Computers, Materials & Continua
DOI:10.32604/cmc.2022.018708
Article

Tech Science Press

Development of PCCNN-Based Network Intrusion Detection System for EDGE Computing

Mohd Anul Haq, Mohd Abdul Rahim Khan[*] and Talal AL-Harbi

Department of Computer Science, College of Computer and Information Sciences, Majmaah University,
Al-Majmaah 11952, Saudi Arabia
[*]Corresponding Author: Mohd Abdul Rahim Khan. Email: m.khan@mu.edu.sa
Received: 18 March 2021; Accepted: 04 May 2021

Abstract: Intrusion Detection System (IDS) plays a crucial role in detecting and identifying the DoS and DDoS type of attacks on IoT devices. However, anomaly-based techniques do not provide acceptable accuracy for efficacious intrusion detection. Also, we found many difficulty levels when applying IDS to IoT devices for identifying attempted attacks. Given this background, we designed a solution to detect intrusions using the Convolutional Neural Network (CNN) for Enhanced Data rates for GSM Evolution (EDGE) Computing. We created two separate categories to handle the attack and non-attack events in the system. The findings of this study indicate that this approach was significantly effective. We attempted both multiclass and binary classification. In the case of binary, we clustered all malicious traffic data in a single class. Also, we developed 13 layers of Sequential 1-D CNN for IDS detection and assessed them on the public dataset NSL-KDD. Principal Component Analysis (PCA) was implemented to decrease the size of the feature vector based on feature extraction and engineering. The approach proposed in the current investigation obtained accuracies of 99.34% and 99.13% for binary and multiclass classification, respectively, for the NSL-KDD dataset. The experimental outcomes showed that the proposed Principal Component-based Convolution Neural Network (PCCNN) approach achieved greater precision based on deep learning and has potential use in modern intrusion detection for IoT systems.

Keywords: IDS; edge computing; machine learning; NSL-KDD; IoT

1 Introduction

In the past decade, the world has witnessed a rapid growth in smart devices, with a rising focus on the Internet of Things (IoT). All small and big devices such as computers, mobile phones, palmtops, smartwatches, and health bands are connected with the internet. They communicate with each other and form bridges to share information amongst them to perform a task. IoT helps resolve various issues for the users, aiding the development and communication of different kinds of digital devices. They ensure smoother and significantly improved lives with work, learning and entertainment. However, most IoT devices have restricted resources sufficient only for transferring data via the internet on the cloud for processing and storage. In this case, the application IoT devices deal with is cloud computing. Importantly, processing and storage in IoT applications produces massive amounts of data, causing congestion between cloud and IoT devices. Near the IoT devices, [1] EDGE computing ensures temporary data storage and processing, which reduces the volume of information to be delivered on the cloud.

Due to real-time processing, it is vital to obtain a faster response from the system. Industry 4.0, an essential IoT area, reduces the operating system cost and enhances industry usability and reliability. IoT Industrial (IIoT) [2] has machines, actuators, sensors, and digital devices in manufacturing and productions to track the automated units of entire chain industries. In IoT, various information types are captured, processed and transmitted by the internet to produce dispersed solutions. Such information is very private and confidential—an essential element to ensure ethical and reliable activities. Here, one faces a high level of difficulty parallel to the system and requires advanced application security. Considering that the system works together, it increases the number of services.

There is a chance that vulnerabilities also get enhanced. The most common attacks include the capture of information and the engagement of services. For example, Dyn in 2016 [3] described an attack that brought down various services such as GitHub, Netflix, Twitter, and Reddit for many hours. In an IoT environment, safety is essential and crucial. The diversity of attacks requires the security of IoT environments as an essential intellectual investment. IoT faces many threats. In this proposed model, there are two critically important types of attacks related to the application and routing of IoT device packets.

The main concern of this paper is the application-related threat, and our aim is to detect the same. The most common attack is Denial of Services (DoS) [4], which renders the target system unavailable. The primary purpose of attacks is to engage the target and create a flood of traffic, thereby reducing memory resources. In IoT devices, the sensor node is targeted by Distributed Denial of Services (DDoS).

Various types of attacks can occur in this scenario, crippling the main target of the system to gain privileged access [5]. IoT devices suffer from vulnerabilities such as attacks targeting the computing devices, modification, interception, polling, and fabrication. A malicious application targeting IoT devices damages the EDGE layer and cloud. Subsequently, it destructs IoT networks interconnected with similar EDGE nodes. Currently, one of the major challenges is intrusion detection in IoT environments [6] and essential key points are detecting unauthorized user attacks. IDS is a very important tool for identifying attacks on computer networks, providing different types of solutions.

While IoT devices have different features, there is no possibility of applying those solutions in the current scenario. In IoT, applications have network distributed devices with limited memory and computation power capabilities. Inappropriately, these devices are not capable of data integrity, can never be able to defend against malicious attacks and can cause system failure.

Many methods have been applied in IoT to ensure intrusion detection, some of which accomplish anomaly analysis [7,8] on IoT nodes. However, such approaches require high processing capability in the IoT node and more specific hardware. Thus, these methods can only handle views of events. The second method is to analyze the traffic of IoT milieus of the cloud's data [9,10]. The detection approach on the centralized cloud has various limitations such as latency, battery power constraints, and bandwidth.

This article's main thrust is on improving the performance of intrusion detection, in terms of reliability, accuracy, safety and suitability performance, for IoT application-based devices. Our paper presents an intrusion detection method that drives in the EDGE computing layer. Our proposed layer has a more advanced feature than the existing IoT device layer. It can detect specific types of attacks and countermeasure the control, altering the network manager for vulnerabilities.

However, multiclass anomaly-based detection is still not precise enough [11–13]. Hence, in this paper, we present a novel detection method with two groups to handle the attack and non-attack events. In the first step, the binary classification of events is intrusive and non-intrusive. In the next step, events classified as intrusive are grouped to specify the attacks and execute countermeasures. Our approach is relevant and more advanced in this regard.

Earlier works are based on two strategies. The first strategy employs various learning algorithms and high performance [14], whereas the other strategy applies hyper-parameters to achieve a higher level of accuracy [11,15,16]. As machine learning approaches are not adequately accurate, it is essential to use hybrid methods to decrease the uncertainty of the model. However, high-tech hybrid approaches [13,17] have failed to attain the requisite level of accuracy and stability with the testing of the database.

Our proposed hybrid approach incorporates binary classification with high precision, showing detection methods. The first stage has recall and a high accuracy rate, which indicates that most of the possible events are classified as intrusive.

The proposed PCA and CNN-based approach has demonstrated promising results. Although earlier studies have used ML and AI-based techniques for IDS, as per our knowledge, a combination of PCA and CNN has not been applied so far for binary and multiclass classification to defend an IoT environment, ensuring high accuracy. The main contributions of the current research are:

(i) We have proposed a new approach that combined the benefits of PCA for feature collection and the deep learning-based CNN classifiers to ensure effectual and accurate intrusion detection in IoT environments.
(ii) Hybrid PCA-CNN multiclass and binary classification method yielded high accuracy in both training and validation phases.
(iii) EDGE computing-based design ensured intrusion detection to defend within IoT environments.
(iv) The main contribution of this paper is to detect an application-specific threat, e.g., Denial of Services (DoS), which is one of the most common attacks.
(v) The proposed approach is reliable, advanced, accurate, safe, and suitable for detecting intrusions in IoT devices.

