




SAD-IoT: Security Analysis of DDoS Attacks in IoT Networks

Prahlad Kumar¹ · Harnoor Bagga² · Bhuneshwar Singh Netam¹ ·
Venkanna Uduthalapally¹ 

Accepted: 8 August 2021 / Published online: 25 August 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

Internet of Things is one of the most versatile technologies in existence today. It has taken over our day to day activities and thus has many applications that are designed to make life easier and simpler. Partly because IoT is new, it is replete with insecurities and vulnerabilities. Due to the lack of fundamental security controls, and the integration of real-world objects with the Internet, IoT devices are facile targets for cyber-criminals and other aggressors. This means that these vulnerabilities can be exploited for hacking, adding to Botnets, and then used to launch DoS and DDoS against organizations. To provide security from DoS and DDoS attacks, various solutions have been proposed. In this paper, Machine Learning, as well as Deep Learning algorithms, have been employed to analyze the DoS and DDoS attacks. The Bot-IoT dataset of the Centre of UNSW Canberra Cyber was used for training purposes. ARGUS software was used to generate the features from the pcap files of UNSW. A testbed was setup using 20 devices and generated dataset. From the result, the best accuracy of attack classification is 99.5% and 99.9% for Deep Learning and Machine Learning algorithms respectively.

Keywords Internet of Things (IoT) · DDoS attacks · Keras · Attack detection · Machine Learning · Deep Learning

✉ Venkanna Uduthalapally
venkannau@iiitnr.edu.in

Prahlad Kumar
prahlad17100@iiitnr.edu.in

Harnoor Bagga
harnoorbagga98@gmail.com

Bhuneshwar Singh Netam
bhuneshwar17100@iiitnr.edu.in

¹ International Institute of Information Technology, Naya Raipur, India

² Vellore Institute of Technology, Chennai, India

1 Introduction

Internet of Things is an enormous interconnection of computing devices, digital and mechanical machines, animals, or people, all of which gather data related to their usage and their surroundings and share it without requiring any human intervention [1]. The data picked up by connected devices enables us to make smart decisions, based on real-time information. The abundance and the ubiquity of the internet, the steadily growing capacity of network connection, and the diversity of connected devices make the IoT adaptable and scalable. But at the same time, the main concern in this area is the security of the data [2]. Attacks on IoT devices can breach their security with the advancement of technology [3]. Some of the very common attacks are DoS, Privilege escalation, Firmware hijacking, Brute-force password attacks, Physical tampering, Malicious node injection, and Eavesdropping.

The main focus of this paper is on DoS and DDoS attacks. DoS or denial-of-service is a malicious attempt in which adversary tries to disrupt the legitimate internet traffic of the victim system, server, network, or service by overwhelming the victim or its surrounding by flooding it with internet traffic. DDoS (Distributed Denial of Service attacks) is a type of DoS attack in which the adversary uses multiple computers to flood the victim with the packets to prevent legitimate users from accessing its resources by overloading it. Many researchers have found different ways to safeguard the network of connected devices. Some of the basic steps that are usually taken to prevent the IoT attacks from happening are making the network more resilient, proper network segmentation and testing, timely software and firmware patching, etc. Other solutions that focus on preventing specific attacks like DoS, DDoS, etc. include Machine Learning [4] or Deep Learning algorithms.

To prevent various attacks a robust method is proposed named as SAD-IoT. This method analyses the DoS and DDoS attacks using various Machine Learning and Deep Learning algorithms to classify attacking traffic from normal traffic. In this approach, the stacking of four Machine Learning algorithms (Decision Trees, Naive Bayes, KNN, and Random Forest) has been implemented on the features set built earlier. The same dataset has been used for the Deep Learning model (Multiclass Neural Networks) as well. All the approaches were then compared using various performance metrics. The BoT-IoT dataset was chosen for training purpose which was taken from the Cyber Range Lab of the center of UNSW Canberra Cyber [5]. It contains more than 72 million records. The dataset includes DDoS, DoS, Keylogging, and Data exfiltration attacks, OS and Service Scan, with the DDoS and DoS attacks further categorized based on the protocols. In this work, only the DoS and DDoS attack dataset was used. Next, for training purpose, the generated testbed consisted of six PCs, six sensors, two smartphones, and four Node-MCUs. Out of six PCs, four were used for attacking purpose. The rest of the PCs along with other devices contributed to the generation of benign traffic.

The rest of the paper is organized as follows: Sect. 2 explains novel contributions of the paper. The existing works on the detection of DoS and DDoS attacks are given in Sect. 3. Section 4, gives background works required. The detailed implementation of proposed solutions is described in Sect. 5. The experimental results and analysis is explained in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Novel Contributions of this Paper

In DoS and DDoS attacks, a series of complex and diverse attacks are employed to afflict the victim's network system, or services which it provides to legitimate users. These attacks may come from different devices and different locations. Hence a detection system is required that can detect diverse DoS and DDoS attacks to secure the data. Given these challenges, a survey on various types of DoS and DDoS attacks and a model for their detection is proposed. The major contributions of the paper are as follows:

- The real testbed was developed using sensors, PCs, smartphones and Node-MCUs.
- The testbed was used for data generation consisting of attack traffic and normal traffic.
- The generated dataset was preprocessed and used for testing purpose.
- For attack traffic and normal traffic classification the proposed work suggests Stacking with three algorithms-Random Forest, KNN, and Decision Trees with Logistic Regression as meta-classifier for Machine Learning, for training-testing purpose.
- Finally various Machine Learning and Deep Learning algorithms performance were compared and analysed.

3 Related Prior Research

The DoS and DDoS attack detection methods keep on evolving as the problem is very critical for the organizations. Most of the researchers are leveraging Artificial Intelligence for detection purposes. Some of the existing solutions are described below.

Rohan Doshi et al. demonstrated that normal and DoS attack traffic can be easily distinguished via packet-level Machine learning algorithms. Their feature selection was based on the hypothesis that network traffic patterns from consumer IoT devices differ from those of well-studied non-IoT networked devices [6]. Parth Bhatt et al. proposed a method that utilized a Hybrid Detection Module based on four Machine learning algorithms. The flooding methods simulated were HTTP Flood, Slowloris, Slow HTTP post, TLS renegotiation, Co-AP Flood Multicast, Co-AP Flood Unicast, Co-AP IP Spoofing, and ARP cache poisoning [7].

Dragan Peraković et al. research used artificial neural networks for classification of pre-defined classes of traffic. The different numbers of neurons used in the hidden layer were (30, 35, 40, 45, 50, and 55). Four publicly available datasets were used [8]. Bayu Adhi Tama et al. proposed a method which addressed deep neural network for classification of IoT network attacks. Three standard datasets were used. The performance metrics used were accuracy, precision, recall, and false alarm rate [9].

McDermott et al. proposed an approach in which they have developed a BLSTM-RNN detection model. They have used the developed model and have compared it to a LSTM-RNN for detecting four attack vectors. Models were evaluated for accuracy and loss. A labeled dataset was generated as a part of their research. Both models returned high accuracy and low loss metrics for the four attack vectors used by the Mirai Botnet malware [10]. The approaches of the above mentioned existing works is summarized in Table 1.

