COMPARISON OF DEEP LEARNING MODELS FOR INTRUSION DETECTION ON NBAIOT DATASET

Session 2016-2020

By

ATTA UR REHMAN

OSAMA AHMAD AFRIDI

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
August, 2022

COMPARISON OF DEEP LEARNING MODELS FOR INTRUSION DETECTION ON NBAIOT DATASET

Session 2016-2020

By

ATTA UR REHMAN

OSAMA AHMAD AFRIDI

Supervised by

MR. QAZI HASEEB YOUSAF

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
August, 2022

COMPARISON OF DEEP LEARNING MODELS FOR INTRUSION DETECTION ON NBAIOT DATASET

By

Atta ur Rehman (7698)
Osama Ahmad Afridi (7625)

CERTIFICATE

A THESIS SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIRMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

We accept this dissertation as conforming to the required standards

(Supervisor) MR. QAZI HASEEB YOUSAF	(Internal Examiner)
(External Examiner)	(Head of the Department)
(Coordinator FYP)	(Approved Date)

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
August, 2022

Dedication

We dedicate this thesis report work to all our teachers, parents, and supervisor. Who motivated, and helped us in completing the project. Without their support, We would not have done it.

Declaration

We hereby declare that We are the sole authors of this thesis. The work submitted in this thesis is the result of our own research, except where otherwise stated.

Atta ur Rehman
Osama Ahmad Afridi
August 2022

Acknowledgment

First and foremost, we express my gratitude to Allah, the Most Gracious and Merciful. we would like to offer our heartfelt thanks to Mr. Qazi Haseeb Yousuf, research supervisor, for his patience and advice. We would also want to thank our parents, friends, and teachers from high school, college, and university. We would not have gotten to this point in our lives without their advice and support. We are also grateful to our brothers and sisters for their unwavering love and support through a challenging time in our lives. We would want to use this occasion to express our gratitude to our other professors for their encouragement, inspiration, kind assistance, and direction. We are grateful to everyone who has helped in any manner with their genuine and persistent support. This thesis work is dedicated to our parents. All our lives, he has been a source of inspiration, unselfish love, and dedication. This study project is dedicated to all of our teachers, parents, and supervisors. Who inspired us and assisted us in finishing my research. We would not have succeeded without their help.

Abstract

The identification of botnet attacks in Internet of Things (IoT) networks has received a lot of attention thanks to deep learning (DL). The conventional centralised deep learning method, however, cannot be utilised to identify the previously unidentified (zero-day) botnet attack without violating the users' right to data privacy. In order to prevent data privacy leakage in IoT-edge devices, we suggest the federated deep learning method in this article. This approach uses an ideal deep neural network (DNN) architecture to classify network traffic. The autonomous training of the DNN models in various IoT edge devices is remotely coordinated by a model parameter server, and local model updates are combined using the federated averaging technique. The Bot-IoT and N-BaIoT data sets are used to mimic zero-day botnet attack scenarios in IoT edge devices. The results of the experiments demonstrate that the FDL model detects zero-day botnet assaults with good classification performance. Ensures data privacy and security. Has minimal communication overhead. Needs less memory for training data storage. Has low network latency. Therefore, in this application scenario, the FDL method outperformed CDL, localised deep learning, and distributed deep learning methods.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AGDs	Algorithmically Generated Domains
ANN	Artificial Neural Networks
C&C	command and control
CNN	Convolutional Neural Network
DL	Deep Learning
DDoS	Distributed Denial of Service
FCNN	Fully Connected Neural Networks
IDS	Intrusion Detection Systems
IoT	Internet of Things
PSD	Power Spectral Density
RGB	Red Green Blue
RNN	Recurrent Neural Network
SHN	Smart Home Network
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
HIDS	Host Intrusion Detection System
NIDS	Network Intrusion Detection System
ML	Machine Learning
SVM	Support Vector Machine
UWSN	Underwater Sensory Network

Chapter 1

Introduction

1.1 Overview

The use of neural network-based autoencoders in anomaly identification issues has grown recently. The identification of botnet attacks in IoT networks has received a lot of attention due to DL. However, it is impossible to detect the previously undiscovered (zero-day) botnet assault using the conventional centralised DL technique without violating the users' right to data privacy [1]. There are two categories of intrusion detection in IoT networks: (i) anomaly detection-based and (ii) malicious traffic signatures-based. In the first way, IoT device normal profiles are discovered using a variety of techniques, and any significant departure from these normal profiles is assumed to be malicious. In the latter strategy, the system executes detections utilising known botnet or malicious traffic signatures or characteristics stored in a database. Unknown botnets or assaults can be stopped using the first strategy. However, there is a chance that infected devices was not found for instance, when malicious traffic imitates a device's legitimate traffic. The latter method could be quite effective in spotting known assaults [2]. The ability to identify unidentified threats makes anomaly detection-based techniques more popular. In this context, a variety of anomaly detection issues have been solved with remarkable success using neural network-based autoencoders [3]. The vulnerabilities of each component technology in an IoT ecosystem may aggregate due to the growing diversity of devices, networks, and services, creating new risks and attack vectors [4–7]. This puts lives and property in danger in addition to the gadgets. Take, for instance, these most recent reports. Critical networks were subjected to DDoS assaults from a sizable collection of internet-connected devices [8]. Another troubling illustration is the Mirai botnet-based assault, which used IoT devices to target several well-known web-based services and platforms and render them unreachable [9]. Critical infrastructure, banks, healthcare facilities, smart cities, and the internet of linked things and vehicles could all be the targets of such DDoS assaults [10–12]. For instance, it has been demonstrated that linked automobiles are susceptible to being commandeered by a remote, hostile attacker who may halt a moving vehicle or lock/unlock doors [13]. Another concerning instance involves the discovery of security and privacy weaknesses in smart (and connected) toys that might be used by an

adversary to maliciously manipulate the toys or seriously violate users' privacy [14]. In response to these instances, academic and industrial researchers are intensifying their efforts to create cutting-edge IoT intrusion detection technologies that will protect IoT from various sorts of intrusions. Regarding their deployment approach, the different IDSs may be generically categorised as either centralised, distributed, or hybrid [15]. The border router or network edge devices house the detection modules or agents in the centralised IDS installation scheme e.g. mobile edge computing servers deployed at eNodeBs, other base stations, or roadside units. On the other hand, with the dispersed method, the IoT devices host the detection modules or agents. The hybrid approach combines the two, with the detection agents or modules being housed at both IoT devices and the network edge in a hierarchical distribution. The capacity to effectively identify threats coming from the outside is one of the key benefits of a centralised IDS housed at the network edge, along with the availability of greater processing, communication, and storage capabilities e.g., via the Internet.

1.2 Background

In order to monitor security and protection, IDS are the systems that automatically identify and examine the strange and scary behaviour within a host or network. The invasion is simply found using intrusion detection. It sometimes specifies the rules for evidence in certain circumstances. An intrusion occurs when a network or computer deviates from regular operation and is used as a pretext for an attempt to steal or corrupt network data [16], [17]. Today, individuals exchange and keep private information on the Internet and other networks. IDS is a cybersecurity tool that a firewall and antivirus software employ, according to [18]. IDS have been actively studied to help the conventional network resist malicious attacks. In literature quite a number of the intrusion detection techniques are developed based on machine learning techniques, based on the assumption that the patterns of the attack packets differ from those of the normal packets. In ANN and SVM are applied to the intrusion detection, using a statistical modeling on a packet data. In a frequency-based encoding method is used for a packet feature in ANN and SVM. The works are based on supervised machine learning techniques, and, thus a number of labeled data sets are required in the training [19]. As we will compare deep learning models for intrusion detection in a system on as NBaIOT dataset. This dataset addresses the lack of public botnet datasets, especially for the IoT. It suggests real traffic data, gathered from 9 commercial IoT devices authentically infected by Mirai and Bashlite. The dataset characteristics are it is a multivariate and sequential dataset and it has real number of attributes containing 115 attributes.

1.3 Problem Statement

There are a variety of ML techniques that is use for the detection of IDS attack and determine its kind. We are going to use DL models to try to find a better solution to these issues. According to previous research, with binary classification of ML, DL, the results are good but with multi-classification, the results are not up to the requirements where we face overfitting and underfitting problems with ML algorithms. we will use DL models to look for a better solution to these problems.

1.4 Proposed Solution

A proposed technique to implement DL models ANN with different architecture as we will also implement CNN and CNN+SVM and comparing these models on NBaIOT dataset.

1.5 Aims and Objectives

The aim and objetive of this study are:

1.5.1 Aims

The aim of this research is to design a model on NBaIOT dataset that accurately classify the IDS using ML techniques. And based on comprehensive analysis of various ML techniques. Nominate the ML technique to give us best result with better accuracy and result.

1.5.2 Objective

- To present the model that classifies the Intrusion Detection and all the others to improve the early prediction and care.
- To present a comparative study between the proposed model and other models to find a better and most efficient methodology for intrusion detection on NBaIot dataset.

