Anomaly Detection for IoT Networks Using Machine Learning

By:

HUSAIN ABDULLA

A Thesis Submitted
in Partial Fulfilment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering, Design and Physical Sciences
BRUNEL UNIVERSITY LONDON

July 2023

Abstract

The Internet of Things (IoT) is considered one of the trending technologies today. IoT affects various industries, including logistics tracking, healthcare, automotive and smart cities. A rising number of cyber-attacks and breaches are rapidly targeting networks equipped with IoT devices. This thesis aims to improve security in IoT networks by enhancing anomaly detection using machine learning.

This thesis identified the challenges and gaps related to securing the Internet of Things networks. The challenges are network size, the number of devices, the human factor, and the complexity of IoT networks. The gaps identified include the lack of research on signature-based intrusion detection systems used for anomaly detection, in addition to the lack of modelling input parameters required for anomaly detection in IoT networks. Furthermore, there is a lack of comparison of the performance of machine learning algorithms on standard and real IoT datasets.

This thesis creates a dataset to test the anomaly binary classification performance of the Neural Networks, Gaussian Naive Bayes, Support Vector Machine, and Decision Trees machine learning algorithms and compares their results with the KDDCUP99 dataset. The results show that Support Vector Machine and Gaussian Naive Bayes perform lower than the other models on the created IoT dataset. This thesis reduces the number of features required by machine learning algorithms for anomaly detection in the IoT networks to five features only, which resulted in reduced execution time by an average of 58%.

This thesis tests CNNwGFC, which is an enhanced Convolutional Neural Network model, in detecting and classifying anomalies in IoT networks. This model achieves an increase of 15.34% in the accuracy for IoT anomaly classification in the UNSW-NB15 compared to the classic Convolutional Neural Network. The CNNwGFC multi-classification accuracy (96.24%) is higher by 7.16 than the highest from the literature.

Keywords: IoT; Machine Learning; Security; Anomaly Detection

Dedication

I dedicate this dissertation to all people who supported me throughout my educational years, especially …

to my mother and father for their words of support and encouragement;

to family members for their inspirational words;

to friends for instilling the importance of hard work and higher education;

to mentors and tutors for their efforts in mentoring and tutoring me.

Declaration

I confirm that this thesis is my original work and is being submitted to the Post-Graduate Research Office for the first time. The research, writing, and review of this thesis were conducted by me, with guidance and supervision from my supervisors in the Department of Electronic and Computer Engineering, College of Engineering, Design and Physical Sciences, Brunel University London UK. All information obtained from other sources has been appropriately cited and acknowledged.

Husain Abdulla

January 2023

Acknowledgements

I would like to extend my sincere gratitude to my supervisors, without whose guidance, support, and encouragement, this thesis would not have been possible:

Prof. Hamed Al-Raweshidy	Dissertation Principal Supervisor, Brunel University
Dr. Wasan Shakir Awad	Dissertation Supervisor, Ahlia University

I am deeply grateful to all of these individuals, as my dissertation would not have been successful without their support and willingness to help. My parents, brothers, and sister have always believed in my ability to succeed and have provided me with everything I needed to get to where I am today. My wife has been an endless source of support and motivation throughout this process. Lastly, I want to thank my friends for helping me through the difficult times with laughter and encouragement.

Table of Contents

Table of Figures

Table of Tables

List of Abbreviations and Acronyms

Abbreviation	Meaning
ANN	Artificial Neural Network
ARP	Address Resolution Protocol
CNN	Convolutional Neural Network
CNNwGFC	Convolutional Neural Network with Global Feature Correlation
DoS	Denial of Service
DRL	Deep Reinforcement Learning
DT	Decision Tree
GFC	Global Feature Correlation
ICA	Independent Component Analysis
IDS	Intrusion Detection Systems
IoT	Internet of Things
KDDCUP99	Knowledge Discovery and Data Mining 1999
KNN	KNN - k-Nearest Neighbors
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron Neural Network
MSE	Mean Squared Error
NB	Naïve Bayes
NIDS	Network-based Intrusion Detection Systems
NN	Neural Network
PCA	Principal Component Analysis
PCAP	Packet Capture
RF	Random Forest
SNMP	Simple Network Management Protocol
SSDP	Simple Service Discovery Protocol
SVD	Singular Value Decomposition

