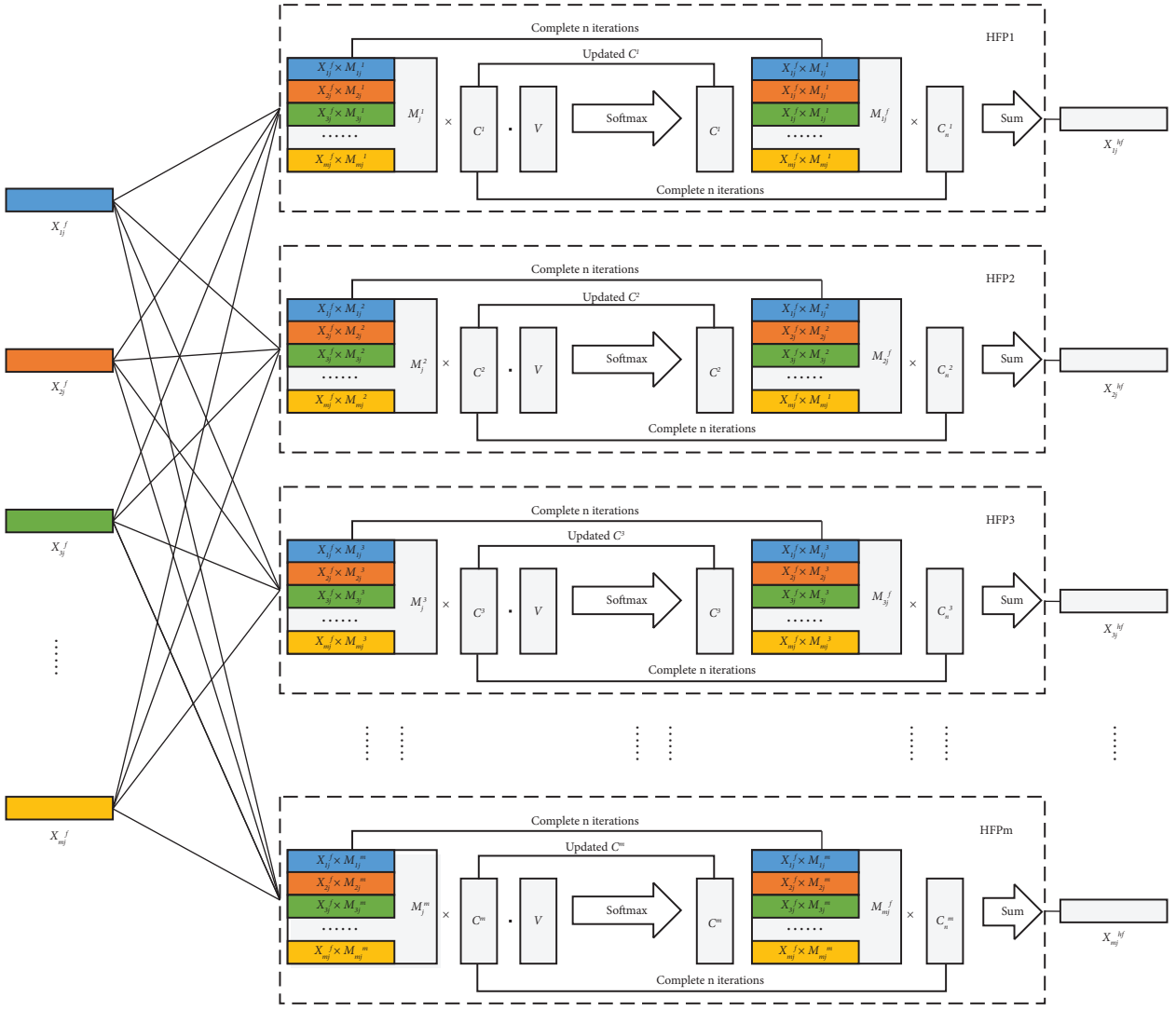

Figure 5: High-energy filtering process.

In the actual implementation process, X_{ij}^{f} can be vertically concatenated to form a spectrum matrix X_j^f. Similarly, M_{ij}^{s} can be vertically concatenated to form a coefficient matrix M_j^s. The Hadamard product is then performed between X_j^f and M_j^s. This process can be represented by (10):

$$E_j^s = X_j^f \times M_j^s. \tag{10}$$

(1) High-Energy Selection Algorithm. In this paper, the element Z_{ij}^{s} in row i of E_j^s is called the primary energy wave corresponding to X_{ij}^{f}. Each element in Z_{ij}^{s} represents the amplitude of the wave at different frequencies. The sum of these amplitudes at each frequency represents the total energy of the wave. A larger amplitude indicates a more energetic wave, which can be considered as the primary component. To identify the main components Z_{ij}^{s} in E_j^s and attenuate the other nonmain components, this paper proposes a high-energy selecting (HSE) algorithm. The operation steps of the HSE algorithm are as follows:

Step 1: Initialize the importance vector $C^s = [C^s[1], C^s[2], \ldots, C^s[i], \ldots, C^s[m]]^T$, initialize the summation vector $V = [V[1], V[2], \ldots, V[i], \ldots, V[m]]^T$, initially C^s and the internal elements of V are all 1, initialize the number of iterations n.

Step 2: Calculate the sum of the amplitudes of each energy wave. Let E_j^s perform Hadamard product operation with C^s, and the process can be expressed as (11). Then, $E_j^s \times C^s$ is subjected to matrix multiplication operation with V to obtain the energy matrix aggregation matrix E_e^s, and the process can be expressed as (12).

$$E_j^s \times C^s = \begin{bmatrix} Z_{1j}^s \times C^s[1] \\ Z_{2j}^s \times C^s[2] \\ \vdots \\ Z_{ij}^s \times C^s[i] \\ \vdots \\ Z_{mj}^s \times C^s[m] \end{bmatrix} = \begin{bmatrix} Z_{1j}^s[0] \times C^s[1] & Z_{1j}^s[1] \times C^s[1] & \cdots & Z_{1j}^s[k] \times C^s[1] & \cdots & Z_{1j}^s[N-1] \times C^s[1] \\ Z_{2j}^s[0] \times C^s[2] & Z_{2j}^s[1] \times C^s[2] & \cdots & Z_{2j}^s[k] \times C^s[2] & \cdots & Z_{2j}^s[N-1] \times C^s[2] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Z_{ij}^s[0] \times C^s[i] & Z_{ij}^s[1] \times C^s[i] & \cdots & Z_{ij}^s[k] \times C^s[i] & \cdots & Z_{ij}^s[N-1] \times C^s[i] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Z_{mj}^s[0] \times C^s[m] & Z_{mj}^s[1] \times C^s[m] & \cdots & Z_{mj}^s[k] \times C^s[m] & \cdots & Z_{mj}^s[N-1] \times C^s[m] \end{bmatrix}, \tag{11}$$

$$E_e^s = E_j^s \times C^s \cdot V = \begin{bmatrix} M_e^s[1] \\ M_e^s[2] \\ \vdots \\ M_e^s[i] \\ \vdots \\ M_e^s[m] \end{bmatrix} = \begin{bmatrix} Z_{1j}^s[0] \times C^s[1] + Z_{1j}^s[1] \times C^s[1] + \cdots + Z_{1j}^s[k] \times C^s[1] + \cdots + Z_{1j}^s[N-1] \times C^s[1] \\ Z_{2j}^s[0] \times C^s[2] + Z_{2j}^s[1] \times C^s[2] + \cdots + Z_{2j}^s[k] \times C^s[2] + \cdots + Z_{2j}^s[N-1] \times C^s[2] \\ \vdots \\ Z_{ij}^s[0] \times C^s[i] + Z_{ij}^s[1] \times C^s[i] + \cdots + Z_{ij}^s[k] \times C^s[i] + \cdots + Z_{ij}^s[N-1] \times C^s[i] \\ \vdots \\ Z_{mj}^s[0] \times C^s[m] + Z_{mj}^s[1] \times C^s[m] + \cdots + Z_{mj}^s[k] \times C^s[m] + \cdots + Z_{mj}^s[N-1] \times C^s[m] \end{bmatrix}. \tag{12}$$

Step 3: The process of processing each element within E_e^s using softmax is carried out with the updated importance vector C^s, which is represented by (13).

$$C^s[i] = \frac{e^{M_e^s[i]}}{\sum_{j=1}^{m} e^{M_e^s[j]}}. \tag{13}$$

Step 4: Repeat steps 2 and 3 for n times, with the importance vector C^s obtained during the nth iteration serving as the final importance coefficients for each primary energy wave.

Step 5: Replace C^s in (11) with C_n^s, then execute (11) to assign importance coefficients to each primary energy wave, resulting in the final energy matrix aggregation matrix E_{ee}^s. Then, calculate the columnwise sum of E_{ee}^s to obtain the high-energy spectrum $X_{ij}^{hf} = \{X_{ij}^{hf}[0], X_{ij}^{hf}[1], \ldots, X_{ij}^{hf}[k], \ldots, X_{ij}^{hf}[N]\}$. The aforementioned process can be represented by (14) and (15).

$$E_{ee}^s = E_j^s \times C_n^s, \tag{14}$$

$$X_{ij}^{hf} = \alpha \cdot E_{ee}^s. \tag{15}$$

In formula (15), $\alpha = \underbrace{[1, 1, \ldots\ldots, 1]}_{m}$.

The pseudocode for the high-energy selection algorithm is shown in Algorithm 2 as follows:

Through the above analysis, it can be observed that the HSE algorithm determines the importance of each energy wave based on the relative magnitude of the 1-norm of the vector. It achieves the objective of assigning distinct importance coefficients to different energy waves through iterative iterations. Consequently, the HSE algorithm is capable of effectively identifying the primary components Z_{ij}^{s} in E_j^s while attenuating the nonprimary components. This process can be succinctly represented by (16).

$$X_{ij}^{\mathrm{hf}} = \mathrm{HSE}\left(E_j^s\right). \tag{16}$$

Input: Primary energy wave matrix E_j^s.
Output: X_{ij}^{hf}
(1) Initialize importance vector C^s, Initialize the summation vector V, Initially, the internal elements of C^s and V are both 1, The number of initial iterations n.
(2) for $t \longleftarrow 1$ to n, do /* Iterate n times */
$E_e^s \longleftarrow E_j^s \times C^s \cdot V$
Update C^s /* Use (13) to update C^s */
$E_e^s \longleftarrow E_j^s \times C^s$
Execute (14) and (15) to obtain X_{ij}^{hf}
end

Algorithm 2: HSE algorithm.

4.2.3. Inverse Fast Fourier Transform inside FNNL. Sections 4.2.1 and 4.2.2 allow for the transformation of network intrusion data eigenvalues from the time domain signal X_{ij}^t to the frequency domain representation X_{ij}^f. Subsequently, filtering is applied to obtain the high-energy frequency spectrum X_{ij}^{hf}. To maintain the invariance of the input eigenvalues' properties, it is necessary to convert the filtered frequency domain data X_{ij}^{hf} back to the time domain data X_{ij}^{ht}, which can be processed by neural networks. This conversion is achieved using the inverse discrete Fourier transform on the high-energy spectrum X_{ij}^{hf}, as shown in (17).

$$X_{ij}^{\mathrm{ht}}[k] = \frac{1}{N} \cdot \sum_{n=0}^{N-1} W_N^{-\mathrm{nk}} \cdot X_{ij}^{\mathrm{hf}}[n], \tag{17}$$

where $W_N^{-\mathrm{nk}} = e^{j\cdot 2\cdot \pi/N \cdot \mathrm{nk}}$, (17) has the same structure as (3), and only requires replacing the input $X_{ij}^t[n]$ with $X_{ij}^{hf}[n]$, the output $X_{ij}^f[k]$ with $X_{ij}^{ht}[k]$, and W_N^{nk} with $W_N^{-\mathrm{nk}}$. Thus, (3) and (17) are completely identical. However, directly using (17) to process X_{ij}^{hf} in terms of computational cost is impractical, as discussed in Section 4.2.1. To address this, a similar approach as in Section 4.2.1 is employed here, using the inverse fast Fourier transform (IFFT) to accelerate the computation process of (17). Since (17) shares the same structure as (3), the operation of IFFT aligns with that of FFT. Initially, X_{ij}^{hf} is processed using Rader's algorithm to obtain the sequence $X_{ij}^{hf}{}_{\text{-end}}$, which will undergo the butterfly operation. Once $X_{ij}^{hf}{}_{\text{-end}}$ is obtained, X_{ij}^{ht} can be derived by applying the butterfly operation rules to $X_{ij}^{hf}{}_{\text{-end}}$. Specifically, this can be achieved by modifying the input of Algorithm 2 to $X_{ij}^{hf}{}_{\text{-end}}$. This entire process can be simplified as shown in (18).

$$X_{ij}^{ht} = \mathrm{IFFT}\left(X_{ij}^{hf}\right). \tag{18}$$

After obtaining X_{ij}^{ht}, it is necessary to perform a summation and compression operation on X_{ij}^{ht}, specifically by performing matrix addition on $X_{1j}^{ht}, X_{2j}^{ht}, \ldots, X_{ij}^{ht}, \ldots$, and X_{mj}^{ht}. This operation allows for the preservation of the extracted feature information from DFNNB while enabling further processing of the extracted feature data by subsequent network structures. The process of obtaining the final output X_j of FNNL through the summation and compression of X_{ij}^{ht}, as shown in Figure 2, is represented by (19).

$$X_j = X_{1j}^{\mathrm{ht}} + X_{2j}^{\mathrm{ht}} + \cdots\cdots + X_{ij}^{\mathrm{ht}} + \cdots\cdots + X_{mj}^{\mathrm{ht}}. \tag{19}$$

