*Computers, Materials & Continua*
DOI:10.32604/cmc.2022.020938
***Article***

Tech Science Press

# DNNBoT: Deep Neural Network-Based Botnet Detection and Classification

**Mohd Anul Haq and Mohd Abdul Rahim Khan***

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: 15 June 2021; Accepted: 30 August 2021

**Abstract:** The evolution and expansion of IoT devices reduced human efforts, increased resource utilization, and saved time; however, IoT devices create significant challenges such as lack of security and privacy, making them more vulnerable to IoT-based botnet attacks. There is a need to develop efficient and faster models which can work in real-time with efficiency and stability. The present investigation developed two novels, Deep Neural Network (DNN) models, DNNBoT1 and DNNBoT2, to detect and classify well-known IoT botnet attacks such as Mirai and BASHLITE from nine compromised industrial-grade IoT devices. The utilization of PCA was made to feature extraction and improve effectual and accurate Botnet classification in IoT environments. The models were designed based on rigorous hyperparameters tuning with GridsearchCV. Early stopping was utilized to avoid the effects of overfitting and underfitting for both DNN models. The in-depth assessment and evaluation of the developed models demonstrated that accuracy and efficiency are some of the best-performed models. The novelty of the present investigation, with developed models, bridge the gaps by using a real dataset with high accuracy and a significantly lower false alarm rate. The results were evaluated based on earlier studies and deemed efficient at detecting botnet attacks using the real dataset.

**Keywords:** Botnet; network monitoring; machine learning; deep neural network; IoT threat

## 1 Introduction

The expansion of the Internet of Things (IoT) network and its applications have risen enormously due to upgrading communication efficiency, low cost, and ever-increasing demand. The IoT devices have been developed and utilized for numerous sectors, including smart cities, smart grid, smart manufacturing and maintenance, intelligent transport, security and surveillance, precision agriculture, utilities such as power, electricity and water, supply chain, and inventory optimization, more. Over the past few years, the number of sensor-based smart devices that can communicate over the internet without human involvement is growing exponentially. It will be reaching around 30 billion by 2050 [1]. However, the massively increasing numbers and global presence of IoT have become an opportunity for hackers to exploit the security and privacy of the IoT network by using anomalous entities such as botnets, as IoT infrastructure still lacks robust security [2]. The primary security challenge in infrastructure is botnet-based attacks where illegitimate users inject malicious scripts into the IoT devices to infect them.

### *1.1 Botnet*

The compromised IoT devices do not show any indications of being hacked and act like zombies for the botmaster to launch the attacks. The size of the botnets might be smaller with hundreds of bots, to a larger botnet contains thousands of bots. Some bots are available on the dark web as cheap as 0.5$ per bot to a massive collection of botnets with a high price. Botnets are of two types, (1) botnets taking commands and in continuous communication with botmaster in a client-server architecture (2) peer to peer bots, which communicated autonomously with each other and launch the attacks after receiving commands from the botmaster. Botmaster used to communicate with bots using the help command-and-control (CnC) server; the bots used to hide until commands from botmaster; this hidden behavior of bots make identification of infected bots and botnet attack a complex task.

### *1.2 Type of Botnet Attacks*

The attacks class are: (1) the scan commands used to find out the vulnerable IoT devices; (2) ACK, SYN, UDP, and TCP flooding; and (3) combo or combination attacks used to open a connection and to transmit the spam to it [3].

#### *1.2.1 Scan Attack*

Botmaster scan IoT device in the network to collect information, including IP scanning, port scanning, etc.

#### *1.2.2 DDoS Attack*

The DDoS attack is one of the major cyberattacks where a hacker sends massive traffic or flooding to the target server from different locations, which results in disruption of the service and no service to legitimate users [4,5]. Botmaster launches the DDoS attack with the botnet, which exhausted the victim server or platform in memory, computing, and resource disruption.

#### *1.2.3 TCP Flooding*

TCP SYN flood is a DDoS attack where the botmaster sends faster TCP syn traffic to exhaust the target's resources and make it unavailable for legitimate requests.

#### *1.2.4 ACK Flooding*

The botmaster sends massive fake acknowledgments in ack flooding to the target server that fake IoT devices received the transmitted data successfully.

#### *1.2.5 UDP Flooding*

In a UDP flood attack, the botmaster sends UDP packets in a massive amount to exhaust the target server or device to lose processing and respond. In addition, the protecting firewall is also exhausted due to the UDP flood, which rejects legitimate requests.

## 2 Previous Studies

Botnet refers to a robot network, which means a network of various bots; aimed to perform unlawful activities against any type of IT infrastructure, including websites or networks. Botnets are controlled by botmaster or herder or cybercriminals using command and control protocol [6]. Therefore, a botnet is one of the significant issues for the security and privacy of IoT networks.

Several studies addressed botnet detection and classification (Tab. 1). Reference [7] proposed an ANN-based botnet detection model with a data resampling in detecting DDoS attacks. Author [8] used machine learning (ML) models with dimensionality reduction for detecting DDoS attacks in IoT systems and observed that k-nearest neighbors (KNN) shown best performance and feature reduction reduced system overhead with has less impact on accuracy. Author [9] used dimensionality reduction and decision tree model to detect botnet attack and stated that dimensionality reduction improved the time efficiency and scalability. Author [10] utilized ML model to detect botnet based on honeypot approach to study the hacker behavior by tempting an attacker to detect new malware attacks in the botnet. Author [11] used novel PCA-firefly based XGBoost classification model for intrusion detection.

Like BClus, CAMNEP, and BotHunter, different methods were compared on the dataset containing normal, background, and Botnet traffic [12]. Also, there is an analysis of different machine learning algorithm for Botnet detection, which has two results Botnet or normal [13]. In [14] a decision tree based framework is used for effective detection of P2P Botnet. The researchers also used decision tree algorithm for the feature extraction. Some of them used a method for detecting IOT Botnet, which uses MQTT protocol [15]. A deep learning neural net has been made to detect the Android malware detection which also uses the recurrent neural network [16]. Some researchers use Deep Learning methods for the detection of Botnet. Binary classification has been done using the Feed Forward backpropagation ANN method [17]. BotShark named framework is used for the detection of Botnet. It is based on the deep learning techniques for the inspection of network transactions which also uses the Convolutional Neural Network (CNNs) [18]. In [19] researchers uses convolutional neural network(CNN) for the feature extraction which will be useful in predictive analysis.

Reinforcement Learning is also used for the classification of a packet into Botnet or Normal. The researchers also tested the model on real world dataset and found a good accuracy [20]. Different feature selection methods are also applied on a Botnet dataset and then finding the best possible combination of features for the binary classification [21].

In the medical field also deep learning is being used widely. Many researchers also used deep learning methods in the combination of autoencoders to detect the premature birth of a child [22]. LSTM is also used for the detection of IoT datasets. IoT devices are also vulnerable to different types of threats and especially Botnet. Researchers show that the bidirectional approach is a better model over time [23]. There is also an approach for detecting Botnet using nodes of a graph. It uses a self-organizing map clustering method on the features [24]. Some researchers also use transfer learning on a dataset obtained by combining the different Botnet datasets [25].

A method has been used by some researchers which uses some multiclass classification techniques. They have used a LSTM based framework for managing multiclass imbalance [26]. In [27] used a method to detect Botnet with network flow. They did a multi class classification on the dataset. Authors [28] used two theorems to show that the method can effectively reduce the compute resources consumption, identify DDoS attacks at their primary stage with higher detection rates, and lower false alarm rates.

