Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2021.017515
Article

Tech Science Press

A Hybrid Deep Learning Intrusion Detection Model for Fog Computing Environment

K. Kalaivani* **and M. Chinnadurai**

Department of Computer Science and Engineering, E.G.S. Pillay Engineering College, Nagapattinam, 611002, India
*Corresponding Author: K. Kalaivani. Email: akilkalai12@gmail.com
Received: 01 February 2021; Accepted: 16 April 2021

Abstract: Fog computing extends the concept of cloud computing by providing the services of computing, storage, and networking connectivity at the edge between data centers in cloud computing environments and end devices. Having the intelligence at the edge enables faster real-time decision-making and reduces the amount of data forwarded to the cloud. When enhanced by fog computing, the Internet of Things (IoT) brings low latency and improves real time and quality of service (QoS) in IoT applications of augmented reality, smart grids, smart vehicles, and healthcare. However, both cloud and fog computing environments are vulnerable to several kinds of attacks that can lead to unexpected loss. For example, a denial of service (DoS) attack can block authenticated users by rendering network resources unavailable and consuming network bandwidth unnecessarily. This paper proposes an intrusion classification model using a convolutional neural network (CNN) and Long Short-Term Memory networks (LSTM) to obtain the advantages of deep learning methods in order to accurately predict such attacks. The proposed integrated CNN with LSTM-based Fog Computing Intrusion Detection ICNN-FCID model is used for multi-class attack classification. Our proposed model is demonstrated using NSL-KDD, a benchmark dataset, and provides attack detection accuracy of about 96.5%. Comparisons of the accuracy of our model with both traditional and other recent deep learning approaches show that our model is superior in performance. The ICNN-FCID model can be used in fog layer devices where network traffic is monitored and the attacks are detected. As a result, the cloud server and fog layer devices can be protected from malicious users and are always available in providing services to IoT devices.

Keywords: IoT; Fog computing; IDS; deep learning; CNN and LSTM

1 Introduction

The Internet of Things (IoT) refers to the interconnected billions of physical devices that store and exchange data from around the world through the internet, where data can be processed and used for many purposes. The large amounts of data generated by the IoT need to be stored, processed, and accessed [1,2]. The cloud computing paradigm can be used for big data storage and analytics. The

sensing data of IoT devices can be stored in the cloud so that the smart devices can be monitored and actuated [3]. This will enable the development of new applications using IoT and smart devices.

Fog computing is an architecture that integrates cloud and IoT technology. The fog architecture acts like a cloud but is closer to the end user, and enables cloud computing facilities to be at the edge of the network, through which connected devices can obtain cloud services [4]. Fog computing enables the operations of cloud computing by means of a control plane and a data plane. A fog node is a device that includes the capabilities of computing, storage, and network connectivity. Multiple fog nodes can be installed to provide support to end devices. Switches, embedded servers, controllers, routers, and cameras can act as fog nodes. Most of the time-sensitive data generated by the end devices are sent to the fog node, where the data are analyzed. The response is sent to the device in a fraction of second. Then the fog node will send a summary of the data and work to the cloud for further analysis. The less time-sensitive data can be processed after seconds or minutes and sent to the aggregate node. After analysis, the aggregate node sends the response via the nearest node to the device. Later the aggregate node will send the report to the cloud for future review.

The IoT network's data that are not time-sensitive can be sent to the cloud, where they are processed, analyzed, and stored. The end devices will wait hours, days, or even weeks for the data. The fog computing architecture also uses private servers to store the confidential data. This local server is also useful for ensuring data security and privacy. The fog node can receive irrelevant data from malicious user during communication. The attackers can produce a flood of data to execute a denial of service (DoS) attack, thereby diminishing the availability of fog nodes and the cloud. A local fog server is vulnerable to several kinds of attacks, including DoS, Remote to Local (R2L), Probe, and User to Root (U2R). This requires effective detection and prevention of various attacks [5]. Since the fog node is constrained by limited resources, if it suffers a DoS attack, it will not be able to provide services for users and the network performance will be greatly reduced. An intrusion detection system (IDS) is used [6] to monitor a network or systems for malicious activity and such activities should be reported either to an administrator or collected centrally. An IDS can be network-based, meaning it is responsible for checking network structures and looking for attack signatures, host-based, which means it is used to monitor host systems, or application-based, meaning it monitors specific applications and programs.

An IDS can be implemented through deep learning models, which are advanced models of machine learning. This model consists of several consecutive layers that are interlinked, and each layer receives the previous layer's output as input. The key advantage of the deep learning algorithm over other machine learning algorithms is its ability to run feature engineering on its own. A deep learning algorithm scans the data to search for correlated features and combines them to enable faster learning without guidelines. Deep learning models are capable of creating new features by themselves. Once the deep learning model is properly trained, it can perform thousands of routine, repeatable tasks within a shorter time frame. A convolutional neural network (CNN) is a deep learning algorithm that can be used in various domains such as image processing, natural language processing (NLP), and biomedical applications. CNN has achieved excellent research results in image classification, sentiment classification, relation classification, textual summarization, and disease diagnosis and detection [7,8], and it has also been applied to various information security use cases, such as classification of malware, detection of intrusion, and Android malware, spam and phishing, and binary analysis. Historically, network intrusion detection was performed by machine learning models. However, these algorithms can cause the program to produce many false positives, causing repetitive work for security teams. Deep learning models can be used to develop more intelligent IDSs that can analyze network traffic more reliably. To address the challenge of intrusion detection in fog computing environments, this paper proposes an integrated CNN with LSTM-based intrusion classification methodology (ICNN-FCID). This methodology can reduce the number of false alerts and help security teams distinguish between bad and good network activities.