Table 1 Summary of related works

| S.no. | Existing works | Methodology | Limitations |
|-------|--|---|--|
| 1 | Machine Learning DDoS Detection for Consumer Internet of Things Devices [6] | <p>Low-cost Machine learning algorithms were applied</p> <p>Flow-based and protocol-agnostic traffic data was used</p> <p>Developed a pipeline of algorithms that performed data collection, feature extraction, and binary classification for traffic detection</p> | <p>Limited no. of features</p> <p>Small dataset</p> |
| 2 | HADS: Hybrid Anomaly Detection System for IoT [7] | <p>Four Machine learning algorithms named Self Organizing Maps (SOM), One-class Support Vector Machine (OCSVM), Gaussian Mixture Model (GMM) and Isolation Forest were used in the hybrid module</p> <p>They simulated 8 different network attacks</p> | <p>High consumption of CPU and RAM</p> <p>Usage of very few features</p> |
| 3 | Artificial Neuron Network Implementation in Detection and Classification of DDoS Traffic [8] | <p>Artificial Neural Network was implemented to detect and classify unwanted DDoS traffic</p> <p>4 pre-defined classes of network traffic were UDP DDoS attack, CharGen DDoS attack, DNS DDoS attack and normal traffic (legitimate traffic)</p> <p>A set of 4986 recordings of network traffic was created from the 4 available datasets</p> | <p>Relatively low accuracy (82.1%) while classifying UDP</p> |
| 4 | Attack Classification Analysis of IoT Network via Deep Learning Approach [9] | <p>Deep Neural Networks model was used for classifying attacks in IoT network traffic</p> <p>Three sampling methods called cross-validation, repeated cross-validation, and subsampling were used</p> <p>Different datasets used were UNSW-NB15, CIDDS-001 and GPRS</p> <p>Grid search was used to discover the best learning parameter</p> | <p>DNN performed efficiently only on CIDDS-001 dataset due to imbalanced dataset</p> <p>No performance differences observed between DNN and other algorithms due to similar architecture</p> |

Table 1 (continued)

| S.no. | Existing works | Methodology | Limitations |
|-------|--|---|---|
| 5 | Botnet Detection in the Internet of Things using Deep Learning Approaches [10] | <p>A Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN) model was used</p> <p>Word Embedding for text recognition and conversion of internet packets into tokenized integer format was implemented</p> <p>Four types of floodings used were UDP flood, ACK flood, DNS flood, and SYN flood</p> | <p>Bidirectional approach added additional overhead to each epoch</p> <p>High Processing time</p> <p>ACK attack vector metrics were shown to be less favourable</p> |

4 Background

This section includes a detailed explanation about the attacks and various Machine Learning and Deep Learning algorithms used in this paper.

4.1 DoS and DDoS Attack

DoS Attack DoS attack deprives legitimate users of accessing a Machine, services, or network. It can be done either via flooding the server with large invalid data, or by sending millions of requests to slow down the server [11]. DoS attacks in IoT devices may render them unresponsive. They may even damage the IoT devices to such extent that it requires a replacement or re-intallation.

DDoS Attack DDoS attack is the malicious attempt to disrupt the normal traffic of the target system. This happens when the bandwidth or the resources of the target system are flooded from the compromised numerous devices that are distributed globally [12]. According to Akamai researchers, about 21% of all the DDoS attacks happen from IoT devices around the world.

SYN Flood It is also known as the half-open attack. The attacker makes the victim's IoT devices unresponsive by consuming its resources by successively sending it syn packets. This is done with the help of spoofed IP addresses. The attacker repeatedly sends connection requests and overwhelms all available ports on the targeted victim's machine, causing the victim machine to respond to legitimate traffic sluggishly or not at all [13].

UDP Flood The attacker floods the victim's machine with UDP packets. In this type of attack, sometimes the firewall which protects the victim gets exhausted by the UDP flood, resulting in a denial-of-service to legitimate traffic. This attacks exhausts the resources of the IoT devices and hence makes them unresponsive [13].

4.2 Machine Learning Algorithms Used in Our Solutions

This paper utilised six Machine Learning techniques in total. They are KNN, Decision Trees, Random Forests, Naive Bayes and a Stacking Technique which took Logistic Regression(sixth technique) as a meta classifier and KNN, Random Forest and Decision Tree as classifiers. KNN being a non-parametric supervised learning algorithm, relies on the labeled input data. This algorithm assumes that similar things lie nearby. It uses feature similarity to classify the testing data [14]. Decision Tree is a non-parametric supervised learning which can be used for both regression and classification tasks. It is used to break down the dataset into smaller subsets according to the decision taken at each step. This results in a tree-like structure with decision nodes and leaf nodes [15]. Random Forest is the advanced version of the Decision Tree. It is used to reduce the problem of overfitting in Decision Tree [7]. It is an ensemble learning algorithm, utilizing a large number of decision trees, which can be used for performing both classification and regression tasks. Naive Bayes works on the principle of Bayes Theorem and is used for classification tasks. It has the assumption that each feature makes an equal and independent contribution to the outcome and due to this reason it is also called idiot Bayes [16]. Logistic Regression is used as a binary classifier and is used to describe data and to predict the probability of a categorical dependent variable. The main benefit of using this algorithm is that it indicates the relationship of the dependent variable with each of its features. It provides the direction of association along with the relevance of its features. Stacking is an ensemble method that

uses meta-algorithms to combine several Machine Learning algorithms into one predictive model to decrease variance (bagging), bias (boosting) or to improve predictions [17]. In this paper, the model used a parallel ensemble method where base learners work independently. The main idea behind stacking is to take average predictions of all the predictors used in the ensemble instead of hard voting.

4.3 Deep Learning Algorithm Used in Our Solution

Multiclass Neural Networks—A famous framework Keras is used to solve the problem of multiclass [18, 19]. The usual choice for multi-class classification is the softmax layer [20]. In this, the softmax function extends the idea of logistic regression. The function takes an input of the vector of K real numbers. Then it normalizes the vector in the range of $(0,1)$ into a probability distribution for each class in a multiclass problem in such a way that the probabilities sum up to 1. This additional constraint helps in faster convergence during training. The number of nodes in the final layer of the neural network is equal to the number of output classes present. Softmax is used just before the output layer with the same number of nodes as those in the final layer. For softmax to work easily, the class labels are applied with one-hot-encoding. This will create a classification model that is effectively a set of weights (multipliers) for each layer of the network. The initial weights are randomized, not starting from a fixed set of numbers for better results. The categorical cross-entropy loss function is used during the compilation of the model to measure the error between any given hypothesis and the original outcome for those inputs present in their training dataset. Adam optimization algorithm is used to adjust the weights to minimize the error in the training set.

5 Proposed Solution: SAD-IoT

The various modules involved in the proposed solution are shown in Fig. 1. The proposed solution is divided into three modules (Dataset Generation (Input Module), Feature Generation (Pre-processing Module), and Testing and Training Phase (Output Module)). The first module explains the need for dataset generation, how the testbed connections were made to collect the data, and about the network analysis done for the same. The second module includes information about data collection from Wireshark application and the generation of features using Argus application in Linux. The final module is about the different algorithms used for testing and training purposes.