**Table 1:** State of the art research focused on botnet detection

| Models | Merit | Demerit | Ref. |
|---|---|---|---|
| XGBoost classification | Used the XGBoost approach to decreased the dataset for classification. This technique gave the better result in presented machine learning techniques. | There is probability to loss of Information due to suppression. | [11] |
| BClus and CAMNEP | Applied the three different methods to botnet detection on real and massive botnet dataset. Applied the error metric designed for botnet detections techniques | Complex to deal dynamically the networks flows and clustering the feature at run time. | [12] |
| Multi-layer botnet detection technique | Extracted the most relevant feature by using decision tree algorithm. In third layer of models, tried to reduce the features, which may enhance the efficiency of classification. | There is probability to loss of Information due to suppression. | [13–15] |
| Deep neural network | Extracted the new feature and original feature by using convolutional neural networks (CNNs) on each layer of autoencoder. On the layer of softmax, classified the predicted malicious traffics. | Distinguish between new original and malware detection is very typical to detect. | [16–18] |
| Reinforcement learning-detection | The reduction of network traffic and applied reinforcement learning technique to improve the efficiency and accuracy of models. | There is probability to loss of Information due to suppression. | [20] |
| P2P botnet detection | Observed the important features which will be more useful in building a botnet detection model. | Specific feature base detection can create problem in future. | [21] |
| DNN with semi-classification | In deep neural network, used the three layers such as stacked sparse autoencoder (SSAE) network with two hidden softmax layers to detect the botnet. Contraction intervals and non-contraction intervals were manually segmented. | Human interaction is required. Not detected good accuracy rate that other machine learning approaches. Just considered only 26 features only. | [22] |

(Continued)

**Table 1:** Continued.

| Models | Merit | Demerit | Ref. |
|---|---|---|---|
| BLSTM-RNN detection | Used the bidirectional long short term memory recurrent neural network (BLSTM-RNN), in conjunction with Word Embedding for botnet detection | Ten attack vectors used by the mirai botnet malware is not enough to deal all malware families of botnet. | [23] |
| Graph-based botnet detection | Authors captured the abnormal behaviors of bots in terms of their graph-based behaviors. On the captured behaviors to applied clustering-based detection algorithm. | This static approach is not suitable for the real datasets. Not existing the capability to deal network flows. | [24] |
| Transfer-learning | The hypothesis is "Predictive Performance can be improved by using transfer learning techniques across datasets containing network traffic from different Botnet | Not achieved the adequate level of accuracy for detection botnet datasets compare to machine learning | [25] |
| Domain generation algorithms (DGA) | In the approached used the LSTM.MI algorithm to combine both binary and multiclass classification models. Experiments are carried out on a real-world collected dataset. | Not considering all existing malware families to test botnet datasets | [26] |
| ANN | The applied a fuzzy logic based feature engineering method. In this models used the different learning models can perform differently on different datasets | There is not used the real datasets of botnet to test the models | [27] |

## 3 Significance of the Problem

Numerous studies performed botnet detection. However, a lack of studies addressed the issues related to feature extraction, dimensionality reduction to suppress duplicate information, overfitting, and rigorous parameters tuning. Most studies used real botnet attack datasets in a real environment. Additionally, studies evaluated ML models for synthetic botnet data without much contribution for feature engineering and in-depth assessment of overfitting. Most research used high imbalanced real-time datasets to study botnet detection. Studies majorly focus on the higher accuracies without addressing limitations of highly imbalanced datasets and obtained illusory accuracy. The data splitting was also a significant issue with proper validation of the model without using unforeseen data from training.

## 4 Our Novel Contribution

The present investigation aimed to resolve the gaps from previous studies with a combined PCA method with DNN. PCA reduced the high dimensionality of the data, and DNN models developed with rigorous hyperparameters tuning using GridSearchCV. Early stopping was also implemented to prevent overfitting. The present investigation does not use entire data during training; some data kept separate from the training process was used to evaluate the model's performance for proper validation. This complete exercise prevents overfitting or underfitting, a notable instance of the ML and DL model's output.

Our proposed approach base hybrid met has the multiclass classification including different types of attacks and non-attacks with high precision. The first stage has a high accuracy rate, which indicates the most extensive amount of botnet attacks possible is classified as a threat. The proposed PCA and DNN-based approach demonstrated promising results. Although earlier studies used ML and AI-based techniques for the botnet, as per our knowledge, a combination of PCA and DNN was not applied so far for multiclass classification of actual botnet attacks to defend an IoT environment, reaching a level of high accuracy. The main contribution of current research are as follows:

(i) We proposed a novel approach that utilizes the benefits of PCA for feature collection and the deep neural classifiers to improve effectual and accurate Botnet detection in IoT environments,
(ii) Two novel developed hybrid methods, DNNBoT1 and DNNBoT2, utilized PCA and demonstrated high accuracy in both training and validation phases with less variance for multiclass classification to detect botnet attacks
(iii) The main contribution of this paper is to detect and classify the application-specific threat, e.g., scan attacks, DDoS, TCP flooding, UDP flooding, and sync flooding, which are some of the most common attacks.
(iv) The proposed approach yields stable, reliable, advanced, accurate, safe, and feasible performance into IoT applications-based approach.

### *4.1 Principal Component Analysis*

PCA is an unsupervised learning technique to find informative, new, and uncorrelated features. It was evident from the correlation heatmap (see Fig. 2) that a large number of features for different devices were correlated well, which can be reduced to new candidate features in less number utilizing PCA without losing any vital information. The PCA process involves finding the mean, covariance matrix with eigenvectors and eigenvalues, selecting PC's with the highest Eigenvalues, and the product of the original data matrix. The steps for the PCA process were given in Eqs. (1)–(8). With a sample of '*n*' observations on a vector of *d*' variables

$$\{x_1, x_2, \cdots, x_n\} \in \Re^d \tag{1}$$

Define the first PC using the linear transformation

$$z_1 = a_1^T x_j = \sum_{i=1}^{d} a_{i1} x_{ij}; \quad j = 1, 2, \cdots, n. \tag{2}$$

where $a_1$ is selected based on $v[z_1]$ is maximum.

where the vector

$$a_1 = (a_{11}, a_{21}, \cdots, a_{d1})\,; \; x_j = \left(x_{1j}, x_{2j}, \cdots, x_{dj}\right) \tag{3}$$

The variance was calculated from Eq. (4).

$$var[z_1] = E((z_1 - \overline{z_1})^2) = \frac{1}{n}\sum_{i=1}^{n}\left(a_1^T x_i - a_1^T \overline{x}\right)^2 = \frac{1}{n}\sum_{i=1}^{n} a_1^T (x_i - \overline{x})(x_i - \overline{x})^T a_1 = a_1^T S a_1 \tag{4}$$

The covariance matrix is given in Eq. (5).

$$S = \frac{1}{n}\sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T \tag{5}$$

where $\overline{x}$ is the mean given as Eq. (6).

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6}$$

To find a1 which maximize the variance subject to a1T a1; Let λ is Lagrange multiplier

$$L = a_1^T S a_1 - \lambda(a_1^T a_1 - 1)\frac{\partial}{\partial a_1} L = S a_1 - \lambda a_1 = 0 \Rightarrow (S - \lambda I)a_1 = 0 \tag{7}$$

The a1 and a2 are eigenvectors of S, which corresponds to the highest and second-highest eigenvalues, respectively.

$$\text{var}[z_k] = a_k^T S a_k = \lambda_k \tag{8}$$

The $k^{\text{th}}$ highest eigenvalue of S is the variance of the $k^{\text{th}}$ PC, and the $k^{\text{th}}$ PC holds the $k^{\text{th}}$ highest fraction of the variation.

## *4.2 Deep Neural Network*

Neural Network (NN) belongs to a feedforward artificial neural network (ANN). It mainly consists of three layers: an input layer, a hidden layer, and an output layer; however, the number of layers can be more, where it becomes DNN. Each node uses a nonlinear activation function except for the input nodes. The advantage of automatic feature extraction in DL-based models such as DNN makes it popular for classification [4,5]. The feature extraction is otherwise difficult to define manually. DL is a variation of ML-based algorithms which consisting of a more significant number of sequential layers. DL's primary advantage is that it automatically selects the features, unlike ML methods where feature extraction needs to be done manually. The DNN is a type of DL model that extracts the feasible features from the input data. DNN, which leads to the identification and classification of elements with less requirement of preprocessing. DNN model generally used three main layers: an input layer, activation function layer, and finally, a classification layer (FCN) used for classification purposes. In the present investigation, DNN models were developed to detect and classify botnet attacks for IoT frameworks.

### *4.2.1 Activation Layer*

A neural network needs an activation function to make the prediction. 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 negative value and output the same value for non-negative values. Another activation function is the Sigmoid or logistic function, which is suitable for binary classification and output between 0 and 1. However, the sigmoid function is not suitable for multiclass classification environments, and it needs the multinomial probability distribution for a mutually exclusive class. Instead, Softmax is a function used to activate the function in the output layer of a neural network to deal with a multiclass classification problem. This activation function predicts a multinomial probability distribution with more than two classes.

If we input {1,2,3}, the max function will output the largest number, 3, in the present example. The argmax will output the index of the largest number, which is 2, 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 have output 0 {0,0,1} in the current example.