4.3. Information Back Propagation and Parameter Update. In FNN, there are two parts: DNN and DFNNB. The back propagation process of gradient information in the DNN part has been extensively discussed in existing literature. Therefore, in this paper, we focus on elaborating the back propagation process of gradient information within the DFNNB. Specifically, we analyze the back propagation process of internal gradient information in the jth DFNNB. The variables that require gradient information calculation in the jth DFNNB include the coefficient matrix M_j^s in FNNL, and the weights W_{jh} and W_{je} in HNN. Let L denote the loss between the network output value and the labeled value. From equations (10), (16), (18), and (19), the gradient relationship between L and the coefficient matrix M_j^s, as presented in Section 4.1, can be expressed by (20).

$$\begin{aligned}\frac{\partial L}{\partial M_j^s} = \frac{\partial L}{\partial X_j}\frac{\partial X_j}{\partial X_{ij}^{\mathrm{ht}}}\frac{\partial X_{ij}^{\mathrm{ht}}}{\partial \mathrm{IFFT}\left(X_{ij}^{\mathrm{hf}}\right)}\frac{\partial \mathrm{IFFT}\left(X_{ij}^{\mathrm{hf}}\right)}{\partial X_{ij}^{\mathrm{hf}}}\frac{\partial X_{ij}^{\mathrm{hf}}}{\partial \mathrm{HSE}\left(E_j^s\right)} \\ \cdot \frac{\partial \mathrm{HSE}\left(E_j^s\right)}{\partial E_j^s}\frac{\partial E_j^s}{\partial\left(X_j^f \times M_j^s\right)}\frac{\partial\left(X_j^f \times M_j^s\right)}{\partial M_j^s}.\end{aligned} \tag{20}$$

In order to be able to calculate the gradient of the coefficient matrix M_j^s in FNNL smoothly on the computer, a further transformation of (20) is required. According to the analysis in Section 4.1, it can be seen that (18) is equivalent to (17), and (17) can be rewritten in the form of matrix operation as shown in (21):

$$X_{ij}^{\mathrm{ht}} = V_N \cdot X_{ij}^{\mathrm{hf}}, \tag{21}$$

where V_N can be expressed by (22):

$$V_N = \frac{1}{N} \cdot \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & \cdots & W_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \cdots & W_N^{-(N-1)(N-1)} \end{bmatrix}. \tag{22}$$

In (22), $W_N = e^{-j2\pi/N}$.

From the analysis in Section 4.2.2 and (14) and (15), it can be deduced that the work accomplished by (16) in the high-energy filtering process is equivalent to (23):

$$X_{ij}^{\mathrm{hf}} = \alpha \cdot \left(E_j^s \times C_n^s\right). \tag{23}$$

From equations (20), (21), and (23), the gradient of the coefficient matrix M_j^s in FNNL can be expressed by (24) as follows:

$$\frac{\partial L}{\partial M_j^s} = \frac{\partial L}{\partial X_j}\frac{\partial X_j}{\partial X_{ij}^{\mathrm{ht}}}\frac{\partial X_{ij}^{\mathrm{ht}}}{\partial\left(V_N \cdot X_{ij}^{\mathrm{hf}}\right)}\frac{\partial\left(V_N \cdot X_{ij}^{\mathrm{hf}}\right)}{\partial X_{ij}^{\mathrm{hf}}}\frac{\partial X_{ij}^{\mathrm{hf}}}{\partial\left(\alpha \cdot \left(E_j^s \times C_n^s\right)\right)}\frac{\partial\left(\alpha \cdot \left(E_j^s \times C_n^s\right)\right)}{\partial E_j^s}\frac{\partial E_j^s}{\partial\left(X_j^f \times M_j^s\right)}\frac{\partial\left(X_j^f \times M_j^s\right)}{\partial M_j^s}. \tag{24}$$

Equation (24) represents the back propagation process of gradient information for the coefficient matrix M_j^s inside FNNL in a DFNNB. X_j^f is a matrix formed by combining X_{ij}^t, where each X_{ij}^f is obtained from the corresponding X_{ij}^t through FFT transformation, which is equivalent to applying DFT to each X_{ij}^t to obtain the corresponding X_{ij}^t. The process of obtaining X_j^f by performing DFT on X_{ij}^t can be expressed by (25).

$$X_j^f = F_N \cdot X_j^t. \tag{25}$$

In (25), FN is represented by (26).

$$F_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}. \tag{26}$$

X_{ij}^t is the output of HNN, and the only trainable parameters in HNN are the weights W_{je} and W_{jh}. According to the chain derivation rule, the gradient information of W_{je} and W_{jh} in HNN can be back propagated. Equations (27) and (28) represent the back propagation process of gradient information for W_{jh} and W_{je} in DFNNB, respectively.

$$\begin{aligned}\frac{\partial L}{\partial W_{jh}} = {} & \frac{\partial L}{\partial X_j}\frac{\partial X_j}{\partial X_{ij}^{\mathrm{ht}}}\frac{\partial X_{ij}^{\mathrm{ht}}}{\partial\left(V_N \cdot X_{ij}^{hf}\right)}\frac{\partial\left(V_N \cdot X_{ij}^{\mathrm{hf}}\right)}{\partial X_{ij}^{\mathrm{hf}}}\frac{\partial X_{ij}^{\mathrm{hf}}}{\partial\left(\alpha \cdot \left(E_j^s \times C_n^s\right)\right)}\frac{\partial\left(\alpha \cdot \left(E_j^s \times C_n^s\right)\right)}{\partial E_j^s} \\ & \cdot \frac{\partial E_j^s}{\partial\left(X_j^f \times M_j^s\right)}\frac{\partial\left(X_j^f \times M_j^s\right)}{\partial X_j^f}\frac{\partial X_j^f}{\partial\left(F_N \cdot X_j^t\right)}\frac{\partial\left(F_N \cdot X_j^t\right)}{\partial X_j^t}\frac{\partial X_j^t}{\partial\left(X_j^e \times W_{jh}\right)}\frac{\partial\left(X_j^e \times W_{jh}\right)}{\partial W_{jh}},\end{aligned} \tag{27}$$

$$\begin{aligned}\frac{\partial L}{\partial W_{je}} = {} & \frac{\partial L}{\partial X_j}\frac{\partial X_j}{\partial X_{ij}^{\mathrm{ht}}}\frac{\partial X_{ij}^{\mathrm{ht}}}{\partial\left(V_N \cdot X_{ij}^{\mathrm{hf}}\right)}\frac{\partial\left(V_N \cdot X_{ij}^{\mathrm{hf}}\right)}{\partial X_{ij}^{\mathrm{hf}}}\frac{\partial X_{ij}^{\mathrm{hf}}}{\partial\left(\alpha \cdot \left(E_j^s \times C_n^s\right)\right)}\frac{\partial\left(\alpha \cdot \left(E_j^s \times C_n^s\right)\right)}{\partial E_j^s}\frac{\partial E_j^s}{\partial\left(X_j^f \times M_j^s\right)}\frac{\partial\left(X_j^f \times M_j^s\right)}{\partial X_j^f} \\ & \cdot \frac{\partial X_j^f}{\partial\left(F_N \cdot X_j^t\right)}\frac{\partial\left(F_N \cdot X_j^t\right)}{\partial X_j^t}\frac{\partial X_j^t}{\partial\left(X_j^e \times W_h\right)}\frac{\partial\left(X_j^e \times W_h\right)}{\partial X_j^e}\frac{\partial X_j^e}{\partial\left(X_{j-1} \cdot W_{je}\right)}\frac{\partial\left(X_{j-1} \cdot W_{je}\right)}{\partial W_{je}}.\end{aligned} \tag{28}$$

Equations (24), (27), and (28) show the back propagation process of gradient information of the to-be-trained variables M_j^s, W_{jh}, and W_{je} in the jth DFNNB. The back propagation process of the gradient information of the parameters to be trained in the remaining DFNNBs is the same as the back propagation process of the gradient information of the variables to be trained in the jth DFNNB.

From equations (24), (27), and (28) combined with the gradient descent algorithm, the updating process of the coefficient matrices M_j^s of FNNL in the jth DFNNB, and the weights W_{jh} and W_{je} in the HNN can be expressed in equations (29), (30), and (31), respectively, as follows:

$$M_j^{s'} = M_j^s - \eta\frac{\partial L}{\partial M_j^s}, \tag{29}$$

$$W_{jh}^{'} = W_{jh} - \eta\frac{\partial L}{\partial W_{jh}}, \tag{30}$$

$$W_{je}^{'} = W_{je} - \eta\frac{\partial L}{\partial W_{je}}. \tag{31}$$

$M_j^{s'}$, $W_{jh}^{'}$, and $W_{je}^{'}$ denote the updated M_j^s, W_{jh}, and W_{je}, respectively, and η is the learning rate, which is able to complete the updating of the parameters in the DFNNB according to equations (29)–(31).

Sections 4.1 and 4.2 elaborate the structure and information forward propagation process of DFNNB in FNN, and Section 4.3 describes the backward propagation process of the information of the parameter gradient to be updated and the parameter updating process in DFNNB.

4.4. Time Complexity Analysis. A DFNNB consists of HNN (Hadamard Neural Network) and FNNL (Fourier Neural Network Layer). In this subsection, the time complexity of executing a DFNNB is analyzed.

First, let us analyze the time complexity of HNN. The computational process of HNN involves equations (1) and (2). Equation (1) represents the matrix multiplication of an $N * 1$ vector with a $1 * M$ vector, with a time complexity of O(NM). Equation (2) involves the Hadamard product of an $N * M$ matrix with another $N * M$ matrix, which also has a time complexity of O(NM). Therefore, the time complexity of HNN is O(NM).

Next, let us consider the time complexity of the Fast Fourier Transform (FFT). As mentioned in Section 4.2, this process requires the fast Fourier transform of M vectors, resulting in a time complexity of O(MNlogN). Through the FFT, a frequency domain representation of the data, X_{ij}^f, is obtained. Subsequently, a high-energy filtering process is performed on X_{ij}^f. In this process, each X_{ij}^f is multiplied elementwise with a coefficient vector M_{ij}^s to obtain the primary energy wave Z_{ij}^s. Therefore, the time complexity of processing one X_{ij}^f is O(N). Since there are M X_{ij}^f in one high-energy filtering process, the time complexity of processing M X_{ij}^f is O(MN).

All Z_{ij}^s together form the primary energy wave matrix E_j^s. E_j^s is then processed using a high-energy selection algorithm that involves K iterations. In each iteration, the key computational steps include (11)–(13). Equation (11) represents the Hadamard product of matrices, with a time complexity of O(MN). Equation (12) represents matrix multiplication, also with a time complexity of O(MN). Equation (13) represents the softmax function, with a time complexity of O(N). Therefore, the time complexity of the high-energy selection algorithm with 1 iteration is O(MN), while the time complexity of the high-energy selection algorithm with K iterations is O(KMN).

Finally, according to the analysis in Section 4.2.3, the inverse transform of the Fast Fourier Transform (FFT) needs to be performed. Its time complexity is the same as the FFT, which is O(MNlogN).

In summary, the time complexity of a DFNNB can be expressed as O(NM) + O(MN) + O(KMN) + O(MNlogN).

5. FNN Detection Steps

Step 1: Perform preprocessing operations on the data, including normalization and splitting the dataset into a training set and a test set.

Step 2: Training FNN.

(1) Determine the structure of the FNN and initialize its parameters.

(2) Input the training set data into the first DFNNB. Using equations (1) and (2), the HNN fits the network intrusion data in the multitemporal space. Then, the fitted samples are processed by FNNL, resulting in filtered and integrated feature data according to equations (7), (16), (18), and (19). These data serve as the output of the DFNNB and are also used as the input for the next DFNNB layer.

(3) Use the output of the previous DFNNB layer as the input for the current DFNNB layer.

(4) Repeat step (3) until the output of the final DFNNB layer is obtained. This output is then used as the input for the DNN part of the FNN, and the output of the DNN part becomes the final output of the FNN.

(5) Calculate the loss value between the FNN's output and the labeled values. Use the gradient descent algorithm to update the trainable parameters in the FNN, reducing the difference between the output and the labeled values. Continue this process until the specified training iterations are reached.

Step 3: FNN testing. Input the test dataset into the trained FNN to obtain the classification results for each test data.

The overall flowchart of the FNN intrusion detection is illustrated in Figure 6.

6. Datasets Description

In order to verify the detection capability of FNN in the face of network intrusion data, this paper not only carries out intrusion detection implementation on the older intrusion detection datasets KDD CUP99 and NSL-KDD but also conducts intrusion detection experiments on the newer intrusion detection datasets UNSW-NB15 and CICIDS2017. This demonstrates the intrusion detection capability of FNN in a more comprehensive way and ensures that the experiments are more convincing.

(1) KDD Cup99 [51]: This dataset is derived from the 1998 DARPA Intrusion Detection Evaluation Program. All network data originate from a simulated United States Air Force local area network, which includes various simulated attacks. The dataset consists of 41 feature attributes and 1 class label. Among the 41 feature attributes, 9 are discrete (symbolic) and the rest are continuous. The class label consists of 5 categories: Normal, Probe, DoS, U2R, and R2L. The dataset suffers from class imbalance, resulting in biased results towards the more frequent data. Table 3 provides a detailed description of the KDD Cup99 dataset.