CNN models are efficient at solving classification problems through future learning. Also, CNN models get an internal representation of the input. One of the main advantages of CNN models is to get comparable performance from feature learning input in time series data. This model never depends on the domain expertise, and it does not learn the feature input manually. Thus, it is suitable and fits the NSL-KDD dataset.

An input vector in 1-d convolutional takes $a \in R^p$ and a filter $b \in R^q$ where q≤p. In the neighboring set of each q elements aa{i:i+q}, one takes bT {i:i+q} and gives 1 node of the convolutional layer. The multiple filters may have been used by the convolution layer. The vectors (one for each filter) obtained from taking all of these inner products of the weights and every q element of the original vector together form the convolutional layer. In 1-D convolution, the kernel moves 1-direction (time-axis) and is used to calculate convolution. The input contains batch, input width, and input channels; the filter contains width, in channels, and out channels; and the output contains batch, the width of filter, and output channels. E.g., if we have an input(k) of {2, 2, 2, 2, 2} with filter(w) {0.35, 0.30, 0.35} then the output will be {2, 2, 2, 2, 2} of 1D shape. Filters are multiple feature representations of an input. In our experiment, the input is 3D ex) 20x14X7; however, the output shape is 1D matrix instead of 3D because the kernel height is equal to the input height.

A neural network needs an activation function in the output layer to make accurate predictions. The rectifier activation function (ReLU) is one of the default activation functions for deep learning applications; it adds nonlinearity to the network. ReLU output 0 for a negative value and output the same value for non-negative values. Another activation function is the sigmoid or logistic function. Output value of the sigmoid function lies between 0 and 1 and is S-shaped, also having similar values. Sigmoid is the most ideal approach for binary classification, getting the result based on the binomial probability distribution. However, the sigmoid function is not suitable for multiclass classification environments as it needs the multinomial probability distribution for mutual exclusive class. Instead, Softmax is the function used to activate the output layer of the neural network to deal with the multiclass classification problem. This activation function predicts a multinomial probability distribution with more than two classes.

If we have an input of {1, 2, 3}, the max function will output the largest number 3. Argmax will output the index of the largest number, which is 2. Also, the Softmax function, which is the probabilistic or "softer" version of the argmax function in which the unit with the largest input has output +1. In contrast, all other units give the output 0 {0, 0, 1} in the current example.

A 1-D CNN model can have single or multiple convolutional hidden layers that operate on a 1D sequence. Generally, the pooling layer comes after the convolution layer; both layers have the same function to predict the outputs based on optimization of the neural network loss. The convolutional and pooling layers are followed by a dense, fully connected layer that interprets the features of the model's convolutional part. A flattening layer is used between the convolutional layers and the dense layer to reduce the feature maps to a single one-dimensional vector. Pooling divides the vector into equal-sized groups and obtains the summary statistic of each group. After that, it presses out noise in local dynamics. The three pooling types are average, minimum and maximum pooling. The maximum value of the batch is selected.

2 Related Works

The main idea of IoT is the existence of an enormous range of intelligence nodes in our daily social life [18]. It requires state-of-the-art methods of intrusion detection in IoT and EDGE computing networks, emphasizing the import of approaches based on Artificial Intelligence. In IoT, digital devices connected to the internet aim to link everyone with smart IoT applications and create network-distributed environments with limited power capability, storage, and memory. IoT embeds the sensor devices into the internet to share resources and information with other connected devices. However, devices related to IoT networks have limited resources and are vulnerable. Intrusion can take advantage of protection blockades to compromise the integrity, confidentiality, and availability of resources [19].

Intrusion Detection System (IDS) identifies the intrusion action and behaviors, and raises the alarm for the administrator to take automated action [20]. The IDS can detect intrusions as per signature methods. In rule detection, the signature is compared to predefined intrusive events in the database [21]. It ensures immediate detection and reduces false alarms, though it has a significant disadvantage: only known intrusions can be detected [22].

All intrusive activity is considered anomalous by Anomaly detection [13,15–17,23–25]. That is, an activity does not match standard treatment as an intrusion. Anomaly-based detection has a significant advantage in detecting the Zero-day attack and variants of known attacks also. Most of the approaches have applied the traditional environment of machine learning to detect intrusions. Robust methods of anomaly detection use applications based on Artificial Neural Networks (ANN) and Deep Learning (DL). This method ignores the limitation of available classical approaches. ANN's features encourage ANN application in various areas and attempt enhancement in intrusion detection [26–28]. These latest approaches are highly useful for modern computing and EDGE computing [16,29–31]. The summary of related work is presented in Tab. 1.

Table 1: Summary of related works

Method	Type	Merits/Demerit	References
Signature-based approach	Intrusion detection	no issue of jitter and latency	9
SVM	Binary and multiclass classification for intrusion	Average accuracy	11
DNN	Binary and multiclass classification for intrusion	Less accuracy for multiclass classification	12
Anomaly detection	Anomaly based IDS for IoT	Low accuracy	13, 32–34
Deep Learning	DoS detection	Generalize well	14
Deep learning	Intrusion detection	High accuracy; however, issue of single-point failure	16
NN	Multiclass IDS classification	Low accuracy	28
KNN	Binary classification and non-attack classification	Low accuracy and insufficient resource for anomaly detection	29
Binary detection	Intrusion detection	lower accuracy	31
Deep Learning-based MHTS	Android and windows malware attack	Multi-platform	35
Signature-based IDS	Detection of novel attack	Overcome from shortcomings	36
Stacked autoencoders	malware classification and anomaly detection	Semantic similarity works well	39
Self-taught learning	Intrusion detection	Less training time and better accuracy	39
Autoencoder+isolation forest	Intrusion detection	Low accuracy	40
DNN+KNN	Intrusion classification	High accuracy and low overhead	41
Genetic algorithm+Random forest model	Binary classification	Improved accuracy	42

(Continued)

Table 1: Continued

Method	Type	Merits/Demerit	References
PCA+XGBoost+Firefly	Intrusion detection	High accuracy and feature detection	43
CANintelliIDS	Vehicle intrusion	High accuracy	44
Grey Wolf with DL and ML	Feature extraction for intrusion detection	High accuracy	45
SP2F based on LSTM	Attack classification	High accuracy	46

3 Data Pre-Processing

The NSL-KDD dataset is extensively used for intrusion detection [32–36]. There are 41 features in the NSL-KDD dataset (Tab. 2), which can be characterized as int64, float64, and nominal. The attacks were categorized into 23 classes (see Tab. 3 and Fig. 1). This dataset comprised of 3 protocols, including TCP, UDP, and ICMP. A correlation heatmap was generated to understand the relationship between the 41 features (Fig. 2). In the pre-processing step, the nominal attributes were converted into discrete attributes using a one-hot encoder (Tab. 3). There were no data gaps in the training and testing dataset; however, one attribute (num_out bound_cmds) column was zero throughout and did not have any significance on training and testing. Therefore, it was removed from the attribute lists. All the attributes were then normalized in the range of {0, 1} by applying max-min scaling. After pre-processing steps, the number of the attributes was expanded from 40 to 119.