2 Related Work

This section lists various accomplishments in the area of intrusion detection, specifically the real-world IDS. Much research has been carried out in the area of network intrusion [9,10]. Yang et al. [11] designed the SVM-RBM algorithm using a support vector machine (SVM) and a restricted Boltzmann machine (RBM) to detect network anomalies. They then used the unsupervised algorithm of RBM to extract useful features from the datasets and trained the SVM classifier in a short time using the Spark gradient descent algorithm. They explored the numbers of hidden units for improving the performance of SVM-RBM. Jiang et al. [12] proposed the use of LSTM recurrent neural networks (LSTM-RNNs) as an intelligent multi-channel attack detection model. They performed multi-channel training with different types of features for preserving attack features of input data. They then classified the attacks and normal data and used a voting algorithm to determine whether the input data are an attack or not with the results of the classifier's attack detection. They have shown that their work was superior to other attack detection methods, such as Bayesian or SVM classifiers.

Peng et al. [13] proposed a decision tree-based IDS system for fog computing environments. They digitized the strings in the KDD Cup dataset using a preprocessing algorithm, and they increased the quality of the input data through data normalization. This improved the efficiency of detection. Gao et al. [14] proposed a deep belief network (DBN), which is a combined form of unsupervised learning networks, a four-layer RBM, and a back propagation network, which is a supervised learning algorithm. Their result is demonstrated with the KDD Cup 1999 dataset. Farahnakian et al. [15] proposed an enhanced IDS model using the deep autoencoder (DAE) method. From the high-dimensional data, features were extracted using AE. They used four autoencoders in their deep autoencoder-based IDS (DAE-IDS), in which the result of the previous layer is used as the input to the next layer in AE. Each layer undergoes greedy unsupervised training to improve the efficiency. After the four autoencoders were trained, they used a softmax layer to classify the inputs to normal and attack. They also used the KDD Cup 1999 dataset in their work for evaluating the efficiency of DAE-IDS.

Wang et al. [16] proposed a hierarchical spatial-temporal features-based intrusion detection system (HAST-IDS). They used deep CNNs to learn low-level spatial features of input data and LSTM networks to learn high-level temporal features from raw data. The deep neural networks automatically completed the entire process of feature learning using the DARPA1998 and ISCX2012 datasets. Kim et al. [17] proposed a DNN-based IDS model for detecting attacks. They used the KDD Cup 1999 dataset. Their DNN model used four hidden layers, 100 hidden units, and a ReLU activation function for the proposed IDS. Potluri et al. [18] used the NSL-KDD dataset to develop an accelerated DNN model for identifying the anomalies in the network data. The input layer contains 41 features that are fed into the DNN and two hidden layers are used for selecting 10 features from 41 features in the dataset. The first two hidden layers come into the pre-training procedure of the DNN. The hidden layer 3 is the softmax layer that will decrease the number of features to five.

Zhang et al. [19] used two hybrid algorithms that combine SVM, RBM, and DBN for the analysis of the false positive rate, accuracy, false negative rate, and testing period with the KDD Cup-99 dataset. Illy et al. [20] proposed using ensemble learners for increasing the accuracy of an IDS. In a Fog of Things environment, they used two classification levels. Othman et al. [21] introduced the Spark-Chi-SVM model for intrusion detection, which used ChiSqSelector for feature selection and an SVM-based intrusion detection model built on the Apache Spark big data platform. They used the KDD99 dataset in their work. In comparing Chi-SVM classifier with Chi-Logistic Regression classifier, the Spark Chi-SVM model demonstrated high performance. Many of the previous works are implemented with KDD cup dataset which consists of redundant records. Also most of the research works on fog computing have focused on architectural aspects except few. The contributions of our research paper are as follows.

- The deep neural network is highly applicable to this field. We have introduced an integrated CNN with LSTM-based intrusion classification model for IDS, called ICNN-FCID, for fog computing environments, where accurate prediction can reduce the number of false alerts. This is primarily because CNN is capable of extracting high-level representations of features that reflect the abstract nature of low-level network traffic communication feature sets and because LSTM is capable of learning long-term dependencies in data.
- The NSL-KDD dataset is used in our proposed model for training and testing.

3 Proposed Methodology

This section describes the architectures of the fog computing model, the CNN-LSTM model, and the proposed architecture and algorithm for intrusion detection.

3.1 Fog Computing Architecture

In our system model, a hybrid IoT network is considered and shown in Fig. 1. It includes the IoT devices D: a set of heterogeneous devices (d), which are equipped with sensing and communication capabilities. Sensing results are periodically reported to the cloud server via the fog device.

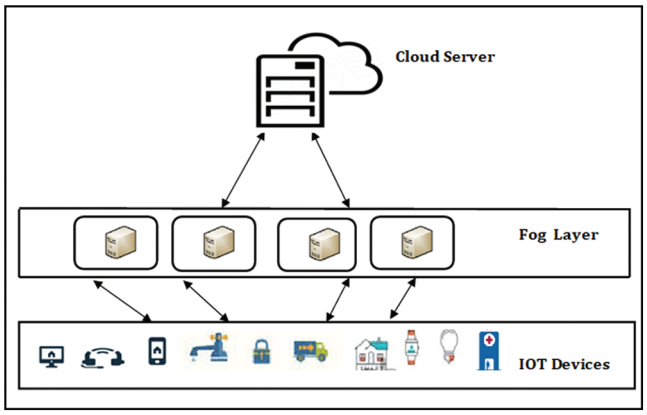

Figure 1: Fog computing architecture

$$D = \{d1, d2, d3 \ldots\ldots dn\} \tag{1}$$

The set D can be further divided into k subsets such as

$$D1, D2, D3 \ldots\ldots\ldots Dk \subseteq D \tag{2}$$

Each subset has m number of IoT devices

$$Di = \{d1, d2, d3 \ldots dm\} \tag{3}$$

where $i = 1 \ldots k$, $Di \cap Dj = \Phi$ for any $i \neq j$ and $|Di| = mi$

The fog layer consists of set of fog nodes and local fog server/cloud. The fog nodes serve as the relay between the IoT devices and the cloud server. F is a set of fog nodes.