(2) NSL-KDD [52]: This dataset is an improvement over the KDD Cup99 dataset, as it removes redundant data and duplicate records present in the original

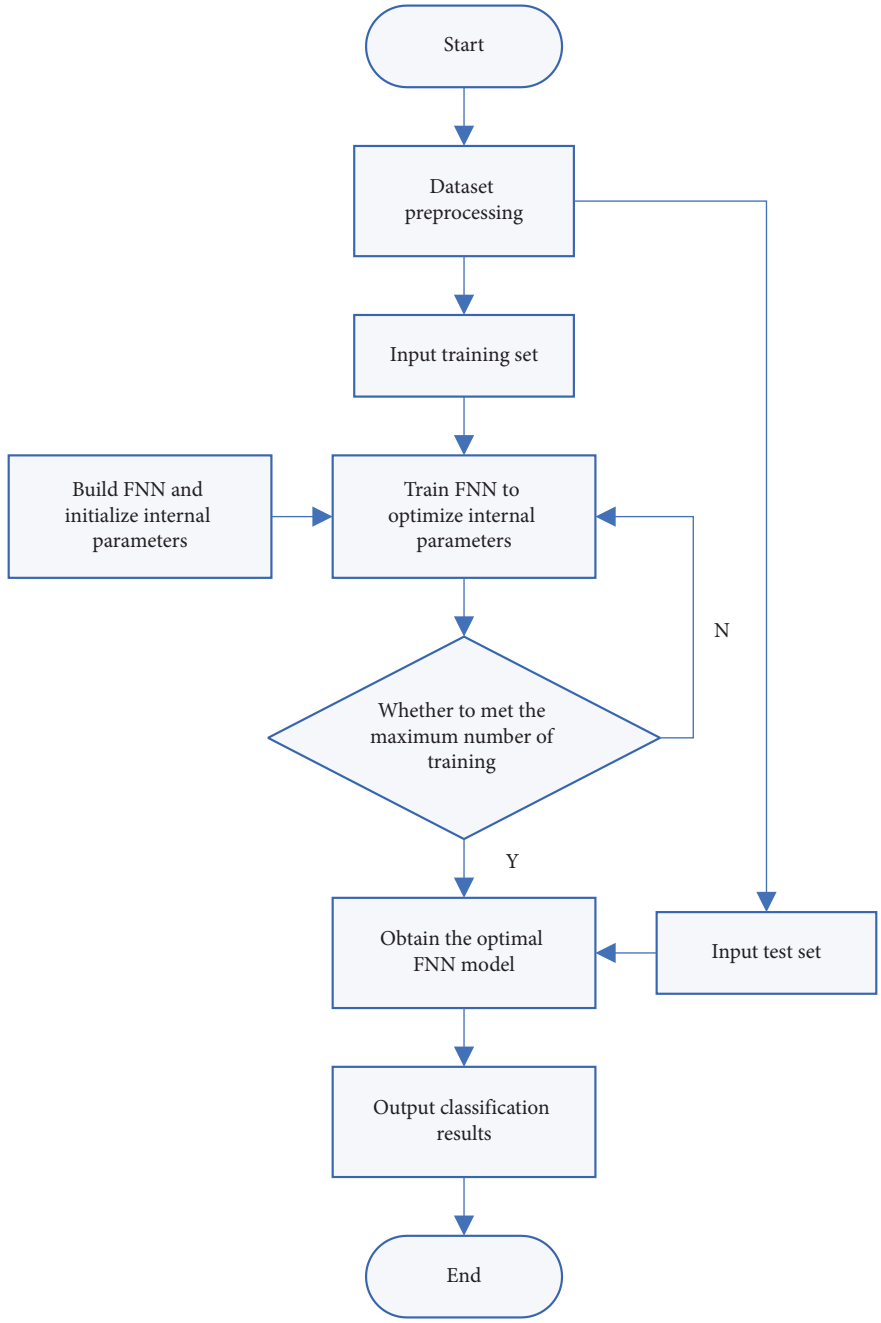

Figure 6: FNN intrusion detection flowchart.

dataset. This ensures that classifiers do not exhibit bias towards more frequent records, resulting in more accurate detection rates. Compared to the KDD Cup99 dataset, this dataset is more suitable for intrusion detection. Table 3 provides a detailed description of the NSL-KDD dataset.

(3) UNSW-NB15 [53, 54]: This dataset was created by the Network Security Laboratory of the Australian Centre for Cyber Security (ACCS) using the IXIA PerfectStorm tool. It aims to provide a more realistic representation of real-world network data and serves as a comprehensive dataset for network attack traffic. The dataset consists of one category representing normal traffic and nine categories representing various types of abnormal traffic. It contains 42 feature attributes and 1 class label attribute. A detailed description of the UNSW-NB15 dataset can be found in Table 4.

(4) CICIDS2017 [55, 56]: This dataset was developed within the Canadian Institute for Cybersecurity and consists of five days of both normal and attack traffic data. It provides labeled, flow-based data for updated attacks and normal usage in a simulated office environment. The dataset also includes network flow analysis results using CICFlowMeter. The dataset is labeled based on

flows, reflecting timestamps, source IP addresses, destination IP addresses, source port numbers, and destination port numbers. It also includes metadata on protocols and the most significant attack types. The dataset encompasses seven attack categories, reflecting recent attack scenarios. A detailed description of the CICIDS2017 dataset can be found in Table 5.

7. Evaluation Index

Due to the imbalance in the intrusion detection dataset, with a significant disparity between the number of normal and abnormal samples, the accuracy rate alone cannot provide a comprehensive evaluation of the intrusion detection algorithm's performance. Therefore, this paper introduces additional evaluation metrics such as accuracy, precision, recall, $F1$-score, and AUC values to assess the effectiveness of the FNN algorithm [57]. These metrics are derived from the confusion matrix presented in Table 6, enabling a more comprehensive analysis of the algorithm's performance.

The above evaluation indicators are defined as follows:

Accuracy: It estimates the ratio of the number of correctly identified samples to the entire test set. The higher the accuracy, the better the performance of the neural network (accuracy $\in [0, 1]$). It is a good metric for test datasets containing balanced classes. The accuracy rate is defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{32}$$

Precision: It estimates the ratio of the number of correctly identified normal samples to the actual predicted normal samples. The higher the precision, the better the performance of the neural network (precision $\in [0, 1]$). The precision is defined as follows, where k is the sample category:

$$\text{precision}_k = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{33}$$

Recall: It estimates the ratio of correctly classified normal samples to the total number of normal samples. If the recall rate is higher, the neural network model is better (recall $\in [0, 1]$). Recall is defined as follows:

$$\text{recall}_k = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{34}$$

$F1$-score: It is the harmonic average of precision rate and recall rate. The higher the $F1$-score, the better the performance of the neural network ($F1$-score $\in [0, 1]$). $F1$-score is defined as follows, where α_k is the weight, representing the proportion of different sample categories:

$$\begin{aligned} F1_K &= \frac{2 \cdot \text{precision}_k \cdot \text{recall}_k}{\text{precision}_k + \text{recall}_k}, \\ F1 - \text{score} &= \left(\sum \alpha_k \times F1_k\right)^2. \end{aligned} \tag{35}$$

ROC (receiver operating characteristic) curve: Its horizontal axis is false positive rate (FPR), and its vertical axis is true positive rate (TPR). The value of AUC is the area under the ROC curve, which is used as a comparison index for neural network models together with ROC. The higher the AUC value, the better the neural network model. The AUC value is calculated as follows:

$$\text{AUC} = \int_0^1 \frac{\text{TP}}{\text{TP} + \text{FN}} d \frac{\text{FP}}{\text{TN} + \text{FP}}, \tag{36}$$

where TP (true positive) is the total number of samples correctly classified into the normal class, TN (true negative) is the total number of samples correctly classified into the attack class, FP (false positive) is the total number of samples that misclassified the normal class as an attack class, and FN (false negative) is the total number of attack samples that are misclassified as normal.

8. Experiments and Discussions

In this section, two sets of comparative experiments were designed with the following objectives: the first group aimed to validate the effectiveness of FNN and explore its performance on different datasets, while the second group aimed to investigate the impact of the number of iterations of the HSE algorithm on the performance of FNN.

In the field of intrusion detection, it is crucial to validate the effectiveness of an intrusion detection method by applying it to various real-world scenarios. To evaluate the effectiveness of FNN, this study utilized FNN for detecting intrusions in the KDD Cup99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets. Multiple evaluation metrics such as accuracy, precision, recall, $F1$-score, ROC curve, and AUC value were employed to assess the performance of FNN.

The deep learning framework used in this study was TensorFlow 2.13.0 (CPU version), and the machine learning library employed was scikit-learn 1.0.2. The programming language used was Python 3.9.1. The hardware configuration for this experiment consisted of an AMD Ryzen 7 5800H processor and 16 GB RAM. The operating system used was Windows 11.

In the specific implementation of the experiments, for the binary classification experiments, normal samples are labeled as 0 and attack samples are labeled as 1. For the multiclassification experiments using the CICIDS2017 dataset as an example, normal samples are labeled as 0, Bot attack samples as 1, BruteForce attack samples as 2, Dos attack samples as 3, Infiltration attack samples as 4, PortScan attack samples as 5, and WebAttack samples as 6. One-hot encoding is employed for the labeling process. The epoch value for both the FNN and the comparative deep learning models during training is set to 50. The Adam optimization function with a learning rate of 0.001 is utilized, and the Categorical Crossentropy is adopted as the loss function.

8.1. Comparative Experiment One. The FNN network structure significantly influences the detection performance of FNN. To validate the effectiveness of FNN, this paper

Table 3: Training and testing connection records of KDD Cup99 and NSL-KDD.

Attack category	Description	10% dataset KDD Cup99		20% dataset NSL-KDD	
		Train	Test	Train	Test
Normal	Normal connection record	97277	60592	13357	9690
Probe	Get detailed statistics for system and network configuration details	4107	4166	2289	2421
DoS	Attack aims to reduce network resources	391438	229825	9234	7435
U2R	Get permission or super user access on a specific computer	52	228	11	200
R2L	Illegal access to remote computer	1126	16189	209	2754
Total		494000	311000	25100	22500

Table 4: Training and testing connection records of UNSW-NB15.

Attack_cat	Description	Train	Test
Normal	Normal connection record	56000	37000
Backdoor	Technology that bypasses security controls to gain access to programs or systems	1746	583
Analysis	An intrusion method to penetrate web applications through ports, emails, and web scripts	2000	677
Fuzzers	An attack method that attempts to find security vulnerabilities in programs, operating systems, or networks by inputting a large amount of random data to make it crash	18184	6062
Shellcode	An attack method that controls the target machine by sending code that exploits specific vulnerabilities	1133	378
Reconnaissance	An attack method that collects computer network information in order to evade security control	10491	3496
Exploit	A piece of code that triggers a vulnerability (or several vulnerabilities) to control the target system	33352	11132
DoS	An attack method that directly or indirectly exhausts the resources of the attacked object, so that the target computer or network cannot provide normal services or resource access	12264	4089
Worms	A malicious computer virus that spreads through the network and actively attacks	130	44
Genertic	A technology that uses a hash function to collide each block cipher without considering the configuration of the block cipher	40000	18871
Total		175300	82332

Table 5: Training and testing connection records of CICIDS2017.

Attack_cat	Description	Train	Test
Normal	Normal connection record	1703490	567830
Bot	Automated large-scale attacks using malicious software or programs to obtain confidential information, cause system downtime, or perform other malicious activities	1467	489
BruteForce	Brute-force a system's security measures by trying a large number of possible password combinations or keys to gain unauthorized access or control privileges	10374	3458
DoS	Depleting the resources of the target by directly or indirectly depleting the resources of the target computer or network so that the target computer or network cannot provide normal services or access to resources	284811	94937
Infiltration	Gain unauthorized access by penetrating the defense layer of the target system and infiltrating, controlling or stealing sensitive information in the system	27	9
PortScan	Scanning the open ports of a target host to detect system vulnerabilities and services in order to find exploitable entry points for unauthorized access or attacks	119103	39701
WebAttack	Attacks against users' online behavior or devices such as web servers	1635	545
Total		2120907	706969

Table 6: Confusion matrix.

Reality	Forecast result	
	NO	YES
Actual: NO	TN (true negative)	FP (false positive)
Actual: YES	FN (false negative)	TP (true positive)

proposes three FNN network structures, namely FNN1, FNN2, and FNN3. Each structure incorporates a varying number of layers of DFNNB (FNN1 with 1 layer, FNN2 with 2 layers, and FNN3 with 3 layers). DNN is interleaved between each DFNNB layer, enabling the learned feature representations from DFNNB to be mapped to different feature spaces. In the first set of experiments, the number of iterations of the HSE algorithm for the different FNN networks is fixed at 3. The detailed structure of FNN1 to FNN3 is presented in Table 7. The structure of DNN, CNN, RNN, and LSTM used in the comparison experiments is shown in Table 8.

When performing binary classification tasks on the data, the number of neurons in the last layer of the DNN network is 1, and the activation function used is the sigmoid function. For multiclassification tasks, the number of neurons in the last layer of the DNN network is 5, 10, and 7 according to the different datasets (the number of neurons in the experiment on the KDD Cup99 and NSL-KDD dataset is 5, the number of neurons in the experiment on the UNSW-NB15 dataset is 10, and the number of neurons in the experiment on the CICIDS2017 dataset is 7), and the activation functions are all softmax functions.