Table 2: NSL-KDD dataset features

S. No.	Features	No. of records	Data type
1	Duration	148517	int64
2	protocol type	148517	Object
3	Service	148517	object
4	Flag	148517	Object
5	src bytes	148517	int64
6	dst bytes	148517	int64
7	Land	148517	int64
8	Wrong fragment	148517	int64
9	Urgent	148517	int64
10	Hot	148517	int64
11	num failed logins	148517	int64
12	logged in	148517	int64
13	num compromised	148517	int64
14	Root shell	148517	int64
15	su attempted	148517	int64
16	num root	148517	int64
17	num file creations	148517	int64

(Continued)

Table 2: Continued

S. No.	Features	No. of records	Data type
18	num shells	148517	int64
19	num access files	148517	int64
20	num outbound cmds	148517	int64
21	is host login	148517	int64
22	is guest login	148517	int64
23	Count	148517	int64
24	siv count	148517	int64
25	setror rate	148517	float64
26	srv seiror rate	148517	float64
27	reiror rate	148517	float64
28	siv mrror rate	148517	float64
29	same srv rate	148517	float64
30	diff srv rate	148517	float64
31	srv diff host rate	148517	float64
32	dst host count	148517	int64
33	dst host srv count	148517	int64
34	dst host same srv- rate	148517	float64
35	dst host diff srv rate	148517	float64
36	dst host same src_port rate	148517	float64

Table 3: Classification of traffic proportions

Class	Label	Traffic proportions
back	0	956
buffer_overflow	1	30
ftp_write	2	8
guess_passwd	3	53
imap	4	11
ipsweep	5	3599
land	6	18
loadmodule	7	9
multihop	8	7
neptune	9	41214
nmap	10	1493
normal	11	67343

(Continued)

Table 3: Continued

Class	Label	Traffic proportions
perl	12	3
phf	13	4
pod	14	201
portsweep	15	2931
rootkit	16	10
satan	17	3633
smurf	18	2646
spy	19	2
teardrop	20	892

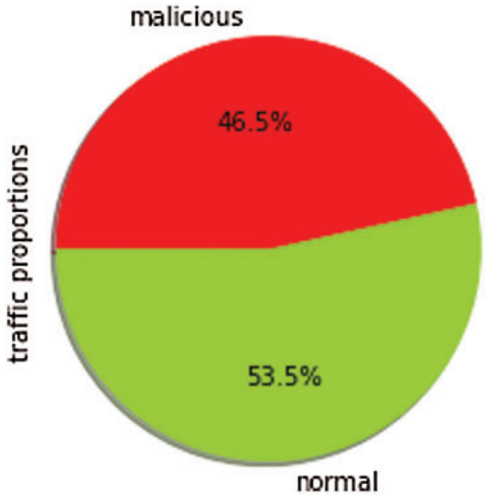

Figure 1: Malicious traffic *vs.* normal traffic

We observed six feature vector values (land, urgent, root_shell, su_attempted, num_shells, is_host_login) to be close to zero. Therefore, applying PCA was a feasible attempt to reduce the feature vector's size based on feature extraction and engineering (Fig. 3). The features were extracted and transformed into principal components, and statistics were computed. The 3-dimensional scatterplot captured only a small portion of the information (Fig. 3). It would take more principal components to capture meaningful information. We could make a few more 3d scatterplots with other principal components, but again, as seen in the explained variance graphs (Fig. 4), these could offer far less insight than the first three components.

The PCA process involves finding the mean, covariance matrix with eigenvectors and eigenvalues, selecting principal components with the higher eigenvalues, and multiplying with the actual (original) data matrix. An important part was to estimate the number of principal components required to describe the data. It was then described based on the CEVR (Cumulative Explained Variance Ratio) as a function of the number of principal components (Fig. 5). Time series graph is given in (Fig. 6).

