Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 3859155, 11 pages
https://doi.org/10.1155/2022/3859155

Research Article

A Metaheuristic Autoencoder Deep Learning Model for Intrusion Detector System

Jay Kumar Pandey,[1] **Sumit Kumar**,[2] **Madonna Lamin**,[3] **Suneet Gupta**,[4] **Rajesh Kumar Dubey**,[5] **and F. Sammy**[6]

[1]*Department of Electronics and Communication Engineering, Shri Ramswaroop Memorial University, Dewa Road, Barabanki, Uttar Pradesh, India*
[2]*Indian Institute of Management, Kozhikode, India*
[3]*Department of Computer Science and Engineering, ITM SLS Baroda University, Vadodara, Gujarat 391510, India*
[4]*Department of CSE, School of Engineering and Technology, Mody University, Lakshmangarh, Rajasthan, India*
[5]*Department of Electrical Engineering, School of Engineering and Technology, Central University of Haryana, Mahendragarh, India*
[6]*Department of Information Technology, Dambi Dollo University, Dembi Dolo, Welega, Ethiopia*

Correspondence should be addressed to F. Sammy; sammy@dadu.edu.et

Received 23 January 2022; Revised 9 February 2022; Accepted 11 February 2022; Published 4 March 2022

Academic Editor: Vijay Kumar

A multichannel autoencoder deep learning approach is developed to address the present intrusion detection systems' detection accuracy and false alarm rate. First, two separate autoencoders are trained with average traffic and assault traffic. The original samples and the two additional feature vectors comprise a multichannel feature vector. Next, a one-dimensional convolution neural network (CNN) learns probable relationships across channels to better discriminate between ordinary and attack traffic. Unaided multichannel characteristic learning and supervised cross-channel characteristic dependency are used to develop an effective intrusion detection model. The scope of this research is that the method described in this study may significantly minimize false positives while also improving the detection accuracy of unknown attacks, which is the focus of this paper. This research was done in order to improve intrusion detection prediction performance. The autoencoder can successfully reduce the number of features while also allowing for easy integration with different neural networks; it can reduce the time it takes to train a model while also improving its detection accuracy. An evolutionary algorithm is utilized to discover the ideal topology set of the CNN model to maximize the hyperparameters and improve the network's capacity to recognize interchannel dependencies. This paper is based on the multichannel autoencoder's effectiveness; the fourth experiment is a comparative analysis, which proves the benefits of the approach in this article by correlating it to the findings of various different intrusion detection methods. This technique outperforms previous intrusion detection algorithms in several datasets and has superior forecast accuracy.

1. Introduction

With the rise of the Internet, artificial intelligence, and big data, network security is facing new and complicated threats. At the moment, people need a more powerful and robust network intrusion detection system (NIDS) because network intrusions are becoming more diverse and complex [1]. Detecting network intrusions is the goal of network intrusion detection systems. They look at the network traffic to see if there is any malicious activity that could be bad. To do this, it needs to build a model that can tell the difference between an attack and normal network traffic. Then, NIDS can turn intrusion detection into pattern recognition and classification, use the same kinds of algorithms to get data, clean it, model it, and classify different network behaviors [2].

After years of research, the current NIDS methods can be broken down into two groups: practises based on feature detection and techniques based on anomaly detection based

on different detection methods [3]. Feature detection is a technique for computing representations of picture details and determining even if there is not an image feature of a certain kind at each image location. A limited image recognition procedure is feature detection. Anomaly detection (also known as outlier detection) [1] is the process of identifying unusual things, activities, or experiences that differ considerably from the entire data. Such instances may raise concerns that they were created by a separate method, or they may emerge to be contradictory to the facts. In the method based on feature detection [4], the first step is to look at different ways of getting into your computer. Then, you look for attack features, add them to the signature database, and use them to find new attacks. When the samples found match the information in the signature database, it can be seen as an attack. Method: this one has a low chance of making a false positive. It also has something called "hysteresis." It can only find the attack patterns that are already in the database. The detection rate of new attacks is very low. The method based on anomaly detection [5] changes the way standard information flow samples are taken and makes a benchmark model by setting up a probability and statistical model. When the pattern of the sample that was found does not match the model, it can be called an "anomaly attack." This type of method has a low false-negative rate and can be very good at spotting new episodes, but it also has a problem with not being able to find everything. Many researchers have come up with ways to improve the accuracy and stability of detection methods because of the current problems with intrusion detection. Most of these methods will improve observation accuracy by diminishing false-positive rates and detecting unspecified attacks [6]. With the rise and development of deep learning, features can be set up by hand and replaced by multilayer networks that can be learned. Intrusion detection tasks can be done with more accuracy and less false positives than with traditional machine learning. So, a lot of deep learning methods are used in NIDS. To optimize the hyperparameters and increase the network's capacity to perceive interchannel dependencies, an evolutionary approach is used to find the optimum architecture set of the CNN model. The CNN model is a type of neural network that allows us to derive higher depictions for image content. Unlike traditional image processing, which requires the user to specify the image characteristics, CNN receives the picture's basic pixel data, develops the model, and then derives the characteristics for improved categorization. The convolutional neural networks (CNN) method was used by Li et al. [6] to come up with a way to find network intrusions. The more training samples you have, the better this method gets. Mehbodniya et al. [7] came up with an automatic intrusion detection system based on a multilayer recurrent neural network to protect fog computing from network attacks, which can be hard to stop. A person named 9 came up with a way to find out if someone was trying to get into your computer. They used a deep convolution neural network to do this (dCNN). The most important thing about this method is that it turns one-dimensional intrusion data into two-dimensional "image data" for training. Detection of intrusions is more accurate because of the network. The accuracy and false alarm rate of NIDS research have always been important, but real-time performance and detection efficiency are also important. Because the autoencoder can effectively reduce the number of features and can be easily combined with other neural networks, it can shorten the time it takes to train a model and improve its detection accuracy. An autoencoder is a type of artificial neural network that can be used for unsupervised learning. It has an encoder function that maps the input to a hidden layer and a sort-out purpose that produces the reconstructed learned information by minimizing the loss purpose. A sort of artificial neural network called an autoencoder is being used to devise effective encoding for unidentified input (unsupervised learning). By seeking to recreate the input from the encoding, the encoding is checked and enhanced. By training the network to disregard inconsequential input ("noise"), the autoencoder establishes a pattern (encoding) for a collection of data, generally for feature extraction. Stacked autoencoders could be used to learn the features of standard samples, and then a support vector machine (SVM) classifier could be used to make the method more accurate [8]. Mahajan et al. [9] used a deep neural network to learn about the features of input data and then used an autoencoder to look for anomalies in the data. This helped improve the chances of correctly classifying a new type of attack. Then, the LSTM network was used to deal with the sequential nature of computer network data so that it can better deal with network attacks that come up out of nowhere. The above methods can be used to get specific results, but they always lose information when they use the autoencoder to compare the original data. People who did this kind of research only used standard samples to train one autoencoder. They did not use attack samples at all. Novel attack samples are less likely to be found. In order to resolve the problems of low detection accuracy, high false alarm rates, and low detection efficiency of new intrusion behaviors, a deep learning procedure based on multichannel autoencoder is suggested. This method combines feature detection and anomaly detection methods to find new intrusion behaviors. When two autoencoders are trained in the unsupervised stage, they use a lot of different types of traffic to make them better. The two autoencoders make two new feature vectors based on the input samples. Then, the two reconstructed feature vectors and the original illustrations are combined to make a multichannel representation of the feature. Finally, a one-dimensional CNN network processes the multichannel characteristic vector representation to learn about the possible connections between channels, as shown in the figure. Our method is very good at combining unsupervised multichannel feature learning and supervised cross-channel feature dependencies to make our models more accurate.