Tables 9–16 presents the accuracy, precision, recall, and $F1$-score values of FNN1, FNN2, and FNN3 in binary classification and multiclassification using classical deep learning and machine learning algorithms, including DNN, CNN, RNN, LSTM, RF, LR, KNN, DT, and SVM, on the KDD Cup99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets. Tables 9–16 demonstrate the superior detection performance of all algorithms. Overall, the performance of each algorithm on KDD Cup99, NSL-KDD, and CICIDS2017 is better than that on UNSW-NB15. This discrepancy can be attributed to the higher data complexity of UNSW-NB15 compared to other datasets. In deep learning methods, CNN, RNN, and LSTM exhibit relatively poor performance. This can be attributed to the absence of translation invariance and clear sequential relationships in network intrusion data. Among traditional machine learning methods, RF and KNN demonstrate more prominent performance. This is primarily due to the fact that RF is an ensemble algorithm that incorporates multiple decision trees, providing certain advantages over individual traditional machine learning algorithms. Additionally, the network intrusion data have a relatively low feature dimension, which contributes to the advantageous performance of KNN in handling such data. Due to the higher performance requirements of algorithms for multivariate classification compared to binary classification, the overall performance of each algorithm decreases in various performance metrics when performing multiclassification on each dataset. This performance decrease is particularly evident in the classification of the UNSW-NB15 dataset, which is more complex than the other three datasets and includes nine types of attacks, including newer attack types. Therefore, it is expected that the algorithms would exhibit a significant decrease in performance on the UNSW-NB15 dataset. In general, each algorithm in the experiments effectively detects the attack samples in the dataset. Analyzing Tables 9–16, FNN demonstrates superior detection performance compared to other algorithms in the majority of cases. Furthermore, the detection performance of FNN continues to improve with an increase in the number of DFNNB layers within the FNN.

In the vast majority of cases, FNN demonstrates superior detection performance compared to traditional methods. In the binary classification test on the KDD Cup99 dataset, each detection method achieves excellent detection results. Specifically, FNN2 and FNN3 exhibit accuracy rates exceeding 0.996, which is a 19.2% improvement compared to the worst-performing LSTM. Moreover, as the depth of FNN increases, both precision and recall also increase. Notably, FNN3 achieves precision and recall rates of 0.999 and 0.997, respectively, surpassing other methods. However, overall, the performance differences between FNN2 and FNN3 are not significant, indicating that further increasing the depth of FNN does not yield substantial improvements in performance. In the multiclassification test on the KDD Cup99 dataset, most algorithms, except LSTM, exhibited a slight decrease in performance. Notably, FNN3 demonstrated the smallest decrease in accuracy. One of the factors contributing to the marginal improvement in LSTM's performance is the use of a multiclassification dataset, which partially alleviates the issue of class imbalance present in the binary classification dataset. This observation suggests that LSTM is less adept at handling significant imbalances in sample quantities. Furthermore, FNN consistently outperforms other algorithms in terms of precision and recall, indicating its superiority. Due to certain limitations of the KDD Cup99 dataset, this study proceeds with further experiments on the NSL-KDD dataset, which is an improved version of the KDD Cup99 dataset. The NSL-KDD dataset provides a better means to evaluate the performance of various algorithms in intrusion detection. Notably, the NSL-KDD dataset has a significantly reduced number of samples in the training set, resulting in fewer data features being learned by the detection methods. As a consequence, all algorithms experience varying degrees of degradation in their detection capabilities. However, when comparing Tables 9–16, it can be observed that FNN demonstrates a relatively smaller decrease in performance compared to other algorithms. Moreover, FNN3 consistently achieves optimal performance indicators when performing both binary classification and multiclassification tasks, indicating its superiority over other algorithms in classifying the KDD Cup99 and NSL-KDD datasets.

When performing binary classification on the UNSW-NB15 dataset, FNN consistently outperforms other algorithms in the majority of cases. Furthermore, increasing the depth of FNN leads to a certain degree of performance improvement. The UNSW-NB15 dataset is relatively new and contains numerous novel attack types, resulting in higher data complexity compared to the NSL-KDD and KDD Cup99 datasets. At this stage, the accuracy rates of DNN, CNN, LSTM, LR, DT, and SVM fall below 0.800, with SVM achieving an accuracy rate of only 0.686. Additionally, the accuracy rates of RNN and KNN are below 0.850. The algorithms with accuracy rates exceeding 0.900 are FNN1,

Table 7: Network structure of FNN.

Network name	Network structure	n	Number of neurons	Activation function
FNN1	DNN	—	42, 64	Tanh
	DFNNB1	3	—	—
	DNN	—	64, 1/5/10/7	Tanh, sigmoid, softmax
FNN2	DNN	—	42, 64	Tanh
	DFNNB1	3	—	—
	DNN	—	64, 128	Tanh
	DFNNB2	3	—	—
	DNN	—	64, 1/5/10/7	Tanh, sigmoid, softmax
FNN3	DNN	—	42, 64	Tanh
	DFNNB1	3	—	—
	DNN	—	64, 128	Tanh
	DFNNB2	3	—	—
	DNN	—	128, 256	Tanh
	DFNNB3	3	—	—
	DNN	—	64, 1/5/10/7	Tanh, sigmoid, softmax

Table 8: Network structure of DNN, CNN, RNN, and LSTM.

Model	Network layer	Number of neurons	Activation
DNN	Hidden layer 1	41	Tanh
	Hidden layer 2	68	Tanh
	Hidden layer 3	128	Tanh
	Output layer (dense)	1/5/10/7	Sigmoid, softmax
CNN	Inception-V1		
LSTM	LSTM	42	Tanh
	Global average pooling	—	—
	Dense	268, 1/5/10/7	Tanh, sigmoid, softmax
RNN	RNN	42	Tanh
	Global average pooling	—	—
	Dense	268, 1/5/10/7	Tanh, sigmoid, softmax

Table 9: KDD Cup99 binary classification test results.

Model	Accuracy	Precision	Recall	$F1$-score
FNN1	0.988	0.997	0.989	0.993
FNN2	0.996	0.999	0.996	0.997
FNN3	0.997	0.999	0.997	0.998
DNN	0.971	0.999	0.965	0.982
CNN	0.961	0.999	0.953	0.975
RNN	0.961	0.992	0.960	0.976
LSTM	0.804	0.804	1.000	0.891
RF	0.967	0.999	0.959	0.979
LR	0.957	0.998	0.948	0.972
KNN	0.991	0.998	0.991	0.994
DT	0.918	0.933	0.967	0.950
SVM	0.945	0.942	0.993	0.967

FNN2, FNN3, and RF, with RF having an accuracy rate of 0.900, which is lower than that of FNN1, FNN2, and FNN3. In terms of precision and recall, FNN1, FNN2, and FNN3 outperform other methods. When performing multiclassification on the UNSW-NB15 dataset, the accuracy rates of FNN1, FNN2, and FNN3 are 0.790, 0.846, and 0.853, respectively. At this stage, the accuracy rates of RF and KNN are 0.809 and 0.802, respectively, which are higher than FNN1 but lower than FNN3 and FNN2. Overall, FNN2 performs similarly to FNN3, and both outperform other algorithms.

Through the analysis of Tables 15 and 16, it is found that FNN's performance in binary classification on the CICIDS2017 dataset is not as good as in multiclass classification. This is because the CICIDS2017 dataset has less noise contamination, and when performing binary classification, there are only two labels, so the influence of data noise on the classification results is relatively small. However, if the filtering frequency of FNN is too high, it will lead to the loss of non-noise information. Therefore, the effectiveness of our designed high-energy selection algorithm cannot be fully exerted. For the selection of the

Table 10: KDD Cup99 multiclass classification test results.

Model	Accuracy	Precision	Recall	*F*1-score
FNN1	0.957	0.957	0.957	0.948
FNN2	0.981	0.982	0.981	0.980
FNN3	0.990	0.990	0.990	0.989
DNN	0.939	0.898	0.939	0.916
CNN	0.928	0.876	0.928	0.899
RNN	0.928	0.874	0.928	0.898
LSTM	0.864	0.834	0.864	0.841
RF	0.920	0.889	0.920	0.894
LR	0.905	0.913	0.905	0.889
KNN	0.942	0.950	0.942	0.925
DT	0.888	0.833	0.888	0.859
SVM	0.932	0.922	0.932	0.909

Table 11: NSL-KDD binary classification test results.

Model	Accuracy	Precision	Recall	*F*1-score
FNN1	0.948	0.954	0.956	0.955
FNN2	0.978	0.976	0.985	0.980
FNN3	0.982	0.982	0.986	0.984
DNN	0.950	0.931	0.984	0.957
CNN	0.833	0.922	0.771	0.840
RNN	0.890	0.961	0.841	0.897
LSTM	0.812	0.865	0.794	0.828
RF	0.938	0.927	0.968	0.947
LR	0.795	0.908	0.712	0.798
KNN	0.977	0.977	0.982	0.980
DT	0.833	0.878	0.820	0.848
SVM	0.853	0.874	0.867	0.870

Table 12: NSL-KDD multiclass classification test results.

Model	Accuracy	Precision	Recall	*F*1-score
FNN1	0.963	0.963	0.963	0.962
FNN2	0.975	0.975	0.975	0.975
FNN3	0.977	0.977	0.977	0.977
DNN	0.933	0.931	0.933	0.930
CNN	0.743	0.580	0.743	0.647
RNN	0.801	0.789	0.801	0.781
LSTM	0.700	0.534	0.700	0.605
RF	0.827	0.867	0.827	0.796
LR	0.725	0.560	0.725	0.629
KNN	0.914	0.912	0.914	0.912
DT	0.780	0.686	0.780	0.727
SVM	0.442	0.409	0.442	0.284

Table 13: UNSW-NB15 binary classification test results.

Model	Accuracy	Precision	Recall	*F*1-score
FNN1	0.927	0.932	0.936	0.934
FNN2	0.944	0.953	0.944	0.949
FNN3	0.944	0.954	0.943	0.949
DNN	0.761	0.867	0.668	0.755
CNN	0.690	0.740	0.674	0.706
RNN	0.813	0.815	0.854	0.834
LSTM	0.749	0.858	0.652	0.740
RF	0.900	0.898	0.924	0.911
LR	0.761	0.862	0.675	0.757
KNN	0.848	0.864	0.859	0.861
DT	0.799	0.826	0.804	0.815
SVM	0.686	0.735	0.673	0.703

Table 14: UNSW-NB15 multiclass classification test results.

Model	Accuracy	Precision	Recall	*F*1-score
FNN1	0.790	0.775	0.790	0.774
FNN2	0.846	0.839	0.846	0.842
FNN3	0.853	0.844	0.853	0.847
DNN	0.676	0.586	0.676	0.592
CNN	0.771	0.685	0.771	0.705
RNN	0.641	0.439	0.641	0.520
LSTM	0.638	0.447	0.638	0.522
RF	0.809	0.809	0.809	0.759
LR	0.792	0.775	0.792	0.775
KNN	0.802	0.785	0.802	0.778
DT	0.764	0.674	0.764	0.698
SVM	0.648	0.669	0.648	0.632

Table 15: CICIDS2017 binary classification test results.

Model	Accuracy	Precision	Recall	*F*1-score
FNN1	0.991	0.979	0.978	0.978
FNN2	0.994	0.985	0.984	0.984
FNN3	0.993	0.984	0.981	0.983
DNN	0.994	0.981	0.991	0.986
CNN	0.964	0.905	0.915	0.910
RNN	0.945	0.890	0.820	0.854
LSTM	0.983	0.947	0.971	0.959
RF	0.998	0.997	0.998	0.997
LR	0.938	0.851	0.832	0.842
KNN	0.995	0.982	0.992	0.987
DT	0.998	0.996	0.997	0.996
SVM	0.969	0.923	0.920	0.922

hyperparameter n in the high-energy selection algorithm, please refer to Section 8.2. From Table 16, it is evident that as the number of DFNNB layers in FNN increases, the FNN detection performance also improves consistently, with FNN3 achieving an high accuracy rate of 0.995. The multiclassification result of CICIDS2017 is greatly affected by data noise, which significantly impacts its multiclassification results. However, the high-energy selection algorithm in FNN effectively filters out the noise information, thereby enhancing the classification performance. Meanwhile, it can be observed that RNN and LSTM exhibit the poorest performance among the deep learning algorithms, indicating that the unoptimized design of RNN and LSTM is not suitable for intrusion detection, which also reflects the absence of clear sequential relationships in network intrusion data. In terms of precision and recall, FNN demonstrates a significant advantage over other algorithms, except RF. RF performs well among machine learning algorithms, highlighting the strengths of RF as an ensemble algorithm.

The *F*1-score is an evaluation metric that combines precision and recall, providing a better reflection of algorithm performance. Tables 9, 11, and 13 indicate that, when performing binary classification on each dataset, FNN2 and FNN3 consistently achieve the optimal *F*1-score across all three datasets, with FNN1 also outperforming most other algorithms. Among the ML algorithms, RF and KNN exhibit

Table 16: CICIDS2017 multiclass classification test results.

Model	Accuracy	Precision	Recall	$F1$-score
FNN1	0.993	0.993	0.993	0.992
FNN2	0.994	0.994	0.994	0.993
FNN3	0.995	0.995	0.995	0.994
DNN	0.969	0.969	0.969	0.968
CNN	0.985	0.985	0.985	0.984
RNN	0.917	0.909	0.917	0.912
LSTM	0.803	0.645	0.803	0.716
RF	0.998	0.998	0.998	0.998
LR	0.963	0.965	0.963	0.963
KNN	0.995	0.995	0.995	0.995
DT	0.998	0.998	0.998	0.998
SVM	0.971	0.973	0.971	0.970

relatively higher $F1$-scores compared to the others. This highlights the superiority of RF as an ensemble algorithm over traditional single algorithms, while also showcasing the advantages of KNN in handling low-dimensional network intrusion data. Overall, FNN demonstrates superior detection performance in binary classification on the KDD Cup99 dataset, the NSL-KDD dataset, and the UNSW-NB15 dataset. Tables 10, 12, 14, and 16 reveal that in most cases, FNN2 and FNN3 exhibit a significant advantage in $F1$-score compared to other algorithms when performing multiclassification on each dataset. This suggests that other algorithms are more prone to false negatives in multiclassification tests on these datasets, whereas FNN performs better. In summary, FNN exhibits superior detection performance in both binary and multiclassification tasks on the KDD Cup99 dataset, the NSL-KDD dataset, the UNSW-NB15 dataset, and the CICIDS2017 dataset. Compared to other algorithms, FNN has a lower likelihood of false negatives during the detection of network intrusion data.