SVM	Support Vector Machine
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNSW-NB15	University of New South Wales-Network-Based 15

1. Chapter One: Introduction

1.1. Background

This thesis discusses anomaly detection for the Internet of Things (IoT) networks using machine learning. This topic includes three key concepts: Anomaly Detection, IoT and Machine Learning. Anomaly detection is also known as outlier detection and novelty detection at times. It refers to detecting unusual objects, occurrences, or observations that differ considerably from a bulk of data and do not correspond to a well-defined typical behaviour [1]. Such occurrences and events may raise concerns that they were generated by a different method or seem inconsistent with the other data [2]. The concept of anomaly detection has several applications, including the prevention of financial fraud and the protection of computer networks from abnormal traffic such as denial of service attacks and ping of death.

The Internet of Things (IoT) promises an optimistic technological future where the physical world is integrated with computer-based systems, resulting in economic benefits and efficiency improvements. The IoT is a network of objects, including devices, home appliances, and vehicles, which may be embedded with electronics, sensors, and software to connect and exchange data [3]. If a 'thing' is referred to as part of the IoT, it is accessible via the internet. The IoT affects a variety of industries, including logistics tracking, healthcare, automotive and smart cities. It is expected that the usage of IoT devices will touch every point of human life.

Machine learning is a subfield of the artificial intelligence field. It is defined as a machine's capability to predict and make decision like a human. Machine learning is a growing field that uses computing algorithms intending to mimic human intelligence through learning from their surroundings [4]. It allows software applications to become more accurate at predicting outcomes. Bayesian Networks, Support Vector Machines, Neural Networks, and Decision Trees are examples of machine learning algorithms.

1.2. Motivation

Despite the indisputable advantages of the IoT, the fact of the matter is that its security is not keeping pace. As the IoT becomes more widespread, its heterogeneity and size are projected to increase existing Internet security vulnerabilities. A wide variety of threats that we now cannot consider will become apparent once humans, sensors and automobiles can easily connect via IoT.

If essential measures are not taken, hackers will exploit the IoT's evasiveness to disrupt communications, achieve substantial financial gains, or physically harm people. For instance, substantial vulnerabilities are found in many IoT baby monitors [5], which hackers might exploit to carry out a variety of illegal acts, such as allowing other users to observe and operate the monitor remotely. Another recent discovery has demonstrated that Internet-connected automobiles can be remotely controlled [6], enabling the driver to perform things such as open doors and even turn off the vehicle's engine while it is in motion. Moreover, there are alarming incidents of IoT hacking that may extend to medical equipment and can have a devastating effect on patients [7, 8]. Another critical point is that most present security countermeasures are computationally intensive and significantly overhead [9]. However, price limits force IoT devices to have restricted memory and computing capability and use small, affordable batteries for energy storage.

To prepare for a future in which the IoT is everywhere and accessible from anywhere, it is more necessary than ever to address significant IoT security concerns. One of the Internet security challenges for IoT devices is anomaly detection. Many negative consequences might result from an abnormal attack on today’s highly linked and interconnected network environment. Artificial intelligence’s growth has led to machine learning’s use in anomaly detection.

1.3. Aim and Objectives

This study aims to improve security in IoT networks by enhancing existing anomaly detection techniques using machine learning. This research will incorporate the following objectives:

- O1: Identify the challenges and research gaps in securing the Internet of Things networks.

- O2: Test the anomaly binary classification (normal/abnormal) performance of the following machine learning algorithm on a real IoT dataset and compare the results with the commonly used dataset from the literature (KDDCUP99): Neural Networks (NN), Gaussian Naive Bayes (NB), Decision Trees (DT), and Support Vector Machine (SVM).