The unique thing about our method is that we use autoencoders to make multichannel representations of traffic data. We use regular traffic and attack traffic to train autoencoder models, and then we use these autoencoders to make basic feature vector representations of network along with features formed by these autoencoders' vector. Then, a convolutional deep learning architecture is used to show the unseen dependencies in the cross-channel feature

representation. This feature representation helps the intrusion detection model separate the attack flow from the normal flow, so the method in this paper can effectively reduce false positives and improve the detection accuracy of unknown attacks, which is what this paper is about.

2. Multichannel Autoencoder Deep Learning

When an autoencoder [10] is used, it learns how to represent data by using the data as a learning target. This is how it does this. The autoencoder network has two parts: there are two ways to get at the hidden representation: first, the encoder f uses the mapping function that takes the input vector and makes it look like the hidden representation h. Then, the decoder uses the mapping function that takes the input vector and makes it look like the hidden representation. When you put the hidden representation into the input space, you obtain a reconstruction vector that has the same parameter as that hidden representation, which is, generally, the functions h and f referring to two dissimilar neural networks, one for each. $M(y,h(f(y))) = M(y,y)$ is the loss purpose of the autoencoder, which is made up of two networks. The loss function of the autoencoder is $M(y,y)$. In math, the penalty term can be written down as $\text{Mse}(y,y) = |y - 2y$.

Neural networks called "convolutional" are a type of feedforward neural network with deep structures and convolutional computations. They are used to process data with grid-like topologies, but they can also process data with other types of topologies. CNNs can use correlation filters to capture spatial or temporal dependencies in data, which leads to good internal representations. CNNs are made up of two parts: the feature extraction part, which has convolutional layers and spatial pooling layers, and the prediction or classification part, which has layers that can be trained and classifiers. Most of the time, when the positional relationship between local features is known, convolutional layers find features with local correlations. Weights (called "convolution kernels") are used to perform an inner product operation on the data in a certain area. The output value is one of the features that was found. The kernel elements that make up the multiplication operation are like the weight matrix in a traditional neural network. All possible kernels that can be trained to have different offsets are used to make feature maps at the convolutional layers. G is the feature map that the convolution layer makes from a linear convolution filter and a nonlinear activation function g.

$$g_{(i,j,k)} = \sigma\left(w_k^T y_{ij}\right). \tag{1}$$

In the formula, (i,j) constitute the location in the feature map; y_{ij} represents the input data whose center is (i,j); and k constitutes the channel index in the feature map.

3. Architecture of Multichannel Autoencoder Deep Learning

Unsupervised multichannel feature building using two autoencoders and supervised cross-channel feature correlations are used in this study to tackle the problem of network intrusion detection, which is a multichannel deep learning problem. The deep learning takes a lot of data to do well than other approaches, and it is quite expensive to train because of the complicated data models. Deep learning also necessitates the use of pricey GPUs and hundreds of workstations. The users' costs will rise as a result of this. Figure 1 depicts the suggested model's general design.