$$F = \{fd1, fd2, fd3 \ldots\ldots\ldots fdx\} \tag{4}$$

The fog layer may be vulnerable to several kinds of attacks. These nodes can monitor the anonymous traffic where our proposed integrated DNN model is deployed to detect intrusive behavior. In this way, the fog layer and cloud server can be protected from malicious users. In addition, it gives fog nodes high availability for more time-sensitive applications.

3.2 CNN-LSTM

A convolutional neural network is a kind of deep neural network and is referred to as CNN or ConvNet [22–24]. The architecture of an ICNN-LSTM is shown in Fig. 2. The input layer contains the input.

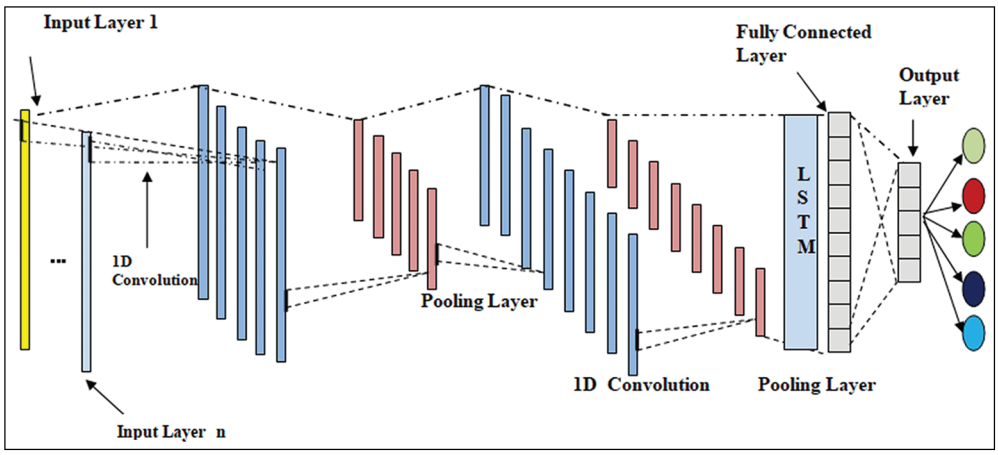

Figure 2: Architecture of ICNN-LSTM model

The primary building unit of a CNN is the convolutional layer, which uses a series of convolution kernels, which are used to identify the features part of the network traffic data. A set of n kernels and biases are $W = \{w_1,w_2,\ldots,w_n\}$ and $B = \{b_1,b_2,\ldots, b_n\}$, respectively, and are convolved with input data at each CNN layer. A new feature map x_k is produced by the convolution between data and each kernel. For each and every convolution layer l, the transformation is defined by:

$$X_k^l = \sigma(W_k^{l-1} * X^{l-1} + b_k^{l-1}) \tag{5}$$

The convolution operation is performed by sliding a filter or kernel over the inputs in the CNN learning process by which the optimized values of weights and bias can be obtained from different features of the input data without considering their position in the input data. The Activation Function, also known as the transfer function, is used to obtain the output of the node. The activation function is applied for every value in this layer. The ReLU rectified linear activation function is one of the common activation functions, and is a piecewise linear function. If the input is positive, it will output the input, else it will output zero and often achieve better performance.

$$f(x) = \max(0, x) \tag{6}$$

The pooling layer is another building block of a CNN. It is used to reduce the spatial size of the representation progressively through dimensionality reduction. In this way, the computational power for processing the data is greatly reduced and it controls overfitting. Max pooling, the most commonly used approach, will return the maximum value from the input, which is covered by the kernel. Max pooling discards the noisy activations. The LSTM layer is a class of recurrent neural network (RNN), which is able to learn long-term dependencies in data. In the fully connected layer, the non-linear combinations of

the high-level features are learned in the output representation of the convolutional layer. The flattened output is input to a feedforward neural network. In every iteration of training, backpropagation is applied. The model can distinguish between dominating features and certain low-level features. These features can be classified using the softmax classification technique over a series of epochs. The output layer compares output values that are predicted with the known labels. After that, it finds the error of the predicted value. The error is sent back through the loss function, through which weights and bias will be updated.

3.3 Proposed Architecture and Algorithm for ICNN-FCID Model

The architecture of the proposed ICNN-FCID classification model is shown in Fig. 3. It consists of three main processes: data preprocessing, training, and testing. These processes are shown in Algorithm 1.