1.6 Thesis Outline

This thesis report is having a total of 5 Chapter. This chapter provides an insight into brief comparison of deep learning models for intrusion detection on NBaIot dataset discussing the problem as well, background of intrusion detection on NBaIot dataset, problem statement of the research, proposed solution for the problem of research, scope and

objectives of the research were also discussed. Chapter 2 is comprised of the literature review of comparison of deep learning models for intrusion detection on NBaIot dataset discussing the previous work done including the techniques used, evaluation metric used and results. Chapter 3 The methodology used in the experimentation process, explain the subsection phases of your research work, about the dataset which is being used, and employed technique also the assesment criteria to be used in this research will be discussed with a properly. Chapter 4 provides the findings from the experimentation process and it also discusses the whole selection process and architecture of the individual technique, tabular and graphical representation of the results will also be provided in that chapter. In the last chapter i.e., Chapter 5 provides the conclusion of the whole thesis report alongside with the future work, and also the threat to validity of this research are also discussed.

Chapter 2

Literature Review

In this chapter, a detailed literature review is described which explain the background of the whole research. A proper understanding of the research techniques, and results obtained from the previous work done helps to elaborate the idea of the research.

2.1 Machine Learning Techniques

ML in the field of computer science is concerned with examining and deciphering data patterns and structures to enable learning, reasoning, and decision-making without the involvement of humans. Prediction, classification, and feature selection are all possible in bioinformatics when using ML algorithms. These methods present both important obstacles and applications with great potential. Medical imaging, the natural language processing of medical records, and genetic data are the three principal applications of machine learning. These fields frequently centre on diagnosis, detection, and prediction.

2.2 Classification Techniques

The IDS on IOT is a generalized research topic and it has gotten much attention in the last two to three decades. The DL techniques are used to predict conventional network resist malicious attacks.

2.3 Related Work

Over the past years, there are a large number of works focusing on botnet detection. According to the resource consumption, they can be classified into two categories, i.e. traditional botnet detection and IoT botnet detection. The traditional botnet detection approaches dedicate to mitigate the botnets constructed by infecting resourceful machines, e.g., web servers, laptops, desktops and so forth. The IoT botnet detection

approaches are developed to detect the botnets consisting of resource-limited IoT devices. In the following, we will summarize these works.

Traditional detection techniques primarily try to spot the coordinated actions or characteristics of bots that are part of the same botnet. The host group that queries the same domain is the major topic of Choi et al's. [20] study. They evaluate a group's resemblance, regularity, and intensity to determine whether or not it is a botnet. This method cannot identify a botnet with several random domains since it focuses on a single domain.

In order to further enhance it, they [21] cluster connected domains. To cluster domains, it makes use of DNS lexical characteristics, query, and response features. The host groups will then be generated using domain groups rather than a single domain.

Gu et al. [22] similarly believe that bots belonging to the same botnet are more likely to have comparable behaviour. In order to capture the spatial-temporal correlation and similarity in network flow, they suggest a technique to detect botnets. They test it using actual data, and the outcome demonstrates good accuracy and a low false-positive rate.

Kwon et al. [23] suggest using the PsyBoG system to examine a bot host's periodicity. They use the PSD signal processing method to do analysis. Many studies focus on the arbitrary DGA-generated domains since attackers frequently use the domain generation algorithm (DGA) to increase the resilience of their botnet CC channels.

Yadav et al's. [24] strategy for detecting domain fluxes involves spotting lexical patterns that are produced by computers rather than by actual people. Phoenix [25] is a system that uses string and IP attributes to distinguish between DGA and non-DGA domains. It can identify unknown produced domains to existing groups and find groupings of DGA domains.

A multi-layer semi-supervised framework for IDS is proposed in [26] based on pure cluster extraction, pattern discovery, fine-grained classification and model updating. They defined "pure" clusters as those in which a vast majority of their samples belong to the same class. They employed a hierarchical semi-supervised k-means algorithm to learn the pure clusters. The samples which did not fall into any pure cluster are then fed into the pattern discovery module in which a clustering-based method is used to find if the patterns are known (normal or intrusions) or unknown patterns. The fine-grained classification module then acts to classify the discovered unknown patterns. However, this

step may require manual inspection to determine the class of the unknown patterns. The task of re-training any modules due to changes in traffic distribution is handled by the model updating module. Their experiments were based on the old KDDCUP99 dataset.

Another work addressing the problem of intrusions in a critical infrastructure monitoring application using wireless sensor networks is of [27] in which the authors investigated the development of an IDS based on a Reinforcement Learning (RL) algorithm called Q-Learning. However, their use of Q-learning as the RL algorithm presents some limitations. For example, the Q-tables could grow very large with a large number of states and/or actions. This would imply that with an increase in the number of states, the memory size required to save and update the Q-table would increase. Moreover, an extremely large amount of time would be required to explore each state for building the Q-table.

Mirai is thoroughly analysed by Antonakakis et al. [28] over the course of seven months. They look at how it started, how it developed, which hosts it infected and attacked, and how it operated. They suggest that weak security in IoT devices will cause significant problems on the Internet.

DDoS assaults will be launched by IoT-based botnets with previously unheard-of bandwidth volume. The supply of the attack codes must be stopped by the discovery of an IoT-based botnet in order to reduce the harm caused by these attacks. the broad use of the domain creation algorithm in botnets as a driving force A lightweight method called ConnSpoiler was developed by Yin et al's. [29]. to quickly identify the stream of AGDs and identify IoT-based botnets. ConnSpoiler can operate well on resource-constrained IoT devices since it only requires a little amount of system resources to operate. ConnSpoiler has a high likelihood (approximately 94%) of identifying infection before the compromised devices connect to CC servers thanks to the use of a potent statistical method called threshold random walk, which can assist to stop further assaults.

A botnet early detection system for the Internet of things called EDIMA introduced by Kumar et al's [30]. will be put at edge gateways in residential networks. In EDIMA, an unique two-stage machine learning (ML)-based detector is included, which first uses ML algorithms to classify aggregate traffic before using ACF-based tests to identify specific bots. EDIMA delivers strong bot scanning detection accuracies with a very low false positive rate, according to performance evaluation data.

IoT devices in SHN are highly vulnerable to complex botnet attacks. Popoola et al. [31] investigate the effectiveness of Recurrent Neural Network (RNN) to correctly classify network traffic samples in the minority classes of highly imbalanced network traffic data. Multiple layers of RNN are stacked to learn the hierarchical representations of highly imbalanced network traffic data with different levels of abstraction.

Summary

In this chapter, we cover the work and contributions of the different researchers in this field. One of the challenging areas for researchers to defend network infrastructure from adversary activities involves detecting abnormal traffic in the Internet of Things (IoT), which is used for a variety of applications, including home and wearable devices, network applications, and even self-driven vehicles. There are flaws in a number of network systems that let hackers exploit malicious communications to gain unauthorised access. Attacks like Distributed Denial of Service (DDoS) and Service Scans necessitate the use of a special automatic system that can spot anomalous traffic patterns early on in order to prevent system damage. There are numerous automatic methods that can find unusual traffic.

Chapter 3

Research Methodology

This chapter discusses the basic methodology for research as well as a summary of the technique used in this study, which is shown with a diagram to help the reader understand the experiment's flow. The methods, methodologies, and data sets used in the research are explained.

3.1 Methodology

To begin research approach, we will acquire a dataset in google collab using pytorch library and will perform a noise cancellation on it after performing noise cancellation we will normalize the data in a dataset using python. After that we will compare different deep learning models on a dataset. This research aims to present the performance analysis of ANN, CNN and CNN+SVM on the NBaIOT dataset. The complete research is prepared via the procedure shown in Figure 3.1. All experimentation is performed on open source ML and tools Excel. After the dataset, the preprocessing steps are applied on each dataset class for removing the extra noise from the dataset to boost the performance of the ML techniques. Then ML techniques are applied to each dataset class, and the outcomes are assessed using different assessment measures to show the better performance of an individual technique. Three assessment measure named precision [32], recall [32], FM [32] and accuracy [33].