3.1. Feature Learning of Autoencoders. As a result of this work, single-channel samples may be converted to multichannel samples using a particular class of autoencoders. Assume that $P = \{(y_i, x_i)\}_{i=1}^{N}$. It is a collection of N training samples, each of which represents an input sample specified on D characteristics, and x_i represents either a normal or an attack sample. $Y = [y_1, y_2, \ldots, y_N] \in S^{N*D}$ represents the data matrix of N D-dimensional random irregular $y_i \in S^D$. In order to differentiate in middle of usual samples and attack samples within Y, we define the following notation: $Y^n = Y_{|yi=n}$ and $Y_a = Y_{|yi=a}$. A pair of independent auto encoders, $hn \bullet fn$ (designated by A_n) and $g_a \cdot f_a$ (designated by A_a), may be learned from samples in Y_n and Y_a. Decoder network activation is viewed as a new learned feature because it corresponds to an output vector that has been reconstructed in the same input space. A new feature vector, $y = h(f(y)) \in S^D$, may be generated by each autoencoder using the samples y. These characteristics can then be used to concatenate several channels of samples. Consequently, a multichannel sample may be used to substitute each sample $y_i \in S^D$:

$$y_i' = \left[y_i, y_i^{\hat{n}}, y_i^{\hat{a}}\right]^T \in S^{D*3}, \tag{2}$$

where $y_i^{\hat{n}} = h_n(f_n(y_i))$ and $y_i^{\hat{a}} = h_a(f_a(y_i))$ represent the reconstructed representation of the single-channel sample y_i, respectively. Therefore, a single-channel data matrix Y can be extended to a multichannel data matrix:

$$Y' = \left[y_1', y_2', \ldots, y_n'\right]^T \in S^{N*D*3}. \tag{3}$$

In this way, the features reconstructed by the autoencoder of standard samples and attack samples are synthesized to enrich the information of pieces y_i. When the works belong to two different distributions, a sample y_i marked as usual should be more similar to the representation of $y_i^{\hat{n}}$ than $y_i^{\hat{a}}$; that is, $||y_i - y_i^{\hat{a}}||^2 < ||y_i - y_a^{\hat{n}}||^2$, and vice versa. The goal in the autoencoder step is to exploit the influence of one channel on every other track in the supervised stage to greatly differentiate the difference between the two classes ordinary and attack.

Three-channel representations can be learned in two ways: there are several ways to resolve this problem, one of them is to employ a filter with a (11) convolution kernel for cross-channel parameter convolution. An alternative technique is to concatenate the learned components into the space S^{3D} instead of $S^{D\times 3}$ and then feed them a fully linked layer in $S^{D\times 3}$ (three-channel expression). In contrast, because the training output is unaffected by the order of the input features, the input topology is totally disregarded in a fully connected layer. Channel-based ranking and new cross-channel attributes may be lost as a result of utilizing this technique.

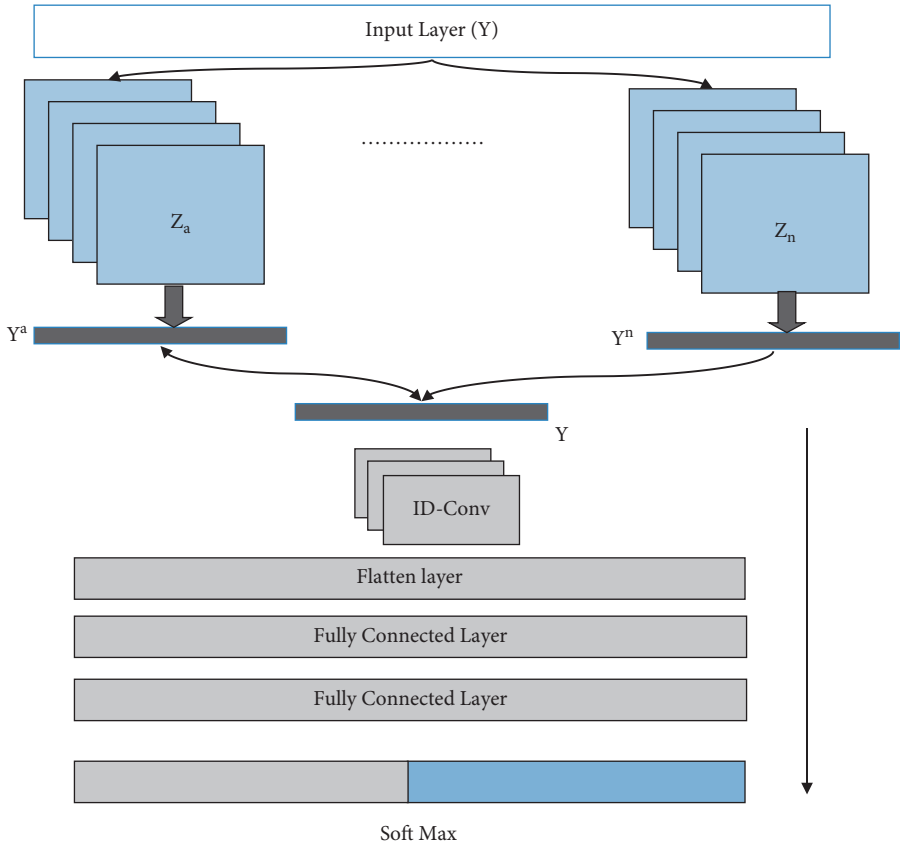

Figure 1: Architecture of the proposed model.