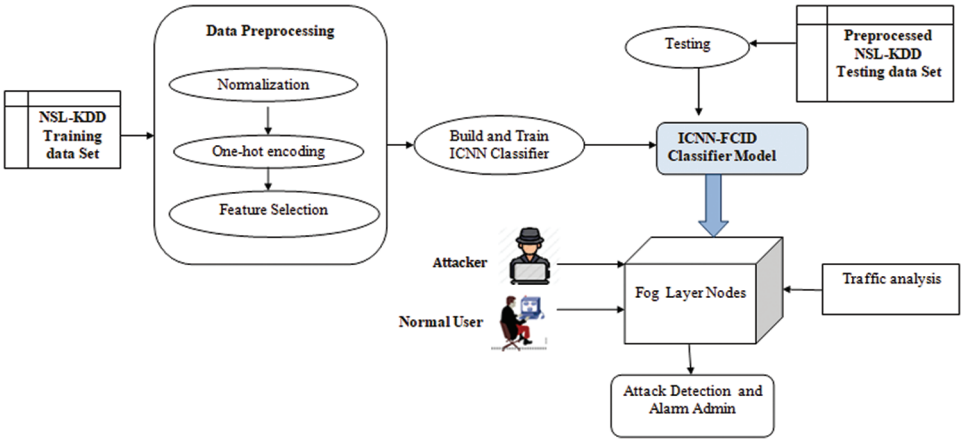

Figure 3: Block diagram of ICNN-FCID Model

During data preprocessing, raw data are transformed into a useful and efficient format. The raw data (real-world data) cannot be applied through a CNN model because errors will result. Data must be preprocessed before it can be used. This step includes the operations of data normalization, feature selection, and one-hot encoding. Data normalization is a technique used to scale the value of each feature having a different range of values to a common scale. In feature selection, the numbers of optimal features are selected. The symbolic features cannot be processed by the CNN model. One-hot encoding is used for converting data into numerical values. Through the feature selection process, the number of features in the NSL-KDD dataset is reduced. Using that subset of features, our ICNN-FCID model is trained and tested. With the CNN/LSTM-based intrusion classification deep learning model, a fog node can easily detect an attack and raise the alarm after processing the network traffic.

3.4 Algorithm 1: ICNN-FCID Classification Model

Input: TD: NSL-KDD dataset with 41 features of raw data.

Output: Well trained model of ICNN-FCID for IDS.

Step 1: Preprocessing of data with TD dataset.

1.1 Normalization of Data for all features in the dataset.

1.2 Selecting optimal features to create subset of features as TD1.

1.3 Feature Encoding using one hot encoding for the features in TD1.

Step 2: Build a Integrated Convolution Neural Network Classifier (ICNN-FCID) for the IDS.

Step 3: Optimize the Classifier using Adam optimizer.

Step 4: Train the classifier.

4.1 Forward propagating the inputs.

4.2 Backpropagation to the output layer.

4.3 Backpropagation to the hidden layer.

4.4 Updating weight

Step 5: Evaluate the performance of the ICNN-FCID classifier using model validation.

Step 6: Calculate the classification accuracy.

4 Experimentation Results and Evaluation

This section evaluates the proposed ICNN–FCID classification model for intrusion detection. In this experiment, the CNN with LSTM is used for five classes of classification (normal, DoS, Probe, R2L, and U2R attacks). Our model is implemented using Python and Keras (the deep learning library of Python) on a computer equipped with an Intel Core i7 CPU, 16 GB of RAM and Windows 10.

4.1 Dataset Description and Analysis

The NSL-KDD dataset used in our proposed methodology [25] is an updated, cleaned up version of the KDD Cup 1999 dataset. The NSL-KDD dataset was created because there were too many redundant records [26] in the KDD Cup 1999 dataset. It was created with the records of internet traffic by a simple intrusion detection network and consists of four types of subdatasets: KDD Train+, KDD Train+_20Percent, KDD Test+, and KDD Test-21. In this dataset, four classes of attacks exist: Denial of Service, Probe, R2L, and U2R. Each record contains 43 features. The first 41 features refer to the input of traffic itself, the 42^{nd} feature refers to whether it is a normal record or attack record, and the 43^{rd} feature refers to the score, i.e., the severity of traffic input.

4.1.1 Dataset Features Classification

The dataset has 32 feature columns containing numeric data, 6 features of each record contain binary data, and 3 features are nominal features. The total number of records in all categories of the NSL-KDD training dataset is 125,973. Of that number, 67,343, 45,927, 11,656, 995, and 52 records belong to the Normal, DoS, Probe, R2L, and U2R attack categories respectively, as shown in Tab. 3. Similarly, the test dataset consists of Normal, DoS, Probe, R2L, and U2R attack records, but contains additional attacks in each class that are not in the training dataset. The test dataset has 37 types of attacks, of which 16 are attacks not available in the training dataset.

4.1.2 Data Preprocessing

The NSL-KDD training and testing dataset must undergo certain preprocessing steps.

4.1.3 Data Normalization

This technique aims to change the numeric values of attributes to a standard scale without distorting differences in value ranges. The values in column y are transformed using Eq. (7) below.

$$z = \frac{y - \min(y)}{\max(y) - \min(y)} \tag{7}$$

4.1.4 Feature Selection

Our dataset, contains 43 features. The last two columns are Packet type and Score. Because some of the features were considered to have no effect on neural network analytical results, fewer resources were needed to complete tasks. The computational cost of the model can also be reduced. In our work we took the first 41 features except the last two columns. In that, the list of {7, 8, 9, 11, 14, 15, 16, 18, 19, 20, 21, 22, 25, 27, 31} features were removed, because they had zero values. This allowed us to reduce the data volume size from 41 to 26 features.

4.1.5 One-Hot Encoding

In the NSL-KDD dataset, there are four attributes with non-numeric values: protocol_type, service, flag, and class. These were converted into numeric values. The protocol_type feature has three value types: TCP, UDP, and ICMP. They were encoded into its binary vectors [1,0,0], [0,1,0], and [0,0,1] by applying one-hot encoding. The service feature has 70 attribute types and the flag feature has 11 attribute types. In this manner transformation was performed. After transformation, the 41-dimensional features were mapped into 112 dimensional features. The predicted targets were mapped with the five categories of classification: normal, DoS attack, Probe attack, R2L attack, and U2R attack. Tab. 1 shows all of the parameters and shapes in each layer in our model. We can see that the total number of parameters is 92,489 and all are trainable. The number of non-trainable parameters is 0.