#### *4.2.2 Dense Layer*

The dense layer is a NN layer, which is deeply connected. Each neuron in the dense layer receives input from all neurons of its preceding layer. It uses a linear operation function to map every input with every output. It can be implemented using Kears as Eq. (9).

$$\text{model.add(Dense}(10, \text{input\_dim} = \text{train\_data\_shape [1]},\ \text{activation} = \text{`relu'})) \tag{9}$$

where relu is the activation function, the shape of training data is the input dimension, with ten units. The units are used to define the shape of output, and its output becomes the input for the successive layer.

#### *4.2.3 Training*

Models such as NN use learning algorithms to minimize the differences between target and output values. One of the main learning algorithms is backpropagation that computes the gradient of a function to fine-tune the network parameters for error minimization.

## 5 Dataset

The N-BaIoT Dataset contains traffic data from 9 Industrial IoT devices, in which seven devices collected instances for 11 classes and the remaining two devices collected data for six classes (Ennio_doorbell and Samsung_SNH_1011_N_Webcam) [29]. The data comprise benign traffic and a variety of malicious attacks such as scan, TCP, UDP, and SYN (Fig. 1). There is a total of 89 csv files in the current version of the dataset with a total size of 7.58 GB and 1486418 instances of normal and attack occurrences. The two botnet attacks MIRAI and BASHLITE were categorized into ten attack and non-attack classes (see Fig. 1). The attacks class are: (1) the scan commands used to find out the vulnerable IoT devices; (2) ACK, SYN, UDP, and TCP flooding; and (3) combo or combination attacks used to open a connection and to transmit the spam to it [3].

## 6 Methodology

Two DNN based models DNNBoT1 and DNNBoT2, were developed in the present investigation to detect botnet attacks based on data from nine industry-grade IoT devices. The primary step to understand the data so that it can be fed for DNN models.

### *6.1 Exploratory Data Analysis (EDA)*

The sklearn, mpl_toolkits, matplotlib, NumPy, pandas, and seaborn libraries were used based on Keras, TensorFlow, and Python language to perform the EDA. The primary objective of EDA is to data cleaning, understand the variables, and analyzing relationships between variables. The columns with NaN were dropped, and the columns were kept, which contains more than one unique value. The statistical correlation of the variables for each device was visualized through a correlation matrix, which is the fastest way to develop an understanding of all variables.

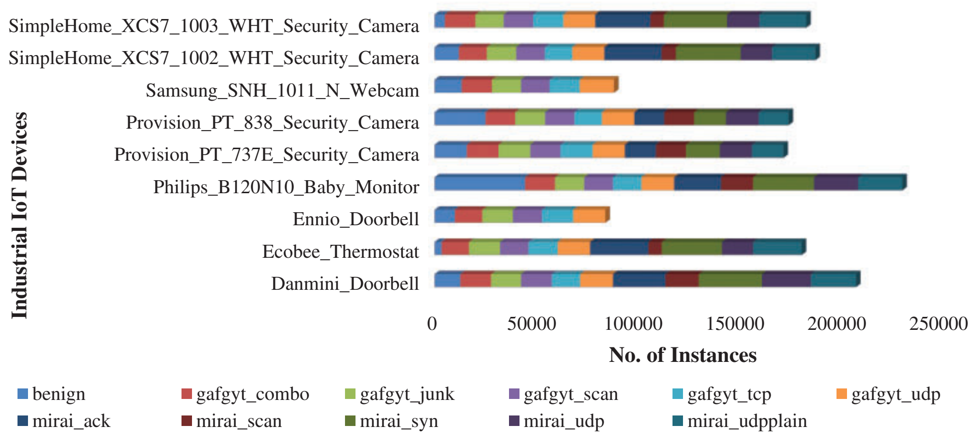

**Figure 1:** Number of classes/instances in each device

### 6.2 *Principal Component Analysis (PCA)*

There were 1486418 instances for ten types of attacks and one non-attack data collected from nine industrial IoT devices. It was observed that some feature vectors have no value close to zero. Therefore, the utility of the PCA was a reasonable idea to synthesize with the DNN models developed in the present investigation to suppress the dimensionality of the feature vectors (see Fig. 3). It was evident from the correlation heatmap (see Fig. 2) that a large number of features for different devices were correlated well, which can be reduced to new candidate features in less number utilizing PCA without losing any vital information. The first step before applying PCA is data standardization. The botnet data have different files and instances; it was required to apply feature scaling to make data of the same scale. The data in the present investigation followed the normal distribution. Therefore standardization was applied. If the data values were skewed, then a normalization function such as min-max scalar could be helpful; to convert data scale-free and ranging between 0 to 1. The PCA process involves finding the mean, covariance matrix with eigenvectors and eigenvalues, selecting PC's with the highest Eigenvalues, and the product of the original data matrix. The number of PCs was selected based on the CEVR (Cumulative Explained Variance Ratio). The steps for the PCA process were given in Eqs. (1)–(8). The standard scalar libraries from sklearn were utilized to obtain the PCA of all nine devices (Fig. 3).

### 6.3 *Deep Neural Network Based Models*

Two novel DNN based models DNNBoT1 and DNNBot2 were developed with six layers each in the present study to detect botnet attacks based on data from 9 industrial IoT. Python 3.8, and Keras 2.3.0 API, Tensorflow 2.0 backend, NumPy, pandas, os, sklearn, matplotlib, and DateTime libraries were used in this research. Data preprocessing and PCA were already applied to the raw dataset; so that the output of both steps can be utilized with developed DNN models. We have used six neural network layers that operate on all ten classes of attacks and one class of non-attack. The rectifier activation function (ReLU) was utilized in starting four neural network layers. The ReLU activation function can be defined as Eq. (10). ReLU acts as a linear function for all positive values and provides zero for all negative values.

$$y = \max(0, x) \tag{10}$$

The fifth layer used the kernel initializer followed by the dense layer with softmax function for classification in the sixth layer. The softmax function can be given as Eq. (11).

$$\sigma\left(\vec{z}\right)_i = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zj}} \tag{11}$$

where K=no of classes, $e^{zi}$ is input vector function, and $e^{zj}$ is output vector function.