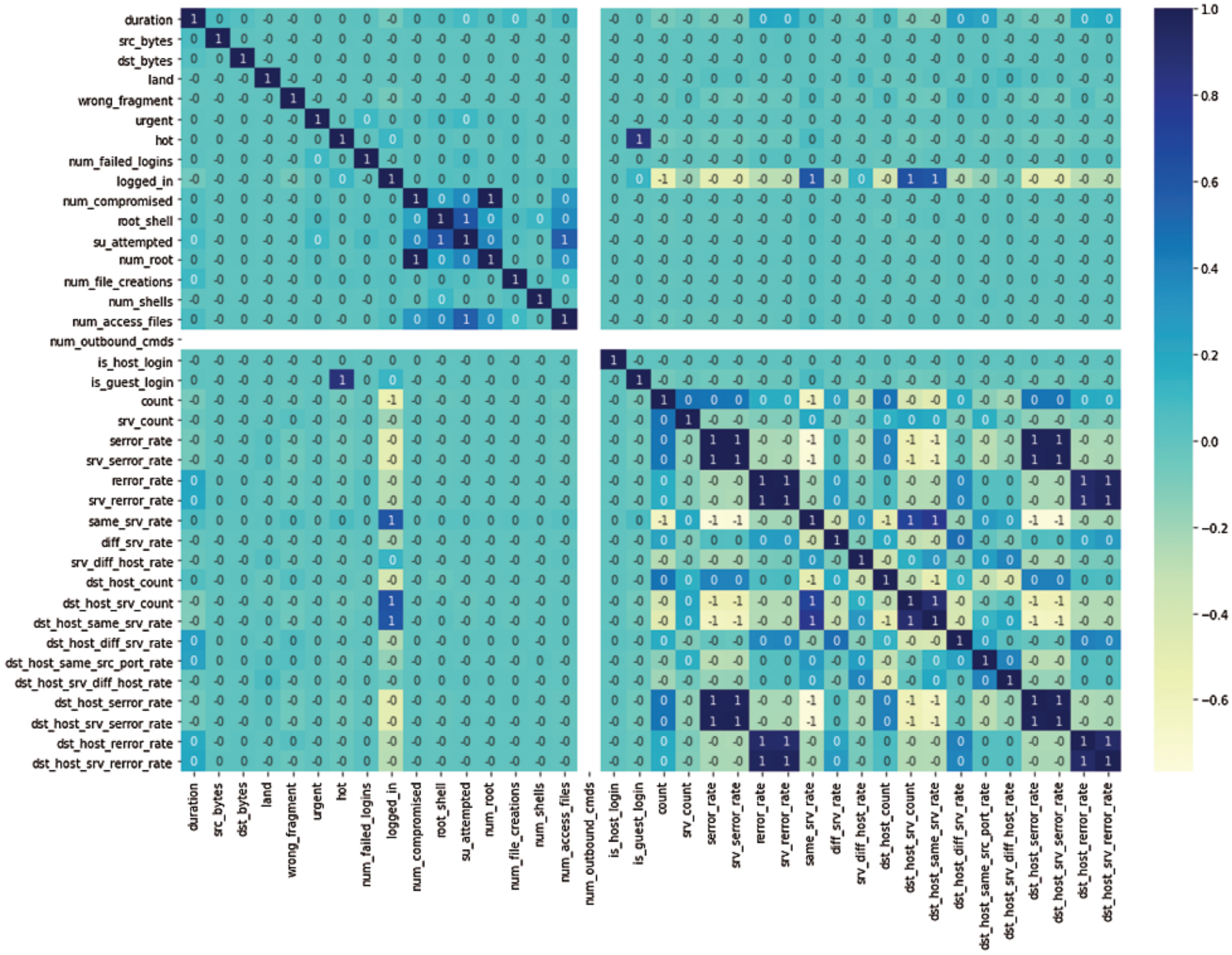

Figure 2: Correlation heatmap between all features

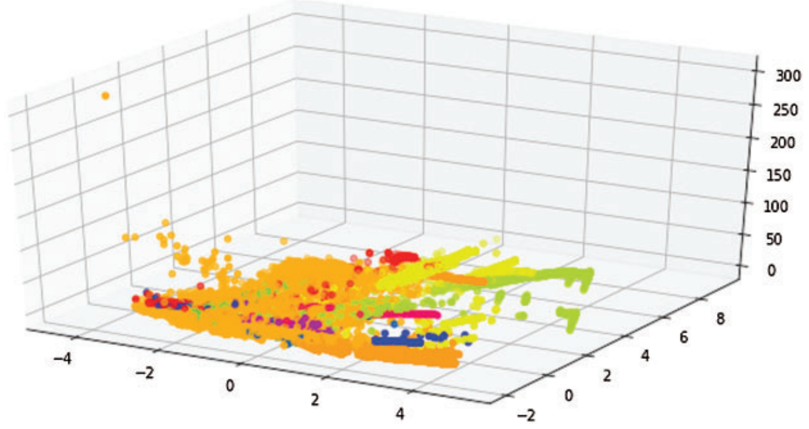

Figure 3: PCA visualization

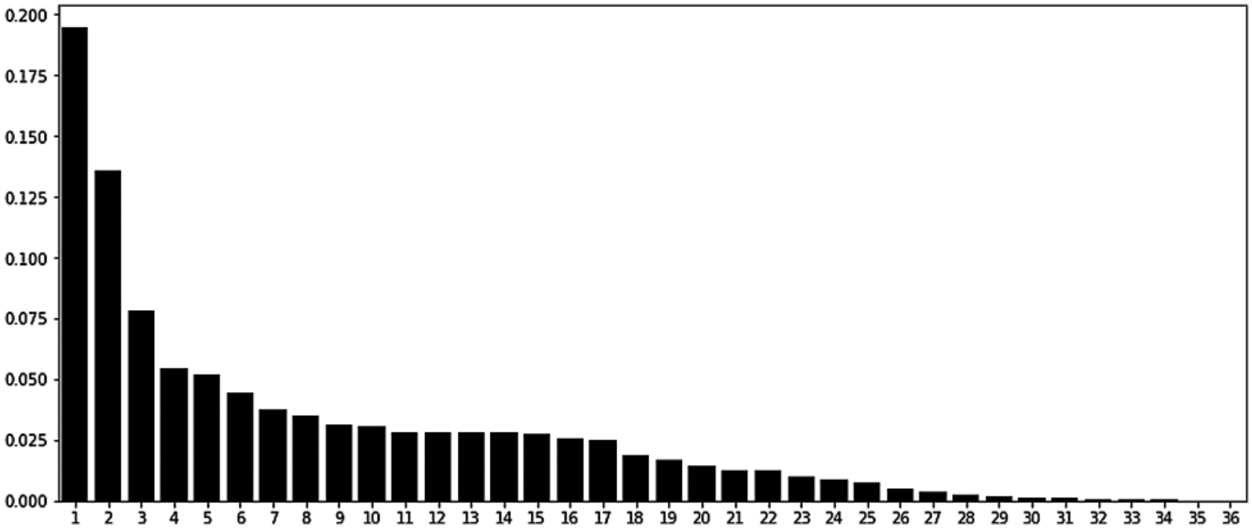

Figure 4: Explained variance ratio

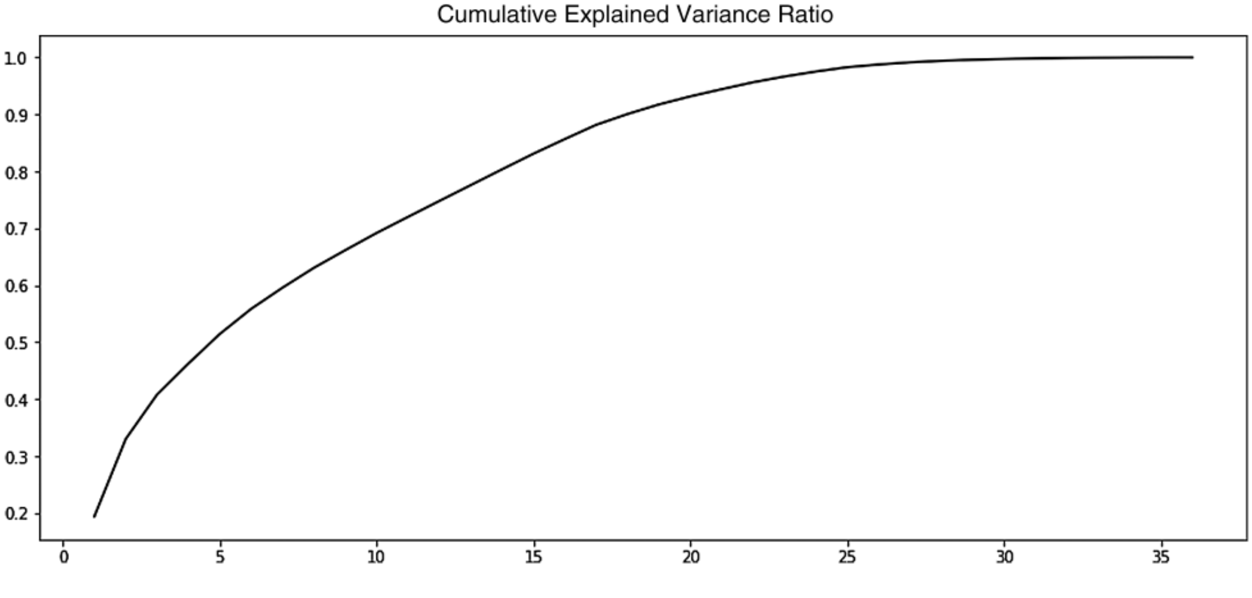

Figure 5: The CEVR curve

The CEVR curve estimated the total, and the 34-dimensional variance was within the first N components. It was observed that the initial 10 components contained around 70% of the variance; however, it required 24 components to describe approximately 100% of the variance. It was also observed that this 2-D projection dropped considerably high information (based on CEVR), and to describe 90% of the variance, it required total 20 components. The CEVR helped understand the presence of ambiguity for a high-dimensional dataset.

4 Methodology

4.1 CNN Modeling

We attempted both multiclass and binary classification. For binary, we clustered all malicious traffic data in a single class. We developed 13 layers of Sequential 1-D Convolutional Neural Network for IDS detection trained on the NSL-KDD dataset (Fig. 7). Python 3.8 and Keras 2.3.0 API with Tensorflow 2.0 backend were used on a single GPU (i9, 10900k, 128 GB 2666 MHz RAM) in this research. Firstly, we carried out the data pre-processing (IDS data). We used two multiple convolutional hidden layers that operated on a 1D sequence. The batch normalization layer was used after the convolution layer to standardize the inputs with mean value and standard deviation as 0 and 1, respectively, to each mini-batch layer. The batch normalization layer functioned to stabilize the process of training and decrease the number of training epochs needed to train the deep CNN networks. The rectifier activation function (ReLU) was used in convolution layers. The fourth layer was a max-pooling layer, followed by a fifth dropout layer. The dropout layer was added between two convolution layers, and outputs of the prior layer were fed to the subsequent layer to prevent overfitting. This worked by "dropping out" or probabilistically removing inputs to a layer, which may be input variables from a previous layer.