Table 1: Summary of the ICNN-FCID model

Layer (type)	Output Shape	Param #
convid_1 (Con1D)	(None, 24, 64)	256
max_pooling1d_1 (MaxPooling1)	(None, 12, 64)	0
convid_2 (Con1D)	(None, 10, 64)	12352
max_pooling1d_2 (MaxPooling1)	(None, 5, 64)	0
lstm_1 (LSTM)	(None, 112)	79296
dropout_1 (Dropout)	(Nine, 112)	0
dense_1 (Dense)	(None, 5)	565
Total params: 92,469 Trainable params: 92,469 Non-trainable params: 0		

4.2 Evaluation Metrics

In this paper, a confusion matrix is used to describe the performance of our model. It includes significant details about actual and predicted output classes. True Positive (TPw) is the value that represents the number of records predicted as attacks that are actually anomalous records. True Negative (TNw) indicates the number of records predicted as normal that are actually normal records. False Positive (FPw) represents the value that indicates the number of records predicted as attacks, but they are actually normal records. False Negative (FNw) is the value that indicates the number of records predicted as normal that they are actually anomalous records. From the confusion matrix, we can define performance metrics mathematically as follows.

False Negative (FNw) is the value that indicates the number of records predicted normal that they are actually anomalous records. From the confusion matrix, we can define performance metrics mathematically as follows.

Accuracy: This is the percentage of the number of records that have been classified correctly to the total number of records.

$$\text{Accuracy (A)} = \frac{\text{TP}_\text{w} + \text{TN}_\text{w}}{\text{TP}_\text{w} + \text{TN}_\text{w} + \text{FP}_\text{w} + \text{FN}_\text{w}} \tag{8}$$

Recall: This is propositional to the true positive rate, and this is the percentage of the number of anomalous records have been correctly detected divided by the total number of anomalous records.

$$\text{Recall (R)} = \frac{\text{TP}_\text{w}}{\text{TP}_\text{w} + \text{FN}_\text{w}} \tag{9}$$

Precision: Precision quantifies the number of attack class predictions that actually belong to the anomalies class.

$$\text{Precision (P)} = \frac{\text{TP}_\text{w}}{\text{TP}_\text{w} + \text{FP}_\text{w}} \tag{10}$$

F-measure: This is the harmonic mean of accuracy and recall, which provides a measurement of derived effectiveness.

$$\text{F} - \text{measure (F} - \text{Score)} = 2 * \frac{\text{R} * \text{P}}{\text{R} + \text{P}} \tag{11}$$

False Alarm Rate: This is the misprediction of normal data as abnormal data.

$$\text{False Alarm Rate (FAR)} = \frac{\text{FP}_\text{w}}{\text{FP}_\text{w} + TN_w} \tag{12}$$

Misclassification Rate: This is the number of records that are incorrectly classified.

$$\text{Misclassification Rate} = \frac{\text{FP}_\text{w} + \text{FN}_\text{w}}{\text{TP}_\text{w} + \text{TN}_\text{w} + \text{FP}_\text{w} + \text{FN}_\text{w}} \tag{13}$$

4.3 Results

In our proposed ICNN-FCID model, the CNN section is composed of an input layer, convolution layer 1 with 64 filters, pooling layer 1 with pooling size 2 and stride size 1, convolution layer 2 with 64 filters, pooling layer 2 with pooling size 2 and stride size 1, LSTM layer 1 with output size 112, fully connected layer 1, and an output layer. Dropout by 0.3 is considered to prevent overflow. The Rectified Linear Unit (ReLU) activation function was used in all of the layers except the last layer. The softmax activation function was used in the last layer. For optimization, the Adaptive Moment Estimation (Adam) method was used.

Experiments were conducted with the ReLU, sigmoid, and hyperbolic tangent (TanH) activation functions, and the performance then compared. With ReLU we achieved the highest accuracy of 96.5%, precision of 85.25%, recall of 91.16%, and F-score of 86.43%. The accuracy, precision, recall, and F-score of our model with the sigmoid activation function were 85.58%, 80.06%, 90.50%, and 87.78%, respectively. The accuracy, precision, recall, and F-score with the TanH activation function were 88.3%, 84.53%, 89.30%, and 85.1%, respectively. All of the results are listed in Tab. 2, which shows that the ReLU activation function in the ICNN-FCID model provided higher accuracy than the sigmoid and TanH

activation functions. The number of epochs was set to 50 and the size of each batch was set to 64. We then assessed the performance of our proposed model by measuring its accuracy. In the experiments, the ICNN-FCID provided improved classification results and the accuracy of our model was approximately 96.5%.