- O3: Reduce the number of features (input size) required to detect anomalies in IoT networks.

- O4: Develop an enhanced deep learning model based on CNN to detect and classify anomalies in IoT networks.

1.4. Contributions

Here is a list of the contributions to the literature:

1. Identified the challenges and the gaps related to securing the Internet of Things networks. The challenges are network size, the number of devices, the human factor as well the complexity of the IoT networks. The gaps identified include the lack of research on the following:
    - Signature-based intrusion detection systems use for anomaly detection.
    - Modelling input parameters required for anomaly detection in IoT networks.
    - Comparison of the performance of machine learning algorithms on the KDDCUP99 and a real IoT dataset [78].
    - High performance machine learning model to classify anomalies in the IoT networks.

2. Tested the binary classification performance including accuracy, precision and F1 Score of the Neural Networks, Gaussian Naive Bayes, Support Vector Machine, and Decision Trees machine learning algorithms and compared their results with the Knowledge Discovery and Data Mining 1999 (KDDCUP99) dataset. The results showed that Support Vector Machines and Gaussian Naive Bayes performs lower than the other models.

3. Reduced the number of features required by machine learning algorithms for anomaly detection in the IoT networks from 41 to 5 features only, which resulted in reduced execution time by an average of 58%.

4. Tested the CNNwGFC, which is an enhanced Convolutional Neural Network model, in detecting and classifying anomalies in IoT networks. This model archives an increase of 9% in the accuracy for IoT anomaly detection in the University of New South Wales-Network-Based 15 (UNSW-NB15) compared to the classic Convolutional Neural Network. The CNNwGFC multi classification accuracy (96.24%) is higher by 7.16 than the highest from the literature [195].

1.5. Thesis Outline

In addition to the Introduction chapter, this thesis includes five other chapters. Here is an overview of each chapter:

- Chapter 2 provides an overview of machine learning algorithms, challenges in securing IoT devices, related studies on the use of Machine Learning in anomaly detection, the gap in research in anomaly detection for IoT networks and an overview of datasets used in this research.

- Chapter 3 tests the binary classification performance of the following Machine Learning algorithms in anomaly detection on the KDDCUP99 and created IoT datasets: Neural Networks, Gaussian Naive Bayes, Decision Trees, and Support Vector Machine.

- Chapter 4 reduces the input size of the KDDCUP99 dataset to five features only and compares the binary classification performance results with Chapter 3.

- Chapter 5 tests the CNNwGFC model, which is an enhanced Convolutional Neural Networks model, in detecting and classifying anomalies in network traffic data.

- Chapter 6 includes a summary of the findings of this research.

1.6. List of Publications

The following papers were accepted for publication:

- H. Abdulla, H. Al-Raweshidy and W. S. Awad, "ARP spoofing detection for IoT networks using neural networks", *SSRN Electronic Journal*, July 24, 2020.

- H. Abdulla, H. Al-Raweshidy and W. Awad, "The Era of Internet of Things: Towards better security using machine learning", in *International Conference on IT Innovations and Knowledge Discovery 2023*, 2023.

- H. Abdulla, H. Al-Raweshidy and W. Awad, "Denial of Service Detection for IoT Networks Using Machine Learning", in *International Conference on Agents and Artificial Intelligence 2023*, 2023.

At the time of writing, the following papers were submitted for review:

- A journal paper titled "CNNwGFC: An Enhanced CNN Model for Network Traffic Anomaly Detection and Classification" submitted to IEEE Access on 30th January 2023.

2. Chapter Two: Related Work and Concepts

2.1. Overview

This chapter aims to identify the gaps related to anomaly detection in IoT networks. First, it details the Machine Learning algorithms used in this research, including Artificial Neural Networks, Decision Trees, Support Vector Machines, Naïve Bayes and Convolutional Neural Networks. It defines two concepts related to the research topic: Intrusion Detection and Time Series. It describes the Internet of Things concept and its security challenges. It describes the public data sets used in this thesis, including KDDCUP99, UNSW-NB15 and a real IoT dataset [78]. Finally, this chapter is concluded with a summary of its contents.