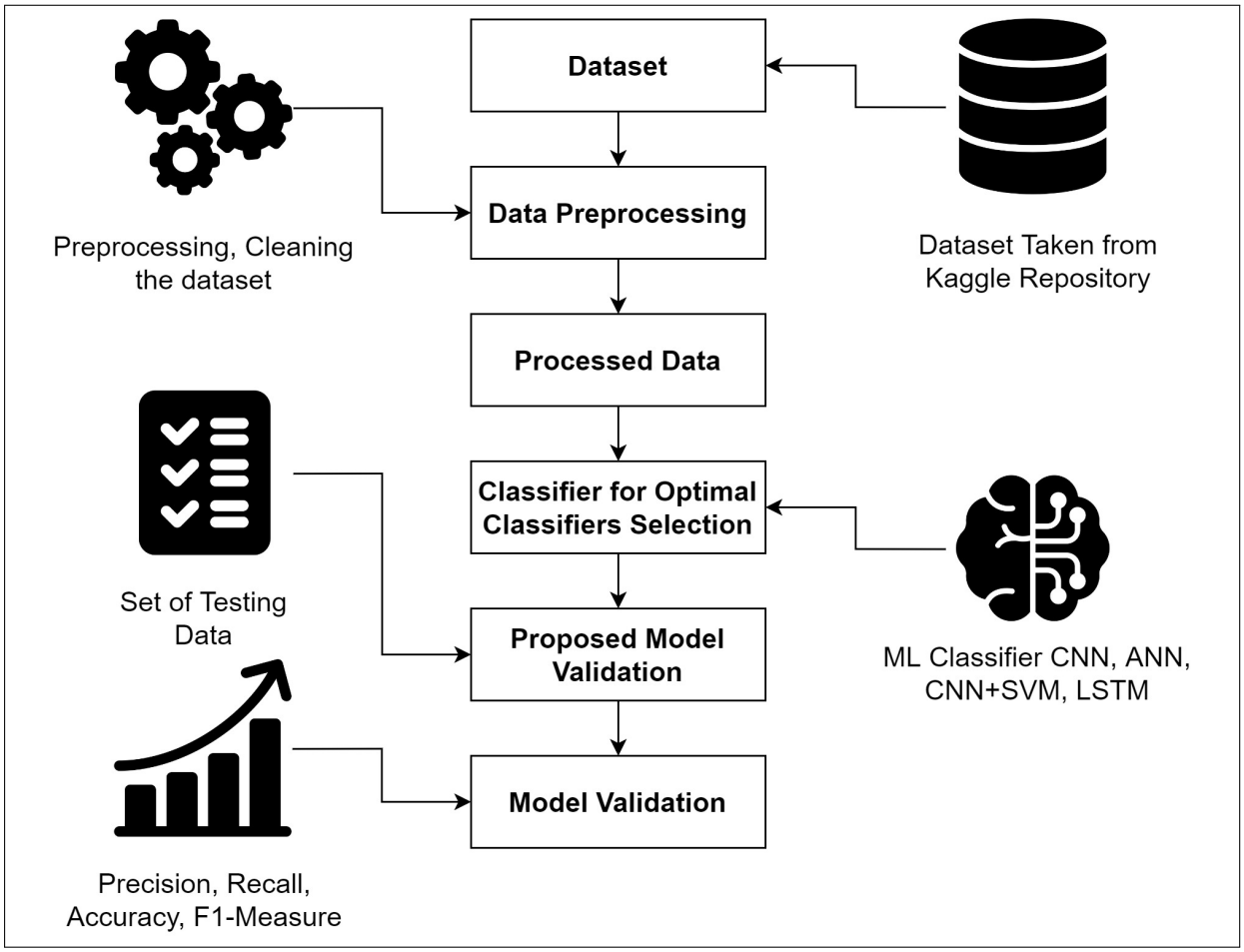

Figure 3.1: Methodology Work-Flow

3.1.1 Datasets Description

Each data set consists of some features and attributes. By using these features and attributes to determine the performance of the model. The datasets that we are going to use have a total of 9 devices and 115 features. For each of the 9 IoT devices, we trained and optimized our model. The following 9 devices are shown in Table 3.1. The 115 features contain the machine weight, mean, and variance. H: Stats summarizing the recent traffic from this packet's host (IP), HH stats summarizing the recent traffic going from this packet's host (IP) to the packet's destination host, HpHp stats summarizing the recent traffic going from this packet's host+port (IP) to the packet's destination host+port, HH_jit stats summarizing the jitter of the traffic going from this packet's host (IP) to the packet's destination host, Time-frame how much recent history of the stream is capture in these statistics L5, L3, L1, The statistics extracted from the packet stream, weight the weight of the stream (can be viewed as the number of items observed in recent

history), radius: The root squared sum of the two streams' variances, magnitude the root squared sum of the two streams' means, cov an approximated covariance between two streams, pcc an approximated covariance between two streams The dataset taken from an online repository called Kaggle [34] Figure 3.2 represent the datasets.

Table 3.1: Samples of IoT Intrusion Dataset

Device ID	**Device Name**
1	Danmini_Doorbell
2	Ecobee_Thermostat
3	Ennio_Doorbell
4	Philips_B120N10_Baby_Monitor
5	Provision_PT_737E_Security_Camera
6	Provision_PT_838_Security_Camera
7	Samsung_SNH_1011_N$_W ebcam$
8	SimpleHome_XCS7_1002_WHT_Security_Camera
9	SimpleHome_XCS7_1003_WHT_Security_Camera

3.1.2 Techniques Employed

For the last few decades, the researcher mainly focused on the detection of IoT Intrusion BotNet attack in early stages. For these purposes, they determine different techniques with higher accuracy and lower error rates. In our research study, we used different techniques including ANN, CNN, CNN + SVm, and LSTM, and have been used for comparisons. The subsection contains a brief detail of each employed technique.

3.1.3 Artificial Neural Network

A set of algorithms known as a neural network aims to imitate the workings of the human brain by determining the connections between sets of data. It is used in many different situations, including regression, classification, image recognition, and many others. Three layers are present in a neural network. the input layer, which is the first layer. The input neurons that transmit data to the buried layer are present there. The computations are done on the input data by the hidden layer, which then sends the results to the output layer. Weight, activation function, and cost function are all included. Weight, which are the numerical values, is the term used to describe the connection between neurons. The neural network's capacity for learning is determined by the weights between neurons. During the learning of artificial neural networks, weight between the neuron varies. Through a learning process, an ANN is tailored for a particular purpose, such as pattern recognition or data classification. Learning mostly requires modifications to the synaptic connections that exist between the neurons.

- **Single Layer Neural Network**

  There are only two layers—input and output—but the input layer is not counted because no processing takes place there. When different weights are given to input nodes and the cumulative effect per node is calculated, the output layer is created. The output layer then receives instructions from all of the neurons to compute the output signals. Architecture of the single layer neural network is show in the Figure 3.2.

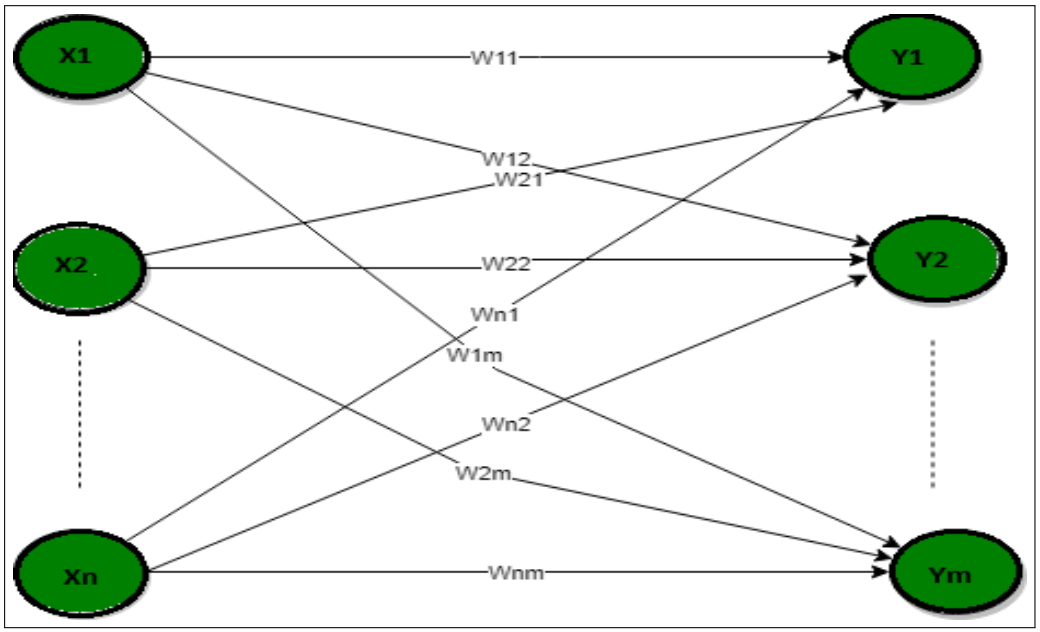

Figure 3.2: Architecture of the ANN with Single Layer

- **Multi Layer Neural Network**

  A hidden layer below this one that is a part of the network but not in direct communication with the external layer also exists. Due to information flow through the input function and the intermediary calculations necessary to calculate the output Z, the presence of one or more hidden layers makes the network computationally stronger. The model's outputs cannot be fed back into it because there are no feedback connections. Architecture of the multi layer network is show in the Figure 3.3.

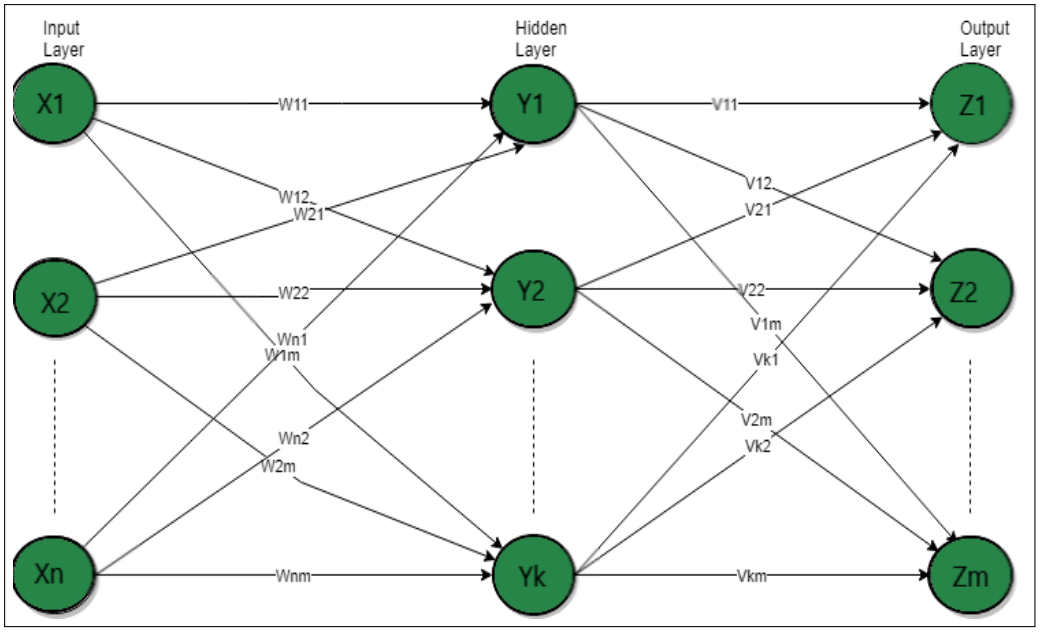

Figure 3.3: Architecture of the ANN with Multiple Layers

3.1.4 Convolution Neural Network

A convolutional layer, a pooling layer, and a fully connected layer are the typical three layers of CNN. The fundamental component of CNN is the convolution layer. It carries the majority of the computational load on the network. This layer creates a dot product between two matrices, one of which is the kernel, and the other is the constrained area of the receptive field. Compared to an image, the kernel is smaller in space but deeper. This indicates that the kernel height and width will be spatially small if the image consists of three RGB channels, but the depth will go up to all three channels. During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride. The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually. Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN This is why it can be computed as usual by a matrix multiplication followed by a bias effect. Architecture of the CNN is show in the Figure 3.4.