The confusion matrix is an analytical table used in machine learning to summarize the predictions made by classification models. It presents the relationship between the true attributes of the sample data and the predicted classification types in the form of a matrix. The confusion matrix is a commonly used method for evaluating the performance of classifiers. It allows for the visualization of classification results and enables the calculation of various evaluation metrics. The confusion matrix provides a clear understanding of the classification results for normal and abnormal samples after being evaluated by the intrusion detection model. Figure 7 illustrates the confusion matrix of FNN, RF, KNN, and their binary classification performance on different datasets. From Figure 7 it is evident that, in most cases, the FNN classification outperforms other algorithms in terms of accuracy, with FNN3 exhibiting the lowest rates of false positives and false negatives.

For the KDD Cup99 and NSL_KDD datasets, the KNN algorithm performs the best overall among the machine learning algorithms. Although KNN outperforms FNN1 in terms of detection performance on these datasets, it falls short compared to FNN2 and FNN3. By comparing Figure 7(i) with Figure 7(q), we can observe that on the KDD Cup99 dataset, FNN3 only misses 75 attack samples and incorrectly identifies 11 normal samples as anomalies. FNN3 demonstrates a clear advantage in terms of both false negatives and false positives. Similarly, comparing Figure 7(j) with Figure 7(r) on the NSL_KDD dataset, FNN3 misclassifies 174 abnormal samples as normal samples, whereas KNN misclassifies 234 abnormal samples as normal samples, indicating a higher false negative rate for KNN. When it comes to detecting the UNSW-NB15 dataset, KNN's performance is noticeably inferior to that of FNN and RF. This is due to the high dimensionality and complexity of the UNSW-NB15 data. For the UNSW-NB15 and CICIDS2017 datasets, the RF algorithm achieves the best overall performance among the machine learning algorithms. Analyzing Figures 7(c), 7(g), and 7(k) together with Figure 7(o), we can observe that RF has a higher number of misclassifications than the FNN1, FNN2, and FNN3 on the UNSW-NB15 dataset, suggesting that RF does not perform as well as FNN. Furthermore, by examining Figures 7(d), 7(h) and 7(l), it is evident that FNN2 has the fewest misclassifications among the FNN models on the CICIDS2017 dataset. Specifically, the number of attack samples misclassified as normal samples accounts for 0.016 of the total attack samples, while the number of normal samples misclassified as attack samples accounts for 0.0038 of the total normal samples. Although Figure 7(p) illustrates that RF has fewer misclassifications than FNN on a single dataset (CICIDS2017), an analysis of the combined confusion matrices shown in Figure 7 indicates that FNN has fewer false positives and false negatives across all the datasets, suggesting its superior performance in detecting various datasets when compared to other algorithms.

To present the multiclassification experiment results more intuitively, we extracted 20,000 samples from each dataset based on the proportion of each type. These samples were processed by the respective FNN models, and the resulting values were passed to t-SNE for visualization. Additionally, the same 20,000 samples were directly passed to t-SNE without any processing for visualization, resulting in Figure 8.

From Figures 8(a), 8(c), and 8(e), it is visually evident that the distribution of the KDD Cup99 dataset is the simplest. In fact, applying t-SNE directly to the KDD Cup99 dataset yields satisfactory segmentation results. However, Figure 8(a) clearly shows that there is significant overlap between the Probe, R2L, U2R data and the Normal, Dos data, indicating that t-SNE alone cannot effectively differentiate the Probe, R2L, U2R data from the KDD Cup99 dataset. Figure 8(b) demonstrates that after processing the data with FNN, t-SNE exhibits an improved ability to distinguish the Probe, R2L, and U2R data, particularly in distinguishing the Probe data. Due to the limited amount of training data for R2L and U2R, the impact of the algorithm on enhancing the discrimination of these two types of data is not very pronounced. Nevertheless, from Figure 8(b), it can be observed that the R2L and U2R data are mostly located at the edges of the Normal and Dos data. This indicates that even with a small amount of training data, FNN still possesses the capability to differentiate attack data.

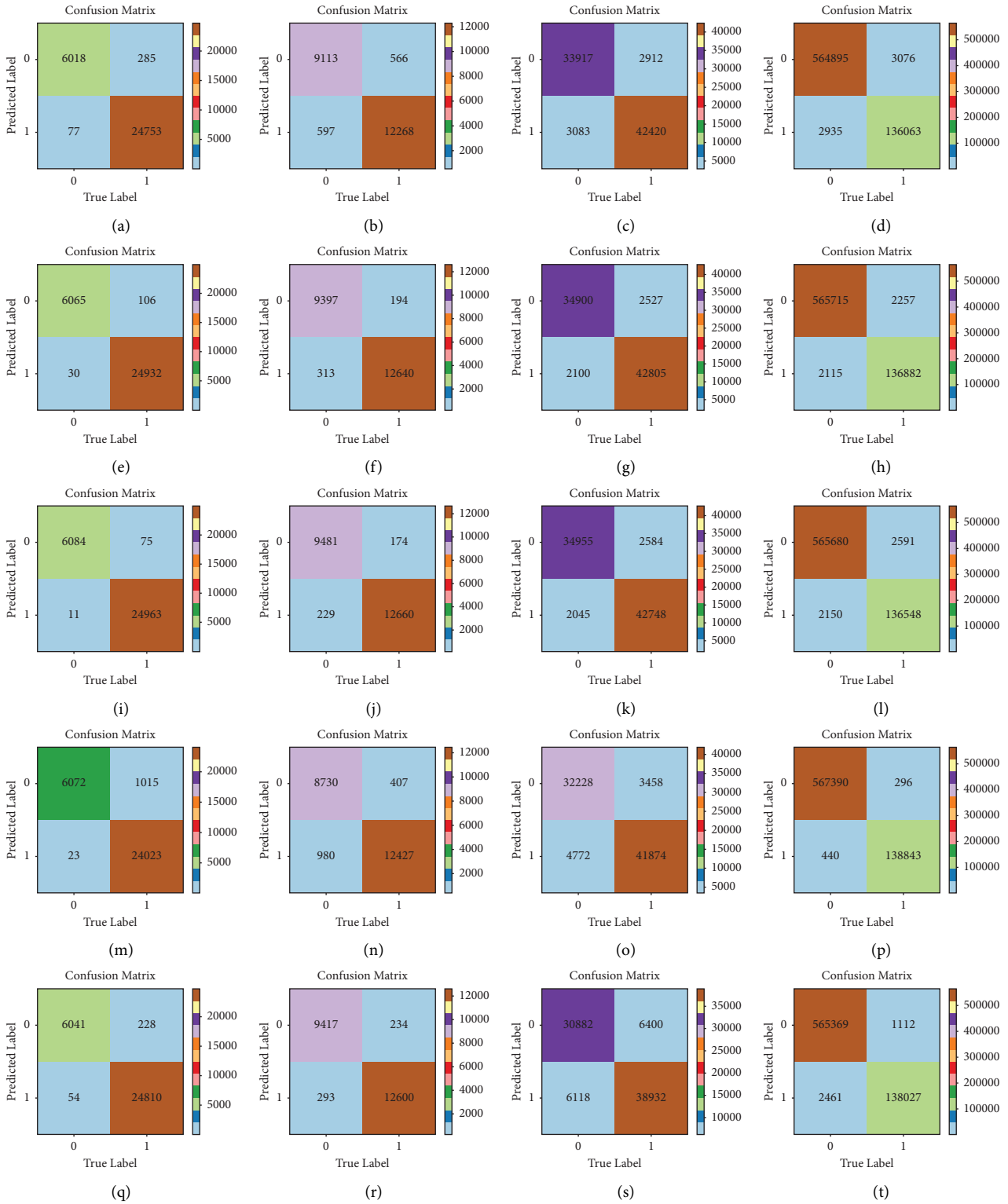

Figure 7: Confusion matrix. (a) KDD Cup99--FNN1. (b) NSL-KDD--FNN1. (c) UNSW-NB15--FNN1. (d) CICIDS2017--FNN1. (e) KDD Cup99--FNN2. (f) NSL-KDD--FNN2. (g) UNSW-NB15--FNN2. (h) CICIDS2017--FNN2. (i) KDD Cup99--FNN3. (j) NSL-KDD--FNN3. (k) UNSW-NB15--FNN3. (l) CICIDS2017--FNN3. (m) KDD Cup99--RF. (n) NSL-KDD--RF. (o) UNSW-NB15--RF. (p) CICIDS2017--RF. (q) KDD Cup99--KNN. (r) NSL-KDD--KNN. (s) UNSW-NB15--KNN. (t) CICIDS2017--KNN.

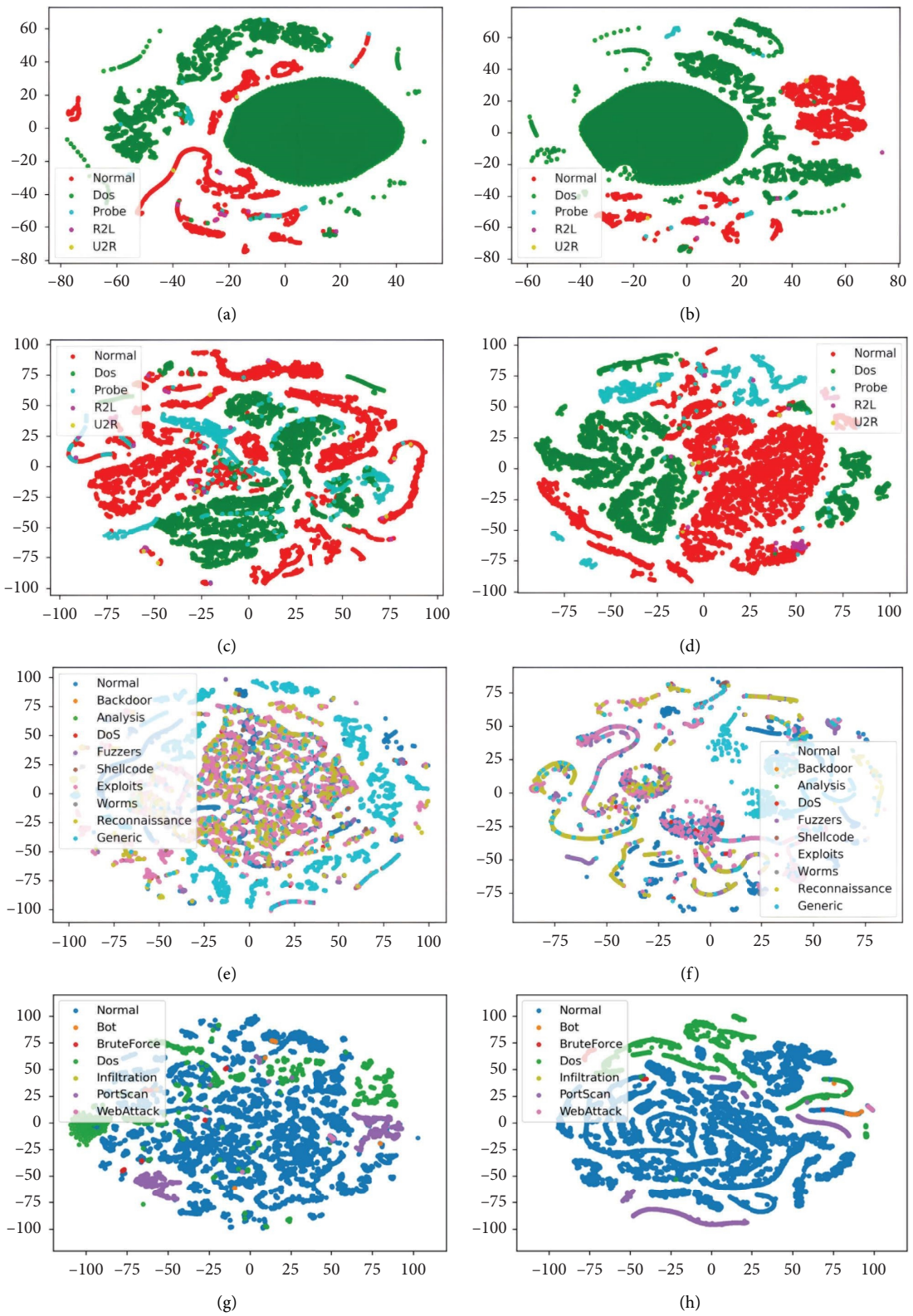

Figure 8: Visualization of multiple classification results. (a) KDD Cup99 original data. (b) KDD Cup99 data processed by FNN3. (c) NSL-KDD original data. (d) NSL-KDD data processed by FNN3. (e) UNSW-NB15 original data. (f) UNSW-NB15 data processed by FNN3. (g) CICIDS2017original data. (h) CICIDS2017data processed by FNN3.