According to this paper's autoencoder, there are three layers where each includes 40 neurons. Place a dropout layer before the decoding layer in sequence to do data regularization and intercept overfitting.

$$\begin{cases} s^l \sim \text{Bernnoulli}\,(p), \\ x^x = s^l * x^l, \\ a_i^{l+1} = v_i^{l+1} * x^y + c_i, \\ x_i^{l+1} = f\left(a_i^{i+1}\right). \end{cases} \tag{4}$$

In the formula, l represents the layer number of the neural network, s^l represents the random vector satisfying the Bernoulli distribution, p represents the retention probability, x^l and $y^{l'}$ represent the output vector of the neuron in the first layer of the neural network and the random block after random blocking, respectively. v_i^{l+1} and a_i^{i+1} represent the weight coefficient and bias of the ith neuron in the lth layer, respectively, f represents the activation function, and a_i^{l+1} and x_i^{l+1} represent the input of the activation function of the ith neuron in the lth layer, respectively, value and output value. The use of dropout in the training process of the autoencoder network is to randomly reset the weights of some neurons to 0 according to the probability p in each training process, that is, to drop some neurons, which can reduce the number of parameters and make the local data clusters different, more noticeable, thus avoiding overfitting. The dropout layer generally chooses the value of the retention probability p to be 0.5 because the dropout layer has the best effect at this time, and the generated network structure is the most abundant. Standard activation functions in neural networks include Sigmoid, tanh, and ReLU. We choose rectified linear unit (ReLU) as the activation purpose for each unseen layer, while for the last layer, we use the linear activation purpose Sigmoid. Multichannel autoencoders use mean square error (MSE) as the loss function.

3.2. Internal Structures of the Network. Since CNN was widely used in image processing at first, most of the network's input is in the form of a two-dimensional matrix, so the internal structures of the network, such as feature maps and convolution kernels, are set to two-dimensional. With the introduction of CNN in language recognition, one-dimensional CNN came into being to adapt to the one-dimensional characteristics of language signals. One-dimensional CNNs process one-dimensional input vectors and the filters in the convolution only slide along one dimension. This paper adopts one-dimensional CNN for feature processing to better utilize the feature combination information across channels. For 1E CNN models, to reduce the dimensionality of the filter size and reduce the amount of computation in the training process, the kernel size of the filter can be limited to 1; that is, a (1×1) convolution kernel is used to reduce the spatial extent. Meanwhile, (1×1)

convolution also combines existing knowledge in the channel dimension to obtain more abstract channel knowledge [10]. Convolution layers with a filter size of 1 will be used in this research to boost cross-channel knowledge, i.e., input the three-channel representation $y_i' = [y_i, y_i^{\hat{n}}, y_i^{\hat{a}}]^T \in S^{D*3}$. The nonlinear cross-channel dependencies are then increased by using more than three filter banks. The flexible field of the one-dimensional complexity is indicated as $y_{i,j,z}^{'}$, and every one filter k is utilized to construct the feature map for the supplied sample y_i. The following is the formula for the medium signals $f_{i,j,k}$:

$$f_{i,j,k} = \sigma\left(v_k^T y_{i,j,z}^{'} + c_k\right). \tag{5}$$

To simplify, the weight and bias coefficients are denoted as $v_k$and the nonlinear activation function as $c_k \in S^3$, respectively, in the formula [11]. The channels of every one feature in y $(i,j,z)'$ are mapped to the feature map using the same shared weight coefficients. It is possible to turn $S^{D\times 3}$ samples into feature maps with $S^{D\times K}$ filters in the convolutional layers. Two fully attached layers are then placed on peak of the 1D convolutional layer's output. For the final classifier module to fulfill its classification function, it connects neurons formed in the convolutional and pooling layers to all other neurons in the upper layers, resulting in the classification probability being output. For this study, the final classification module is the Softmax classifier, and the likelihood of categorizing y in the regression model as class j is

$$\begin{aligned} t &= \left(x^i = j | y^i; \theta\right) \\ &= \frac{e^{\theta_i y^i}}{\sum_k e^{\theta_i y^i}}. \end{aligned} \tag{6}$$

In the formula, θ_j represents the jth weight vector, and x^i is the data sample.

3.3. Chromosome Structure in CNN Optimization Model. Deep learning approaches based on CNNs have made major strides in a variety of disciplines of study. This is despite the fact that deep learning is capable of learning features and optimizing weight parameters using data. Human interaction is required to alter the network architecture of input and output variables and learnt parameters [12]. It is, however, difficult to analyze all conceivable network structures to derive ideal weight values since the number of structures rises exponentially with the network's depth. According to this study, an approach based on a genetic algorithm may be used to automatically determine a CNN model's best topology (GA).

For the most part, the GA algorithm is used in this phase, which optimizes the topology while focusing on the CNN's feature extraction process [13]. The convolution and pooling layers of the CNN recognize patterns and extract essential characteristics for the given input as the kernel passes over the raw data. The performance of large networks can be improved by choosing the right weight parameters for these layers. Hyperparameters such as these might change over time; thus, it is essential to employ a methodical technique to determine them. When the convolution kernel is large, it is difficult for the network to take into account the specific properties of the input data [14]. Too much information might be confusing for a little seed. Kernels in each convolutional layer have an effect on feature learning since each seed generates a distinct feature map and acts as a detector of features with respect to different points of view. A new perspective emerges when the number of processing cores grows. In addition, when the number of cores grows, it is vital to identify an appropriate value that allows the input data to learn features well and reduces computational cost. Empirical performance rather than theory is used to make the decision for network topology in most CNN research [13]. The topology of the one-dimensional CNN model is optimized using a genetic algorithm in this study in order to increase prediction performance for intrusion detection. In order to develop a CNN structure that is optimum, all architectural factors must be tweaked at the same time. The GA method optimizes the number of kernels, the size of the kernels, and the size of the pooling window for convolution and pooling layer operations. Each layer component's size should be encoded as a binary string, as seen in the second image.