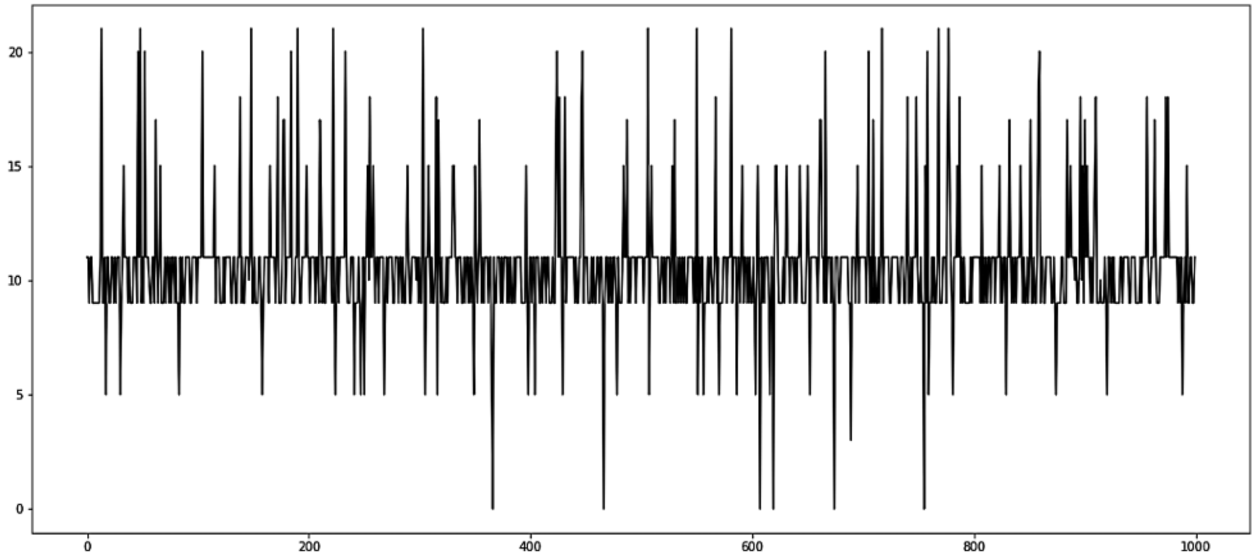

Figure 6: Traffic time series data: only 1000 values (x-axis) were selected for better visual clarity, labeled class (y-axis)

A value of 0.5 was chosen with two dropout layers. Layer 2 to layer 5 was repeated as layer 6 to 9. A flattening layer was added as the 10th layer, as it was required to utilize the fully connected layers after convolutional/max-pool layers. The flattening layer combined all the observed local features of the previous convolutional layers. First fully-connected dense layers (11th) and second fully-connected LVQ layers were added, and then layer 11 and 13 were separated by a dropout (12th) layer. The dense layers acted as an artificial neural network (ANN) classifier. In the proposed architecture, the 13th layer, which was a LVQ layer, was used as the output layer to predict and specify the output's transformation and structure.

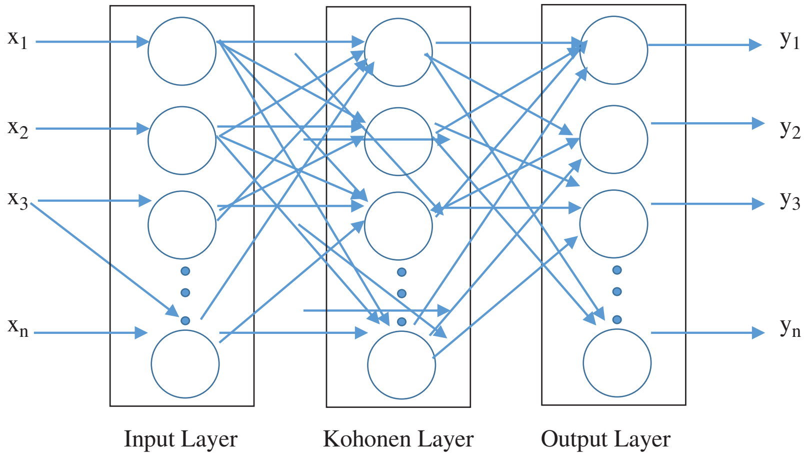

Figure 7: Architecture of LVQ

The model compilation was the next step after adding the layers. Compilation requires an optimizer, a loss function, and a metric function to evaluate the model accuracy. The Adam optimization algorithm was used for optimization, which is an extension of stochastic gradient descent and has many benefits such as fewer memory requirements and faster and straightforward computation. The binary cross-entropy loss function was utilized to compute the rate of error between the actual and the m values for binary classification, such as

$$\text{Loss} = -\frac{1}{O_s}\sum_{i=0}^{O_s} a_i \cdot \log \hat{t}_i + (1 - a_i) \cdot \log(1 - \hat{t}_i) \tag{1}$$

where O_s *is the* output size, a_i and $\hat{t}_i$ are the target output values, respectively.

The metric function "accuracy" and F-1 measure were used to evaluate our model's performance. This metric function is similar to the loss function, except that the metric evaluation results are not used when training the model. During the training process, the weights in the CNNs are optimized to improve the accuracy; however, these improvements attempt to correlate positively with the number of runs and reach a point where overfitting takes place and results in lower generalization performance.

Dropouts were used efficiently to reduce the overfitting; however, the early stopping technique was also added during CNN model fitting. The early stopping technique was implemented using the tf.keras. Early Stopping callback function: In CNN classification, the 14th epoch resulted in better training accuracy but lower validation accuracy than the previous (13th). Thus, the training was terminated at the 14th epoch, notwithstanding that the number of maximum epochs was set to 30. The data was reshaped to 3-dimensional so that it could be fed to CNN. The training and testing size were 100778 and 25195 for training and testing, respectively (Tab. 4).

4.2 Convolution Neural Network with Learning Vector Quantization (CNN-LVQ) Algorithm

Learning Vector Quantization (LVQ), a well-established heuristic technique, was utilized to assimilate CNN. The LVQ layer was added as a second fully connected layer in the proposed CNN-LVQ model. LVQ is primarily a 3-layer neural network that utilizes competitive and supervised learning to solve classification problems. The three layers include the input layer, the Kohonen layer (or competition layer) and the output layer. The input layer neurons collect the values from the input variables, while each neuron of the output layer represents a class of input. The Kohonen and output layers are connected partially, while the input layer and the Kohonen layer are fully connected. The learning takes place in the Kohonen layer, and the classification results are passed to the output layer. The LVQ architecture is shown in Fig. 7. Complete network is given in Fig. 8.

Table 4: Number of trainable and non-trainable parameters for both binary and multiclass classifications

Parameters	Binary classification	Multiclass classification
Total params	1,018,241	2,344,127
Trainable params	1,017,473	2,343,167
Non-trainable params	768	960

In the proposed method, weighting parameters were selected by using the LVQ technique for classification. In LVQ, the first step was setting the initial synaptic weight for random values with the interval between 0 and 1. Then the learning rate of 0.01 was used, and the input vector a_i was initiated with the random value.