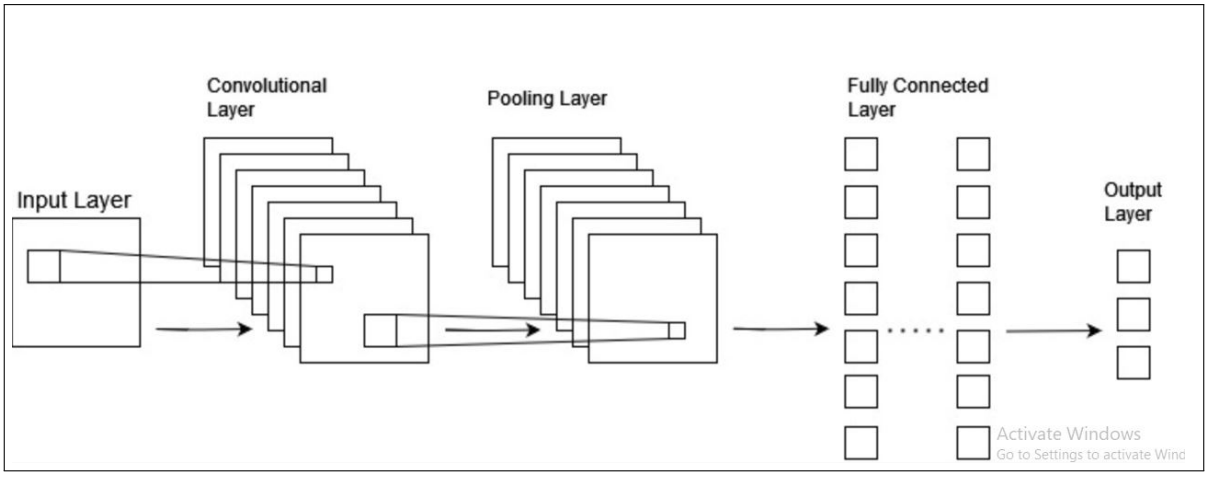

Figure 3.4: Architecture of the CNN

3.1.5 Convolution Neural Network + Support Vector Machine

The architecture of the CNN-SVM model was constructed by replacing the last output layer of the CNN model with an SVM classifier. The outputs of the final dense layer in the CNN are ten values between 0 and 1 computed by the relu activation function. The input of the activation function is the linear combination of the outputs from the previous hidden layer with trainable weights, plus a bias term. The output values of the hidden layer not only make sense to the CNN model, but also can be treated as input features for other classifiers. The SVM takes the outputs from the hidden layer as a new feature vector for training. Once the SVM classifier finishes training, it will be used to perform classification tasks. Architecture of the CNN + SVM is show in the Figure 3.5.

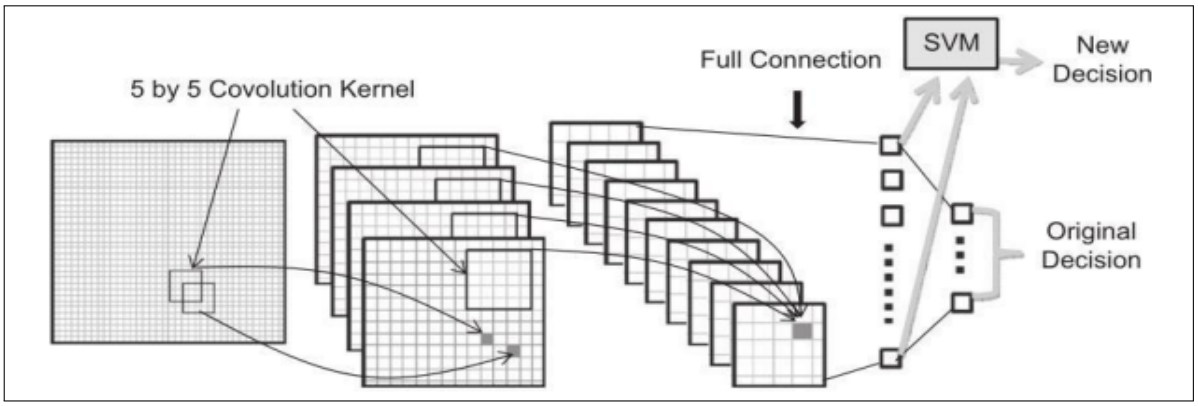

Figure 3.5: Architecture of the CNN + SVM

3.1.6 Long Short-Term Memory

A popular RNN (Recurrent Neural Network) type for learning sequential data prediction problems is the LSTM (Long Short-Term Memory) network. Like every other neural network, LSTM has a few layers that aid in pattern recognition and learning for improved performance. The basic operation of LSTM can be considered to hold the required information and discard the information which is not required or useful for further prediction. As the name suggests the forward pass and backward pass LSTM are unidirectional LSTM which process the information in one direction either on the forward side or on the backside where the bidirectional LSTM processes the data on both sides to persist the information. All the above-given LSTM types work on a basic structure. Updating the basic structure causes the difference between various LSTM. Next, in the article, we will see different components of a basic LSTM model architecture. Architecture of the long short-term memory is show in the Figure 3.6.

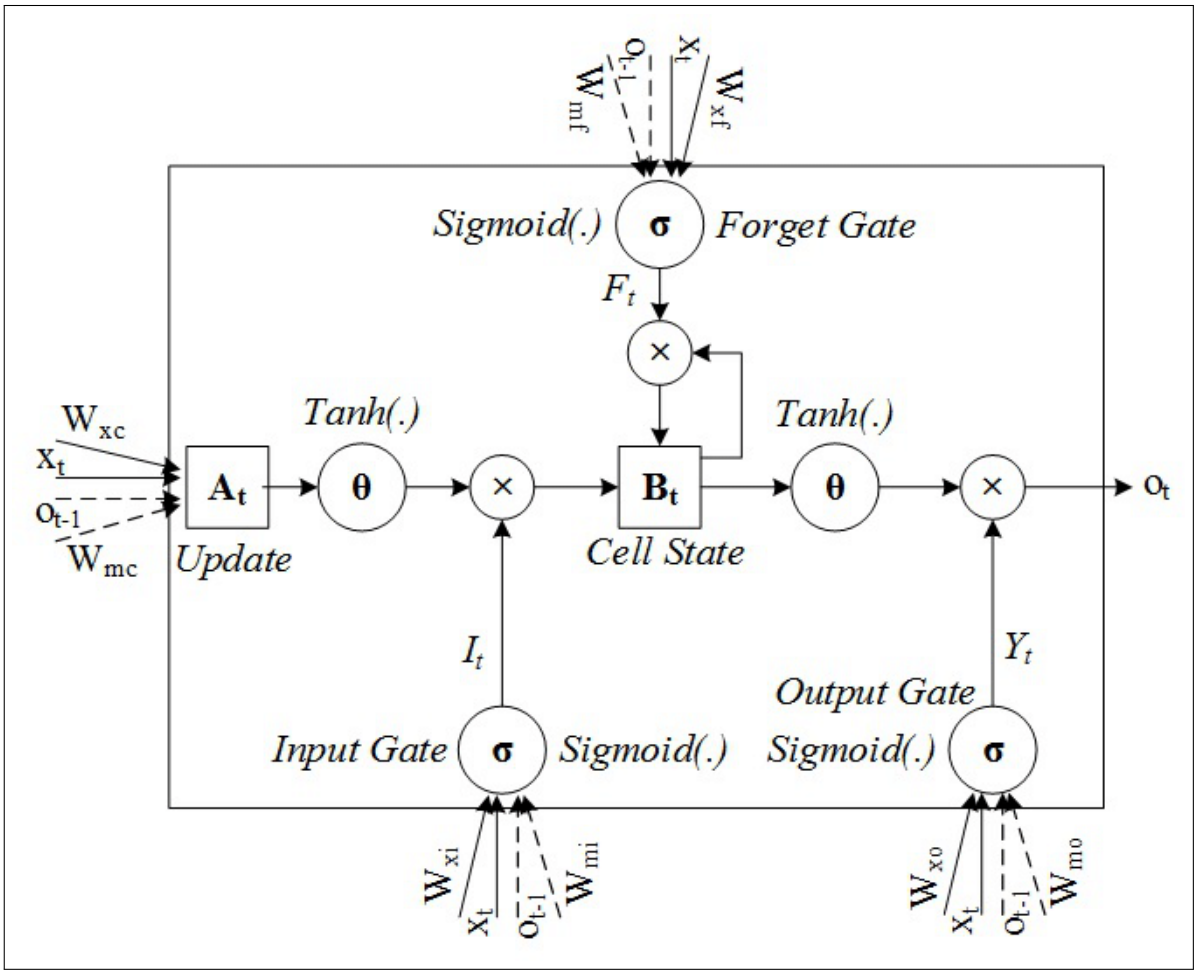

Figure 3.6: Architecture of the EfficientNet

3.1.7 Assessment Criteria

I have to discuss about the performance matrices in order to know in what extent our model works accurately. In this section we will discuss about techniques that we consider to evaluate. I need to familiarize with some terms regarding the model evaluation. Assessment metrics involved namely accuracy, precision, recall, and f1-measure.