These are the steps that make up the CNN optimization process for GA-based CNNs:

(a) Make a fresh start with the population: to begin with, the genetic algorithm generates a list of potential outcomes. As a result, the most important role of a genetic algorithm is to articulate possible solutions to the issue that chromosomes are tackling and to set performance standards. Each chromosome should accurately reflect the features of the target problem because the expression of viable solutions influences all genetic changes. As a result, the number of kernels per convolutional layer, the size of the kernel, and the potential value associated with the window size are all encoded in each chromosome.

(b) To determine how well each chromosome performs, a fitness function must be established. Intrusion detection categorization accuracy is used as the fitness function in this work. It is predicted that chromosomes with better classification accuracy would replicate more often than those with lower classification accuracy.

(c) In order to create a new generation, the genetic operators of selection, crossover, and mutation are used once the fitness value of the entire population has been computed. Genetic operators offer individuals with more knowledge, and the GA tends to arrive at an optimal or near-optimal solution as a result of these operations. The other optimization techniques may be more efficient in terms of fast convergence for particular optimization issues and

problem cases than genetic algorithms. Evolutionary techniques, evolutionary programming, simulated annealing, Gaussian adaptation, hill climbing, and swarm optimization (e.g., ant colony optimization, particle swarm optimization), as well as integer linear programming approaches, are examples of alternative and complimentary algorithms. Genetic algorithms' applicability is determined by how well the problem is understood; well-known issues commonly have superior, more specialized techniques.

(d) Get the final subset of CNN's hyperparameters using GA search to produce the optimum solution.

3.4. Intrusion Detection Method. As shown in Figure 2, the intrusion detection method based on a multichannel autoencoder mainly comprises a multichannel autoencoder and one-dimensional CNN. The model combines the advantages of feature-based detection and anomaly detection methods, using regular traffic and attack traffic to train the autoencoder models of their respective channels, and then utilizes a 1D CNN network to extract the hidden interrelationships in the multichannel description, thereby reducing the imbalance [15]. The influence of data on the model dramatically improves the detection accuracy of the sample, and because the model uses an unsupervised autoencoder and a lightweight 1D CNN network, the detection efficiency of the model is effectively improved. The specific training process is given below:

(a) Use data preprocessing to initialize the training sample set.

(b) Use the processed sample set to train the autoencoder of each channel.

(c) Reconstruct the feature vectors calculated by different channel autoencoders and use them as the input of 1D CNN.

(d) Use the reconstructed vector to train the CNN, and use the GA algorithm to optimize the hyperparameters of the CNN.

(e) Use the trained model to detect the samples and output the classification results based on the Softmax classifier.

4. Performance Evaluation

The overall performance of the proposed method is measured by analyzing the perfection and *F*-score of the trained intrusion detection model, which can be obtained from the obfuscation table, where perfection is the ratio of accurately tagged traffic, defined as

$$\mathrm{ACC} = \frac{\mathrm{UQ} + \mathrm{UM}}{\mathrm{UQ} + \mathrm{UM} + \mathrm{GQ} + \mathrm{GM}}. \tag{7}$$

In the formula, UQ and UM represent the number of samples that correctly predict traffic as usual and attack types, respectively, GQ and GM represent the number of models that incorrectly predict traffic as usual and attack types, and UQ + UM + GQ + GM is the total number of samples [16]. The higher the accuracy of the test results, the higher the algorithm's performance incorrectly predicting the type of traffic.

The *F*-score is the harmonic mean of precision and recall, where precision p measures the capability of the intrusion detection system to recognize only attacks, and recall r can be seen as the system's capability to detect all attacks, which is defined as

$$P = \frac{\mathrm{UQ}}{UQ + GQ},$$

$$R = \frac{\mathrm{UQ}}{\mathrm{UQ} + \mathrm{GM}}, \tag{8}$$

$$\mathrm{F\ score} = \frac{2 * q * s}{s + q}.$$

The higher the *F*-score, the better the balance between precision and recall achieved by the method.

4.1. Experiment on Multichannel Autoencoder. The method in this paper uses four experiments to verify the superiority of the performance. The first experiment is the ablation study, by setting up four network structures: neural network (NN), artificial neural network (ANN), CNN, and artificial convolution neural network (ACNN) to verify the multichannel in this paper. The effectiveness of the deep learning algorithm: the second experiment is robustness, and the robustness of the method in this paper is verified by testing in an unbalanced dataset; the third experiment is to explore the process of reconstructing samples from autoencoders z_n and z_a, and analyze [17]. The effectiveness of the multichannel autoencoder; the fourth experiment is a comparative study, which verifies the advantages of the method in this paper by comparing it with the results of several other intrusion detection algorithms.

4.1.1. Architectures of Networks' Structures. First, the architectures of 4 networks' structures, NN, ANN, CNN, and ACNN, are given:

(1) The NN network model consists of an input layer and the last four layers of the architecture in Figure 2 (1 flatten layer, 2 FC layers, and 1 Softmax layer), and the input samples are $x^i \in \mathrm{RD}$; namely,$Z^{(D)} \longrightarrow$ Flatten (320) $\longrightarrow$ GE (160) $\longrightarrow$ GE (2) $\longrightarrow$ Softmax.

(2) The architecture of the ANN network is the same as that of the NN, but the input samples use the combined information of two autoencoders $z_i \oplus z_i^{\hat{n}} \oplus z_i^{\hat{a}} \in S^{3E}$.