2.2. Machine Learning

2.2.1. Overview of Machine Learning

Machine learning is a branch of artificial intelligence that enables a computer to learn how to create output without being explicitly programmed [10]. Machine learning technology is commonly used for data analysis to construct prediction models. The popularity of machine learning stems from the fact that it should do two different jobs. First are the tasks that machines can perform, followed by those that are impossible for humans to execute. Some academics divide the process of machine learning into two parts: learning and prediction. The learning element concerns feeding the machine learning algorithm with the training data, while the prediction element concerns the machines' predictions. Supervised and unsupervised learning are the two most typical types of machine learning.

- **Supervised Learning**

Supervised machine learning is used when a dataset has labelled data. Supervised learning aims to train the computer to predict values accurately or classify an input example. Classification and regression are the most popular products of supervised learning [11].

- **Unsupervised Learning**

Unsupervised learning aims to describe hidden patterns from input data. Unsupervised learning is used when a labelled dataset is not available. Sometimes, unsupervised learning is used to categorize unlabelled datasets, and the resulting labelled dataset is used for supervised learning. Dimensionality reduction and clustering are two common examples of unsupervised learning [12]

As this research is done on a labelled dataset, it is opted to employ supervised learning for this thesis. A model of class label distribution is trained using supervised learning algorithms, and this model can predict class labels for testing samples. Figure (2-1) is an example of a process flowchart for supervised learning algorithms; this whole procedure is also known as classification, which

forms the basis for the created prediction model. It is crucial to choose the appropriate classification model for a given task.