a. **Accuracy**

The number of correctly classified data instances divided by the total number of data instances is known as accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

b. **Precision**

Precision is the ratio of True Positive (TP) and the sum of True Positive (TP) and True Negative (TN) rates to determine how good the model is at detecting negatives.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

c. **Recall**

Recall is the ratio of the True Positive (TP) rate and the sum of True Positive and True Negative (TN) rates to determine how good the model is at detecting the positives.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

d. **F1-Measure**

F1-Measure is the ratio of two-time multiplication of Precision, Recall, and subtraction of Precision, and Recall also.

$$F1 - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.4}$$

e. **Mean Absolute Error**

Is the average of all the instances in the test set of individual prediction errors calculated as:

$$MAE = \frac{1}{2}\sum_{j=1}^{n} |y_i - y| \tag{3.5}$$

where n is the number of errors, $|y_i - y|$ is the absolute error.

3.2 Tools

Mention below is all the tools that are employed in this research study.

3.2.1 PyTorch

PyTorch is a Torch and Python-based Deep Learning tensor library that is mostly utilised in CPU and GPU applications. Since PyTorch uses dynamic computation graphs and is entirely pythonic, it is preferred over other deep learning frameworks like TensorFlow and Keras. It enables real-time testing and execution of certain sections of code by researchers, programmers, and neural network debuggers. Users can therefore check whether a portion of the code works or not without having to wait for the implementation of the complete program. To build, train, and assess neural networks in the Python programming language, developers and researchers needed an effective, simple-to-use library. This was because deep learning had begun to change almost all fields of computer science.

3.2.2 Pandas

The most often used open source Python library for data science, data analysis, and machine learning activities is called Pandas. It is constructed on top of Numpy, a different package that supports multi-dimensional arrays. In the Python ecosystem, Pandas is one of the most widely used data wrangling packages. It integrates well with a variety of other data science modules, and it is typically available in all Python distributions, including those sold by commercial vendors like ActiveState's ActivePython and those that come with your operating system.

3.2.3 Google Colab

Colab is a free jpyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your

notebook.

3.2.4 Kaggle

Data scientists and machine learning enthusiasts can connect online at Kaggle. Users of Kaggle can work together, access and share datasets, use notebooks with GPU integration, and compete with other data scientists to solve data science problems. With the help of its potent tools and resources, this online platform—which was established in 2010 by Anthony Goldbloom and Jeremy Howard and was bought by Google in 2017—aims to assist professionals and students in achieving their objectives in the field of data science. Over 8 million users are registered on Kaggle as of today (2021).

Summary

This chapter defines the core methodology, subsection phase, dataset, and strategies used to evaluate the workflow of the entire research process and to uncover the outcomes. The tools and approaches that are being employed were described. The Assessment Criteria, which comprise accuracy, recall, f1-measure, and the specified confusion matrix derived from each approach, were also described, along with correct equations to determine how well our techniques categorized the records.

Chapter 4

Results

This chapter gives the experimental results of the techniques applied to the Intrusion detection. Firstly all the results obtained from the experiments are compared with each other than select the best from the rest with a high accuracy rate and low error rate. Regarding accuracy, we keep in mind all the aspects that affect the performance e.g. over-fitting, under-fitting and etc. The following results are presented in tabular and graphical forms and comparison are made.

4.1 Experimental Results

This section will present outcomes obtained through our analysis and implementation. The results of the different implemented techniques are mentioned with values and also illustrate it using tables and figures.

4.1.1 Results Analysis

We experimented on IoT Intrusion Detection dataset by STEFANOS T [34]. The dataset is publicly available, and addresses the lack of public botnet datasets, especially for the IoT. It suggests real traffic data, gathered from 9 commercial IoT devices consists of 93 csv file data contains multivariate sequential, number of Instances: 7062606, under the area of computer attribute characteristics, real Number of attributes 115, associated Tasks classification, clustering with null missing values. Originally we aimed at distinguishing between benign and Malicious traffic data by means of anomaly detection techniques. However, as the malicious data can be divided into 10 attacks carried by 2 botnets, the dataset can also be used for multi-class classification: 10 classes of attacks, plus 1 class of 'benign'. The study's results for each of the 9 IoT devices we trained and optimized a deep autoencoder on 2/3 of its benign data (i.e., the training set of each device). This was done to capture normal network traffic patterns. The test data of each device comprised of the remaining 1/3 of benign data plus all the malicious data. On each test set we applied the respective trained (deep) autoencoder as an anomaly detector. The detection of anomalies (i.e., the cyberattacks launched from each of the above IoT devices)

concluded with 100% TPR. The following attributes information H stats summarizing the recent traffic from this packet's host (IP), HH stats summarizing the recent traffic going from this packet's host (IP) to the packet's destination host, HpHp stats summarizing the recent traffic going from this packet's host+port (IP) to the packet's destination host+port. HH_jit stats summarizing the jitter of the traffic going from this packet's host (IP) to the packet's destination host. Time-frame (The decay factor Lambda used in the damped window) how much recent history of the stream is capture in these statistics L5, L3, L1,. The statistics extracted from the packet stream weight The weight of the stream (can be viewed as the number of items observed in recent history), radius: The root squared sum of the two streams' variances, magnitude the root squared sum of the two streams' means, cov an approximated covariance between two streams, pcc an approximated covariance between two streams. In the end, we were compared to choose the one with a better accuracy value as described in Table 4.1, Table 4.2, and Table 4.3 respectively. The graphical representation of each technique can be observed in Figure 4.1.

4.1.2 Standard Deviation Bar

Standard deviation bars quantify the scatter among the values. Looking at whether the error bars overlap lets you compare the difference between the mean with the amount of scattering within the groups. If the samples were larger with the same means and same standard deviations, the value would be much smaller and vice versa. In other words, the standard deviation bar shows the spread of data. The range will tell you how to spread out the data is, from the lowest to the highest values [35].

Table 4.1: Results Analysis

S.NO	**Techniques**	**Precision**	**Recall**	**F1-Measure**
1	ANN 1	0.98	0.98	0.98
2	CNN 1	0.99	0.99	0.99
3	RNN 1	0.99	0.99	0.99
4	ANN 2	0.99	0.99	0.99
5	CNN 2	0.99	0.99	0.99
6	RNN 2	0.98	0.98	0.98
7	ANN 1	1	1	1
8	CNN 1	1	1	1
9	RNN 1	0.99	0.99	0.99

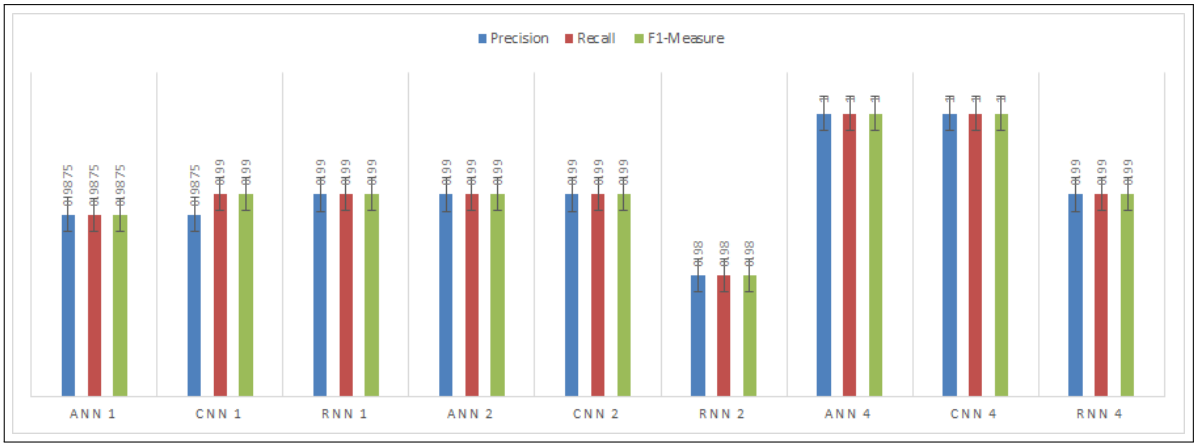

Figure 4.1: Comparison based on Precision, Recall, F-measure

Table 4.2: Results Analysis

S.NO	Techniques	Precision	Recall	F1-Measure
1	ANN 1	0.99	0.99	0.99
2	CNN 1	0.99	0.99	0.99
3	RNN 1	0.99	0.99	0.99
4	ANN 2	0.99	0.99	0.99
5	CNN 2	0.99	0.99	0.99
6	RNN 2	0.92	0.89	0.89
7	ANN 4	0.97	0.96	0.96
8	CNN 4	0.99	0.99	0.99
9	RNN 4	1	1	1