(3) The architecture of the CNN network is to add a one-dimensional convolutional layer to the NN, and the architecture is:

(4) The architecture of the ACNN network is the same as that of the CNN, but the input data adopts the

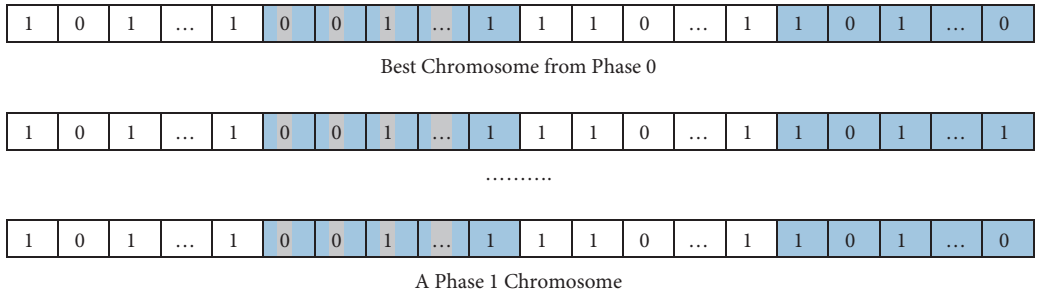

Figure 2: Chromosome structure in CNN optimization model.

combined information of the autoencoder, and the architecture is X(3D) ⟶ Flatten (320) ⟶ FC (160) ⟶ FC (2) ⟶ Softmax.

(5) The architecture of the CNN network is to add a one-dimensional convolutional layer to the NN, and the architecture is $Z^{(D)}$ ⟶ Flatten (320) ⟶ GE (160) ⟶ GE (2) ⟶ Softmax.

(6) The architecture of the ACNN network is the same as that of the CNN, but the input data adopts the combined information of the autoencoder, and the architecture is $Z^{(D)}$ ⟶ Conv1E (64) ⟶ Flatten (320) ⟶ GE (160) ⟶ GE (2) ⟶ Softmax.

Figures 3 and 4 along with Tables 1 and 2 illustrate the accuracy and *F*-scores of the proposed technique and four networks NN, ANN, CNN, and ACNN. With our technique, we exceed all baseline methods in terms of accuracy and *F*-score, proving the usefulness of using autoencoders in conjunction with 1E convolution and multichannel input to increase the accuracy of intrusion detection tasks [18]. It is vital to remember that autoencoders that are not tied to convolutions do not always produce better results. Dense convolutional layers, instead of autoencoders, may typically enhance intrusion detection accuracy. In any scenario, great accuracy and *F*-scores may be reached when using convolutions on data rich in autoencoders. Using many channels to compute convolutions rather than a single channel built by concatenation, the investigation indicated that the model superiority resulting from convolution relied on finding both the original variable and its autoencoder-based counterpart feature.

In addition, this paper also compares the number of parameters estimated by several networks, as shown in Table 3. As can be seen from the table, the higher accuracy of our method usually comes at the expense of a more significant number of estimated parameters. At the same time, the smaller the proportion of attack samples, the more parameters are required.

4.1.2. Robustness Measurement of the Proposed Method. The second experiment is mainly used to analyze the robustness of our method in solving the problem of imbalanced data. For the analysis data of this experiment, this paper uses the CICIDS2017 dataset, which is based on unbalanced data collected from real-world network scenarios, including 80% regular traffic and 20% attack traffic [19]. To verify the robustness of the method in this paper to imbalanced data, the usual traffic and attack traffic in the dataset are combined to obtain 5 subsamples that account for 100%, 75%, 50%, 25%, and 5% of the total sample. Set for testing Figure 5 with table presents the *F*-score test results of our method and four networks NN, ANN, CNN, and ACNN in 5 sample subsets. It can be seen from the figure that the *F*-score corresponding to the method in this paper has a minor decrease, and the *F*-score is still the highest among all forms, which shows that the manner in this paper is suitable for intrusion detection of unbalanced data.

4.1.3. Reconstruction Error Analysis. The third experiment is an analysis of multichannel autoencoders to explore the process by which autoencoders Zn and Za accurately reconstruct samples from both normal and attack classes. It can be seen from the autoencoder that Zn is more accurate in reconstructing standard samples than attacking samples, while Za has the opposite effect performance [20]. Thus, it is demonstrated that using a multichannel autoencoder to train standard samples and attack samples separately can launch information that helps to distinguish the two classes. Furthermore, it is also observed from the figure that our method has superior advantage to the number of autoencoder-based samples in both categories.

4.1.4. Intrusion Detection Algorithms. The fourth experiment compares the method in this paper with several intrusion detection algorithms, such as DNN, AIDA, CNN-1D, Gray-scale, WnD, and MDPCA-DBN.

Figures 6–8 show the different intrusion detection algorithms in three data along with Tables 4 and 5.

Comparison of test results on the set: as can be seen from the figure, the accuracy and *F*-score results of the method in this paper are generally better than those of the comparison methods on the three datasets, which fully demonstrates the effectiveness and superiority of the way in this paper. Furthermore, the only exception observed from the figure is on the KDDCUP99 dataset, where the results of our method are in suboptimal solutions, and the DNN model has the best accuracy and *F*-score.

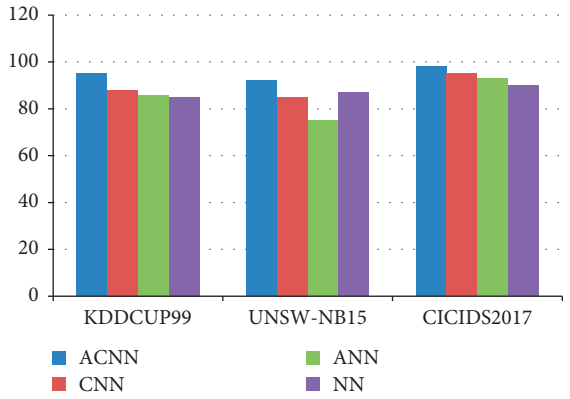

Figure 3: Accuracy of different network models on 3 datasets.