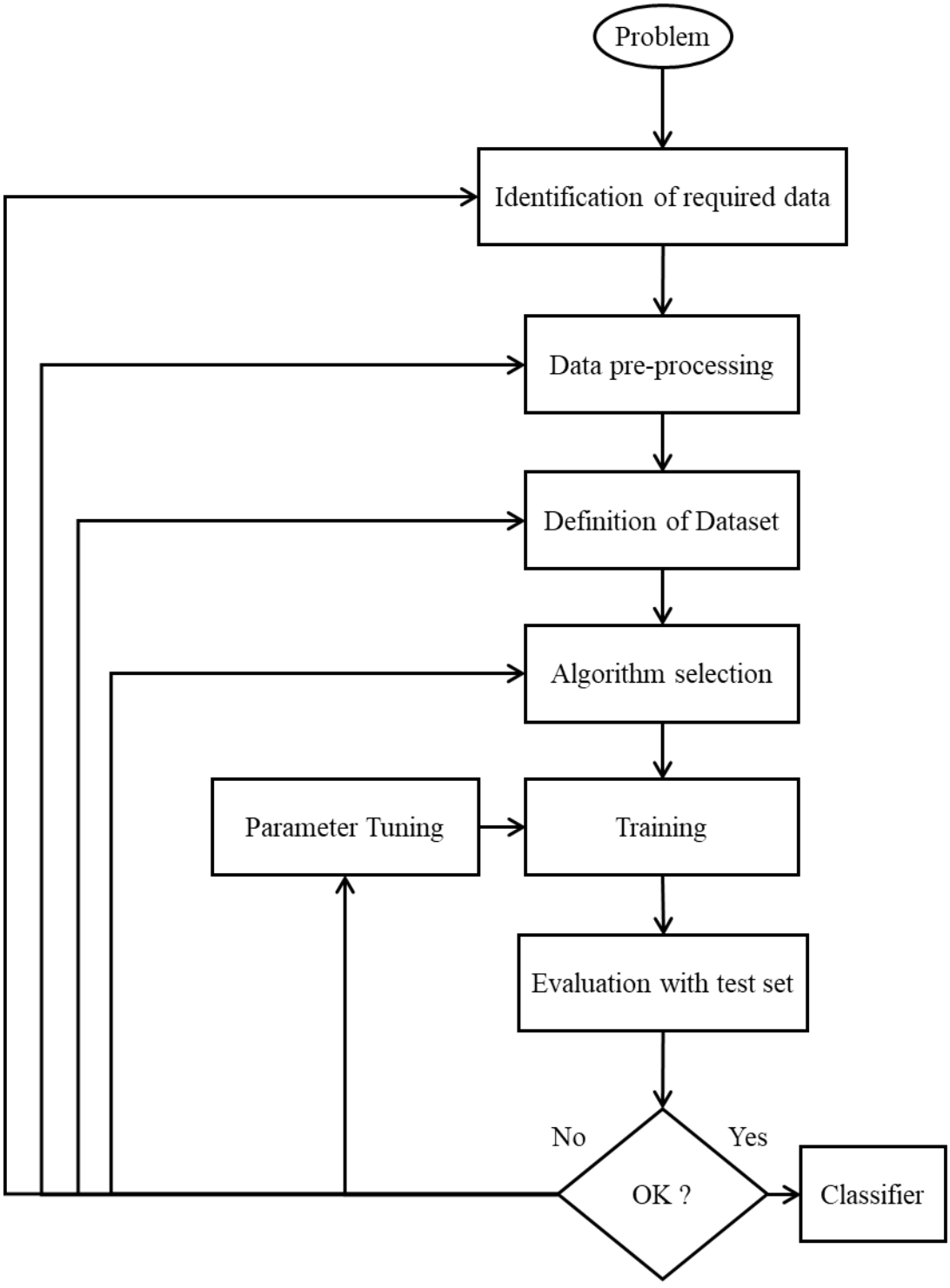

Figure 2-1: The process of supervised learning [15].

The supervised learning algorithms include the following categories: Regression and Classification, as shown in Figure (2-2).