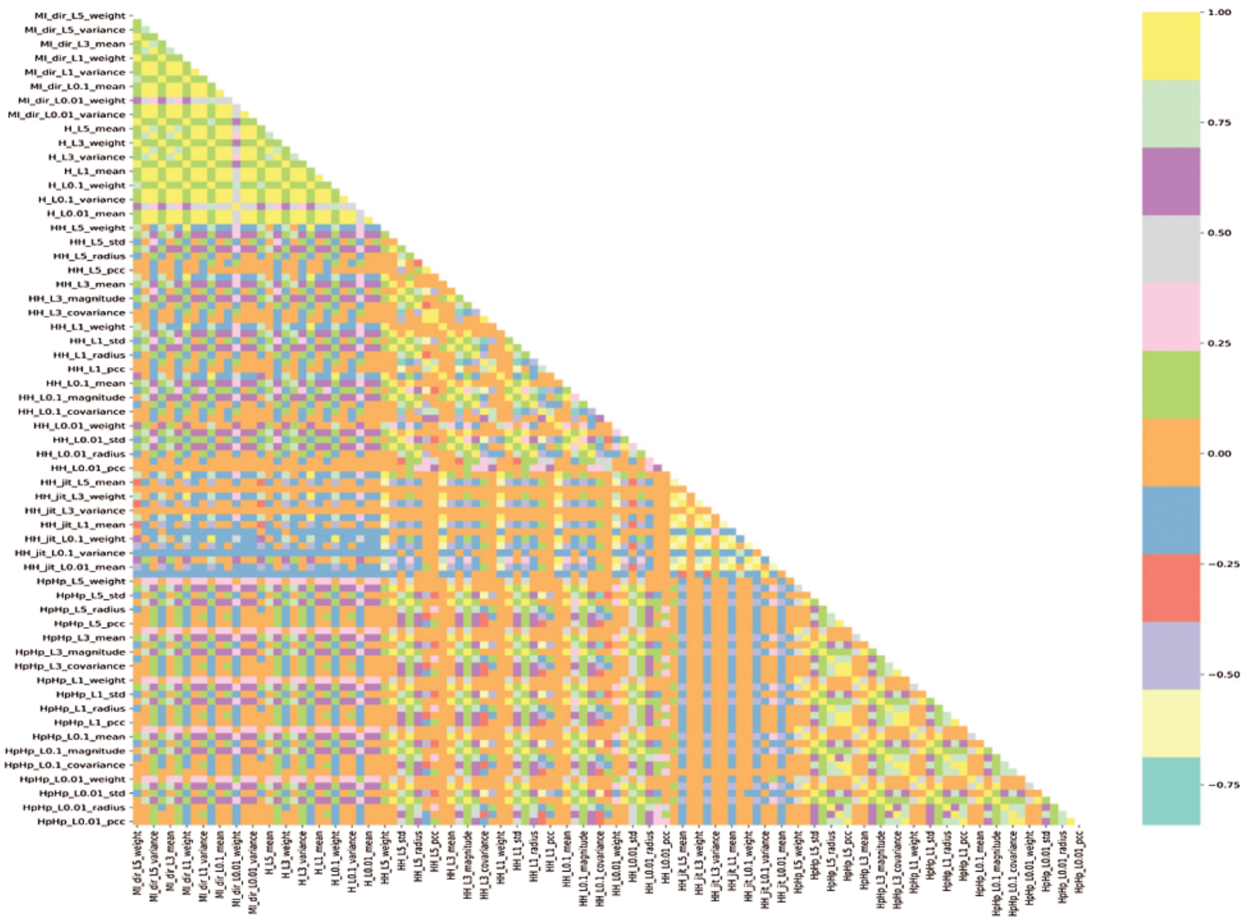

**Figure 2:** The correlation matrix between non-attack and attack instances for provision _PT_737E_security_camera device

Early stopping was used to reduce the learning rate through Keras callbacks function to prevent overfitting. The number of epochs was automatically chosen using the early stopping of

Keras callback functions based on validation loss, minimum delta value, and patience. In DNN model training, the number of parameters such as iterations, learning rate, batch size, and the activation function was obtained using GridSearchCV. Deep learning models such as DNN might be complex, and data splitting is also a significant issue; while tuning the parameters.

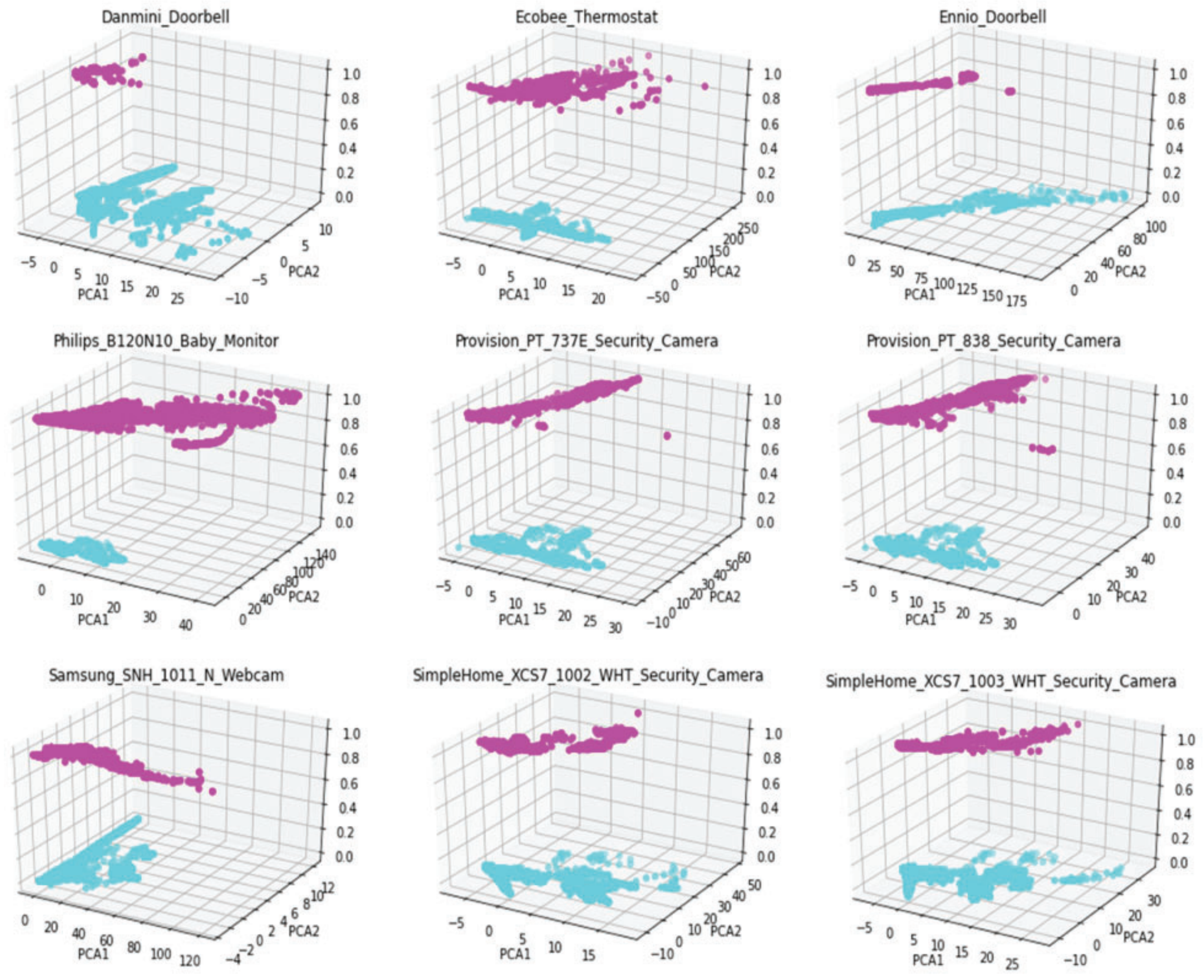

**Figure 3:** 3-D PCA for all 9 IoT devices, cyan data shown normal traffic instances and pink color data shown attack instances

Using GridsearchCV, testing different parameter branches was performed to select the best combination of data splitting ratios for the training and testing. Other important hyperparameters such as batch size, number of neurons, and activation functions such as relu and sigmoid were also assessed utilizing the GridsearchCV. Based on the GridsearchCV utility, the best parameters for both DNNBoT1 and DNNBoT2 models were obtained (Fig. 4). There was a total 2653 number of parameters, and all parameters were trainable using DNNBoT1. For DNNBoT2, there were 8981 parameters, and all parameters were trainable.

Both models were compiled after defining the model. The Adam optimizer was used with a decay of 1e–3; the learning rate was automatically selected dynamically using a callback monitor. The present problem was multiclass classification; therefore, the loss was measured based on categorical cross-entropy; it used to compute the rate of error between the actual and the m values for classification, such as Eq. (12).

$$\text{Loss} = -\frac{1}{O_s}\sum_{i=0}^{O_s} a_i \cdot \log \hat{t}_i + (1 - a_i) \cdot \log\left(1 - \hat{t}_i\right) \tag{12}$$

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

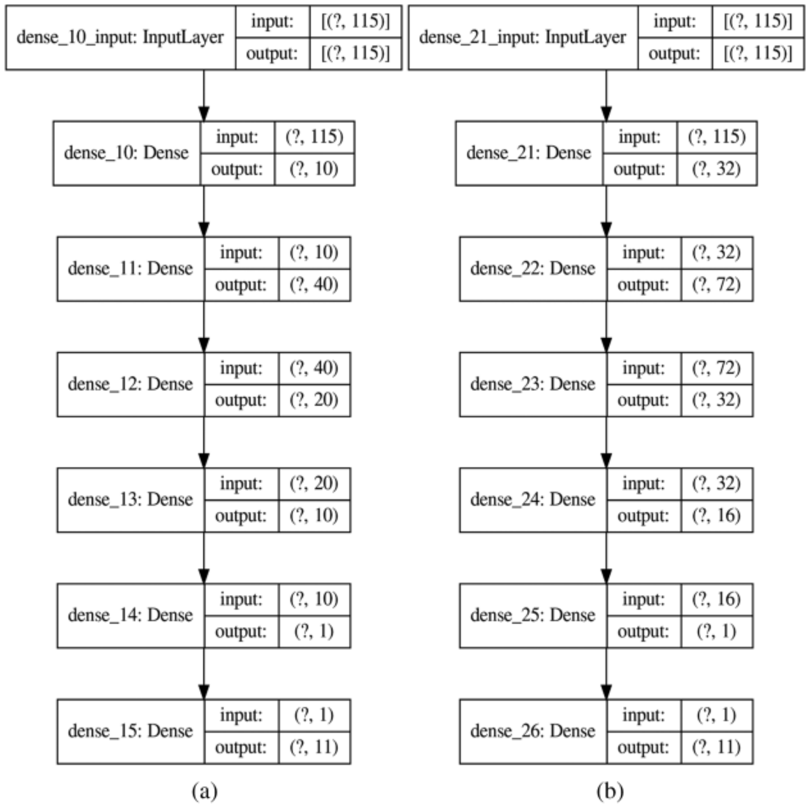

**Figure 4:** (a) Developed DNN models DNNBoT1 and (b) DNNBoT2 to detect botnet attacks for ten attack classes and one non-attack class

## 7 Results and Discussions

In this section, the results were discussed based on the training accuracy, validation accuracy, training loss, and validation loss. The performance of both developed models was assessed with the unforeseen data kept separate from during the training process for proper assessment and evaluation of the developed models. The computation time and accuracies compared to the results of other studies.