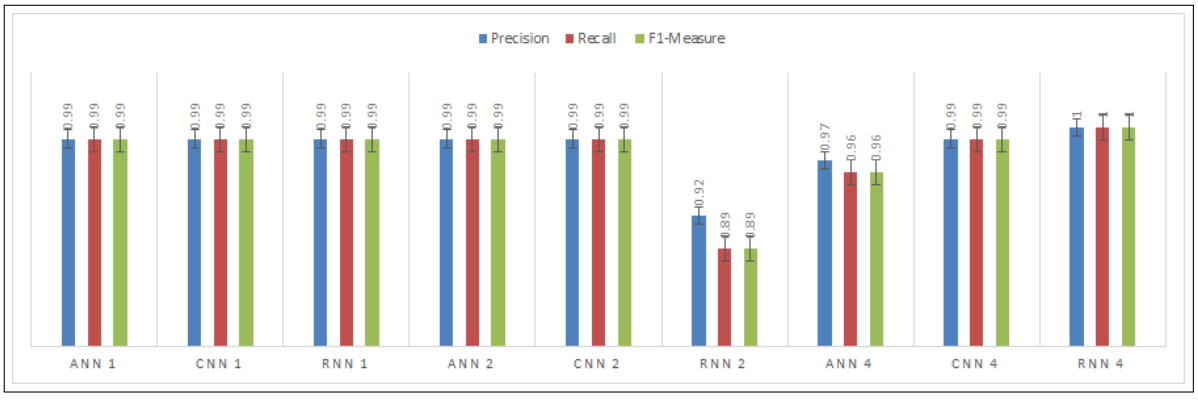

Figure 4.2: Comparison based on Precision, Recall, F-measure

Table 4.3: Results Analysis

S.NO	Techniques	Precision	Recall	F1-Measure
1	ANN 1	0.91	0.83	0.77
2	CNN 1	0.91	0.82	0.77
3	RNN 1	0.74	0.83	0.77
4	ANN 2	0.85	0.82	0.76
5	CNN 2	0.91	0.83	0.77
6	RNN 2	0.98	0.98	0.98
7	ANN 4	0.91	0.83	0.77
8	CNN 4	0.91	0.83	0.77
9	RNN 4	0.87	0.83	0.78

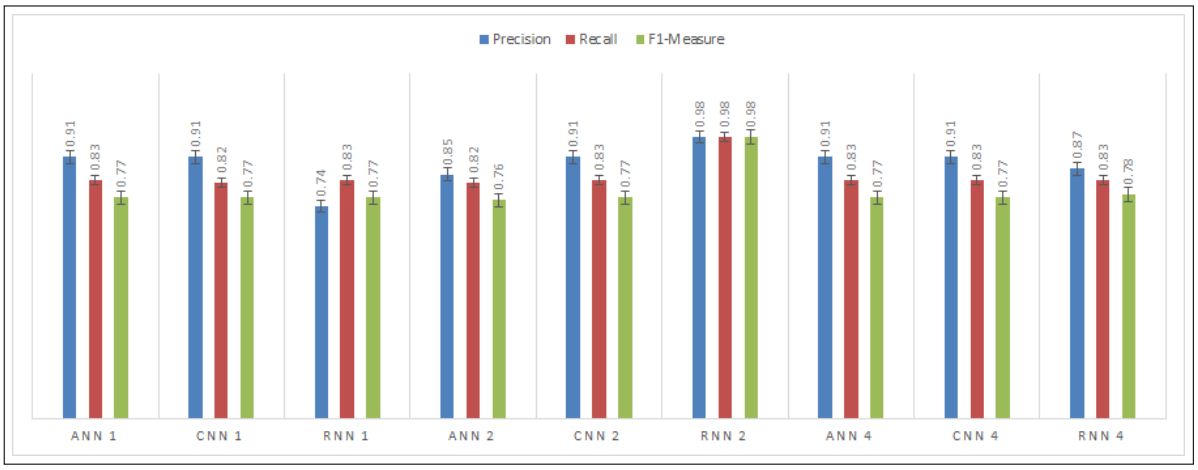

Figure 4.3: Comparison based on Precision, Recall, F-measure

4.1.3 Technique Accuracies

Our proposed model showed 99% accuracy for ANN model which is trained on 400 epochs with a batch size of 244 compare the accuracy of our ANN model with others. CNN showed 99% accuracy with 400 epoch, RNN showed the same accuracy of 99% with 400 epoch for class 4. On class 5 showed 99% accuracy for ANN model which is trained on 400 epochs with a batch size of 244 compare the accuracy of our ANN model with others. CNN showed 99% accuracy with 400 epoch, RNN showed the same accuracy of 99% with 400 epoch. On class 6 showed 82% accuracy for ANN model which is trained on 400 epochs with a batch size of 244 compare the accuracy of our ANN model with others. CNN showed 83% accuracy with 400 epoch, and RNN showed the same accuracy of 83% with 400 epoch. Table 4.4 display techniques accuracies and Figure 4.2, displays the accuracy graph of our proposed ANN, CNN, and RNN for class 1, class 2 and class 3 model.

Table 4.4: Techniques Accuracy Analysis

S.NO	Techniques	Accuracy
1	ANN 1	0.99
2	CNN 1	0.99
3	RNN 1	0.99
4	ANN 2	0.99
5	CNN 2	0.99
6	RNN 2	1
4	ANN 3	0.82
5	CNN 3	0.83
6	RNN 3	0.83

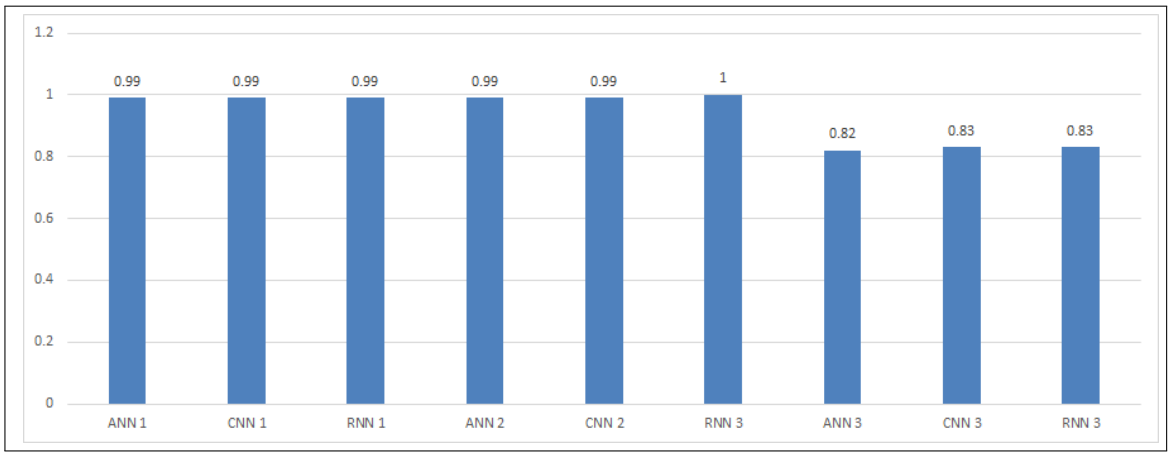

Figure 4.4: CNN Training and Testing Validation Accuracy

4.2 Results Discussion

This study focuses on various DL models for Instrusion Detection on NBaiot Detection. The said models are evaluated using some standard measures including accuracy, precision, recall, and F-M. The experimental outcomes show the better performance of ANN, CNN and RNN with an accuracy of 0.99%, for class 1 and on the other hand, class 4 ANN presents the weakest performance with an accuracy of 0.82%. Figure 4.3 through Figure 4.10 show the training and testing validation accuracies of all the classes ANN, CNN, and RNN respectively. These analyses and visual representations also show the better performance of ANN 99% in class 1, same in class 2 performance of is ANN 99%, CNN 99% and RNN is 100%. CNN and RNN automatically detect the important features, without any intervention from outside. Moreover, ANN, CNN and RNN is computationally efficient. The special convolution and pooling operations and, parameter sharing, Enables the CNN model to run on any device and makes them universally attractive.