5.1 Dataset Generation (Input Module)

The dataset of UNSW incorporates both normal IoT-related and other network traffic, along with various types of attack traffic commonly used by Botnets [5]. A portion of dataset of DoS and DDoS attack traffic from the UNSW dataset server was used for training purpose. Whereas for testing purpose, real time dataset was generated using a total of 20 devices in an isolated IoT environment. The real time dataset included both IoT and non-IoT devices. The list of softwares and hardware components used are shown in Tables 2 and 3 respectively.

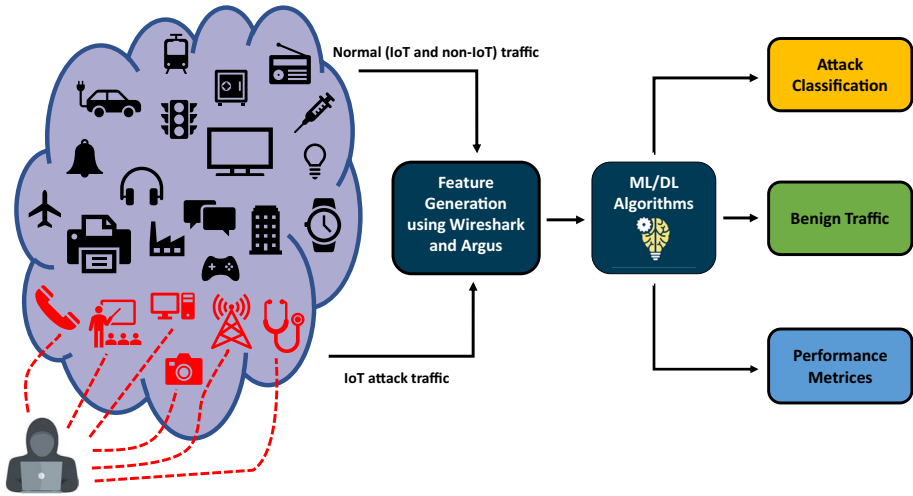


Fig. 1 Modular diagram

Table 2 Softwares utilised

| Tasks | Softwares |
|---------------------------------|--------------------------|
| Feature generation | Argus (in Linux) |
| Internet traffic capturing | Wireshark |
| Preprocessing | Microsoft Excel , Python |
| Machine Learning, Deep Learning | Python |
| DoS and DDoS attack simulation | LOiC, CMD |

Table 3 Hardwares utilised

| Tasks | Components |
|---|--|
| MicroController to connect sensors with the network | NodeMCU |
| To establish a network | Router |
| Traffic generation (Attack and Benign) | PC, Laptops, Mobile Phones |
| IoT Data generation | Sensors (PIR, Infrared, LM35, Flex, Sound, DHT11, Ultra-sonic) |

5.1.1 Need for Dataset Generation

Given below are some of the reasons for generation of new dataset:

- There are limited datasets available online, to train and test different Machine Learning models as per our requirements.
- Existing datasets incorporates limited number of attacks.

- Referred literature suggests that previous works lag in real time attack data capturing environment.

5.1.2 IoT Network Setup for Creation of Dataset

The schematic diagram of the process of generation of a dataset for DoS and DDoS attacking packets in the IoT network is shown in Fig. 2. The router acts here as a bridge between the isolated testbed and the internet. At the start of the simulation, all the PCs generated normal internet traffic and then after 15 min, the four attacking PCs started DDoS attacks first and later DoS attacks. The attacking period was 30 s. After every 30 s, the attack was halted and normal internet traffic generation was done in order to simulate real attacks. If the attacks are prolonged for a longer time in the network, there’s a high chance that the attacking devices are identified by the firewall or the other detectors.

A total of 20 devices were used for the testbed. These devices were located in a separate network using two routers, one of which was connected to the internet through a LAN. Rest of the ports of the routers were utilised to connect the PCs. The testbed also consists six PCs, four sensor nodes with multiple sensors and, two smartphones. These sensors were used for development of IoT sensor Network shown in Fig. 3. The data collected from the sensor was sent to the ThingSpeak cloud. Sample of collected data from the sensor is depicted in Fig. 4. From the above network setup few nodes were made the victims of the DoS and DDoS attack in the testbed. These sensor nodes were connected to the internet via their inbuilt WiFi modules to one of the non attacking PC which was used for WiFi hotspot.

The testbed implementation with the devices utilised and their connections is shown in Fig. 5. The proposed testbed consisted of four desktops, two laptops (six PCs), two smartphones, two routers, four NodeMCUs(sensor nodes) and six sensors. The laptops were non attacking PCs in the testbed and the desktops acted as attackers. The smartphones generated normal traffic by surfing internet, playing videos on the youtube and downloading apps from playstore. These smartphones were connected to WiFi hotspot