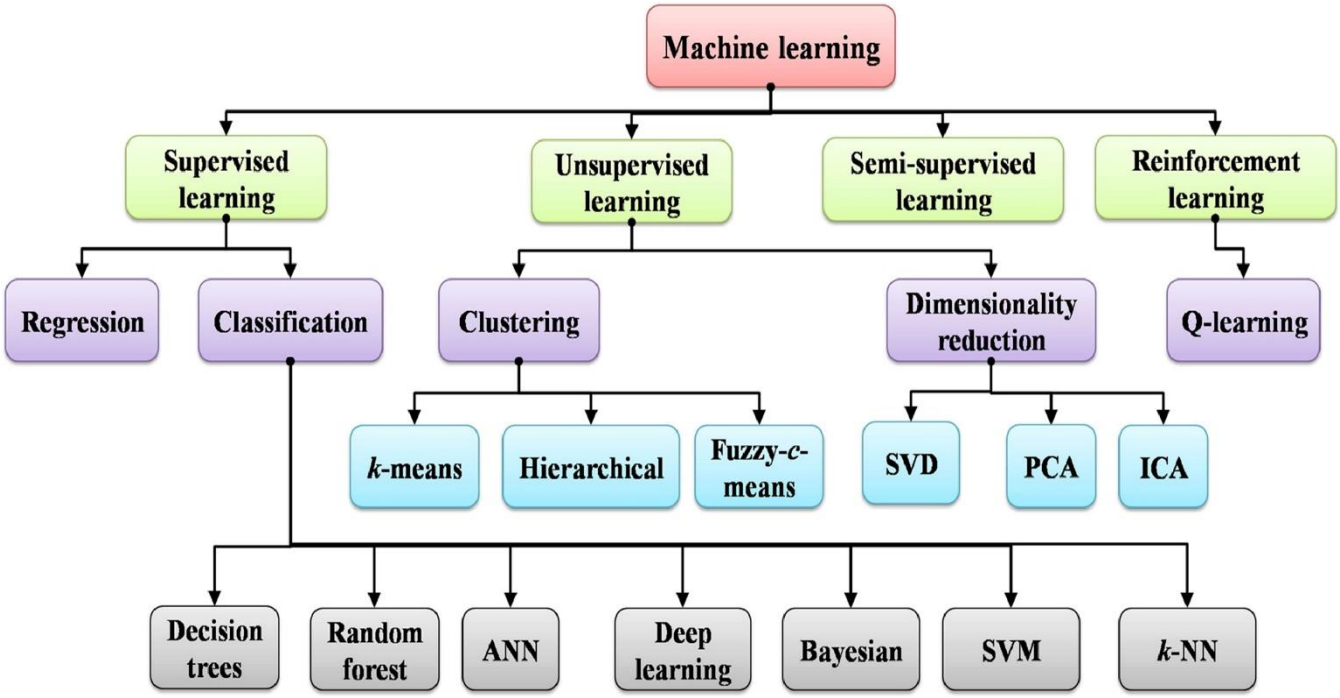

Figure 2-2: Taxonomy of ML techniques [13].

Since the problem targeted in this research is a classification problem (Normal/Abnormal Traffic), classification machine learning algorithms are used. According to [14] [15], supervised machine learning algorithms, which are mainly used for classification, include: Random Forest (RF), Naive Bayes Classifier, Logistic Regression, Linear Classifier, Neural Networks, K-Means Clustering, Boosting, Perceptron, Decision Tree, Support Vector Machines, Quadratic Classifiers; Bayesian Networks. Among the classification-supervised machine learning algorithms, the following are the best suited for anomaly detection according to [16][17]: Neural Networks, Support Vector Machines, Naïve Bayes, Decision Trees, and Deep Learning. These algorithms are reviewed in the following sub-sections.

2.2.2. Artificial Neural Networks

Artificial Neural Network (ANN) is a supervised machine learning method proposed fifty years ago. ANN are a type of machine learning algorithm that mimics the structure and function of the human brain. They analyse data by grouping raw input, identifying patterns, and labelling information. Since neural networks can only process numerical data, real-world input such as images, text, and sounds must be converted into numerical format before being analysed by the network [18]. The two main parts of a neural network are:

- Connections (weights): These are the links between the neurons in a neural network and hold values that are adjusted during the training process.

- Neurons (nodes): Neurons receive inputs from other neurons via the connections, which have weight values. They multiply them by their corresponding weights and sum them up as shown in Equation (2-1). This result is then passed through an activation function, which determines whether the neuron "fires" or not. This process is mathematically represented in Equation (2-1). The output of this process, Y, is then processed by the activation function (2-2).

$$Y = \sum(input.weight) + bias \quad (2\text{-}1)$$

$$Output = f(Y) \quad (2\text{-}2)$$

Fire means to activate; the word is derived from the brain's fundamental processes. One neuron must have an activation function for the binary classification task. The Sigmoid function is an example of an activation function. The equation for the sigmoid function is:

$$f(z) = 1 / (1 + \exp(-z)) \quad (2\text{-}3)$$

where:

- f(z) is the output value between 0 and 1,
- z is the input to the sigmoid function, which can be any real number,
- exp() is the exponential function, and
- 1 / (1 + exp(-z)) is the formula that maps z to the range (0, 1).

Figure (2-3) shows a Sigmoid function graph.