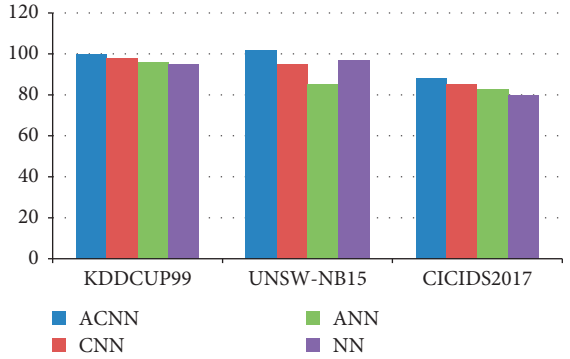

Figure 4: *F*-score of different network models on 3 datasets.

Table 1: Accuracy of different network models on 3 datasets.

Network	ACNN	CNN	ANN	NN
KDDCUP99	95	88	86	85
UNSW-NB15	92	85	75	87
CICIDS2017	98	95	93	90

Table 2: *F*-score of different network models on 3 datasets.

Network	ACNN	CNN	ANN	NN
KDDCUP99	100	98	96	95
UNSW-NB15	102	95	85	97
CICIDS2017	88	85	83	80

Table 3: *F*-score of different network models on unbalanced data.

Network	ACNN	CNN	ANN	NN
KDDCUP99	95	88	76	67
UNSW-NB15	98	95	65	68
CICIDS2017	85	68	75	80

This is because DNN learns an intrusion detection model through a deep neural network and text representation and then captures context- and sequence-related knowledge in system calls. The model goes through an optimization process to find the network's best parameters and topology. Therefore, the higher accuracy of DNNs on the KDDCUP99 dataset can be attributed to the text representation method and the topology and parameter settings of the determined architectures. These accuracy and *F*-score values are shown in Table 6.

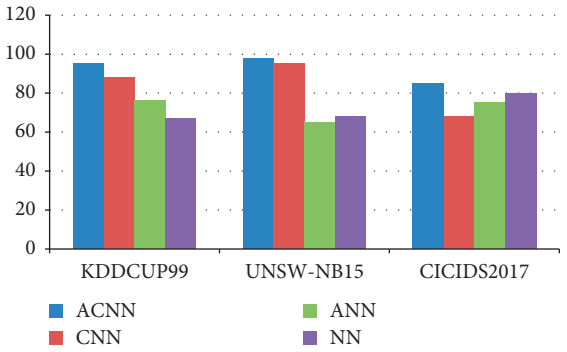

Figure 5: F-score of different network models on unbalanced data.

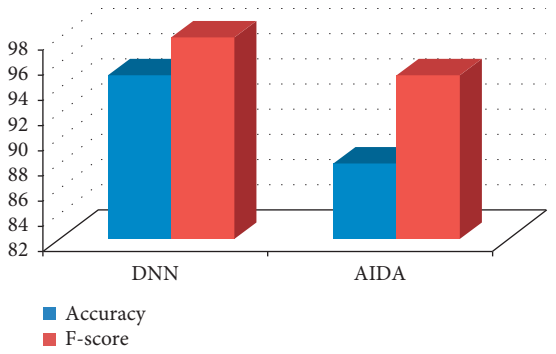

Figure 6: Comparison of different algorithms on KDDCUP99.

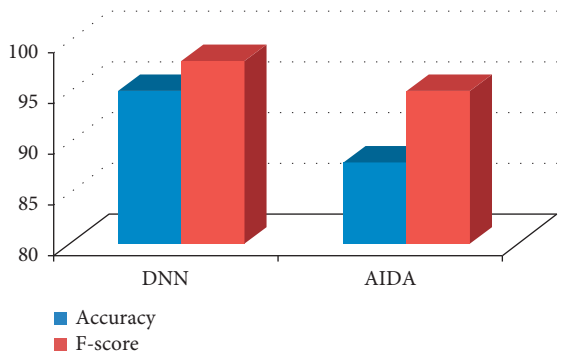

Figure 7: Comparison of different algorithms on UNSW-NB15.

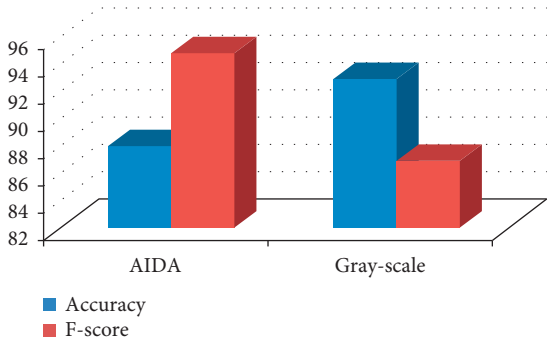

Figure 8: Comparison of different algorithms on CICIDS2017.

Table 4: Comparison of different algorithms on KDDCUP99.

Algorithms	DNN	AIDA
Accuracy	95	88
F-score	98	95

Table 5: Comparison of different algorithms on UNSW-NB15.

Algorithms	DNN	AIDA
Accuracy	95	88
F-score	98	95

Table 6: Comparison of different algorithms on CICIDS2017.