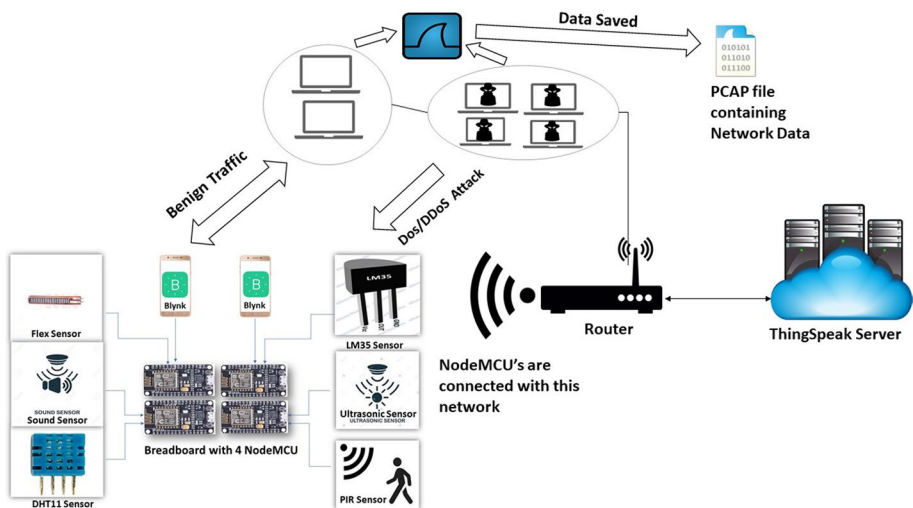


Fig. 2 Schematic diagram of our testbed

Fig. 3 IoT network setup

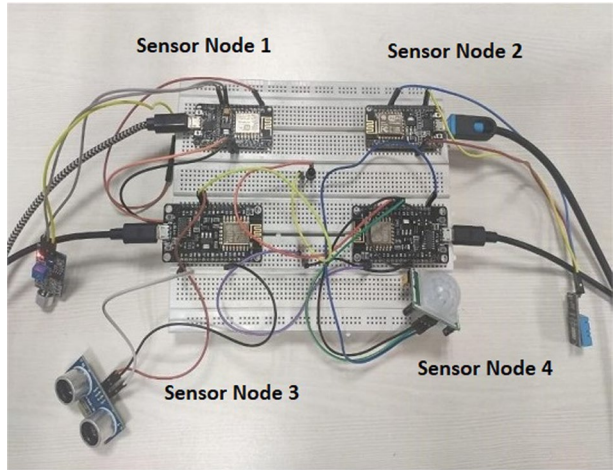


Fig. 4 Flex sensor reading snapshot from ThingSpeak server

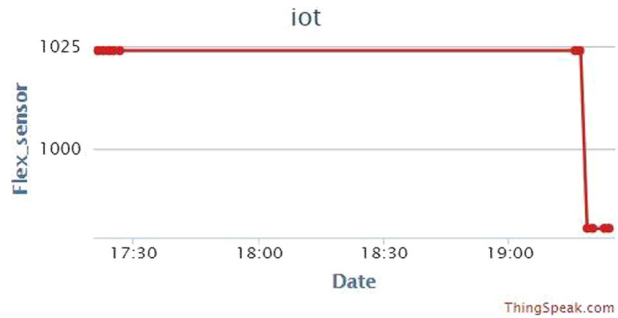
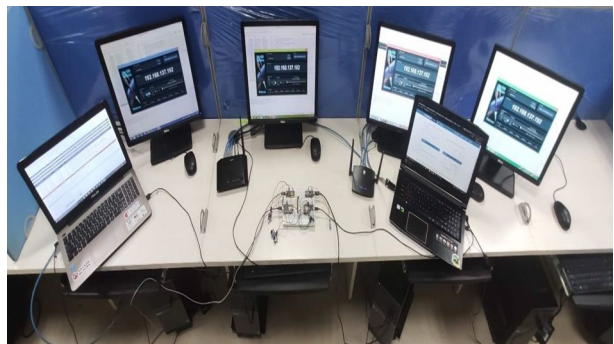


Fig. 5 Testbed implementation



hosted by one of the non-attacking PCs. The simulated testbed was isolated from the institute’s network and a separate network was established using two routers and eighteen other devices. Isolation was done for the safety purpose. Normal data was collected by performing transfer of files and internet browsing. For the DDoS attack simulation, four desktops were used to attack the sensor nodes(victims) with DDoS (TCP and UDP flood) traffic simultaneously. LOIC application and Command Prompt’s ping of death

were used for generating malicious traffic (TCP and UDP flood). These data traffic packets were captured using Wireshark.

5.2 Data Collection and Feature Generation (Pre-processing Module)

For training purpose, UNSW's DDoS and DoS raw dataset was collected, which was later preprocessed as per our requirements [5]. The dataset was in the pcap format. Similarly, for testing purpose a testbed with 20 devices was setup in real-time environment. The internet traffic was captured using Wireshark and was dumped in pcap format. The pcap file contains all the information associated with the internet packets but the information is initially hidden and only few columns are shown by default in Wireshark, like start time, end time, packet id etc. Since the essential columns like number of ports activated during connection could not be extracted directly, Argus application was used to manually extract 20 new features. These new features were then mapped for every packet. After this, 8 new features were generated for identification of DoS and DDoS attack traffic. For example "Tpkt Saddr" represented total packets sent by each source IP address. Thus if the source IP address belongs to the attacker the number of packets generated will be much higher. Similarly "rate" of the transfer will also be high. Thus, these features are of importance to the learning models. Data cleansing was then done using python to fill the empty cells and to remove any unwanted string values occurring in the dataset. For example, asterisk was appended at the end of the "dur" column, which was later removed. After properly cleaning the dataset, this dataset was normalised in range - 1 to 1 for better training using z-normalisation method. The final labelled dataset consisted of four classes i.e. DDoS TCP, DDoS UDP, DoS and Benign internet traffic. It was then fed to the Machine Learning models [21]. The description of all the features is listed in Table 4.

5.3 Training and Testing Phases (Output Module)

For training the Machine learning models, the dataset was pre-processed and the discrepancies like null values, mis-matched data types, etc were corrected [22]. Before feeding the dataset to the models, the dataset was normalised in order to keep outliers in check. The first model to be employed was Decision Tree classifier. This model is very fast to train and it is easier to interpret. But the accuracy came out to be 98.599%. The problem with Decision Tree is that it tends to overfit. To overcome this problem, Random Forest was employed. It gave an accuracy of 98.756%. Since KNN is preferred in case of large dataset, the next model used for comparison was KNN. It is robust but it took a lot of time to train on the dataset. However the accuracy came out to be 99.466%. The training accuracy was very good but the training time was compromised. Hence, Naive Bayes was employed next as it is simpler due to its quicker convergence and its ability to find the influencing features. However it performed the worst among all the aforementioned algorithms. Its accuracy came out to be 74.274%. Finally, the Stacking algorithm was generated with Decision Tree, Random Forest and KNN as its predictors and Logistic Regression as its meta-classifier. This algorithm lived up to the expectation and performed the best among all the models and gave an accuracy of 99.611% but in cost of taking the highest time to train the model.

For Deep Learning [23], Keras framework was used to solve the problem of multi-class [19]. A neural network model was created and it was ensured that the input layer had the right number of features. Figure 6 shows the diagrammatic representation of this

Table 4 Feature description

| Feature name | Description |
|-----------------|---|
| rank | Packet sequence ID |
| tpkts | Total packets sent and received by an ip |
| tbytes | Total bytes sent and received by an ip |
| spkts | Packets sent by a source ip |
| dpkts | Packets sent from destination ip |
| sbytes | Bytes sent from Source ip |
| dbytes | Bytes sent from Destination ip |
| saddr | Source ip address |
| daddr | Destination ip address |
| Tportcnt_Saddr | Total no of ports to which a source is connected in 1 pcap file |
| Tportcnt_Daddr | Total no of ports to which a destination is connected in 1 pcap file |
| Tbytes_Saddr | Total bytes sent by the source by source |
| Tbytes_Daddr | Total bytes received from the destination |
| dur | Total Duration for which an IP address participated in a connection |
| srate | Rate at which source is sending packets |
| drate | Rate at which destination is sending the packets (or source is receiving the packets) |
| rate | Rate at which data transfer is happening |
| Tsaddrcnt_Daddr | total no of destinations to which a source is connected |
| Tdaddrcnt_Saddr | total no of sources to which a destination is connected |
| load | Bits per second |
| sport | Source port |
| dport | Destination port |
| runtime | Total active flow run time |
| stime | Record start time |
| ltime | Record end time |
| Tpkt_Saddr | Total packets sent by each Source IP ADDRESS |
| Tpkt_Daddr | Total packets sent by each Destination IP ADDRESS |
| proto | Protocol |

model where X_1, X_2, \dots upto X_{23} represents the input features. The rectified linear unit activation function referred to as ReLU was implemented on the input layer. It was compared with other activation functions as well, these are further discussed in detail in this paper. When the inputs are transmitted between neurons, the weights are applied to the inputs to control the signal between the neurons of the hidden layers. These weights are represented as $W^{(1)}$, $W^{(2)}$, and $W^{(3)}$ in Fig. 6. The Softmax function was used on the final layer. It was used to classify the output into 4 classes with the help of probability distribution. The compilation of the model was done using cross-entropy as a loss argument and Adam as an optimizer. The model was fitted on the dataset using 30 epochs and then evaluated by calculating different performance measures. It was observed that if the model networks are too deep and computation is difficult, then ReLU can be preferred. Leaky ReLU can be used as a solution for the problems of vanishing gradients in ReLU but computation will be extensive. It was concluded that the activation function