From Figure 8(c), it is evident that the NSL-KDD dataset, as an improvement over the KDD Cup99 dataset, is more complex. Unlike the KDD Cup99 dataset, which exhibits more distinct differentiation results after direct processing with t-SNE, the distribution of various data types in the NSL-KDD dataset is closer after t-SNE processing, with the R2L and U2R class data showing significant overlap with other data types. Figure 8(d) clearly demonstrates that after undergoing FNN processing, all data types exhibit significant improvements compared to the nonprocessed data. At this stage, the Normal, Dos, and Probe data show more distinct differentiation, while the R2L and U2R data, although overlapping with other data types, are mostly located at the edges of the other data clusters.

From Figure 8(e), it is evident that the UNSW-NB15 dataset is more complex compared to the KDD Cup99 and NSL-KDD datasets, with a greater variety of attack types and a more chaotic distribution of various data types. Furthermore, Figure 8(f) clearly demonstrates a significant and effective differentiation between different types of attack data after FNN processing.

Figure 9(g) illustrates that the CICIDS2017 dataset encompasses a wide range of attack types, resulting in a confusing distribution of these attack types. Notably, Bot, BruteForce, and WebAttack exhibit significant overlap with the Normal class, indicating a potential misclassification as Normal. Upon observing Figure 9(h), it becomes apparent that FNN processing significantly enhances the data compared to the unprocessed data. Specifically, a clear differentiation emerges between the Normal, DoS, and PortScan categories. Moreover, Bot, BruteForce, and WebAttack form distinct clusters and reside at the periphery of the clusters comprising other data types.

In conclusion, the visualized images derived from the original dataset and processed by the FNN reveal a distinct clustering pattern of different data types in the low-dimensional space. The proposed FNN model in this study demonstrates accurate identification of various types of anomalous network traffic. While the FNN exhibits superior detection performance in the field of intrusion detection, as a novel neural network, there remains significant room for further enhancement in its capabilities.

Figures 9 and 10 depict the accuracy curves and accuracy box plots for each dataset, respectively. The accuracy curves provide a clear visualization of the convergence of each algorithm during the training process, while the box plots offer insights into the presence of outliers and the distribution characteristics of the data.

From Figure 9(a), it can be observed that in the binary classification of the KDD Cup99 dataset, all deep learning algorithms, except for CNN, converge at a faster pace. However, CNN exhibits a sudden increase in accuracy at the 8th iteration, indicating an unstable optimization process for its parameters. This substantial jump in performance suggests instability within CNN. Furthermore, Figure 10(a) reveals that the accuracy distribution of DNN and FNN is more concentrated, indicating a comparatively stable training process for these algorithms. Combining this with Figure 9(a), it becomes evident that the training process of DNN and FNN is smoother compared to other deep learning algorithms. In the case of multiclassification on the KDD Cup99 dataset, CNN still experiences a jump in accuracy at the 8th iteration, further highlighting the instability of its training process. Notably, LSTM achieves an accuracy exceeding 0.950 during training; however, Table 10 demonstrates that LSTM's final accuracy is only 0.864, indicating overfitting. It is worth mentioning that the accuracy trends of the remaining algorithms show no significant deviation from Figure 9(a).

When binary classification is performed on the NSL-KDD dataset, it can be seen from Figure 9(c) that FNN has the fastest convergence speed among the deep learning algorithms, and its accuracy converges to an optimal value at the early stage of training. Figure 10(c) shows that the accuracy of FNN is stably distributed around an optimal value, and there are only a few outliers. As in the case of binary classification on KDD Cup99, the accuracy of CNN also shows large jumps in the training process when binary classifying the NSL-KDD dataset, and it can be seen from Figure 10(c) that the accuracy of CNN is distributed in a large range, which indicates that the CNN training process is extremely unstable. In Figure 10(c), there are more outliers in the accuracy rates of LSTM and RNN, which also indicates the instability of their training process. From Figure 9(d), it can be seen that when multiclassification is performed on the NSL-KDD dataset, the instability of various algorithms in the training process is significantly improved, but CNN still shows large fluctuations in its accuracy in the early stage of training.

Analyzing Figures 9(e) and 10(e), it can be observed that during binary classification of the UNSW-NB15 dataset, the training processes of all neural networks, except for DNN, exhibit varying degrees of fluctuations. Additionally, Figure 10(e) displays a higher number of outliers, which can be attributed to the continuous improvement of algorithm accuracy with increasing training iterations. Figure 9(e) reveals that the accuracy of DNN stabilizes at a relatively low value early in the training process, indicating that DNN does not effectively capture more useful data features in subsequent training stages. In the case of multiclassification on the UNSW-NB15 dataset, LSTM and RNN initially show improvements in accuracy during early training stages, but quickly stabilize at a lower level. This suggests that LSTM and RNN fail to learn significant data features through training at this point. Moreover, Figure 10(f) clearly illustrates the concentrated distribution of LSTM and RNN accuracy at a lower level.

Analyzing Figure 9(g), it can be observed that during binary classification of the CICIDS2017 dataset, the accuracy of various deep learning algorithms, except for DNN, continues to improve with increasing training iterations. Each neural network's training process exhibits varying degrees of fluctuations, with Figure 10(g) displaying more outliers. FNN tends to stabilize in the later stages of training, indicating that the model has learned useful data features, reached convergence, and achieved optimal performance. On the other hand, RNN reaches its peak accuracy in the early stages of training but experiences significant fluctuations with increasing training iterations, suggesting that the optimization process

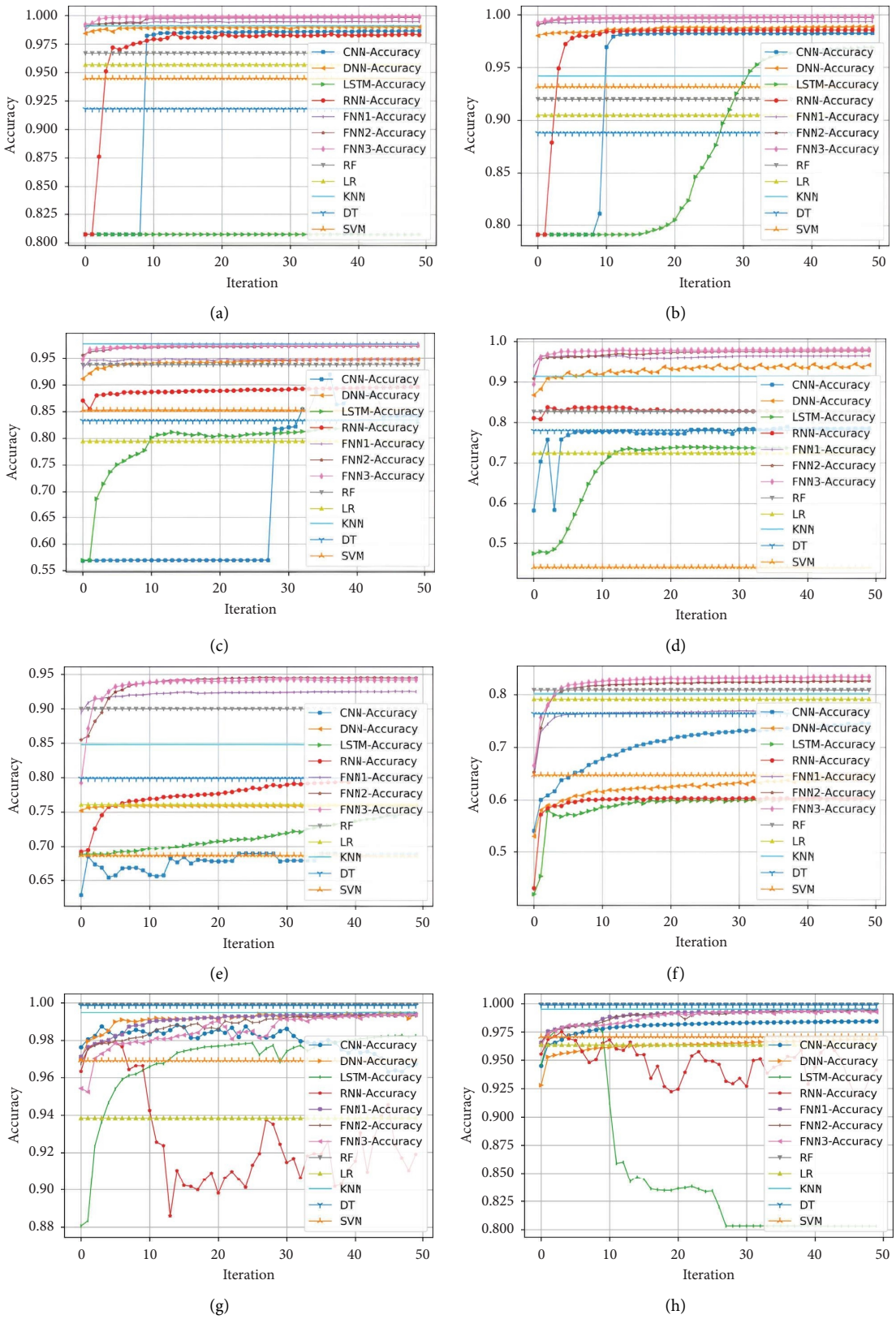

Figure 9: Dataset classification accuracy. (a) KDD Cup99 binary classification. (b) KDD Cup99 multiple classification. (c) NSL-KDD binary classification. (d) NSL-KDD multiple classification. (e) UNSW-NB15 binary classification. (f) UNSW-NB15 multiple classification. (g) CICIDS2017 binary classification. (h) CICIDS2017 binary classification.

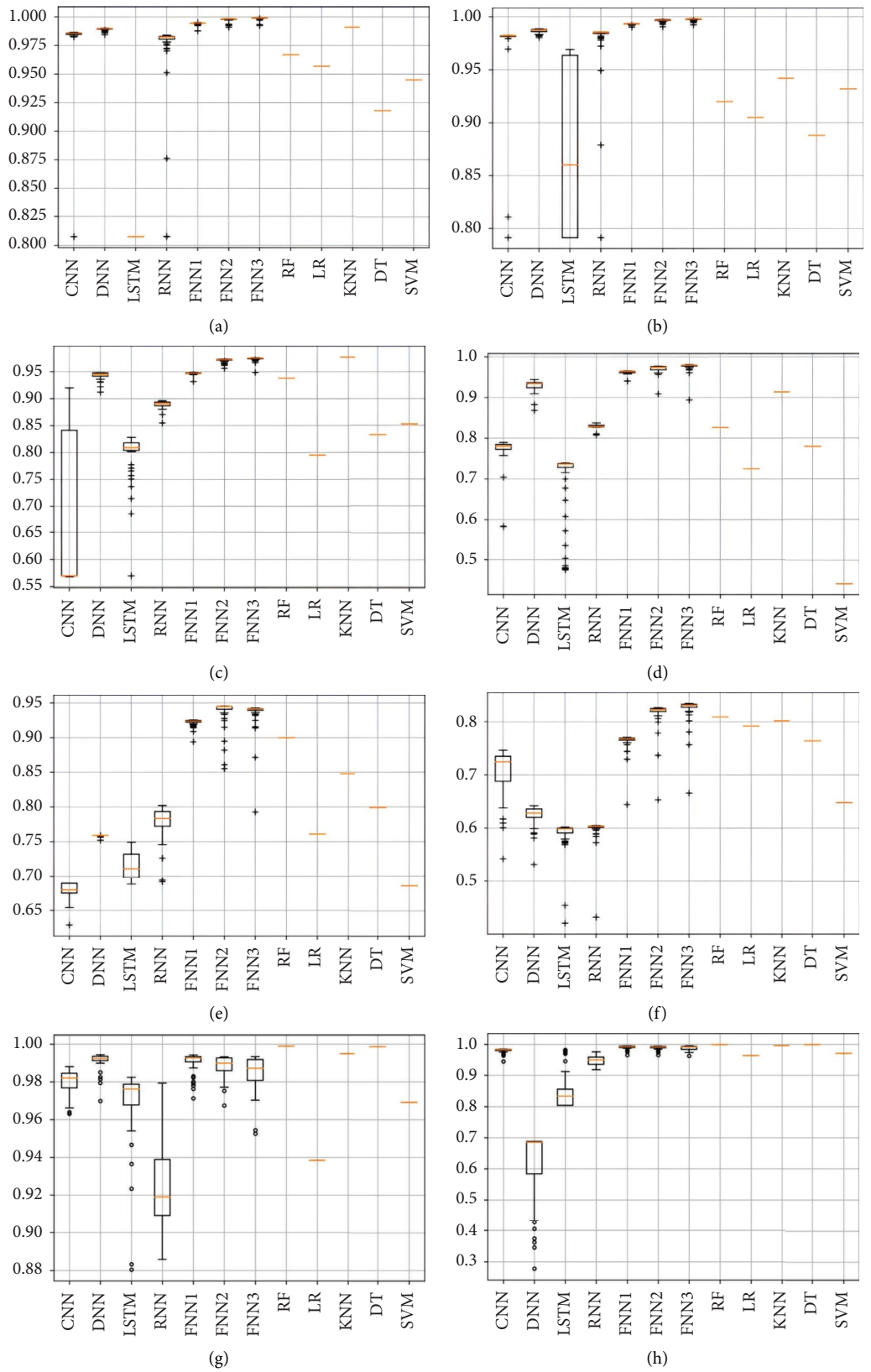

Figure 10: Dataset classification accuracy boxplot. (a) KDD Cup99 binary classification. (b) KDD Cup99 multiple classification. (c) NSL-KDD binary classification. (d) NSL-KDD multiple classification. (e) UNSW-NB15 binary classification. (f) UNSW-NB15 multiple classification. (g) CICIDS2017 binary classification. (h) CICIDS2017 binary classification.

for RNN's parameters is highly unstable. From Figure 10(h), it can be observed that during multiclassification of the CICIDS2017 dataset, CNN and FNN demonstrate a more concentrated distribution of accuracy, indicating a more stable training process. This observation, combined with Figure 9(h), clearly indicates that the training process of CNN and FNN is comparatively smoother than that of other deep learning algorithms. Figure 10(h) illustrates that DNN's accuracy is centrally distributed at a lower level, indicating that DNN fails to learn useful data features through training. On the other hand, LSTM and RNN exhibit significant fluctuations with increasing training iterations, with a higher number of outliers in their accuracy rates, further highlighting the instability of their training process.