Table 2: Performance evaluation of ICNN-FCID with activation functions

Activation Function	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)
ReLu	96.5	85.25	91.16	86.43
Sigmoid	85.58	80.06	90.5	85.6
Tanh	88.3	84.53	89.3	85.1

Table 3: Detailed performance of ICNN-FCID model on test dataset

Label	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)	False Alarm Rate (%)	Misclassification Rate (%)
Normal	97.73	96.45	98.36	97.40	2.73	2.27
DoS	98.93	98.56	98.22	98.39	0.70	1.07
Probe	98.34	97.67	86.61	91.81	0.24	1.66
R2L	98.94	98.16	93.10	95.57	0.24	1.06
U2R	98.53	35.41	80.00	49.00	1.29	1.47

We labeled DoS, Probe, R2L, and U2R attacks as 1, 2, 3, and 4, respectively, while the normal connections were labeled as 0. The confusion matrix for the NSL-KDD testing dataset of the ICNN-FCID model is shown in Fig. 4. In our model, as the epoch increases, the accuracy of the training and testing sets also increases. With the increase in epoch, there is a decrease in the loss of the training and testing sets. As the training rounds increase, the accuracy increases and the loss is decreases, but in the end, it appears to be flat. To find a better value of epoch, we tested every 10 epochs, from 10 to 50. The result is shown in Figs. 5a and 5b. Tab. 3 shows the performance of the ICNN-FCID model on the test dataset, as well as the accuracy, precision, recall, F-Score, false alarm rate, and misclassification rate for each class. The accuracy and false alarm rate for the DoS attack are 98.93% and 0.7%, respectively. The accuracy of the Probe, R2L, and U2R attacks is 98.34%, 98.94%, and 98.53%, respectively. The false alarm rate of the Probe, R2L, and U2R attacks is 0.24%, 0.24%, and 1.29%, respectively. Fig. 6 compares the performance analysis of the ICNN-FCID model with the three activation functions.

4.4 Comparison with Existing Work

The accuracy of our ICNN-FCID model was compared with conventional machine learning models and the latest algorithms of deep learning models. From Yin et al. [27] the literature reports that the traditional models of J48, Naive Bayesian, and Random Forest for intrusion detection have been implemented using the NSL-KDD dataset.

The accuracies in five class classifications of J48, Naïve Bayesian, and Random Forest were 81.05%, 76.56%, and 80.67%, respectively. Next, we analyzed the performance of our model with the implementation of the latest deep learning models. The deep learning classification model called stacked NDAE was proposed by Shone et al. [28] and had accuracy of 85.42%. Li et al. [29] proposed a

multi-CNN IDS fusion method that demonstrated 81.33% accuracy for multiclass classification. The accuracies of these models are shown in Tab. 4. Fig. 7 shows the results of the comparison of accuracies of our ICNN-FCID model with traditional J48, Naive Bayesian, Random Forest, Stacked NDAE, and multi-CNN fusion models. The experiment demonstrated that our ICNN-FCID model performs classification with high accuracy on the NSL-KDD dataset.

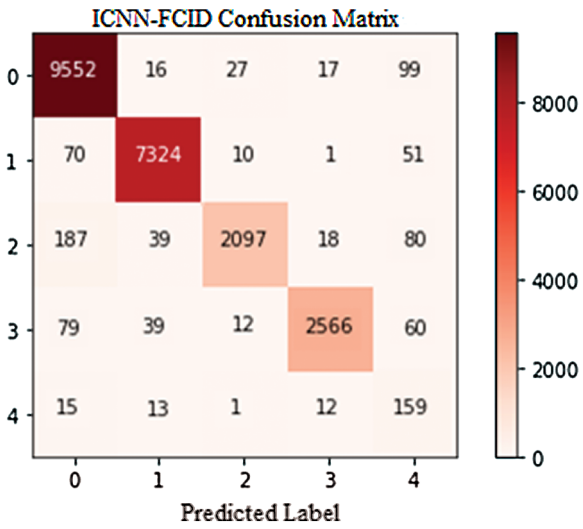

Figure 4: Confusion Matrix for NSL-KDD testing dataset

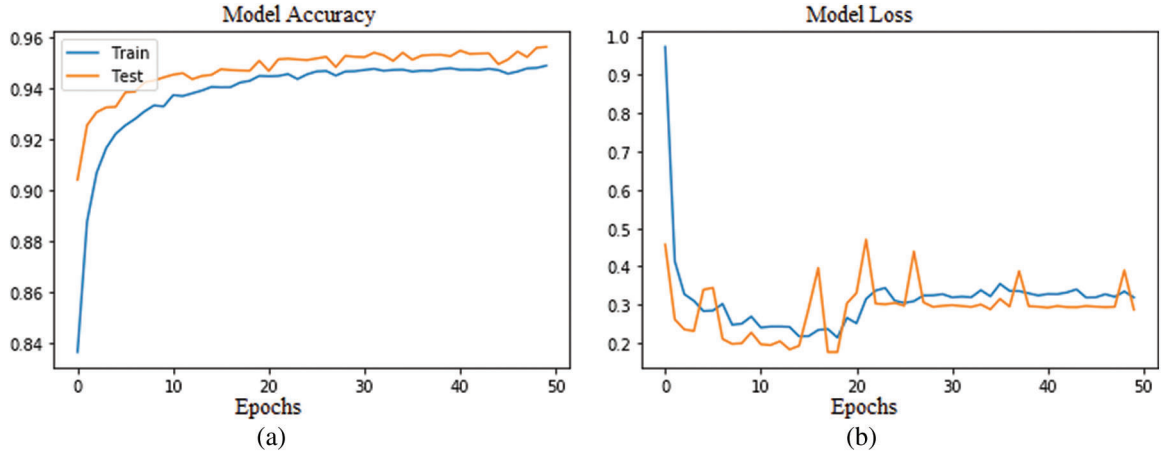

Figure 5: (a) Accuracy of ICNN-FCID model (b) Loss of ICNN-FCID model

4.5 Real Time Classification

Virtualization technology was used for simulating our experiments. We considered only DoS attacks. Fig. 8 shows the virtualization framework that was implemented for attacker-fog-cloud structure. All of the traffic to the cloud passes through the intermediate fog layer. Attack traffic can easily be classified on the fog layer through the ICNN-FCID model, so that the malicious traffic can be dealt with easily before it reaches the cloud server. Therefore, it provides efficient utilization of cloud resources and time. Using open-source tools and scripts on various operating systems and cloud servers, malicious traffic was

created to enforce this scenario, along with an intermediate fog layer through which the traffic was directed to the cloud. The malicious attack traffic was detected to save processing power both the cloud and fog node.