### 7.1 Accuracy Assessment

We evaluated the performance of both DNNBoT1 and DNNBoT2 models based on loss and accuracy (Tab. 2). These metrics are defined as follows: Accuracy of a method on a test dataset is the percentage used to correctly identifies the test occurrences, and it is computed as Eq. (13).

**Table 2:** Performance of developed models based on training and validation accuracies for all 9 IoT devices

| Performance | Device 1 | Device 2 | Device 3 | Device 4 | Device 5 | Device 6 | Device 7 | Device 8 | Device 9 | Avg. Accu |
|---|---|---|---|---|---|---|---|---|---|---|
| T_Accu_ DNNBoT1 | 0.8940 | 0.9115 | 0.8017 | 0.9188 | 0.9012 | 0.9064 | 0.7815 | 0.9240 | 0.8940 | 0.9071 |
| T_Accu_ DNNBoT2 | 0.9270 | 0.9004 | 0.8112 | 0.9368 | 0.9045 | 0.9185 | 0.8277 | 0.9111 | 0.9023 | 0.9144 |
| V_Accu_ DNNBoT1 | 0.8925 | 0.9095 | 0.7913 | 0.9157 | 0.9010 | 0.9055 | 0.7810 | 0.9216 | 0.8921 | 0.9054 |
| V_Accu_ DNNBoT2 | 0.9197 | 0.9001 | 0.8109 | 0.9360 | 0.9016 | 0.9176 | 0.8198 | 0.9102 | 0.9015 | 0.9124 |

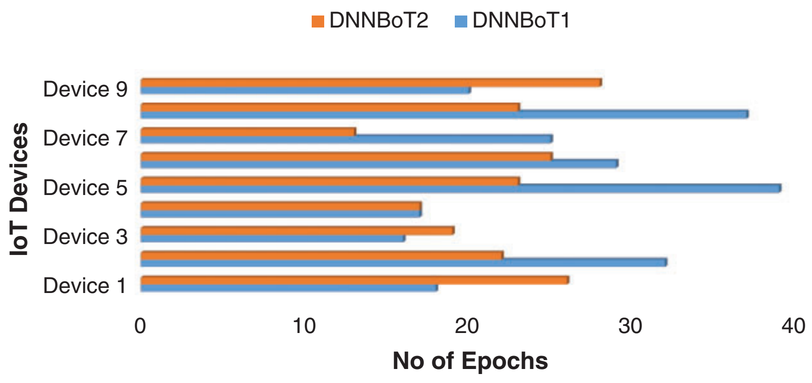

**Figure 5:** Number of epochs for both models for each device

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})} \tag{13}$$

An attempt was made to see if both the model was overfitted. Overfitting can be detected if training loss is comparatively less than validation loss or 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 the overfitting did not exist. It was observed that the training loss was higher since it was more challenging for the network to provide the correct representation. However, all of the units were available during validation so that the network can utilize its full computational power, and therefore, it may perform better than in training. The high accuracy was obtained on the training sets and validation sets with significantly low variance for both models applied on nine devices. Therefore, there was no overfitting.

**Table 3:** Comparison of proposed DNNBoT1 and DNNBoT2 with other studies

| Model | Classification/ Detection | Accuracy | Limitation | Source |
|---|---|---|---|---|
| Deep autoencoder | Detection | Detection accuracy 100% | Data splitting is done manually no classification performed | [1] |
| Semi-supervised | Detection | Detection accuracy 80% | Low accuracy and high speed | [3] |
| SMOTE-Recurrent neural network (DRNN) | | Detection accuracy 99.98% | Data pre processing is required, and no classification | [4] |
| Deep learning ANN technique | Detection | Detection accurcy 99.6%, | This method is not suitable for real-time analysis, no classification | [17] |
| Deep autoencoders | Detection | Detection accuracy 84% | Cannot detect unknown botnets | [29] |
| ZeroR, OneR classifier | Classification | Classification accuracy 85% | 10 to 60 attributes only | [30] |
| Domain generation algorithm(DGA) based on deep learning | Classification | Classification accuracy 90% | Not suitable for real-time | [31] |
| DNNBoT (Proposed) | Detection, classification | Accuracy 90.71%, 91.44% | - | (Proposed) |

As mentioned earlier, both models were hyperparameters tuned using GridsearchCV, and callbacks were utilized to automatically select the optimal number of epochs to prevent overfitting for all nine devices. The automatically selected number of epochs for both DNNBoT1 and DNNBoT2 were given in Fig. 5. DNNBoT1 was efficient in several epochs, which were 233 and 196 for DNNBoT1 and DNNBoT2, respectively, Figs. 6, 7. The utilization of callbacks to select an optimal number of epochs was computation and time efficient instead of fixing the number of epochs for different models or datasets, consuming more computing resources and time.