Algorithms	AIDA	Gray-scale
Accuracy	88	93
F-score	95	87

5. Conclusion

Multichannel autoencoder deep learning is proposed here as an intrusion detection approach to address the low accuracy and high false-positive rate that currently plague intrusion detection systems. A successful detection model is built by combining an unsupervised stage in which two autoencoders are trained with average traffic and an attack traffic stage in which cross-channel feature dependency is supervised. The architecture of a one-dimensional CNN model is improved using a genetic technique in this study. As a result of the experiments detailed in this article, the accuracy of this method in intrusion detection systems was greatly enhanced when compared to other methods studied. The encoders then generate new feature vectors, which are combined into a multichannel feature vector representation and fed into a one-dimensional CNN network to learn about any channel-to-channel correlations. For improved intrusion detection performance while preserving the model's flexibility, this research employs a genetic approach to optimize the one-dimensional CNN model's topology. Compared to other approaches tested, this one's accuracy in intrusion detection systems was significantly improved as a consequence of the experiments described in this study.

Data Availability

The data shall be made available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

References

[1] S. Araki, T. Hayashi, M. Delcroix, M. Fujimoto, K. Takeda, and T. Nakatani, "Exploring Multi-channel features for denoising-autoencoder-based speech enhancement," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 116–120, South Brisbane, QLD, Australia, April 2015.

[2] K. Zmolikova, M. Delcroix, L. Burget, T. Nakatani, and J. H. Černocky, "Integration of variational autoencoder and spatial clustering for adaptive multi-channel neural speech separation," in *Proceedings of the Spoken Language Technology Workshop (SLT)*, pp. 889–896, Shenzhen, China, January 2021.

[3] D. Ayata, Y. Yaslan, and M. Kamasak, "Multi channel brain EEG signals based emotional arousal classification with unsupervised feature learning using autoencoders," in *Proceedings of the 2017 25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, Antalya, Turkey, May 2017.

[4] S. Thakkar, C. Cao, L. Wang, T. J. Choi, and J. Togelius, "Autoencoder and evolutionary algorithm for level generation in lode runner," in *Proceedings of the 2019 IEEE Conference on Games (CoG)*, pp. 1–4, London, UK, August 2019.

[5] O. Nikisins, A. George, and S. Marcel, "Domain adaptation in multi-channel autoencoder based features for robust face anti-spoofing," in *Proceedings of the 2019 International Conference on Biometrics (ICB)*, pp. 1–8, Crete, Greece, June 2019.

[6] L. Li, H. Kameoka, and S. Makino, "Fast MVAE: joint separation and classification of mixed sources based on multichannel variational autoencoder with auxiliary classifier," in *Proceeding of the ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 546–550, Brighton, UK, May 2019.

[7] A. Mehbodniya, I. Alam, S. Pande et al., "Financial fraud detection in healthcare using machine learning and deep learning techniques," in *Security and Communication Networks*, C. Chakraborty, Ed., vol. 2021, Article ID 9293877, 8 pages, 2021.

[8] Q. Yin, B. Duan, M. Shen, and X. Qu, "Stacked sparse autoencoder based fault detection and location method for modular five-level converters," in *Proceedings of the IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 1580–1585, Beijing, China, October 2017.

[9] K. Mahajan, U. Garg, and M. Shabaz, "CPIDM: a clustering-based profound iterating deep learning model for HSI segmentation," in *Wireless Communications and Mobile Computing*, V. Shanmuganathan, Ed., vol. 2021, Article ID 7279260, 12 pages, 2021.

[10] K. Chen, J. Hu, and J. He, "Detection and classification of transmission line faults based on unsupervised feature learning and convolutional sparse autoencoder," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1748–1758, 2016.

[11] S. Sanober, I. Alam, S. Pande et al., "An enhanced secure deep learning algorithm for fraud detection in wireless communication," in *Wireless Communications and Mobile Computing*, V. Shanmuganathan, Ed., vol. 2021, Article ID 6079582, 14 pages, 2021.

[12] S. Li, B. Qin, J. Xiao, Q. Liu, Y. Wang, and D. Liang, "Multi-channel and multi-model-based autoencoding prior for grayscale image restoration," in *IEEE Transactions on Image Processing*, vol. 29, pp. 142–156, IEEE, 2020.

[13] S. Seki, H. Kameoka, L. Li, T. Toda, and K. Takeda, "Generalized multichannel variational autoencoder for underdetermined source separation," in *Proceedings of the 2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5, A Coruna, Spain, September 2019.

[14] M. S. Almahirah, N. S. Vijayalakshmi, M. Jahan, S. Sharma, and S. Kumar, "Role of market microstructure in maintaining

economic development," *Empirical Economics Letters*, vol. 20, no. 2, pp. 1–14, 2021.

[15] S. S. Embrandiri and M. Ramasubba Reddy, "Maximum contrastive networks for Multi-channel SSVEP detection," in *Proceedings of the 2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, pp. 992–995, Montpellier, France, April 2015.

[16] S. Kumar, "Relevance of buddhist philosophy in modern management theory," *Psychology and Education*, vol. 58, pp. 2104–2111, 2020.

[17] J. Yang, Z. Ma, J. Wang, and Y. Fu, "A novel deep learning scheme for motor imagery EEG decoding based on spatial representation fusion," *IEEE Access*, vol. 8.

[18] L. Pfeifenberger, M. Zöhrer, and F. Pernkopf, "DNN-based speech mask estimation for eigenvector beamforming," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 66–70, New Orleans, LA, USA, March 2017.

[19] G. Roland, S. Kumaraperumal, S. Kumar, A. D. Gupta, S. Afzal, and M. Suryakumar, "PCA (principal component analysis) approach towards identifying the factors determining the medication behavior of Indian patients: an empirical study," *Tobacco Regulatory Science*, vol. 7, no. 6, pp. 7391–7401, 2021.

[20] B. Li, H. Yu, and A. Sano, "Toward end-to-end prediction of future wellbeing using deep sensor representation learning," in *Proceedings of the 2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pp. 253–257, Cambridge, UK, September 2019.