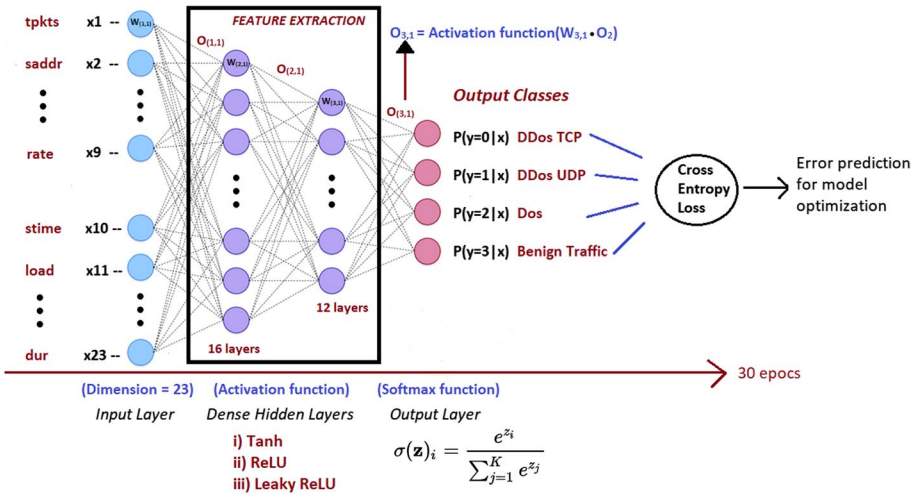


Fig. 6 Deep neural network model

plays a major role in the optimization of the problem by observing all requirements and information of the deep neural network model [24].

6 Security and Result Analysis

This paper analyzed detection of DDoS and DoS attacks in IoT in two scenarios. The first scenario utilised Machine Learning algorithms including Random Forest, Decision Trees, KNN and Naive Bayes. Initially these algorithms were used separately. Later the four algorithms- KNN, Random Forest, Decision Trees and Logistic Regression were stacked together and its performance was compared with aforementioned separate algorithms. The second scenario included neural networks for better and more informative analyses of the attack detection. Comparison was made using different activation functions and then the result was analysed. The performance metrics used were Accuracy, Undetected Rate False Alarm Rate, Precision, Recall and F1-Score. These are described below briefly.

- *Precision* Ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false alarm rate.
- *Recall* Ratio of correctly predicted positive observations to the all observations in actual class. It is also known as sensitivity.
- *F1-Score* Weighted average of Precision and Recall.
- *False Alarm Rate* The probability of false detection during testing or training. Lower value signifies better performance.
- *Undetected Rate* The ratio of number of incorrect detections to the total number of input samples.
- *Accuracy* The ratio of number of correct predictions to the total number of input samples.

Table 5 Dataset composition

| Classes | No of samples |
|----------|---------------|
| DoS | 3,00,000 |
| DDoS UDP | 2,50,000 |
| DDoS TCP | 2,50,000 |
| Normal | 2,00,000 |

Table 6 Accuracy of Machine Learning models

| Machine Learning model | Accuracy (%) |
|--|--------------|
| Decision Trees | 98.599 |
| Random Forest | 98.756 |
| KNN | 99.466 |
| Naive Bayes | 74.274 |
| Stacked (Decision Trees, Random Forest, KNN) | 99.611 |

6.1 Analysis

The composition of the dataset is shown in Table 5. The dataset for testing purpose was designed in such a way to keep the classes balanced and to reduce any misclassification that may occur due to imbalanced dataset. The performance was compared on the basis of parameters like Accuracy and Loss function.

Scenario 1: Security Analysis Using Various Machine Learning Approach.

Initially, four separate algorithms were utilised for training and testing purposes. They were Decision Trees, Random Forests, KNN and Naive Bayes. After this a Stacked algorithm comprising of three classifiers: KNN, Random Forest, Decision Trees along with Logistic Regression as a meta-classifier was utilised. The meta-classifier is used to do majority voting on the outputs of the classifiers used in the Stacking algorithm. Table 6 shows the detection accuracy values of the Machine learning algorithms employed. It was observed that all the models gave the accuracy score close to each other except for Naive Bayes hence, it was not considered in the Stacking algorithm for achieving better results. However, Stacking algorithm performed the best due to the fact that it utilised decision making capability of four algorithms at once. The meta-classifier chose the majority output of the classifiers and gave the final output, which made it more efficient than the existing separate classifier's results.

Further, performance measures like Precision, Recall, F1-Score, False Alarm Rate, and Undetected Rate values for all the Machine learning models are compared in Table 7. It is shown classwise for better comparison. Lower value of False Alarm Rate signifies the better performance of the algorithm. The same applies for Undetected Rate as well. The lower the value, the better the algorithm. Thus from the analysis carried out on the above mentioned parameters it was observed that KNN performed the best in all the parameters and was very close to the results of Stacking algorithm. KNN algorithm is simple and works really good on the non-linear type of data due to the fact that it is versatile and makes no assumption about the data. This can be seen when comparing Table 7(c) and (e). Whereas from Table 7(d), it was observed that Naive Bayes performed very poorly due to the it's nature of assuming that all the features are independent and hence was discarded while

Table 7 Classwise analysis of different machine learning models

| Classes/performance measures | Precision (%) | Recall (%) | F1-Score (%) | FAR | UR |
|---|---------------|------------|--------------|--------|--------|
| <i>(a) Decision Trees</i> | | | | | |
| DDoS TCP | 100 | 98.0 | 99.0 | 0.108 | 1.712 |
| DDoS UDP | 97.0 | 100 | 98.0 | 1.205 | 0.100 |
| DoS | 99.0 | 97.0 | 98.0 | 0.601 | 3.155 |
| Benign Traffic | 100 | 100 | 100 | 0 | 0 |
| <i>(b) Random Forest</i> | | | | | |
| DDoS TCP | 100 | 99.0 | 99.0 | 0.065 | 1.192 |
| DDoS UDP | 97.0 | 100 | 98.0 | 1.154 | 0.264 |
| DoS | 99.0 | 97.0 | 98.0 | 0.473 | 2.930 |
| Benign Traffic | 100 | 100 | 100 | 0 | 0 |
| <i>(c) KNN</i> | | | | | |
| DDoS TCP | 100 | 99.0 | 100 | 0.150 | 0.509 |
| DDoS UDP | 99.0 | 100 | 99.0 | 0.430 | 0.014 |
| DoS | 100 | 99.0 | 99.0 | 0.136 | 1.337 |
| Benign Traffic | 100 | 100 | 100 | 0.004 | 0.006 |
| <i>(d) Naive Bayes</i> | | | | | |
| DDoS TCP | 76.0 | 20.0 | 31.0 | 2.145 | 80.361 |
| DDoS UDP | 51.0 | 100 | 67.0 | 32.889 | 0.006 |
| DoS | 100 | 81.0 | 90.0 | 0 | 18.599 |
| Benign Traffic | 100 | 100 | 100 | 0.022 | 0 |
| <i>(e) Stacked (Decision Trees, Random Forest, KNN)</i> | | | | | |
| DDoS TCP | 100 | 99.0 | 100 | 0.150 | 0.509 |
| DDoS UDP | 99.0 | 99.0 | 99.0 | 0.430 | 0.014 |
| DoS | 100 | 99.0 | 99.0 | 0.136 | 1.337 |
| Benign Traffic | 100 | 100 | 100 | 0.004 | 0.006 |