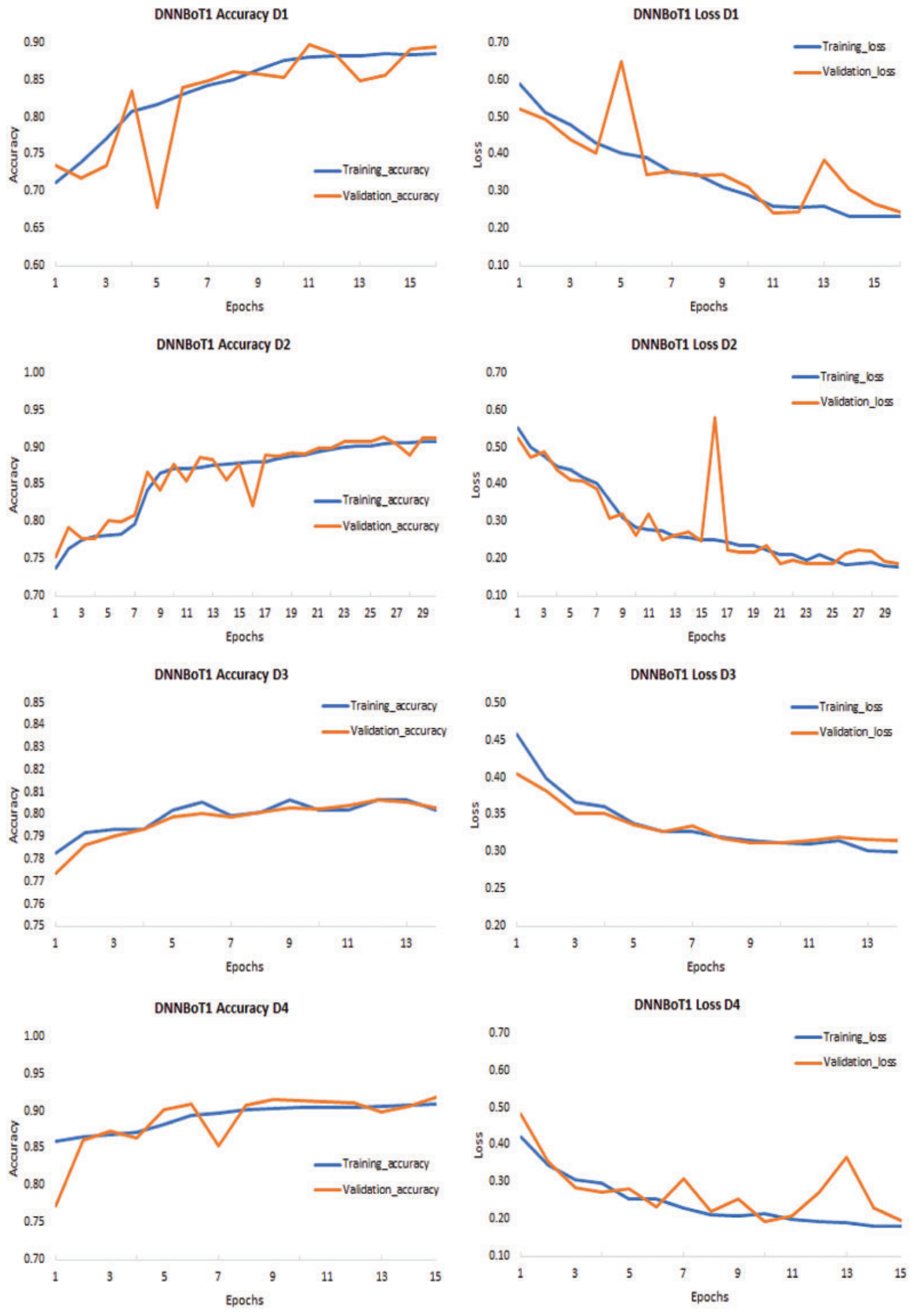

**Figure 6:** (Continued)

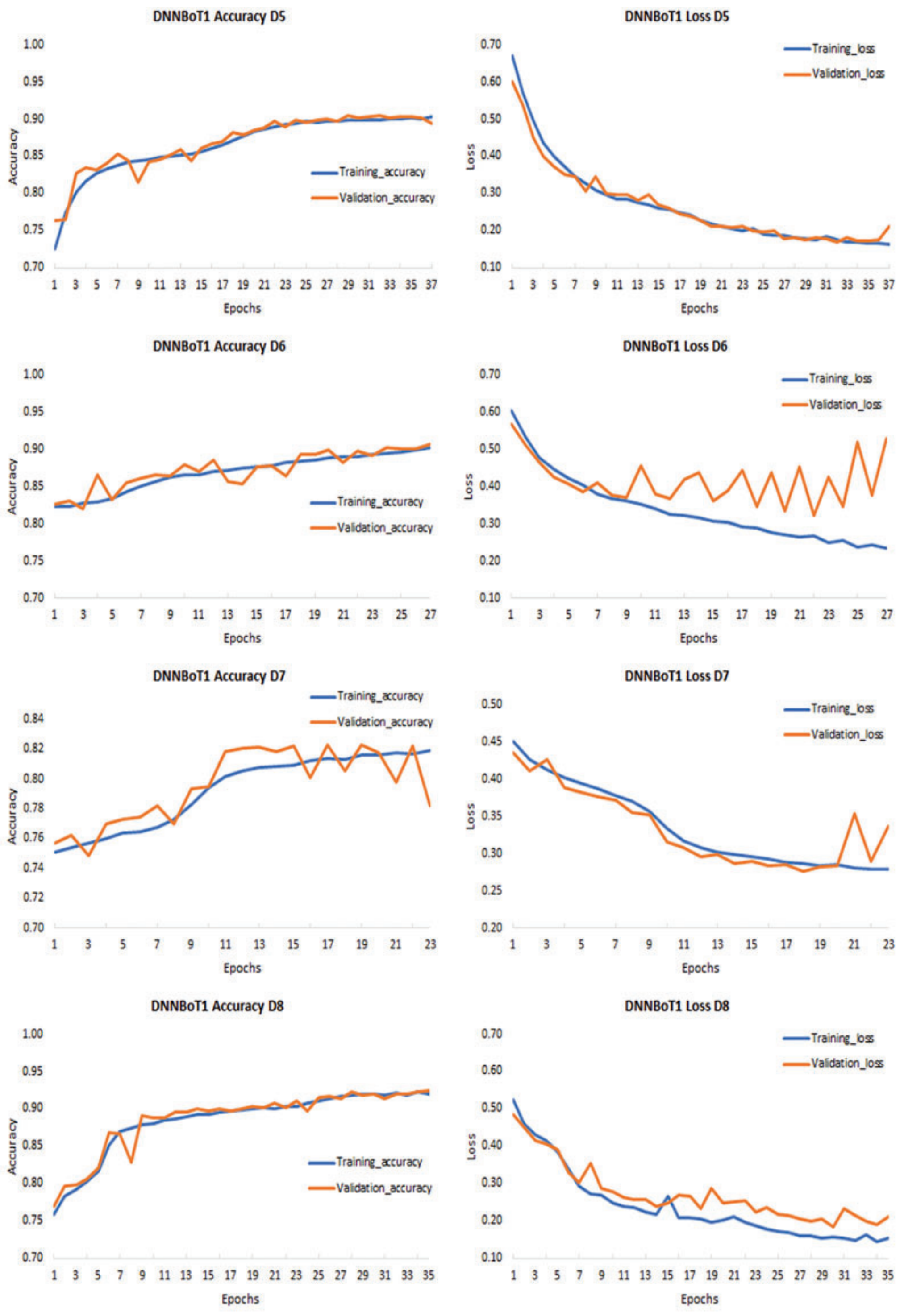

**Figure 6:** (Continued)

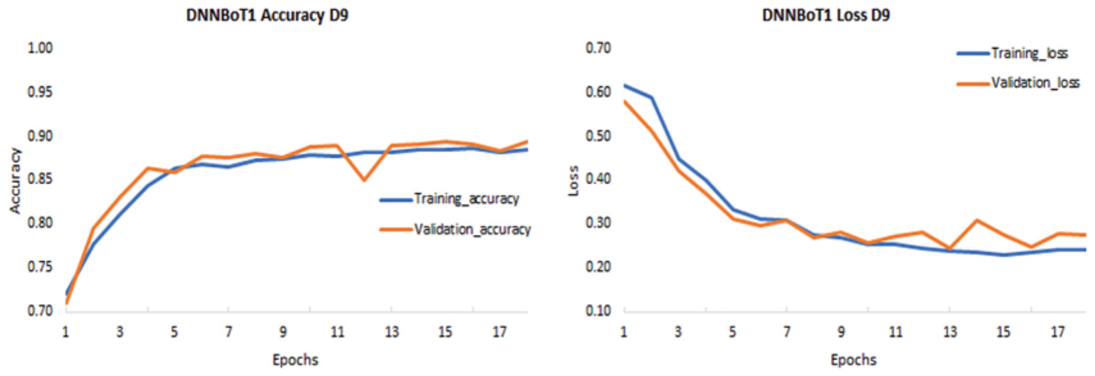

**Figure 6:** Performance evaluation of DNNBoT1 model for all 9 IoT devices (D1–D9) based on accuracy and loss of training and validation

The developed models, i.e., DNNBoT1 and DNNBoT2, demonstrated good training and validation accuracy; however, devices 3 and 9 showed average training and validation accuracy of 80%. The significantly lower performances of both models for device 3 (Ennio doorbell) and device 7 (Samsung_SNH_1011_N_Webcam) can be correlated with fewer data, i.e., data of only six classes out of a total of eleven classes. The average training accuracy of DNNBoT1 and DNNBoT2 was 90.71% and 91.44%, respectively; the average validation accuracy of DNNBoT1 and DNNBoT2 was 90.54% and 91.24%, respectively. Thus, DNNBoT2 performed slightly better than DNNBoT1 based on the training and validation accuracies. Interestingly, the variance between training and validation accuracy was significantly lower in both models; and there was no sign of overfitting or underfitting. Therefore, the accuracies of both models for multiclass botnet classification are higher than in previous studies [3,29–31] based on DNNBoT1 and DNNBoT2 datasets.

## *7.2 Computational Complexity of the Present Study*

The developed models, i.e., DNNBoT1 and DNNBoT2converge quicker with an automated and optimized number of epochs using callback functions. The number of parameters for both models was 2663 and 8991, respectively; all the parameters were trainable. It was evident that DNN models were having a smaller number of parameters. The present investigation utilized Tensor Processing Units (TPUs) v2–8. These TPU's are Google's application-specific circuits, which accelerate the AI models training workflows. There were eight cores and 64 GiB memory in the TPU v2–8 used in the present investigation. DNNBoT1 took 64 s and 37 ms with 26 epochs on average, whereas MLP used only 58 seconds and 16 ms to train the model with 22 epochs.