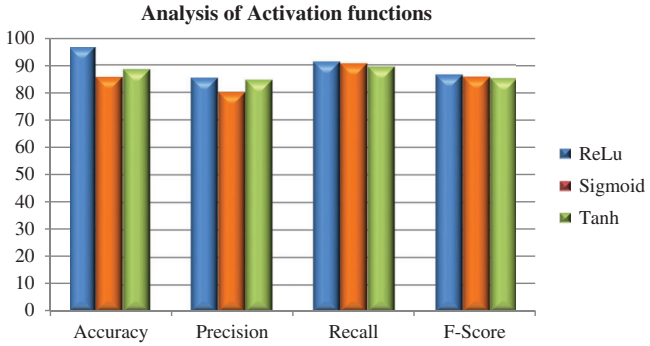

Figure 6: Comparative analysis of different activation functions

Table 4: Accuracies of our ICNN-FCID model, traditional and latest models for 5 class classification

Model	Accuracy of KDD Test+
J48	81.05%
NB	76.56%
RF	80.67%
Stacked NDAE [28]	85.42%
multi-CNN fusion [29]	81.33%
Our Model	96.50%

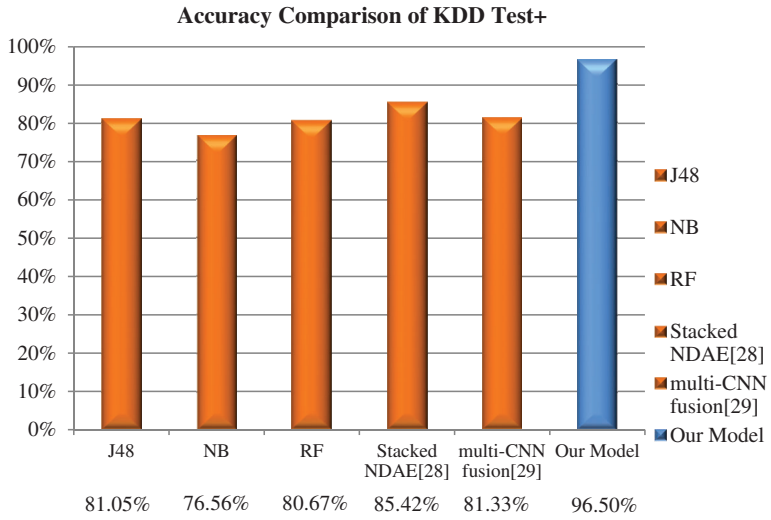

Figure 7: Performance of our ICNN-FCID model and other models for 5 class classification

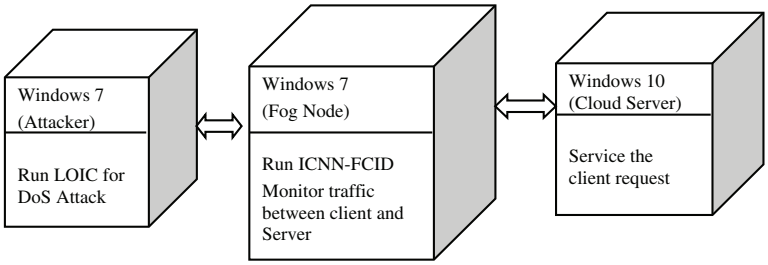

Figure 8: Real time classification framework

The VMware ESXi hypervisor was installed on the server and configured. Then the three nodes—two Windows 7 systems and one Windows 10 system—were deployed in the virtual environment using the web interface of the ESXi server, and the firewall was shut down to make the DoS attack easier. The proposed ICNN-FCID was deployed in one Windows 7 virtual machine (VM). This system was used to monitor all of the traffic. We used the open-source Ethereal Network Analyzer to capture the packets. We used the Low Orbit Ion Cannon (LOIC) network stress-testing and DoS attack application tool in another Windows 7 VM attacker system for creating a SYN flood to attack the target machine. After that, the network traffic was preprocessed we formed the dataset by adding 400 connection records. A total of 500 records were given as input to the ICNN-FCID model, which classified the packets with expected accuracy. Fig. 9 shows the real-time classification performance of the ICNN-FCID model.

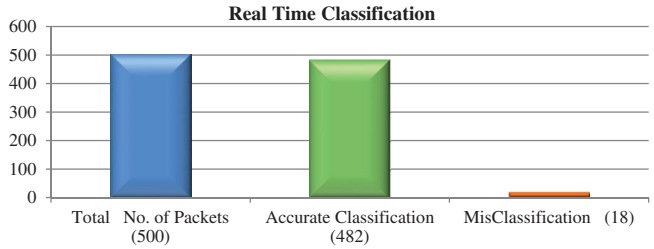

Figure 9: Real time classification performance of our ICNN-FCID model

5 Conclusion

We performed an experiment to demonstrate the feasibility of using CNN and LSTM for a Network Intrusion Detection System (NIDS), in order to exploit the power of deep learning to identify network intrusions. In this paper, a hybrid classification model combining CNN with LSTM, called ICNN-FCID, is proposed, then implemented and trained for intrusion detection in fog computing environments. To improve the accuracy of our model, we used normalization, one-hot encoding for the features in the dataset. We trained our hybrid model with the KDD Train + dataset, and tested it using the KDD Test + dataset. With a testing accuracy of 96.5%, our model outperforms traditional machine learning methods and other recent deep learning algorithms. The efficiency of our model for the classification of DoS, Probe, R2L, and U2R attacks has been demonstrated for fog computing environments. We conducted our test using virtualization technology to detect DoS attacks and it achieved the expected results. This means that the proposed ICNN-FCID classification model can function efficiently in real-time environments. The direction that our future work will take will be to provide an attack detection model using multiple CNNs with more real-time traffic.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Botta, W. Donato, V. Persico and A. Pescape, "Integration of cloud computing and Internet of Things: A survey," *Future Generation Computer Systems*, vol. 56, no. 1, pp. 684–700, 2016.