choosing the algorithms for Stacking purpose. The Stacking algorithm gave satisfactory results as it utilised the voting method for better predictions. It can also be observed from Table 7 that while classifying the Benign Traffic from Attack Traffic all the algorithms had 100% score in Precision, Recall, and F1-Score columns. Thus, the features chosen for the algorithms performed truly well.

The importance of each feature on the output class is depicted in Fig. 7. Random Forest algorithm was implemented for the purpose of generating this graph. It was done in order to understand which features were more significant than others while making a decision during splitting of a node while training the model. Graph shows that "ltime" has the highest importance and "drate" has the least importance on the output class.

The impact of feature Tportent_Daddr on the output classes is shown in Fig. 8.

Deep learning models are preferred over Machine Learning models because they can solve a complex query involving a huge amount of data. Since data traffic increases with increase in duration of time, hence deep learning models will give better classification results in systems with high internet usage. Deep learning automatically tries to learn features which are important for the classification purpose without any human intervention. In Machine learning models, the features are provided manually. Thus, a comparison was

Fig. 7 Feature importance graph

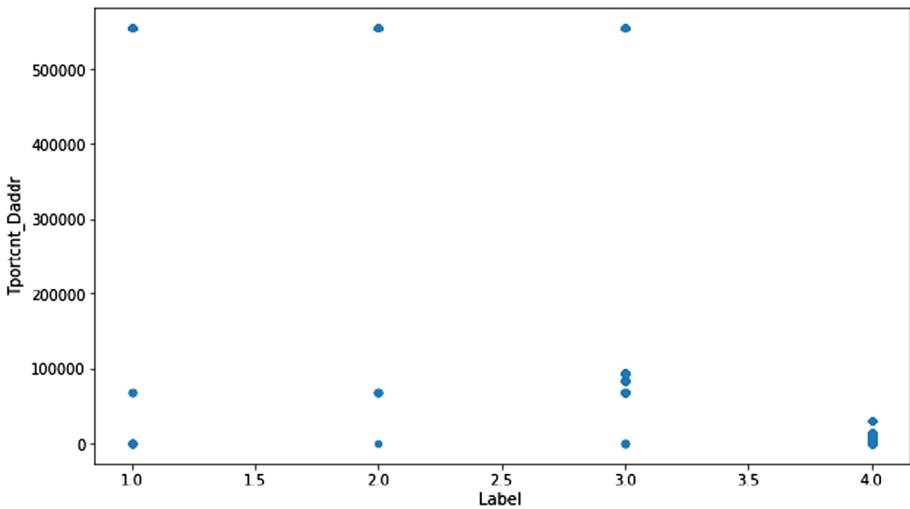
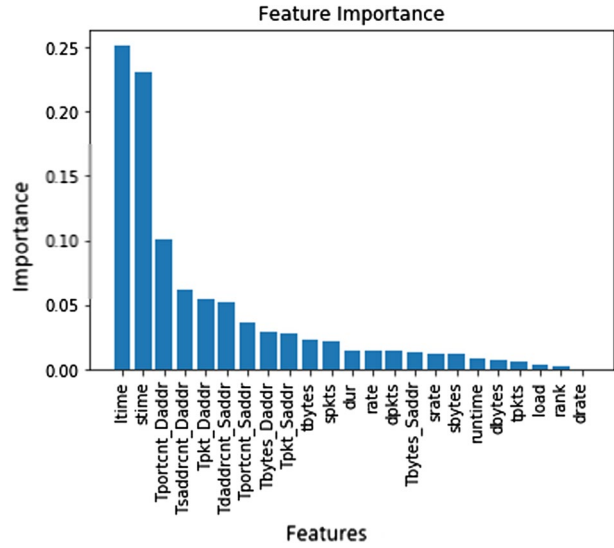


Fig. 8 Impact of Tportcnt_Daddr on label

made with Deep learning to check whether the features chosen by this model outperforms Machine learning results. The corresponding result analysis is given for the following section.

Scenario 2: Security Analysis Using Deep Neural Networks.

Deep learning methodology involves multiplication of the input variable with a weight, then a bias is added to the product. An activation function is then applied on the result. Activation functions play an important role in deep learning model, without it the neural network would just perform like linear regression model. The network will use forward and back propagation methods without any non linear transformation. Back propagation

Table 8 Accuracy of Deep Learning Algorithm using different activation functions

| Activation functions | Accuracy (%) |
|-----------------------------|--------------|
| Hyperbolic Tangent function | 99.483 |
| ReLU | 99.529 |
| Leaky ReLU | 99.438 |

Table 9 Classwise analysis of different activation functions

| Classes/performance measures | Precision (%) | Recall (%) | F1-Score (%) | FAR | UR |
|--|---------------|------------|--------------|-------|-------|
| <i>(a) Hyperbolic Tangent Function</i> | | | | | |
| DDoS TCP | 98.390 | 99.989 | 99.183 | 0.003 | 1.609 |
| DDoS UDP | 99.568 | 99.863 | 99.715 | 0.045 | 0.431 |
| DoS | 99.983 | 98.427 | 99.199 | 0.686 | 0.016 |
| Benign Traffic | 99.992 | 100 | 99.996 | 0 | 0.007 |
| <i>(b) ReLU Function</i> | | | | | |
| DDoS TCP | 99.081 | 99.479 | 99.279 | 0.172 | 0.918 |
| DDoS UDP | 99.562 | 99.869 | 99.715 | 0.043 | 0.437 |
| DoS | 99.564 | 98.979 | 99.270 | 0.441 | 0.435 |
| Benign Traffic | 99.997 | 100 | 99.998 | 0 | 0.002 |
| <i>(c) Leaky ReLU Function</i> | | | | | |
| DDoS TCP | 98.322 | 99.967 | 99.138 | 0.010 | 1.677 |
| DDoS UDP | 99.470 | 99.865 | 99.667 | 0.044 | 0.529 |
| DoS | 99.968 | 98.304 | 99.129 | 0.740 | 0.031 |
| Benign Traffic | 99.997 | 99.138 | 99.993 | 0.002 | 0.002 |

in neural networks is used to calculate and the error values related to the weights. Table 8 shows the accuracy of the model when employed with different activation functions. It was observed that ReLU function gave the best accuracy results.