If the class label a_i and the weight vector ΔVW are close, the LVQ will move in the direction of a_i. If labels have different average values. for class a_i and weight vector ΔVW, the LVQ will move away from a_i. Assuming that the weight vector in parallel $\Delta VW(t)$ is close to the input vector x_i, the mathematical equation representation is as follows for ΔVW Eq. (2).

$$\Delta VW(t) = argmin||a_i - \Delta VW(t)||_2 \tag{2}$$

Let the class of $\Delta VW(t)$ be denoted as CVW and the class of a_i be stated as Ca_i. The weight vector $\Delta VW(t)$ is adjusted as follows: if both classes are similar $CVW = Ca_i$, then ANN parameters are represented as Eq. (3).

$$\Delta VW(t+1) = VW(t) + \eta(t)(a_i - VW(t)) \tag{3}$$

where $\eta(t)$ is denoted as the learning rate of the adaptation procedure. If both classes are different $CVW \neq Ca_i$, then LVQ is denoted by Eq. (4).

$$\Delta VW(t+1) = VW(t) - \eta(t)(a_i - VW(t)) \tag{4}$$

Based on the condition, either Eqs. (3) or (4) can be used to update the weighting function in ANN. Using the weighting function, the output of the predicted value is determined by Eq. (5).

$$b_i = f(a_i \times VW(t+1)) \tag{5}$$

The output b_i classifies the NSL-KDD threats for multiclass and binary classification.

5 Accuracy Assessment

We evaluated our model's performance based on accuracy and f-measure as the new version of Keras removed recall and precision (Fig. 9). These metrics are defined as follows: *Accuracy* of a method on a test dataset is the percentage used to correctly identify the test occurrences and it is computed as

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{TN} + \text{FN}) \tag{6}$$

F-measure was applied to obtain the testing accuracy, which is computed based on the harmonic mean of the precision and recall:

$$\text{F-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

An attempt was made to see if the models were overfitted. Overfitting can be detected if training loss is comparatively less than validation loss or there is a significant variance between the validation and training loss. It was observed that the variance between validation loss and training loss was significantly lower; therefore, it indicated that underfitting was absent. The dropouts were also utilized to prevent the overfitting issues. The main features of dropout were to disable neurons so that some information loss might occur for each sample, and the next layers attempt to construct the representation based on incomplete representations. It was observed that the training loss was higher since it was harder for the network to provide the correct representation. However, all of the units were available during validation so that the network could utilize its full computational power-therefore, it could perform better than in training. The training accuracy and validation accuracy for both binary and multiclass classification were also significantly promising.

The proposed method showed a high accuracy of 99.13% for multiclass classification and 99.34% for binary classification. Accuracy is useful based on the consideration of true positives and true negatives, while F1-measure uses the false negative and false positive, which are crucial to assess any model's performance. In real-world problems, due to imbalanced class data, F1-score provides a better metric to evaluate a model. In this study, the F-measure values were 99.27% and 99.78% for multiclass and binary classification, respectively.

6 Comparison with Other Studies

Intrusion detection is a highly investigated topic, and the previous body of works provides an opportunity to compare our results with similar studies. Previous studies have employed algorithms ranging from conventional machine learning such as Random Forest and SVM to sophisticated Deep Learning methods such as CNN and RNN. A comparative analysis was made in this section with other contemporary intrusion detection methods based on machine learning and deep learning techniques with reference to the accuracy observed for the NSL-KDD dataset. Tab. 5 advocated that methods based on Deep Neural Network (DNN) and CNN showed significantly better performance than other methods, based on the accuracy values.

A major limitation of the present method is the computing time required to calculate the CNN models, which is an important parameter for any real-time solution. The future scope of the proposed model is to reduce the computation time by adding an extra pre-processing technique, utilizing GANs (Generative Adversarial Networks) for IDS in FOG or EDGE environments and using many available datasets. Another important way forward will be to use mobile CNN for IoT infrastructure as it requires real-time solutions with fewer computing requirements.