Overall, in most cases, FNN demonstrates faster convergence towards a higher accuracy rate than other algorithms. The accuracy distribution of FNN remains more stable during the training process, with only a few outliers in the accuracy values. This indicates that FNN exhibits a more consistent and stable performance.

In conclusion, the FNN exhibits superior detection performance in binary and multivariate classification of network intrusion data than traditional deep learning and machine learning algorithms. It is worth noting that we have validated the FNN model on multiple datasets, further enhancing the credibility of the experimental results. This indicates that the exceptional performance of the FNN model is not solely attributed to chance or specific features of a particular dataset but possesses general applicability and stability. Therefore, the FNN model holds significant potential for application in the field of network intrusion detection.

8.2. Comparative Experiment Two. In Section 4.2.2, we proposed a high-energy selection algorithm. In this algorithm, the number of iterations n needs to be manually selected. Based on the analysis in Section 4.2.2, it is observed that a larger value of n leads to a stronger filtering effect of the HSE algorithm on the noise signal wave, but at the cost of losing more information. Conversely, a smaller value of n results in a weaker filtering effect on the noise signal wave but with less information loss. Excessive noise signals can degrade the algorithm's final performance, and a higher degree of information loss can also lead to a decline in performance. To investigate the impact of n on the final performance of FNN, experiments were conducted with different values of n for FNN3: $n = 1, 2, 3, 4, 5, 6$. The performance metrics of FNN3 on the KDD Cup99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets were recorded under each condition. The detailed experimental results can be found in Tables 17–24.

As shown in Tables 17–24, the experimental results of classifying the KDD Cup99, NSL-KDD, UNSW-NB15, and CICIDS2017 datasets indicate that the detection performance of FNN initially increases and then decreases with the increase in the number of iterations (n) of the HSE algorithm. Specifically, the detection performance of FNN usually reaches its peak when n is set to 3 or 4. When classifying the KDD Cup99 dataset, the experimental results demonstrate that the detection performance of FNN is optimal when n is set to 3. On the other hand, for the NSL-KDD dataset, the peak detection performance of FNN occurs at different values of n. For the binary classification task, the best detection performance of FNN is achieved when n is set to 3, while for the multiclassification task, the optimal detection performance is observed when n is set to 4. When classifying the UNSW-NB15 dataset, the experimental results suggest that the detection performance of FNN is highest when n is set to 4. Regarding the binary classification task of the CICIDS2017 dataset, the best detection performance of FNN is achieved when n is set to 2, while for the multiclassification task, the optimal detection performance is observed when n is set to 3.

From the above observations, it can be noted that increasing the value of n within a certain range effectively filters out noise signals through the HSE algorithm, thereby improving the final detection performance of the FNN. However, when the value of n exceeds a certain range, excessive information filtering by the HSE algorithm may lead to information loss and a subsequent decrease in the FNN's detection rate. Further analysis reveals that for complex datasets, the FNN's detection performance reaches its peak at larger values of n. For instance, in the UNSW-NB15 dataset, the optimal value of n is found to be 4. The identification of this phenomenon provides valuable guidance for future research, enabling the selection of an appropriate value of n in the FNN based on the complexity of the dataset and specific application requirements, ultimately enhancing the classification performance of the FNN.

8.3. Comparison of Performance with Other Studies. Table 25 provides a comparison of different models in terms of accuracy and time consumption. Each row represents a model proposed for intrusion detection in different references. The table includes the accuracy, precision, recall, $F1$-score, training time, inference time, and loss of our proposed models on different datasets, including KDD Cup99, NSL-KDD, UNSW-NB15, and CICIDS2017, for comparison with existing studies. It is important to note that the selection and division of datasets in different references may have an impact on the training time and inference time of the corresponding models. In our experiments, we specifically recorded the training and inference time for FNN1, FNN2, and FNN3 for each dataset. The training time represents the time required to train each epoch of the training set, while the inference time represents the time required to make predictions on the test set.

In order to determine the superiority of the proposed FNN compared to the other studies listed in Table 25, it is necessary to analyze the different metrics provided, including model accuracy, training time, and inference time. By comparing these metrics, we can gain insights into the advantages of the proposed FNN. The FNN achieves high accuracy on multiple datasets, with FNN3 achieving the highest accuracy of up to 99.7%. This demonstrates the model's effectiveness in accurately classifying network traffic and detecting intrusions. In comparison to other studies, the proposed FNN consistently maintains high accuracy, surpassing many other models listed in the table. Regarding

Table 17: Binary classification test results for KDD Cup99 at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.992	0.998	0.992	0.995
2	0.994	0.999	0.994	0.996
3	**0.997**	**0.999**	**0.997**	**0.998**
4	0.989	0.997	0.989	0.993
5	0.987	0.997	0.987	0.992
6	0.987	0.995	0.988	0.992

The bold values indicate the best performance in these set of experiments.

Table 18: Multiclass classification test results for KDD Cup99 at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.953	0.957	0.953	0.941
2	0.974	0.975	0.974	0.972
3	**0.990**	**0.990**	**0.990**	**0.989**
4	0.976	0.977	0.976	0.974
5	0.970	0.971	0.970	0.967
6	0.944	0.902	0.944	0.921

The bold values indicate the best performance in these set of experiments.

Table 19: Binary classification test results for NSL-KDD at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.959	0.956	0.972	0.964
2	0.962	0.966	0.968	0.967
3	**0.982**	**0.982**	**0.986**	**0.984**
4	0.955	0.956	0.967	0.961
5	0.959	0.953	0.977	0.965
6	0.944	0.932	0.973	0.952

The bold values indicate the best performance in these set of experiments.

Table 20: Multiclass classification test results for NSL-KDD at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.962	0.962	0.962	0.962
2	0.968	0.966	0.968	0.967
3	0.977	0.977	0.977	0.977
4	**0.982**	**0.982**	**0.982**	**0.981**
5	0.976	0.976	0.976	0.975
6	0.955	0.955	0.955	0.954

The bold values indicate the best performance in these set of experiments.

Table 21: Binary classification test results for UNSW-NB15 at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.909	0.928	0.906	0.917
2	0.936	0.961	0.921	0.941
3	0.944	0.954	0.943	0.949
4	**0.953**	**0.969**	**0.945**	**0.957**
5	0.952	0.962	0.951	0.956
6	0.935	0.958	0.922	0.940

The bold values indicate the best performance in these set of experiments.

training time, the proposed FNN exhibits relatively fast training times compared to other models. The size and characteristics of the datasets, as well as the division of the training set, can influence the duration of the training process. The training time for the FNN ranges from 1 second to 717 seconds. Although the table does not provide training times for all models, the training time of the proposed FNN appears competitive, as it falls within a reasonable range when compared to other models in the table. Inference time refers to the time required for the model to make predictions on new and unseen attacks. In our experiments, we recorded the prediction time of the test set as the inference time. The proposed FNN demonstrates efficient inference times, ranging from 1 to 152 seconds for different datasets. Similar to the training time, the inference time of the proposed FNN is competitive with other models in the table.

Table 22: Multiclass classification test results for UNSW-NB15 at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.779	0.742	0.779	0.754
2	0.836	0.811	0.836	0.821
3	0.843	0.825	0.843	0.833
4	**0.847**	**0.829**	**0.847**	**0.836**
5	0.779	0.743	0.779	0.754
6	0.780	0.743	0.780	0.754

The bold values indicate the best performance in these set of experiments.

Table 23: Binary classification test results for CICIDS2017 at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.991	0.979	0.978	0.978
2	**0.994**	**0.985**	**0.984**	**0.984**
3	0.993	0.984	0.981	0.983
4	0.977	0.925	0.960	0.942
5	0.992	0.971	0.986	0.977
6	0.993	0.984	0.978	0.981

The bold values indicate the best performance in these set of experiments.

Table 24: Multiclass classification test results for CICIDS2017 at different n values.

n	Accuracy	Precision	Recall	$F1$-score
1	0.992	0.992	0.992	0.992
2	0.992	0.992	0.992	0.992
3	**0.995**	**0.995**	**0.995**	**0.994**
4	0.993	0.993	0.993	0.992
5	0.990	0.990	0.990	0.989
6	0.992	0.992	0.992	0.992

The bold values indicate the best performance in these set of experiments.

As shown in Table 25, some studies rely on a single dataset to validate their proposed models. However, this approach is not ideal as it does not ensure the generalization of the model. This is because a single dataset may possess unique features that the model can learn and overfit to, resulting in higher accuracy scores on that particular dataset but lower performance on other datasets. To test the model's generalizability, it is crucial to evaluate it on multiple datasets with varying characteristics. Models that perform well across different datasets demonstrate their applicability to diverse environments, which is a key requirement for an effective IDS. Our proposed model has been validated on four distinct datasets, including KDD Cup99, NSL-KDD, UNSW-NB15, and

Table 25: Performance comparison with other studies.

Reference	Model	Dataset	Classification	Accuracy (%)	Precision (%)	Recall (%)	*F*1 (%)	Loss	Training times/ epoch	Inference times
Hnamte and Hussain [58]	DCNNBiLSTM	CICIDS2018	Multiple	1	—	—	—	0.0000	3245	1712.03
		Edge_IIoT	Multiple	0.9962	—	—	—	0.0081	8421	4177.53
Alotaibi et al. [59]	DT-PCA-DNN	NSL-KDD	Binary	0.8864	—	0.8456	—	—	14.15 (ms)	57.86 (ms)
			Multiple	0.8329	—	—	—	—	17.36	—
Hnamte et al. [60]	LSTM-AE	CICIDS2017	Binary	0.9999	—	—	—	0.0005	184	53.66
		CSE-CICDIS2018	Binary	0.991	—	—	—	0.0040	462	128.24
Naveed et al. [61]	DNN	NSL-KDD	Binary	0.9973	0.9975	0.9973	0.9972	—	138	2.7
Hnamte and Hussain [62]	DCNN	CICIDS2017	Multiple	0.9996	0.9996	0.9996	0.9996	0.0015	40	29.36
		CICIDS2018	Multiple	1	1	1	1	0.0000	15	9.91
Bashar et al. [63]	Multilayer LSTM	NSL-KDD	Binary	0.950	0.950	0.950	0.950	—	—	—
			Multiple	0.960	0.830	1	0.930	—	—	—
Sangeetha et al. [64]	Multilayered IDS	CICIDS2018 + BoT-IoT + ToN-IoT	Binary	0.981	0.998	0.998	—	—	—	—
Our model	FNN1	KDDCUP99	Binary	0.988	0.997	0.989	0.993	0.0238	6	8
			Multiple	0.957	0.957	0.957	0.948	0.0977	6	8
		NSL-KDD	Binary	0.948	0.954	0.956	0.955	0.0989	1	1
			Multiple	0.963	0.963	0.963	0.962	0.0848	1	1
		UNSW-NB15	Binary	0.927	0.932	0.936	0.934	0.1427	1	2
			Multiple	0.790	0.775	0.790	0.774	0.5224	2	2
		CICIDS2017	Binary	0.991	0.979	0.978	0.978	0.0257	27	18
			Multiple	0.993	0.993	0.993	0.992	0.0150	27	18
	FNN2	KDDCUP99	Binary	0.996	0.999	0.996	0.997	0.0042	30	15
			Multiple	0.981	0.982	0.981	0.980	0.0045	33	15
		NSL-KDD	Binary	0.978	0.976	0.985	0.980	0.0649	6	2
			Multiple	0.975	0.975	0.975	0.975	0.0795	6	2
		UNSW-NB15	Binary	0.944	0.953	0.944	0.949	0.1288	9	4
			Multiple	0.846	0.839	0.846	0.842	0.4235	9	4
		CICIDS2017	Binary	0.994	0.985	0.984	0.984	0.0171	385	84
			Multiple	0.994	0.994	0.994	0.993	0.0148	358	82
	FNN3	KDDCUP99	Binary	0.997	0.999	0.997	0.998	0.0041	54	20
			Multiple	0.990	0.990	0.990	0.989	0.0044	60	21
		NSL-KDD	Binary	0.982	0.982	0.986	0.984	0.0663	11	3
			Multiple	0.977	0.977	0.977	0.977	0.0738	11	3
		UNSW-NB15	Binary	0.944	0.954	0.943	0.949	0.1306	16	6
			Multiple	0.853	0.844	0.853	0.847	0.3960	7	10
		CICIDS2017	Binary	0.993	0.984	0.981	0.983	0.0186	717	152
			Multiple	0.995	0.995	0.995	0.994	0.0145	681	136

CICIDS2017, all of which have yielded impressive results. Hence, our FNN model can be considered more generalizable, indicating its ability to perform well on previously unseen datasets. In comparison to many other models in the table, our model exhibits strong performance across multiple datasets, making it more reliable than models that solely excel on a single dataset. Overall, the proposed FNN stands out due to its high accuracy, competitive training time, and efficient inference time, making it a superior model compared to many other studies mentioned in Table 25.