## *7.3 Comparison from Other Studies*

The present study used the N-BaIoT dataset, one of the available real industrial IoT datasets compromised with two well-known botnet attacks, i.e., MIRAI and BASHLTE. The accuracy of developed novel models DNNBoT1 and DNNBoT2 was compared with other established studies using presented in Tab. 3.

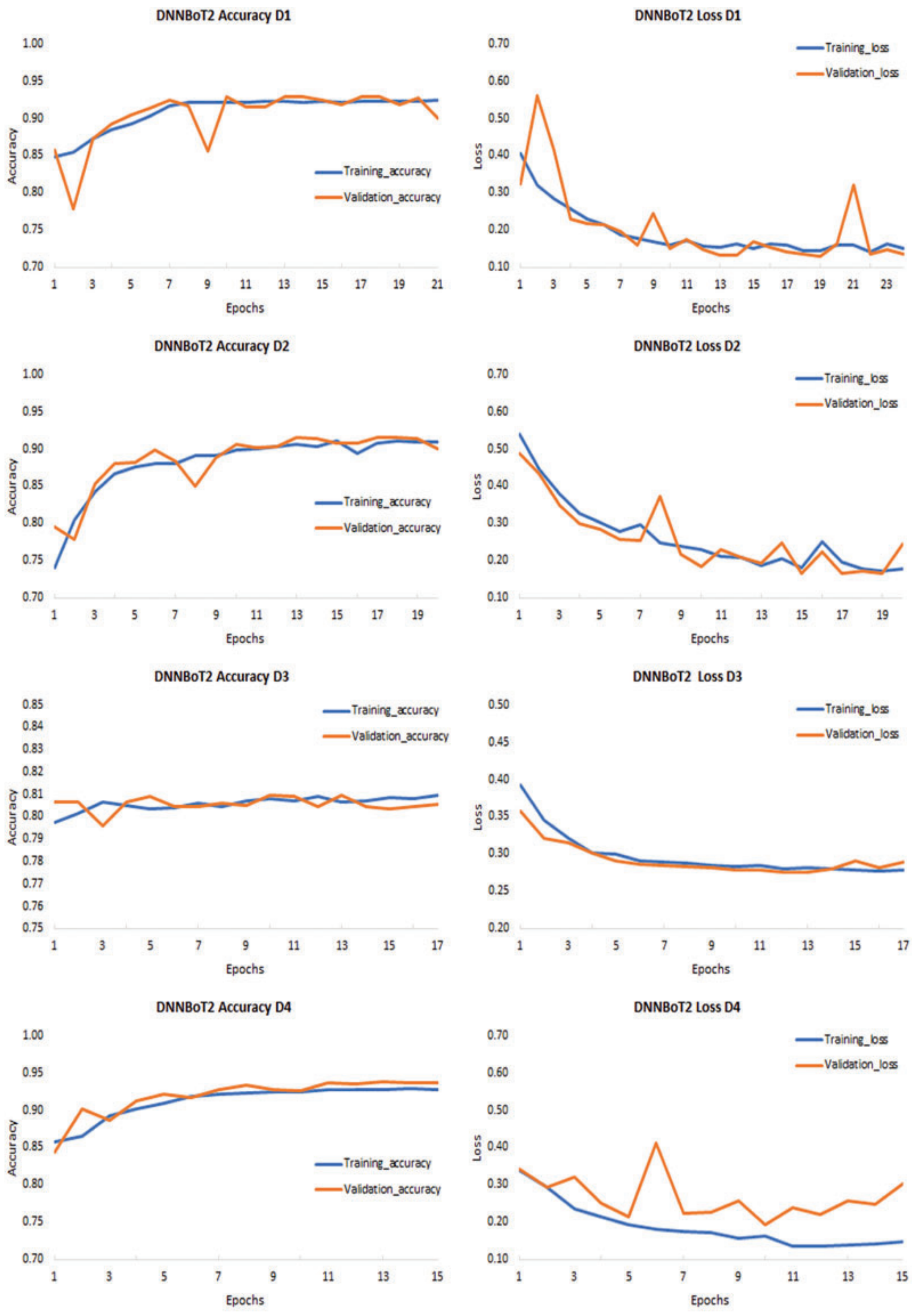

**Figure 7:** (Continued)

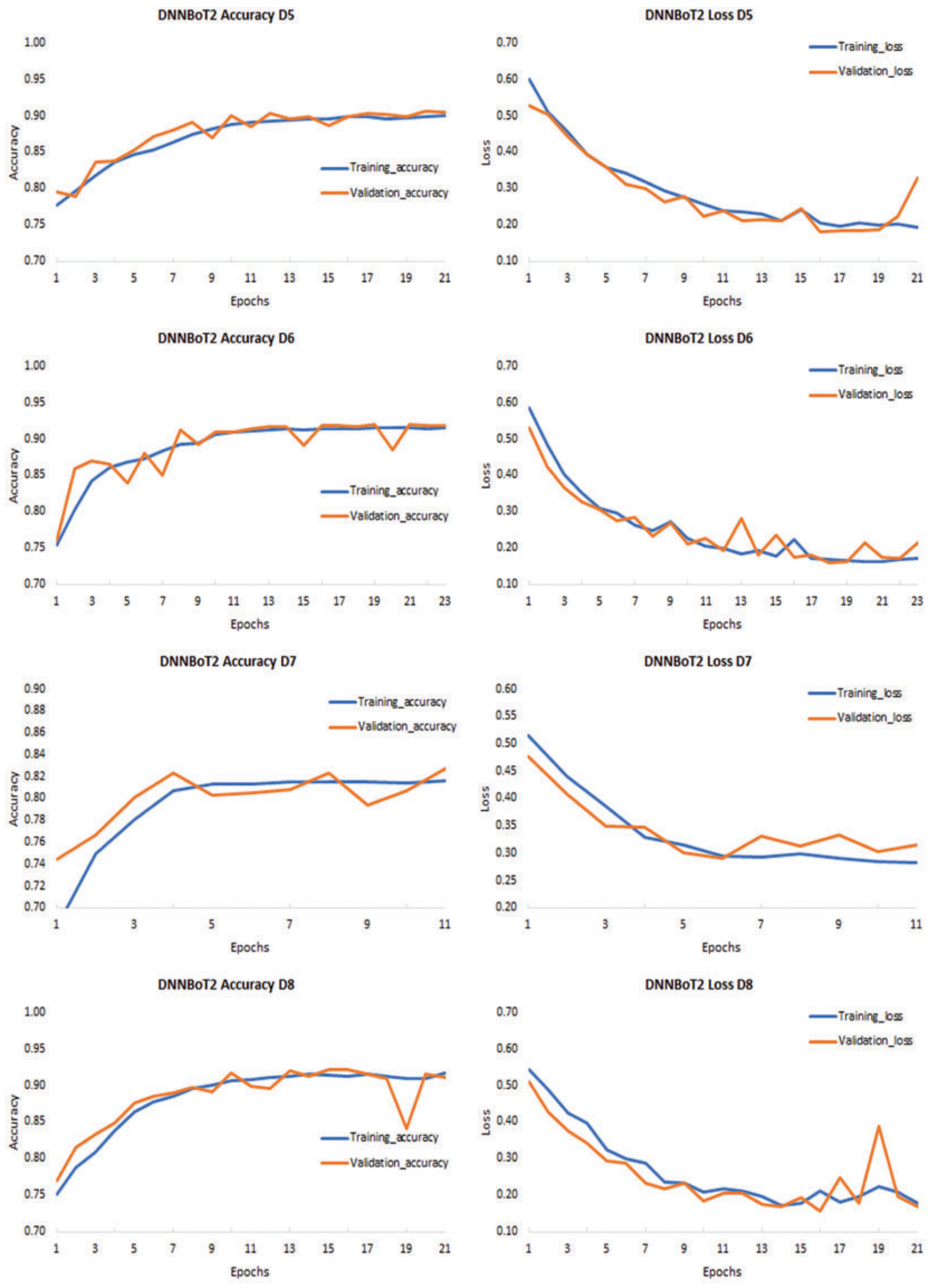

**Figure 7:** (Continued)

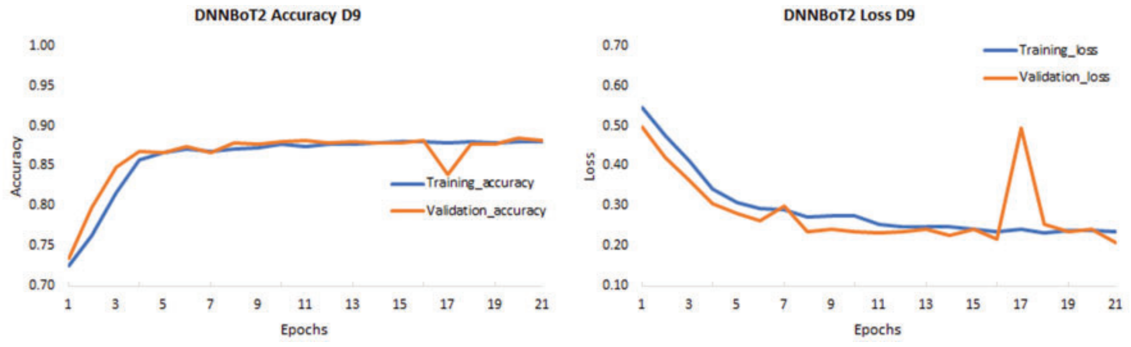

**Figure 7:** Performance evaluation of DNNBoT2 model for all 9 IoT devices (D1–D9) based on accuracy and loss of training and validation models' performance based on computation