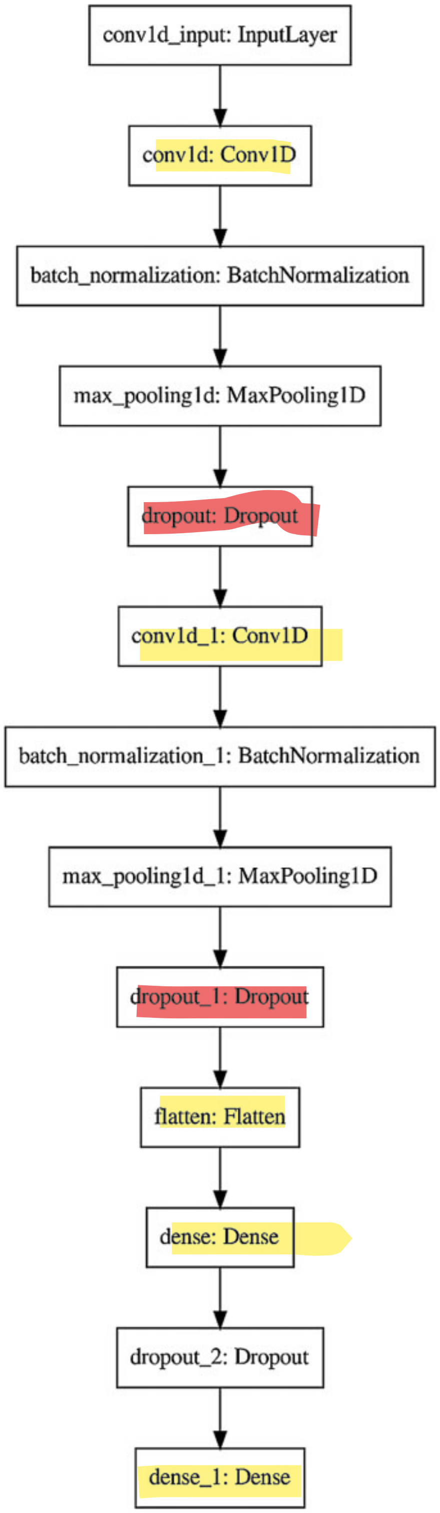

Figure 8: Proposed CNN architecture

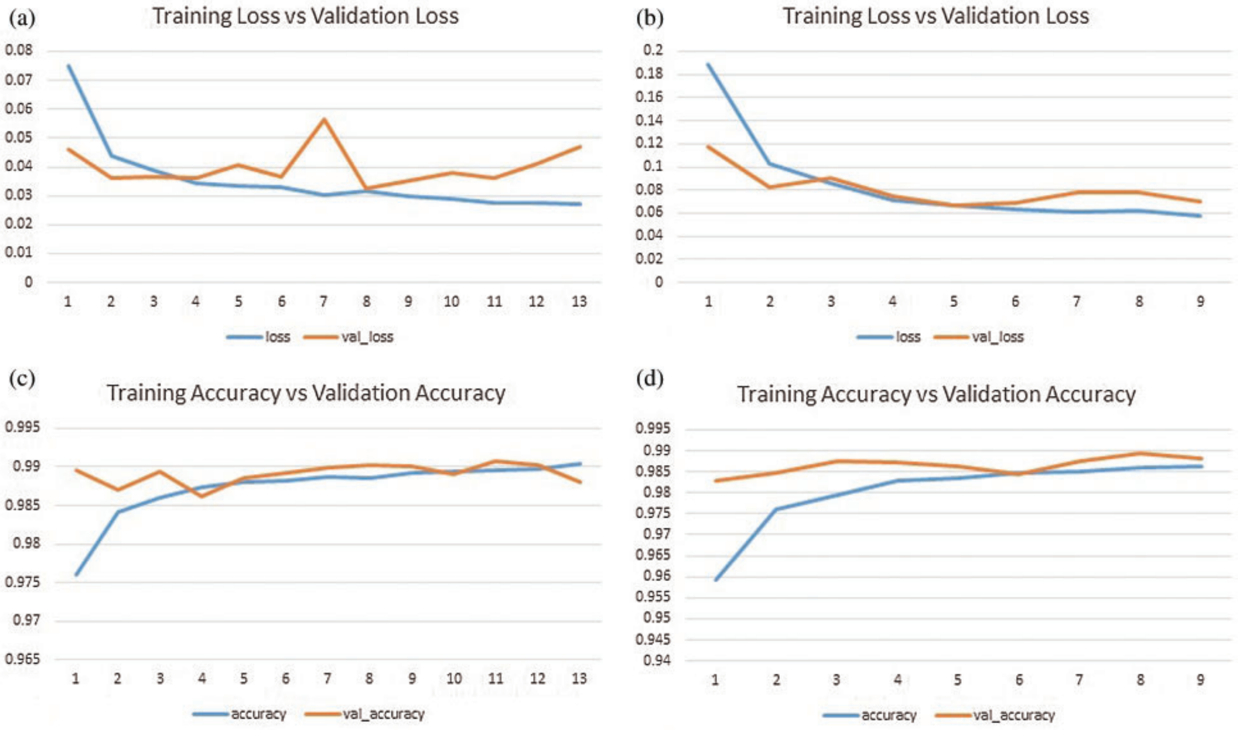

Figure 9: CNN model performance (a) Training loss and validation loss for binary classification, (b) Training loss and validation loss for multiclass classification, (c) Training accuracy and validation accuracy for binary classification, (d) Training accuracy and validation accuracy for multiclass classification

Table 5: Comparison of different deep learning and machine learning-based techniques used for IDS based on the NSL-KDD dataset

No.	Method	Accuracy (%)	Type
1	Naïve Bayes [36]	75.56	DL
2	Multi-layer perceptron [36]	77.41	DL
3	Random forest [36]	80.67	ML
4	NB-Tree [36]	82.02	ML
5	Recurrent neural network [37]	83.28	DL
6	Autoencoder and Naïve Bayes [38]	83.34	DL
7	Fuzzy approach [27]	84.12	ML
8	Self-taught learning + SVM [39]	84.96	DL
9	Autoencoder and Isolation Forest [40]	95	DL
10	DAE+DNN [35]	98.6	DL

(Continued)

Table 5: Continued

No.	Method	Accuracy (%)	Type
11	Current method Multiclass Classification)	99.13	DL
12	Current method (Binary Classification)	99.34	DL
13	DNN+KNN [41]	99.77	DL
14	Genetic Algorithm+Random forest model [42]	97.20	ML
15	KNN+PCA+Firefly [43]	99.40	ML
16	Naïve Bayes-PCA and Firefly [43]	84.20	ML
17	Random Forest+PCA+Firefly [43]	99.80	ML
18	SVM+PCA+Firefly [43]	97.5	ML
19	CANintelliIDS (Known intrusion) [44]	97.54	DL
20	CANintelliIDS (Test intrusion) [45]	93.94	DL
21	DNN+PCA+GW [46]	99.90	DL
22	KNN+PCA+GW [46]	99.80	ML
23	NB+PCA+GW [46]	86.20	ML
24	RF+PCA+GW [46]	99.90	ML
25	SVM+PCA+GW [46]	98.20	ML
26	SLSTM for DoS [47]	99.54	DL
27	SLSTM for DDoS [47]	98.87	DL

7 Conclusion

IDS (Intrusion Detection System) is essential for detecting and identifying attacks on IoT devices. In this investigation, a solution was designed based on PCA and CNN (Convolutional neural network) to detect intrusion in EDGE Computing. Two categories were proposed to handle the attack and non-attack events in the system. The current investigation demonstrated a benchmark for both multiclass and binary classification. We developed 13 layers of sequential 1-D CNN for IDS detection trained on the NSL-KDD dataset. PCA was implemented before applying CNN to reduce the feature vector's size based on feature extraction and engineering. The proposed approaches applied on the NSL-KDD dataset demonstrated the accuracy values of 99.34% and 99.13% for the binary classification and the multiclass classification, respectively. The experimental results showed that the proposed PCCNN approach achieved greater precision based on deep learning and can be used for current advancements in intrusion detection for EDGE systems. The present investigation's future scope is to apply GANs (Generative Adversarial Networks) for IDS in FOG or EDGE environments and utilize other available datasets.

Funding Statement: Mohd Anul Haq would like to thank the Deanship of Scientific Research at Majmaah University for supporting this work under Project No. R-2021-117.

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

References

[1] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proc. First Edition of the MCC Workshop on Mobile Cloud Computing ACM*, Helsinki Finland, pp. 13–16, 2012.

[2] A. C. Panchal, V. M. Khadse and P. N. Mahalle, “Security issues in IIoT: A comprehensive survey of attacks on IIoT and its countermeasures,” in *2018 IEEE (GCWCN)*, Lonavala, India, pp. 124–130, 2018.

[3] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, “DDoS in the IoT: Mirai and other botnets,” *Computer,* vol. 50, no. 7, pp. 80–84, 2017.

[4] J. Arshad, M. A. Azad, M. Mahmoud Abdellatif, M. H. Ur Rehman and K. Salah, “COLIDE: A collaborative intrusion detection framework for internet of things,” *IET Networks,* vol. 8, no. 1, pp. 3–14, 2019.

[5] F. Muhammad, W. Anjum and K. S. Mazhar, “A critical analysis on the security concerns of internet of things (IoT),” *International Journal of Computer Applications,* vol. 111, no. 7, pp. 1–6, 2015.

[6] R. Roman, J. Lopez and M. Mambo, “Mobile edge computing, a survey and analysis of security threats and challenges,” *Future Generation Computer System,* vol. 78, pp. 680–698, 2018.

[7] N. Berjab, H. H. Le, C. Yu, S. Kuo and H. Yokota, “Hierarchical abnormal-node detection using fuzzy logic for ECA rule-based wireless sensor networks,” in *Proc. 2018 IEEE 23rd Pacific Rim Int. Symp. on Dependable Computing*, Taipei, Taiwan, pp. 289–298, 2018.

[8] F. De Almeida Florencio, E. D. Moreno Ordonez, H. Teixeira Macedo, R. J. Paiva De Britto Salgueiro, F. Barreto Do Nascimento *et al.*, “Intrusion detection via MLP neural network using an arduino embedded system,” in *Proc. 2018 VIII SBESC*, Salvador, Brazil, pp. 190–195, 2018.

[9] M. Rebbah, D. El Hak Rebbah and O. Smail, “Intrusion detection in cloud internet of things environment,” in *Proc. ICMIT*, Adrar, Algeria, pp. 65–70, 2017.

[10] B. Hajimirzaei and N. J. Navimipour, “Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm,” *ICT Express,* vol. 5, no. 1, pp. 56–59, 2019.

[11] X. Tang, S. X. D. Tan and H. B. Chen, “SVM based intrusion detection using nonlinear scaling scheme,” in *Proc. 14th IEEE ICSICT, IEEE*, Qingdao, China, pp. 1–4, 2018.