9. Conclusion and Future Work

This paper hasproposed a novel approach called the Fourier Neural Network (FNN). It utilizes Fast Fourier Transform to convert network intrusion data into the frequency domain space, applies filtering to the converted data, and can subsequently convert the filtered data back to the time domain space. By enabling processing of network data in both the time and frequency domain spaces, FNN enhances the neural network's capability to handle complex data and extract features. To eliminate noisy signals, this study has also introduced a high-energy filtering process (HFP), which further enhances the performance of FNN in intrusion detection by filtering out low-energy noise signal waves based on their amplitude. The experimental results have shown that FNN has significant advantages over existing classical neural network algorithms and traditional machine learning algorithms in dealing with intrusion detection problems. In addition, this study has explored the effect of the number of iterations n in HSP on the performance of FNN, and the results have shown that choosing the right value of n is crucial for achieving the best performance. In summary, the main contributions of this study are twofold. First, the FNN framework was developed to enhance the ability of neural networks to process complex data using Fourier transform. Second, HSP was introduced to effectively eliminate the noisy signals and further improve the performance of FNN in intrusion detection. As a novel neural network, FNN has certain limitations in its application in other domains. For instance, currently, FNN is only capable of performing Fourier transform on one-dimensional data, thereby restricting its ability to handle images and videos. Furthermore, the FNN structure is specifically suited for classification tasks and not applicable to regression problems. In the future, further enhancements will be made to FNN to enable effective processing of image and video data, as well as its applicability to regression problems. Additionally, improvements will be made to the HSE algorithm to develop a more efficient filtering algorithm, thereby enhancing the performance of FNN.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors have no relevant financial or nonfinancial interests to disclose.

Authors' Contributions

Zhendong Wang performed the conceptualization, data curation, formal analysis, resources, supervision, project administration, and review and editing. Jingfei Li performed the conceptualization, methodology, software, validation, investigation, original draft writing, and review and editing. Zhenyu Xu performed the conceptualization, methodology, visualization, and review and editing. Shuxin Yang performed the conceptualization and review and editing. Daojing He performed the conceptualization and review and editing. Sammy Chan performed the conceptualization and review and editing.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (grant nos. 62062037, 61562037, 62376074, and 72261018), the Natural Science Foundation of Jiangxi Province (grant nos. 20212BAB202014 and 20171BAB202026), and the Shenzhen Science and Technology Program under (grant nos. KCXST20221021111404010, JSGG20220831103400002, and JSGGKQTD20221101115655027).

References

[1] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion detection systems: a cross-domain overview," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3639–3681, 2019.

[2] M. Ozkan-Okay, R. Samet, Ö. Aslan, and D. Gupta, "A comprehensive systematic literature review on intrusion detection systems," *IEEE Access*, vol. 9, pp. 157727–157760, 2021.

[3] M. H. Nasir, S. A. Khan, M. M. Khan, and M. Fatima, "Swarm intelligence inspired intrusion detection systems—a systematic literature review," *Computer Networks*, vol. 205, Article ID 108708, 2022.

[4] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks," *Internet of Things*, vol. 16, Article ID 100462, 2021.

[5] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: datasets and comparative study," *Computer Networks*, vol. 188, Article ID 107840, 2021.

[6] M. Choraś and M. Pawlicki, "Intrusion detection approach based on optimised artificial neural network," *Neurocomputing*, vol. 452, pp. 705–715, 2021.

[7] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with naïve Bayes feature embedding," *Computers and Security*, vol. 103, Article ID 102158, 2021.

[8] E. U. H. Qazi, M. Imran, N. Haider, M. Shoaib, and I. Razzak, "An intelligent and efficient network intrusion detection system using deep learning," *Computers and Electrical Engineering*, vol. 99, Article ID 107764, 2022.

[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[10] D. Psaltis, A. Sideris, and A. A. Yamamura, "A multilayered neural network controller," *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 17–21, 1988.

[11] H. Aldarmaki, A. Ullah, S. Ram, and N. Zaki, "Unsupervised automatic speech recognition: a review," *Speech Communication*, vol. 139, pp. 76–91, 2022.

[12] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," in *Proceedings od the ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5904–5908, IEEE, Toronto, Canada, June 2021.

[13] M. H. Guo, T. X. Xu, J. J. Liu et al., "Attention mechanisms in computer vision: a survey," *Computational visual media*, vol. 8, no. 3, pp. 331–368, 2022.

[14] W. Lu and J. Chen, "Computer vision for solid waste sorting: a critical review of academic research," *Waste Management*, vol. 142, pp. 29–43, 2022.

[15] H. Li, N. Zeng, P. Wu, and K. Clawson, "Cov-Net: a computer-aided diagnosis method for recognizing COVID-19 from chest X-ray images via machine vision," *Expert Systems with Applications*, vol. 207, Article ID 118029, 2022.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[18] L. Su, J. Guo, L. Wu, and H. Deng, "BamnetTL: bidirectional attention memory network with transfer learning for question answering matching," *International Journal of Intelligent Systems*, vol. 2023, Article ID 7434058, 11 pages, 2023.

[19] Y. Zhang, J. Wang, and X. Zhang, "Conciseness is better: recurrent attention LSTM model for document-level sentiment analysis," *Neurocomputing*, vol. 462, pp. 101–112, 2021.

[20] N. Bensalah, H. Ayad, A. Adib, and A. Ibn El Farouk, "CRAN: an hybrid CNN-RNN attention-based model for Arabic machine translation," in *Networking, Intelligent Systems and Security: Proceedings of NISS 2021*, pp. 87–102, Springer, Singapore, 2022.

[21] F. Yuan, Z. Zhang, and Z. Fang, "An effective CNN and Transformer complementary network for medical image segmentation," *Pattern Recognition*, vol. 136, Article ID 109228, 2023.

[22] G. Lokku, G. H. Reddy, and M. G. Prasad, "OPFaceNet: OPtimized Face Recognition Network for noise and occlusion affected face images using Hyperparameters tuned Convolutional Neural Network," *Applied Soft Computing*, vol. 117, Article ID 108365, 2022.

[23] L. Sun, W. Shao, Q. Zhu, M. Wang, G. Li, and D. Zhang, "Multi-scale multi-hierarchy attention convolutional neural network for fetal brain extraction," *Pattern Recognition*, vol. 133, Article ID 109029, 2023.

[24] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the internet of things," *IEEE Access*, vol. 7, pp. 42450–42471, 2019.

[25] M. Safaldin, M. Otair, and L. Abualigah, "Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 1559–1576, 2021.

[26] A. Ponmalar and V. Dhanakoti, "An intrusion detection approach using ensemble support vector machine based chaos game optimization algorithm in big data platform," *Applied Soft Computing*, vol. 116, Article ID 108295, 2022.

[27] E. K. Boahen, B. E. Bouya-Moko, and C. Wang, "Network anomaly detection in a controlled environment based on an enhanced PSOGSARFC," *Computers and Security*, vol. 104, Article ID 102225, 2021.

[28] H. Ding, L. Chen, L. Dong, Z. Fu, and X. Cui, "Imbalanced data classification: a KNN and generative adversarial networks-based hybrid approach for intrusion detection," *Future Generation Computer Systems*, vol. 131, pp. 240–254, 2022.

[29] M. Yousefnezhad, J. Hamidzadeh, and M. Aliannejadi, "Ensemble classification for intrusion detection via feature extraction based on deep Learning," *Soft Computing*, vol. 25, no. 20, pp. 12667–12683, 2021.

[30] M. Ayar, A. Isazadeh, F. S. Gharehchopogh, and M. Seyedi, "Chaotic-based divide-and-conquer feature selection method and its application in cardiac arrhythmia classification," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 5856–5882, 2022.

[31] F. S. Gharehchopogh, A. Ucan, T. Ibrikci, B. Arasteh, and G. Isik, "Slime mould algorithm: a comprehensive survey of its variants and applications," *Archives of Computational Methods in Engineering*, vol. 30, no. 4, pp. 2683–2723, 2023.

[32] M. Alazab, R. A. Khurma, A. Awajan, and D. Camacho, "A new intrusion detection system based on Moth–Flame Optimizer algorithm," *Expert Systems with Applications*, vol. 210, Article ID 118439, 2022.

[33] Z. Halim, M. N. Yousaf, M. Waqas et al., "An effective genetic algorithm-based feature selection method for intrusion detection systems," *Computers and Security*, vol. 110, Article ID 102448, 2021.

[34] A. Thakkar and R. Lohiya, "Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system," *Information Fusion*, vol. 90, pp. 353–363, 2023.

[35] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Computing*, vol. 24, no. 22, pp. 17265–17278, 2020.

[36] J. J. Fu and X. L. Zhang, "Gradient importance enhancement based feature fusion intrusion detection technique," *Computer Networks*, vol. 214, Article ID 109180, 2022.

[37] V. Ravi, R. Chaganti, and M. Alazab, "Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system," *Computers and Electrical Engineering*, vol. 102, Article ID 108156, 2022.

[38] Z. Wang, Y. Zeng, Y. Liu, and D. Li, "Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection," *IEEE Access*, vol. 9, pp. 16062–16091, 2021.

[39] Y. Shen, C. Zhang, F. Soleimanian Gharehchopogh, and S. Mirjalili, "An improved whale optimization algorithm based on multi-population evolution for global optimization and engineering design problems," *Expert Systems with Applications*, vol. 215, Article ID 119269, 2023.

[40] F. S. Gharehchopogh, "Quantum-inspired metaheuristic algorithms: comprehensive survey and classification," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5479–5543, 2023.

[41] P. Rajesh Kanna and P. Santhi, "Unified deep learning approach for efficient intrusion detection system using integrated spatial–temporal features," *Knowledge-Based Systems*, vol. 226, Article ID 107132, 2021.

[42] P. R. Kanna and P. Santhi, "Hybrid intrusion detection using mapreduce based black widow optimized convolutional long short-term memory neural networks," *Expert Systems with Applications*, vol. 194, Article ID 116545, 2022.

[43] S. Balasubramaniam, C. Vijesh Joe, T. A. Sivakumar et al., "Optimization enabled deep learning-based DDoS attack

detection in cloud computing," *International Journal of Intelligent Systems*, vol. 2023, Article ID 2039217, 16 pages, 2023.

[44] G. Yang, L. Wang, R. Yu, J. He, B. Zeng, and T. Wu, "A modified gray wolf optimizer-based negative selection algorithm for network anomaly detection," *International Journal of Intelligent Systems*, vol. 2023, Article ID 8980876, 23 pages, 2023.

[45] D. Sundararajan, *The Discrete Fourier Transform: Theory, Algorithms and Applications*, World Scientific, Singapore, 2001.

[46] H. J. Nussbaumer and H. J. Nussbaumer, *The Fast Fourier Transform*, Springer, Berlin, Germany, 1982.

[47] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proceedings of the IEEE*, vol. 56, no. 6, pp. 1107-1108, 1968.

[48] R. T. M. A. C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*, Springer, Switzerland, 1989.

[49] S. Impedovo, T. Simone, and G. Dimauro, "Integration of the cooley, rader and Winograd-Fourier algorithms for a faster computation of the DFT," *Recent Issues in Pattern Analysis and Recognition*, pp. 52–57, 1989.

[50] S. Engelberg, "Elementary number theory and rader's FFT," *SIAM Review*, vol. 59, no. 3, pp. 671–678, 2017.

[51] Kdd Cup 99 dataset, "Kdd cup99 dataset[online]," 2022, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[52] Nsl-Kdd dataset, "Nsl-kdd dataset[online]," 2021, http://users.cis.fiu.edu/%7Elpeng/Datasets_detail.html.

[53] Unsw-Nb 15 dataset, "Unsw-nb15 dataset[online]," 2019, https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.

[54] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proceedings of the 2015 military communications and information systems conference (MilCIS)*, pp. 1–6, IEEE, Canberra, Australia, November 2015.

[55] Cic-Ids 2017 dataset, "Cic-ids2017 dataset[online]," 2017, https://www.unb.ca/cic/datasets/ids-2017.html.

[56] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*, vol. 1, pp. 108–116, Funchal- Madeira, Portugal, January 2018.

[57] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2020, https://arxiv.org/abs/2010.16061.

[58] V. Hnamte and J. Hussain, "DCNNBiLSTM: an efficient hybrid deep learning-based intrusion detection system," *Telematics and Informatics Reports*, vol. 10, Article ID 100053, 2023.

[59] S. D. Alotaibi, K. Yadav, A. N. Aledaily et al., "Deep neural network-based intrusion detection system through PCA," *Mathematical Problems in Engineering*, vol. 2022, Article ID 6488571, 9 pages, 2022.

[60] V. Hnamte, H. Nhung-Nguyen, J. Hussain, and Y. Hwa-Kim, "A novel two-stage deep learning model for network intrusion detection: lstm-ae," *IEEE Access*, vol. 11, pp. 37131–37148, 2023.

[61] M. Naveed, F. Arif, S. M. Usman et al., "A deep learning-based framework for feature extraction and classification of intrusion detection in networks," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 2215852, 11 pages, 2022.

[62] V. Hnamte and J. Hussain, "Dependable intrusion detection system using deep convolutional neural network: a Novel framework and performance evaluation approach," *Telematics and Informatics Reports*, vol. 11, Article ID 100077, 2023.

[63] G. M. H. Bashar, M. A. Kashem, and L. C. Paul, "Intrusion detection for cyber-physical security system using long short-term memory model," *Scientific Programming*, vol. 2022, Article ID 6172362, 11 pages, 2022.

[64] S. K. Sangeetha, P. Mani, V. Maheshwari, P. Jayagopal, M. Sandeep Kumar, and S. M. Allayear, "Design and analysis of multilayered neural network-based intrusion detection system in the internet of things network," *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 9423395, 7 pages, 2022.