[2] C. Stergiou, K. E. Psannis, B. G. Kim and B. Gupta, "Secure integration of IoT and cloud computing," *Future Generation Computer Systems*, vol. 78, no. 3, pp. 964–975, 2018.

[3] P. P. Ray, "A survey of IoT cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1–2, pp. 35–46, 2016.

[4] A. Alrawais, A. Alhothail, C. Hu and X. Cheng, "Fog computing for the internet of things: Security and privacy issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[5] X. An, J. Su, X. Lu and F. Lin, "Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, pp. 1–10, 2018.

[6] N. A. Azeez, T. J. Ayemobola, S. Misra, R. Maskeliunas and R. Damasevicius, "Network intrusion detection with a hashing based Apriori algorithm using Hadoop MapReduce," *Computers*, vol. 8, no. 4, pp. 1–10, 2019.

[7] Y. Sun, B. Xue, M. Zhang and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 1–10, 2019.

[8] D. Połap, "Analysis of skin marks through the use of intelligent things," *IEEE Access*, vol. 7, no. 1, pp. 1–10, 2019.

[9] N. A. Azeez, O. J. Asuzu, S. Misra, A. Adewumi, R. Ahuja *et al.,* "Comparative evaluation of machine learning algorithms for network intrusion detection using weka," *Towards Extensible and Adaptable Methods in Computing*, vol. 1, no. 1, pp. 195–208, 2018.

[10] A. P. K. da Costa, P. J. Papa and O. C. Lisboa, "Internet of Things: A survey on machine learning-based intrusion detection approaches," *Computer Networks*, vol. 151, no. 1, pp. 147–157, 2019.

[11] J. Yang, J. Deng, S. Li and Y. Hao, "Improved traffic detection with support vector machine based on restricted Boltzmann machine," *Soft Computing*, vol. 21, no. 11, pp. 3101–3112, 2015.

[12] F. Jiang, Y. Fu, B. B. Gupta, F. Lou, S. Rho *et al.,* "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Transactions on Sustainable Computing*, vol. 1, no. 1, pp. 2377–3782, 2018.

[13] K. Peng, V. C. M. Leung, L. Zheng, S. Wang, C. Huang *et al.,* "Intrusion detection system based on decision tree over big data in fog environment," *Wireless Communications and Mobile Computing*, vol. 2018, no. 1, pp. 1–10, 2018.

[14] N. Gao, L. Gao, Q. Gao and H. Wang, "An intrusion detection model based on deep belief networks," in *Proc. IEEE Advanced Cloud and Big Data*, Huangshan, China, pp. 247–252, 2014.

[15] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proc. IEEE Advanced Communication Technology*, Chuncheon-si, Gangwon-do, Korea (South), pp. 178–183, 2018.

[16] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye *et al.,* "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 2018, no. 6, pp. 1792–1806, 2018.

[17] J. Kim, N. Shin, S. Y. Jo and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Big Data and Smart Computing*, Jeju, South Korea, pp. 313–316, 2017.

[18] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," in *Proc. IEEE Emerging Technologies and Factory Automation*, Berlin, Germany, pp. 1–8, 2016.

[19] X. Zhang and J. Chen, "Deep learning based intelligent intrusion detection," in *Proc. IEEE Communication Software and Networks*, Guangzhou, China, pp. 1133–1137, 2017.

[20] P. Illy, G. Kaddoum, C. M. Moreira, K. Kaur and S. Garg, "Securing Fog-to-Things environment using intrusion detection system based on ensemble learning," in *Proc. IEEE Wireless Communications and Networking (WCNC)*, Marrakesh, Morocco, pp. 1–7, 2019.

[21] S. Othman, F. Ba-Alwi1, T. N. Alsohybe1 and Y. A. Al-Hashida, "Intrusion detection model using machine learning algorithm on big data environment," *Journal of Big Data*, vol. 5, no. 1, pp. 1–10, 2018.

[22] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proc. Int. Conf. on Engineering and Technology*, Antalya, Turkey: Akdeniz University, pp. 274–279, 2017.

[23] P. Deepan and L. R. Sudha, "Fusion of deep learning models for improving classification accuracy of remote sensing images," *Journal of Mechanics of Continua and Mathematical Sciences*, vol. 14, no. 1, pp. 189–201, 2019.

[24] J. Yang and J. Li, "Application of deep convolution neural network," in *Proc. IEEE Computer Conf. on Wavelet Active Media Technology and Information Processing*, Chengdu, China, pp. 229–232, 2017.

[25] NSL-KDD, 2009. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[26] KDD Cup 1999, 2018. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/.

[27] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, no. 1, pp. 21954–21961, 2017.

[28] N. Shone, T. N. Ngoc and V. D. Phai, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[29] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng *et al.,* "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," *Measurement*, vol. 154, no. 2, pp. 107450, 2019.