[12] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat *et al.*, “Deep learning approach for intelligent intrusion detection system,” *IEEE Access,* vol. 7, pp. 41525–41550, 2019.

[13] H. H. Pajouh, R. Javidan, R. Khayami, D. Ali and K. K. R. Choo, “A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks,” *IEEE Transactions on Emerging Topics in Computing,* vol. 7, no. 2, pp. 314–323, 2019.

[14] M. Roopak, G. Yun Tian and J. Chambers, “Deep learning models for cyber security in IoT networks,” in *Proc. IEEE 9th Annual CCWC*, Las Vegas, NV, USA, pp. 452–457, 2019.

[15] S. Prabavathy, K. Sundarakantham and S. M. Shalinie, “Design of cognitive fog computing for intrusion detection in internet of things,” *Journal of Communications and Networks,* vol. 20, no. 3, pp. 291–298, 2018.

[16] A. A. Diro and N. Chilamkurti, “Distributed attack detection scheme using deep learning approach for internet of things,” *Future Generation Computer Systems,* vol. 82, pp. 761–768, 2018.

[17] M. E. Pamukov, V. K. Poulkov and V. A. Shterev, “Negative selection and neural network based algorithm for intrusion detection in IoT,” in *2018 41st Int. Conf. on Telecommunications and Signal Processing*, Athens, Greece, IEEE, pp. 1–5, 2018.

[18] L. Atzori, A. Iera and G. Morabito, “The internet of things: A survey,” *Computer Networks,* vol. 54, no. 15, pp. 2787–2805, 2010.

[19] R. Heady, G. Luger, A. Maccabe and M. Servilla, “The architecture of a network level intrusion detection system,” Technical Report, University of New Mexico, Department of Computer Science, 1990.

[20] R. Bace and P. Mell, "NIST special publication on intrusion detection systems," Technical Report, Booz-Allen and Hamilton Inc Mclean Va, 2001.

[21] J. Arshad, M. A. Azad, M. Mahmoud Abdellatif, M. H. Ur Rehman and K. Salah, "COLIDE: A collaborative intrusion detection framework for internet of things," *IET Network,* vol. 8, no. 1, pp. 3–14, 2019.

[22] S. Northcutt, M. Cooper, M. Fearnow and K. Frederick, "*Intrusion Signatures and Analysis,*" New Riders Publishing, New Jersey, 2001.

[23] M. G. Raman, N. Somu, K. Kirthivasan, R. Liscano and V. S. Sriram, "An efficient intrusion detection system based on hypergraph-genetic algorithm for parameter optimization and feature selection in support vector machine," *Knowledge-Based Systems,* vol. 134, pp. 1–12, 2017.

[24] K. Atefi, H. Hashim and T. Khodadadi, "A hybrid anomaly classification with deep learning (DL) and binary algorithms (ba) as optimizer in the intrusion detection system (IDS)," in *2020 16th IEEE CSPA*, Langkawi, Malaysia, IEEE, pp. 29–34, 2020.

[25] Y. Zhong, W. Chen, Z. Wang, Y. Chen, K. Wang *et al.*, "A novel network anomaly detection model based on heterogeneous ensemble learning," *Computer Network,* vol. 169, pp. 107049, 2020.

[26] K. Vieira, A. Schulter, C. Westphall and C. Westphall, "Intrusion detection for grid and cloud computing," *IT Professional,* vol. 12, no. 4, pp. 38–43, 2009.

[27] R. A. R. Ashfaq, X. Z. Wang, J. Z. Huang, H. Abbas and Y. L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences,* vol. 378, pp. 484–497, 2017.

[28] M. G. Raman, N. Somu, K. Kirthivasan and V. S. Sriram, "A hypergraph and arithmetic residue-based probabilistic neural network for classification in intrusion detection systems," *Neural Network,* vol. 92, pp. 89–97, 2017.

[29] L. Li, Y. Yu, S. Bai, Y. Hou and X. Chen, "An effective two-step intrusion detection approach based on binary classification and k-nN," *IEEE Access,* vol. 6, pp. 12060–12073, 2017.

[30] J. Ni, K. Zhang, X. Lin and X. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys & Tutorials,* vol. 20, pp. 601–628, 2018.

[31] G. D'angelo, F. Palmieri, M. Ficco and S. Rampone, "An uncertainty-managing batch relevance-based approach to network anomaly detection," *Applied Soft Computing,* vol. 36, pp. 408–418, 2015.

[32] P. Illy, G. Kaddoum, C. Miranda Moreira, K. Kaur and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," in *Proc. 2019 IEEE Wireless Communications and Networking Conf.*, Marrakesh, Morocco, pp. 1–7, 2019.

[33] Y. Otoum and A. Nayak, "AS-Ids: Anomaly and signature based ids for the internet of things," *Journal of Network and Systems Management,* vol. 29, no. 3, pp. 1–26, 2021.

[34] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," *Simulation Modelling Practice and Theory,* vol. 101, pp.1 02031, 2020.

[35] M. AL-Hawawreh, N. Moustafa and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications,* vol. 41, pp. 1–11, 2018.

[36] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2009 IEEE Symp. on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, pp. 1–6, 2009.

[37] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access,* vol. 5, pp. 21954–21961, 2017.

[38] M. Yousefi-Azar, V. Varadharajan, L. Hamey and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. on Neural Networks*, Anchorage, AK, United States, pp. 3854–3861, 2017.

[39] M. Al-Qatf, Y. Lasheng, M. Al-Habib and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access,* vol. 6, pp. 52843–52856, 2018.

[40] K. Sadaf and J. Sultana, “Intrusion detection based on autoencoder and isolation forest in fog computing,” *IEEE Access,* vol. 8, pp. 167059–167068, 2020.

[41] C. A. de Souza, C. B. Westphall, R. B. Machado, J. B. M. Sobral and G. D. S. Vieira, “Hybrid approach to intrusion detection in fog-based IoT environments,” *Computer Network,* vol. 180, pp. 1–18, 2020.

[42] A. Adel, “Anomaly classification using genetic algorithm-based random forest model for network attack detection,” *Computers, Materials & Continua,* vol. 66, pp. 767–778, 2021.

[43] S. Bhattacharya, S. R. Krishnan, P. K. R. Maddikunta, R. Kaluri, S. Singh, T. R. Gadekallu *et al.*, “A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU,” *Electronics,* vol. 9, no. 2, pp. 1–16, 2020.

[44] A. Rehman, S. Ur Rehman, M. Khan, M. Alazab and T. R. G, “CANintelliIDS: Detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU,” *IEEE Transactions on Network Science and Engineering,* vol. 4697, no. c, pp. 1–11, 2021.

[45] R. M. S. Priya, P. K. R. Maddikunta, M. Parimala, S. Koppu *et al.*, “An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture,” *Computer Communications,* vol. 160, no. 2, pp. 139–149, 2020.

[46] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, T. R. Gadekallu *et al.*, “SP2F: A secured privacy-preserving framework for smart agricultural unmanned aerial vehicles,” *Computer Networks,* vol. 187, no. 1, pp. 1–17, 2021.

[47] M. D. Hossain, H. Ochiai, D. Fall and Y. Kadobayashi, “LSTM-Based network attack detection: Performance comparison by hyper-parameter values tuning,” in *Proc. EdgeCom*, New York, NY, USA, pp. 62–69, 2020.