Detailed information about how the activation functions performed in terms of Precision, Recall, and F1-Score is shown in Table 9.

The above analysis was performed on the activation functions which were used for the hidden layers of the deep learning model. First choice of the activation function was tanh. It was chosen because it produces zero centred output thereby aiding the back-propagation process. The benefit of using tanh function was that it mapped negative inputs (the values of features were in range -1 to 1 after normalization) strongly negative and zero near to it. Since, it had the problem of vanishing gradient as well as production of dead neurons, another function called ReLU was applied to the hidden layers. This function is faster in computation and also used in almost all the neural networks due to the fact that both the function and its derivative are monotonic. Hence, it is differentiable and the range of the function lies between 0 to infinity. The issue with ReLU is that it maps all the negative values to zero which becomes problematic in the case if the features contain negative values hence Leaky ReLU was tried, which was an attempt to solve the problem of dying ReLU. The range of Leaky ReLU is minus infinity to infinity which should have benefitted this case but still ReLU performed better because of the following reasons:

- The parameters for Leaky ReLU does not change during training phase. It is predefined.
- The Leaky ReLU function is not differentiable at 0, which may cause values to change abruptly during backpropagation.

From Table 9(b), it can also be interpreted that the ReLU's FAR and UR values were the lowest making it the better performer among the rest of the activation functions. The ReLU in the hidden layer and Softmax in the outer layer makes a the best pair in this case for the multiclass classification. The undetected rate in all the activation function was highest for the DDoS TCP class making it the most tricky attack with variable attacking pattern.

The Loss and Accuracy graphs of different activation functions used in the Deep Learning model are plotted in Fig. 9.

The categorical cross-entropy was used as the loss function for this model. It was chosen because of it's property to quantify the difference between two probability distribution. First probability distribution is the one predicted by the model and second is the true distribution of the classes. In the above model the predicted probability distribution is given by the softmax layer applied on the outer layer. It results in values between 0 and 1 for each of the classes which all sum up to 1. For the calculation of the error values related to the weights, the back-propagation algorithm of the artificial neural network is applied. It is necessary to determine the correct optimization strategy to minimize the error rate. Thus, Adam optimizer was used as it is computationally efficient and has an adaptive learning rate.

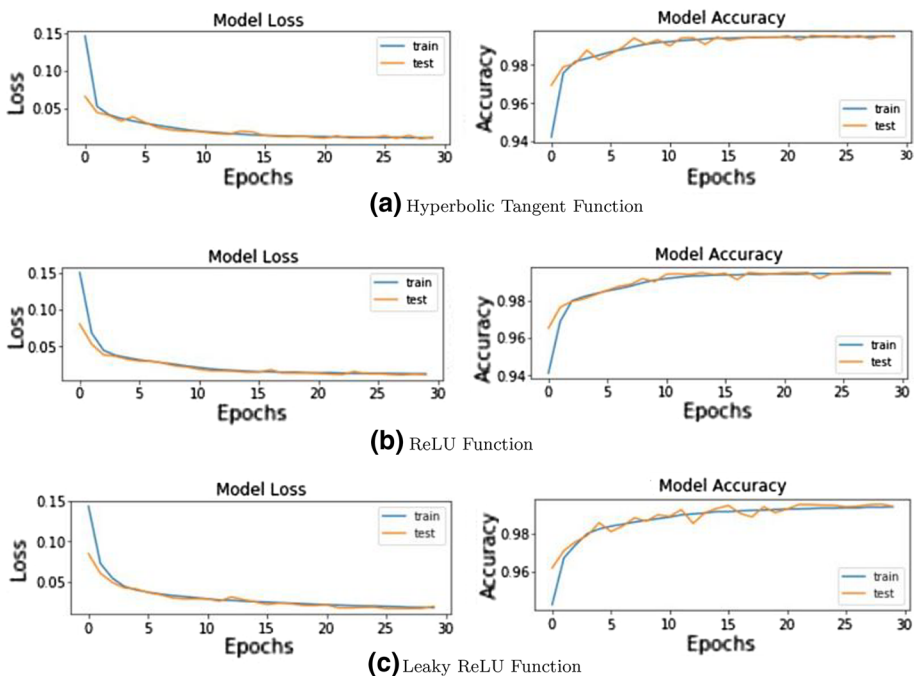


Fig. 9 Loss and accuracy graph using different activation functions

Table 10 Comparison between existing solution and proposed solution

| Existing works | Model used | Real testbed | Hybrid model | Feature selection | Accuracy (%) |
|--|--|--------------|--------------|-------------------|--|
| Machine Learning DDoS Detection for Consumer Internet of Things Devices [6] | Machine Learning | Yes | No | Yes | KN = 99.9 LSVM = 99.1 DT = 99.9 RF = 99.9 NN = 99.9 |
| HADS: Hybrid Anomaly Detection System for IoT [7] | Machine Learning (Hybrid) | Yes | Yes | Yes | HDM = 98.70 SOM = 98.55 Isolation Forest = 98.08 GMM = 96.03 OCSVM = 98.26 ANN = 95.6 |
| Artificial Neuron Network Implementation in Detection and Classification of DDoS Traffic [8] | Deep Learning | No | No | No | |
| Attack Classification Analysis of IoT Network via Deep Learning Approach [9] | Deep Learning | No | No | Yes | For different datasets UNSW-NB15 = 94.16 CIDDS-001 = 99.99 GPRS-WEP = 82.89 GPRS-WPA2 = 94.0 |
| Bonnet Detection in the Internet of Things using Deep Learning Approaches [10] | Deep Learning | Yes | No | Yes | Mirai = 99.998992 UDP = 98.582144 ACK = 93.765198 DNS = 98.488289 |
| SAD-IoT: Security Analysis of DDoS Attacks in IoT Networks | Machine Learning (Hybrid) Deep Learning | Yes | Yes | Yes | DT = 98.599 RF = 98.756 KNN = 99.466 NB = 74.274 Stacking = 99.611 ANN (with ReLU) = 99.529 |

Bold signifies proposed solution

6.2 Comparison Between Existing Solution and Proposed Solution

A detailed comparison is shown in Table 10. The criterias chosen for this were based on the drawbacks of other existing works. This paper tried to overcome these drawbacks by using both Machine Learning as well as Deep Learning model to identify which of the model performs better. Both Dos and DDoS attacks were considered for testing using a real time testbed for better analysis.