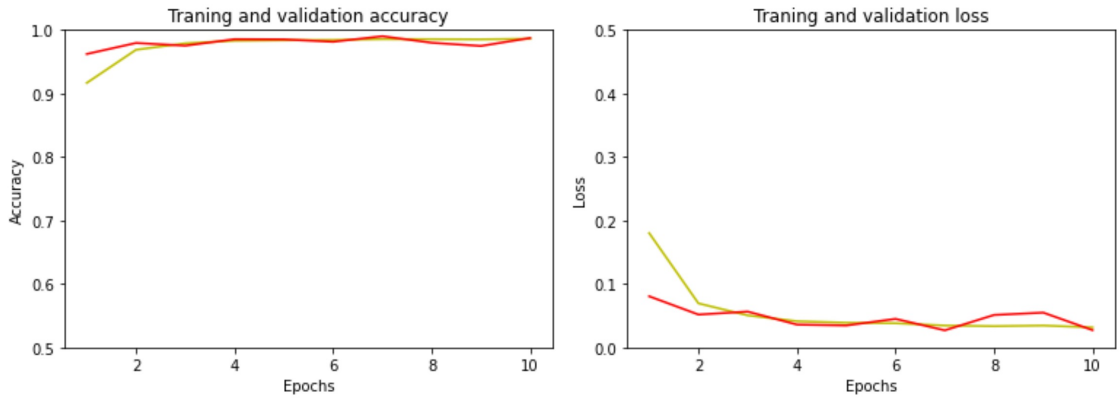

Figure 4.5: Class 4 ANN Training and Testing Validation Accuracy

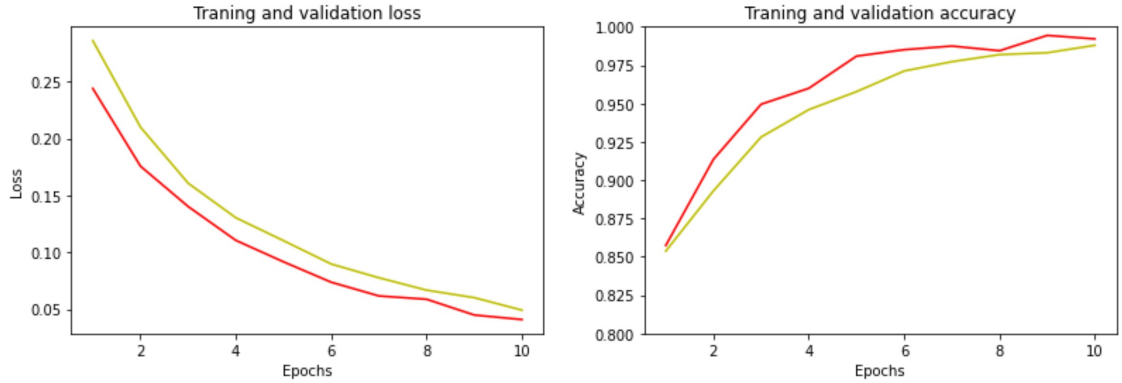

Figure 4.6: Class 4 CNN Training and Testing Validation Accuracy

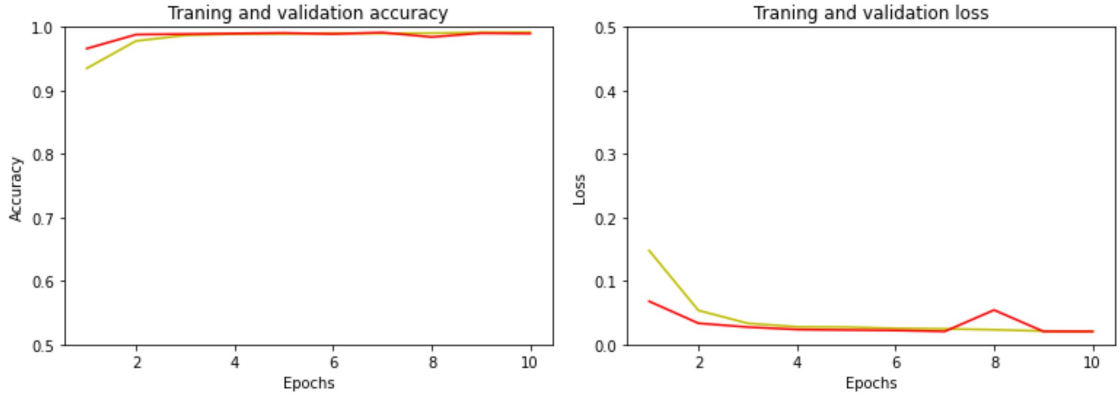

Figure 4.7: Class 4 RNN Training and Testing Validation Accuracy

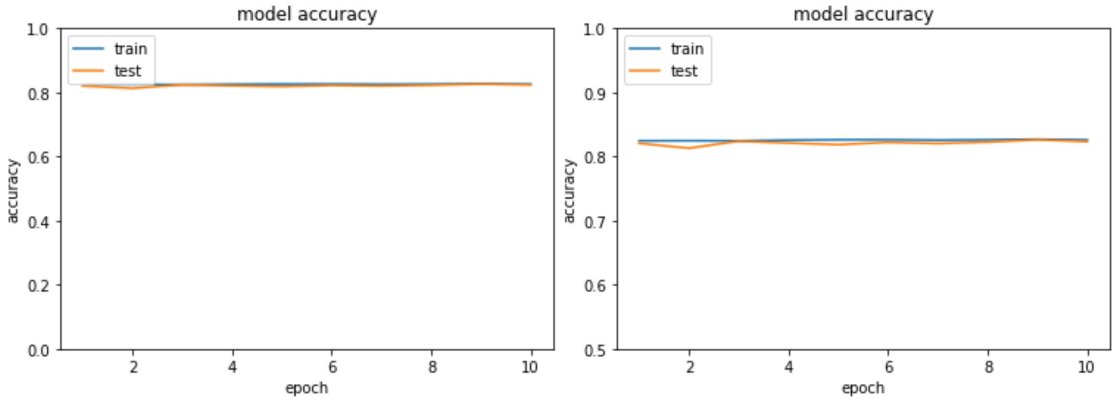

Figure 4.8: Class 6 ANN Training and Testing Validation Accuracy

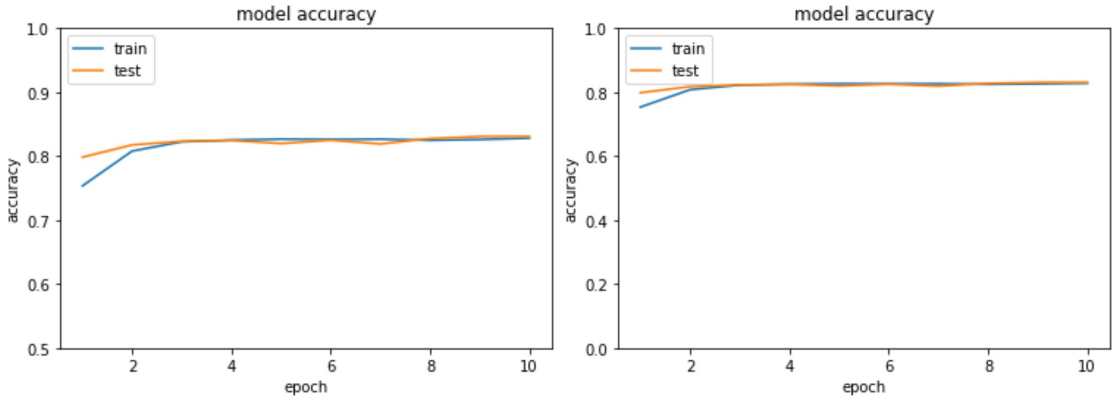

Figure 4.9: Class 6 CNN Training and Testing Validation Accuracy

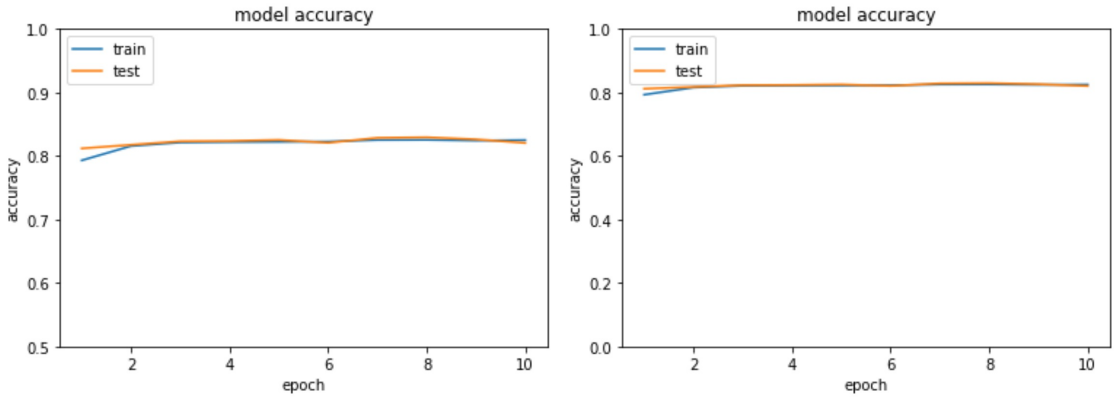

Figure 4.10: Class 6 RNN Training and Testing Validation Accuracy

Now another question may arise here why the performance of class 1 ANN, CNN, and RNN is better than other employed models? To this end, we have some major advantages of CNN over the rest of said models. CNN is basically the type of ANN used for prediction and classification in same case RNN detect the sequential properties of the data and make use of patterns to forecast the upcoming likely scenario and tasks. In CNN convolution represents the understanding of features within data. To extract these features, a filter is used. These are some concepts used in CNN that make its prediction better than other models. Unlike other types of neural networks, RNN have special capabilities that provide its users a wide range of opportunities while also posing certain difficulties. The only neural network having memory and two data processing stages is the RNN. Multiple inputs and outputs can be mapped out by it. The advantage of RNN is that it can map out many to many, one to many, and many to one input and outputs, in contrast to other algorithms that produce one output for one input.

4.2.1 Threats to Validity

It is important to evaluate the reliability of your research in this area. Furthermore, the validity of your evaluation or study design is important. The internal or external validity of your evaluation design may be at hazard under certain circumstances. By avoiding this type of validity, the emphasis will be placed on the research validity. Some of the validity are listed below.