## 8 Conclusion

We aimed to develop two novel deep neural network-based botnet detection and classification models for IoT devices. The main contribution of this research is to detect and classify the application-specific threat, e.g., scan attack, DDoS, TCP flooding, UDP flooding, and sync flooding; which are one the most common attack. Models developed in the present investigation used complete attack datasets, unlike previous approaches, which assumed that attacks constitute only a small subset of the whole dataset. Presented models utilized the PCA process to reduce the dimensionality; deep neural network-based models run with optimized parameters obtained from rigorous GridsearchCV based hyperparameters tuning. The callback function was utilized for early stopping to optimize the number of epochs and avoid overfitting. Both models keep learning with time to detect and classify botnet attacks. The results based on unforeseen data kept separate from the training process were used to evaluate the performance of the two developed models. Both models showed good training and validation accuracy with minimal loss and time efficiency. The future scope of the present investigation is to integrate more datasets, apply novel model such as firefly with deep neural networks to understand the internal learning process so that efficiency can be further enhanced in the future [32,33].

**Funding Statement:** Authors would like to thank the Deanship of Scientific Research at Majmaah University for supporting this work under Project No. R-2021-220.

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

## References

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[2] L. Markowsky and G. Markowsky, "Scanning for vulnerable devices in the internet of things," in *Proc. IDAACS*, Warsaw, Poland, pp. 463–467, 2015.

[3] K. Naveed and H. Wu, "A semi-supervised framework to detect botnets in IoT devices," in *Proc. IFIP*, Paris, France, pp. 649–651, 2020.

[4] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh and K. Anoh, "SMOTE-DRNN: A deep learning algorithm for botnet detection in the internet-of-things networks," *Sensors*, vol. 21, no. 9, pp. 2985, 2021.

[5] V. Kansal and M. Dave, “DDoS attack isolation using moving target defense,” in *Proc. ICCCA*, Greater Noida, India, pp. 511–514, 2017.

[6] S. Almutairi, S. Mahfoudh, S. Almutairi and J. S. Alowibdi, “Hybrid botnet detection based on host and network analysis,” *Journal of Computer Networks and Communications*, vol. 2020, no. 1, pp. 1–16, 2020.

[7] Y. N. Soe, P. I. Santosa and R. Hartanto, “DDoS attack detection based on simple ANN with smote for IoT environment,” in *Proc. ICIC*, Semarang, Indonesia, pp. 4, 2019.

[8] M. Aamir and S. M. A. Zaidi, “DDoS attack detection with feature engineering and machine learning: The framework and performance evaluation,” *International Journal of Information Security*, vol. 18, no. 6, pp. 761–785, 2019.

[9] H. Bahsi, S. Nomm and F. B. La Torre, “Dimensionality reduction for machine learning-based IoT botnet detection,” in *Proc. ICARCV*, Singapore, pp. 1857–1862, 2018.

[10] C. Dietz, R. L. Castro, J. Steinberger, C. Wilczak, M. Antzek *et al.,* “IoT-botnet detection and isolation by access router,” in *Proc. NOF*, Poznan, Poland, pp. 88–95, 2018.

[11] S. Bhattacharya, 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.

[12] S. García, M. Grill, J. Stiborek and A. Zunino, “An empirical comparison of botnet detection methods,” *Computers and Security*, vol. 45, pp. 100–123, 2014.

[13] C. Joshi, V. Bharti and R. K. Ranjan, “Botnet detection using machine learning algorithms,” in *Proc. PCCDS*, Singapore, pp. 717–727, 2020.

[14] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, N. A. Golilarz *et al.,* “An adaptive multi-layer. An adaptive multi-layer botnet detection technique using machine learning classifiers,” *Applied Sciences*, vol. 9, no. 11, pp. 2375, 2019.

[15] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Munoz-Castañeda, I. Garcia *et al.,* “Multiclass classification procedure for detecting attacks on MQTTIoT protocol,” *Complexity*, vol. 2019, pp. 1–11, 2019.

[16] X. Pei, L. Yu and S. Tian, “AMalNet: A deep learning framework based on graph convolutional networks for malware detection,” *Computers and Security*, vol. 93, no. 6, pp. 101792, 2020.

[17] A. Ahmed, W. A. Jabbar, A. S. Sadiq and H. Patel, “Deep learning-based classification model for botnet attack detection,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 2020, pp. 1–10, 2020.

[18] S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha and R. Khayami, “BoTShark: A deep learning approach for botnet traffic detection,” *Advances in Information Security*, vol. 70, pp. 137–153, 2018.

[19] C. Chen, P. Zhang, Y. Liu and J. Liu, “Financial quantitative investment using convolutional neural network and deep learning technology,” *Neurocomputing*, vol. 390, pp. 5–11, 2020.

[20] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem *et al.,* “An efficient reinforcement learning-based botnet detection approach,” *Journal of Network and Computer Applications*, vol. 150, no. 11, pp. 102479, 2020.

[21] C. Joshi, V. Bharti and R. K. Ranjan, “Analysis of feature selection methods for p2p botnet detection,” in *Proc. ICACDS-2020*, Singapore, pp. 272–282, 2020.

[22] L. Chen and H. Xu, “Deep neural network for semi-automatic classification of term and preterm uterine recordings,” *Artificial Intelligence in Medicine*, vol. 105, pp. 101861, 2020.

[23] C. D. McDermott, F. Majdani and A. V. Petrovski, “Botnet detection in the internet of things using deep learning approaches,” in *Proc. IJCNN*, Brazil, Rio de Janeiro, pp. 1–8, 2018.

[24] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang *et al.,* “Botnet detection using graph-based feature clustering,” *Journal of Big Data*, vol. 4, no. 1, pp. 1–23, 2017.

[25] B. Alothman and P. Rattadilok, “Towards using transfer learning for botnet detection,” in *Proc. ICITST*, Cambridge, UK, pp. 281–282, 2017.

[26] D. Tran, H. Mac, V. Tong, H. A. Tran and L. G. Nguyen, “An LSTM based framework for handling multiclass imbalance in DGA botnet detection,” *Neurocomputing*, vol. 275, no. 8, pp. 2401–2413, 2018.

[27] L. Mathur, M. Raheja and P. Ahlawat, “Botnet detection via mining of network traffic flow,” *Procedia Computer Science*, vol. 132, pp. 1668–1677, 2018.

[28] J. Cheng, R. M. Xu, X. Y. Tang, V. S. Sheng and C. T. Cai, “An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment,” *Computers, Materials & Continua*, vol. 55, no. 1, pp. 95–119, 2018.

[29] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai *et al.,* “N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders,” *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[30] S. S. Chawathe, “Monitoring IoT networks for botnet activity,” in *Proc. NCA*, Cambridge, MA, USA, pp. 1–8, 2018.

[31] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. V. Pham, S. K. Padannayil *et al.,* “A visualized botnet detection system based deep learning for the internet of things networks of smart cities,” *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4436–4456, 2020.

[32] T. R. Gadekallu, N. Khare, S. Bhattacharya, S. Singh, P. K. R. Maddikunta *et al.,* “Early detection of diabetic retinopathy using PCA-firefly based deep learning model,” *Electronics*, vol. 9, no. 2, pp. 1–16, 2020.

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