7 Conclusion and Future Work

In this paper, the Machine Learning and Deep Learning algorithms and features which are significant for the detection of DoS and DDoS attacks in a network, are analyzed. The findings of the research suggested that KNN performed quite close to the Stacking algorithm, which actually turned out to be the best performer in every aspect among all Machine Learning algorithms mentioned in this paper, but they took a lot of time for training the model. Random Forest and Decision Trees gave similar detection accuracy results which were quite good considering the time taken to train the model, along with the parameters which were considered for performance comparison. As for the Deep Learning model, Deep Neural Networks was implemented using different activation functions. ReLu activation function in the inner layers pairing with Softmax in the outlier layer turned out to be the best performer among others and it's detection accuracy was very close to the Stacking algorithm. Hence, it can be concluded that both Machine Learning algorithms and Deep Learning models can be employed for detection purpose but Deep Learning models are preferred with systems having abundant resources and huge data transfer platform for security purposes as they utilise more resources as they learn over time and will be ideal for detecting new threats. Machine Learning models can be ideally implemented in systems with less resources as their resource utilisation is less and they are fairly consistent and they perform better in relatively less data traffic.

References

1. Balaji, S., Nathani, K., & Santhakumar, R. (2019). IoT technology, applications and challenges: A contemporary survey. *Wireless Personal Communications*, 108(1), 363–388.
2. Tweneboah-Koduah, S., Skouby, K. E., & Tadayoni, R. (2017). Cyber security threats to IoT applications and service domains. *Wireless Personal Communications*, 95(1), 169–185.
3. Harbi, Y., Aliouat, Z., Harous, S., Bentaleb, A., & Refoufi, A. (2019). A review of security in internet of things. *Wireless Personal Communications*, 108(1), 325–344.
4. Verma, A., & Ranga, V. (2020). Machine learning based intrusion detection systems for IoT applications. *Wireless Personal Communications*, 111(4), 2287–2310.
5. Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100, 779–796.
6. Doshi, R., Apthorpe, N., & Feamster, N. (2018). Machine learning ddos detection for consumer internet of things devices. *IEEE Security and Privacy Workshops (SPW)*. IEEE.
7. Bhatt, P., & Morais, A. HADS: Hybrid anomaly detection system for IoT environments. In *2018 international conference on internet of things, embedded systems and communications (IINTEC)* (pp. 191–196). IEEE.

8. Peraković, D., Periša, M., Cvitić, I., & Husnjak, S. Artificial neuron network implementation in detection and classification of DDoS traffic. In *2016 24th Telecommunications Forum (TELFOR)* (pp. 1-4). IEEE.
9. Tama, B., & Rhee, K. (2017). Attack classification analysis of IoT network via deep learning approach. *Research Briefs on Information & Communication Technology Evolution: ReBICTE*, 3, 1–9.
10. McDermott, C. D., Majdani, F., & Petrovski, A. V. Botnet detection in the internet of things using deep learning approaches. In *2018 international joint conference on neural networks (IJCNN)* (pp. 1-8). IEEE.
11. Rahal, R., Korba, A., & Ghoualmi-Zine, N. (2020). Towards the development of realistic DoS dataset for intelligent transportation systems. *Wireless Personal Communications*, 115, 1415–1444.
12. Kumar, U., Navaneet, S., Kumar, N., & Chandra Pandey, S. Isolation of ddos attack in iot: A new perspective. *Wireless Personal Communications*, 114, 2493–2510.
13. De Donno, M., Dragoni, N., Giaretta, A., & Spognardi, A. (2018). DDoS-capable IoT malwares: Comparative analysis and Mirai investigation. *Security and Communication Networks*, 2018, 1–31.
14. Yusof, A., Udzir, N., & Selamat, A. (2016). *An evaluation on KNN-SVM algorithm for detection and prediction of DDoS attack*. Cham: Springer.
15. Lakshminarasimman, S., Ruswin, S., & Sundarakantham, K. Detecting DDoS attacks using decision tree algorithm. In *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)* (pp. 1-6). IEEE.
16. Fouladi, R. F., Eren Kayatas, C., & Anarim, E. Frequency based DDoS attack detection approach using naive Bayes classification. In *2016 39th International Conference on Telecommunications and Signal Processing (TSP)* (pp. 104-107). IEEE.
17. Ouyang, Z., Sun, X., Chen, J., Yue, D., & Zhang, T. (2018). Multi-view stacking ensemble for power consumption anomaly detection in the context of industrial internet of things. *IEEE Access*, 6, 9623–9631.
18. Blanco, R., Malagon, P., Cilla, J., & Moya, J. (2018). Multiclass network attack classifier using cnn tuned with genetic algorithms. In *Optimization and simulation (PATMOS)*. IEEE.
19. Manaswi, N. (2018). *Understanding and working with Keras. Deep learning with applications using python* (pp. 31–43). Berkeley, CA: Apress.
20. Farsad, N., & Goldsmith, A. (2017). Detection algorithms for communication systems using deep learning. [arXiv:1705.08044](https://arxiv.org/abs/1705.08044).
21. Jagannath, J., Polosky, N., Jagannath, A., Restuccia, F., & Melodia, T. (2019). Machine learning for wireless communications in the Internet of Things: A comprehensive survey. *Ad Hoc Networks*, 93, 101913.
22. Chu, X., Ilyas, I. F., Krishnan, S., & Wang, J. (2016). Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 international conference on management of data* (pp. 2201-2206).
23. Yavuz, F., Ünal, D., & Gül, E. (2018). Deep learning for detection of routing attacks in the internet of things. *International Journal of Computational Intelligence Systems*, 12(1), 39–58.
24. Agostinelli, F., Hoffman M., Sadowski, P., & Baldi, P. (2014). Learning activation functions to improve deep neural networks. [arXiv:1412.6830](https://arxiv.org/abs/1412.6830).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Prahlad Kumar is an undergraduate in Computer Science from International Institute of Information Technology, Naya Raipur. He will graduate with B.Tech. in Computer Science in 2021. His projects are focused on Android development, Machine Learning and Web development. His interests include Machine Learning, Internet of Things and full stack development. He has been working with CGNet Swara and has been developing software's for the organization since May, 2020.



Harnoor Bagga completed her B.Tech. degree in Computer Science and Engineering from Vellore Institute of Technology in 2020. Her projects include working on Python, Django, Hadoop, and MySQL. She has worked as an intern in Larsen & Toubro Limited, Chennai and Schlumberger Pune, India. Her work area as intern included Software Development in GoLang and Azure DevOps. Her interest areas include Internet of Things (IoT), Machine and Deep Learning, NLP and DevOps.



Bhuneshwar Singh Netam is an undergraduate in Computer Science from International Institute of Information Technology, Naya Raipur. He will graduate with B.Tech. in Computer Science in 2021. His projects are focused on Web Development, Machine Learning and Image Processing. His interests include Internet of Things, Network Security and full stack development. He has worked as an intern in Web Addiction (startup incubated by 36inc Raipur) and has been developing Web Sites.



Venkanna Uduthalapally obtained his Ph.D. degree by the National Institute of Technology, Tiruchirappalli (NITT), in 2015. Since 2005, he has been in the teaching profession and currently he is an Assistant Professor in the Department of Computer Science and Engineering, Dr. Shyama Prasad Mukherjee International Institute of Information Technology, Naya Raipur (IIIT-NR). He has 8 years of teaching and research experience. His research interests include Internet of Things (IoT), Software Defined Networks, Network Security, Wireless Ad hoc, and Sensor network. He has to his credit of publishing 30 research papers including IEEE Transactions.