Internal Validity: This paper's analysis is based on a variety of well-known assessment criteria that have been applied in previous investigations. Several of these criteria are used to assess the rate of error, while others are used to assess accuracy. As a result, the introduction of new evaluation criteria as a substitute for previously used standards may reduce accuracy. Furthermore, several of the methodologies utilized in this study can be replaced by newer ones.

External Validity: On dataset from the Kaggle repository, we conduct experimental analysis. If we compare the predicted methods to real data provided by organizations, or if we replace these dataset with another dataset, we risk disrupting the conclusions while raising error rates. Similarly, utilizing alternative dataset, the projected approach may not be capable of producing improved predicted results.

Construct Validity: On the intrusion detection classification dataset taken from the Kaggle repository based on multiple assessment metrics, several ML algorithms are applied in this work. The variety of strategies used in this study is at the core of their advancements above other techniques used by researchers in recent decades. However, there is a risk that if we add more new methods, these new techniques would exhaust the anticipated techniques. Additionally, dividing the dataset into training and testing,

as well as increasing or lowering the number of fold validations for the experiments, can all help to lower the error rate.

Summary

This chapter provides the detailed walk through of the whole experimental findings and future work of the Intrusion Deteciton on NBaIoT Dataset. For proper understanding, Table 4.1, Table 4.2, and Table 4.3 represent the result analysis, and Table 4.4 accuracy analysis and Figure 4.5 through Figure 4.10 represent training and testing accuracy.

Conclusion and Future Work

Conclusion

The main focus of this research is to identify the best datasets for wireless IoT IDS. A comparison was made between dataset with various machine learning algorithms. Besides that, with the simulated annealing approach with different combinations of features, the most prominent features necessary for IDS are identified. IDS is a monitoring system that detects suspicious activities and generates alerts when they are detected. According to the previous research with binary classification of ML, DL the results are good but with multi-classification the results are not up to the requirements where we face overfitting and under fitting problem with machine learning algorithms. We proposed to implement deep learning models ANN with different architecture as we will also implement CNN and CNN+SVM and comparing these models on NBaIOT dataset. After completing our research, we expect to have a system which will be able to detect intrusions with high accuracy.

Future Work

Deep learning has caught the interest of researchers since its beginnings, due to its contributions in fields as diverse as bioinformatics, computer vision, pattern recognition, and many others, as well as how deep learning might tackle our most difficult challenges. Many problems have been covered but there are many opportunities in the intrusion detection domain.

1. In future investigations, we hope to focus on capturing primary data of sufficient size in order to make the data as realistic as feasible.

2. Large neural networks have millions of hidden layers and nodes with their respective weights. Therefore making an extensible and efficient model with low computational cost and more generalized.

3. Large data, resemble the architecture that demands fewer hardware requirements and provides a suitable output in a reasonable amount of time.

4. There is also the chance for using categorical classification problems for the identification of intrusion types and detecting attacks.

5. Future work will focus on lowering the number of parameters and computing time without compromising on performance.

References

[1] A. J. Siddiqui and A. Boukerche, "Adaptive ensembles of autoencoders for unsupervised iot network intrusion detection," *Computing*, vol. 103, no. 6, pp. 1209–1232, 2021.

[2] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in iot-edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2022.

[3] G. E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy," *Advances in neural information processing systems*, vol. 6, 1993.

[4] A. Boukerche, K. R. L. Jucá, J. B. Sobral, and M. S. M. A. Notare, "An artificial immune based intrusion detection model for computer and telecommunication systems," *Parallel Computing*, vol. 30, no. 5-6, pp. 629–646, 2004.

[5] C. Ioannou and V. Vassiliou, "An intrusion detection system for constrained wsn and iot nodes based on binary logistic regression," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 259–263, 2018.

[6] R. A. Nofal, N. Tran, C. Garcia, Y. Liu, and B. Dezfouli, "A comprehensive empirical analysis of tls handshake and record layer on iot platforms," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 61–70, 2019.

[7] A. J. Siddiqui and A. Boukerche, "Encoded .flow features for network intrusion detection in internet of things," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, 2020.

[8] "DDoS.." https://www.trendmicro.com/vinfo/us/security/news/ddos, 2022. [Online; accessed on 29 May 2022].

[9] "Brain Tumor Basics.." https://www.kaspersky.com/about/press-releases, 2022. [Online; accessed on 29 May 2019].

[10] M. Aloqaily, S. Otoum, I. Al Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Networks*, vol. 90, p. 101842, 2019.

[11] A. Boukerche, R. B. Machado, K. R. Jucá, J. B. M. Sobral, and M. S. Notare, "An agent based and biological inspired real-time intrusion detection and security model for computer network operations," *Computer Communications*, vol. 30, no. 13, pp. 2649–2660, 2007.

[12] R. B. Machado, A. Boukerche, J. B. M. Sobral, K. R. L. Jucá, and M. S. M. A. Notare, "A hybrid artificial immune and mobile agent intrusion detection based model for computer network operations," in *19th IEEE international parallel and distributed processing symposium*, pp. 8–pp, IEEE, 2005.

[13] G. S. Tandel, M. Biswas, O. G. Kakde, A. Tiwari, H. S. Suri, M. Turk, J. R. Laird, C. K. Asare, A. A. Ankrah, N. Khanna, *et al.*, "A review on a deep learning perspective in brain cancer classification," *Cancers*, vol. 11, no. 1, p. 111, 2019.

[14] C. Osborne and Z. Day, "The most interesting internet-connected vehicle hacks on record," *ZDNet Available at: https://www. zdnet. com/article/these-are-the-most-interesting-ways-to-hack-internet-connected-vehicles/.(Accessed: 19th July 2019).*

[15] S. Shasha, M. Mahmoud, M. Mannan, and A. Youssef, "Playing with danger: A taxonomy and evaluation of threats to smart toys," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2986–3002, 2018.

[16] J. Yang, T. Li, G. Liang, W. He, and Y. Zhao, "A simple recurrent unit model based intrusion detection system with dcgan," *IEEE Access*, vol. 7, pp. 83286–83296, 2019.

[17] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, 2019.

[18] B. Liao, Y. Ali, S. Nazir, L. He, and H. U. Khan, "Security analysis of iot devices by using mobile computing: a systematic literature review," *IEEE Access*, vol. 8, pp. 120331–120350, 2020.

[19] Y. Li, M. Zhu, X. Luo, L. Yin, and Y. Fu, "A privacy-preserving botnet detection approach in largescale cooperative iot environment," *Neural Computing and Applications*, pp. 1–13, 2022.

[20] H. Choi, H. Lee, and H. Kim, "Botgad: detecting botnets by capturing group activities in network traffic," in *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, pp. 1–8, 2009.

[21] H. Choi and H. Lee, "Identifying botnets by capturing group activities in dns traffic," *Computer Networks*, vol. 56, no. 1, pp. 20–33, 2012.

[22] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," 2008.

[23] J. Kwon, J. Lee, H. Lee, and A. Perrig, "Psybog: A scalable botnet detection method for large-scale dns traffic," *Computer Networks*, vol. 97, pp. 48–73, 2016.

[24] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 48–61, 2010.

[25] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: Dga-based botnet tracking and intelligence," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 192–211, Springer, 2014.

[26] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu, "Msml: A novel multilevel semi-supervised machine learning framework for intrusion detection system," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1949–1959, 2018.

[27] S. Otoum, B. Kantarci, and H. Mouftah, "Empowering reinforcement learning on big sensed data for intrusion detection," in *Icc 2019-2019 IEEE international conference on communications (ICC)*, pp. 1–7, IEEE, 2019.

[28] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

[29] L. Yin, X. Luo, C. Zhu, L. Wang, Z. Xu, and H. Lu, "Connspoiler: Disrupting cc communication of iot-based botnet through fast detection of anomalous domain queries," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1373–1384, 2020.

[30] A. Kumar, M. Shridhar, S. Swaminathan, and T. Joon Lim, "Ml-based early detection of iot botnets," in *International Conference on Security and Privacy in Communication Systems*, pp. 254–260, Springer, 2020.

[31] "Stacked recurrent neural network for botnet detection in smart homes," *Computers Electrical Engineering*, vol. 92, p. 107039, 2021.

[32] "Precisiona, Recall, and F1-Measure.." https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/, 2022. [Online; accessed on 29 July 2022].

[33] "Accuracies.." https://machinelearningmastery.com/confusion-matrix-machine-learning/#:~:text=A%20confusion%20matrix%20is%20a,two%20classes%20in%20your%20dataset., 2022. [Online; accessed on 30 July 2022].

[34] "IoT Intrusion Dataset.." https://www.kaggle.com/code/stefanost/iot-intrusion-detection/data?select=features.csv., 2022. [Online; accessed on 30 July 2022].

[35] "What is standard deviation bar error?.." https://www.biologyforlife.com/interpreting-error-bars.html#:~:text=An%20error%20ba\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{%\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\let\typeout\protect\begingroup\def\MessageBreak{Ω(Font)}\let\protect\immediate\write\m@ne{LaTeXFontInfo:oninputline192.}\endgroup\endgroup\relax\let\ignorespaces\relax\accent23%\egroup\spacefactor\accent@spacefactor20is%20a,deviation%20of%20a%20data%20set./,., 2022. [Online; accessed on 30 July 2022].