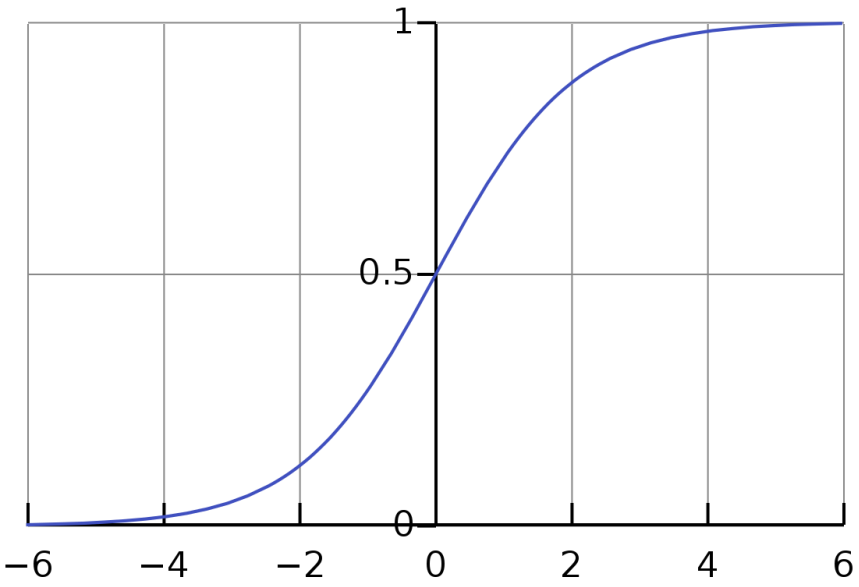

Figure 2-3: Sigmoid Function.

According to [19], a simple neural network or multi-layer perceptron consists of 3 layers as shown in Figure (2-4):

- Input Layer: It is the first layer and responsible for receiving external data.

- Hidden Layer: It is located between the input layer and output layer to transform the input data into output.

- Output Layer: It is the last layer in a neural network to provide the output.

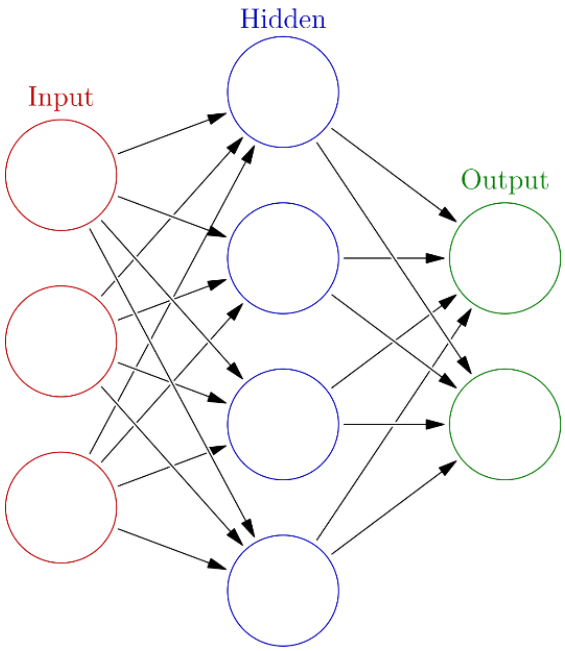

Figure 2-4: Example of a Neural Network Diagram.

Using the backpropagation formula, the weights of neurons inside each layer are adjusted [20]. Equation (2-3) shows the back-propagation formula: W is the change in the edge weight at time t (or $t-1$ for the previous iteration), alpha is the learning rate, and the gradient is the derivation within fraction.

$$\Delta W_t = \alpha * \frac{\partial MSE}{\partial W_t} + \mu * W_{t-1} \quad (2\text{-}4)$$

During training, loss functions are used to calculate the loss between the predicted variable and the output, hence assisting in the training of a neural network [19]. The Mean Squared Error (MSE) function, seen at Equation (2-4), is a loss function. $\widehat{y_i}$ is the predicted result, and n is the number of output classes, y_i is the outcome of the learning process.

$$MSE = \frac{1}{n} \sum_{n}^{i=1} (\widehat{y_i} - y_i)^2 \quad (2\text{-}5)$$

2.2.3. Decision Tree

A Decision Tree (DT) demonstrates how to make a decision based on a specific attribute collection. Figure (2-5) shows that a decision tree has internal decision nodes and terminal leaves. Each decision node, y, implements $f(y(x))$ function with binary outcomes to label the branches by the function output. Any given Decision Tree is fully deterministic; however, specific algorithms may modify their trees based on extra information [21].

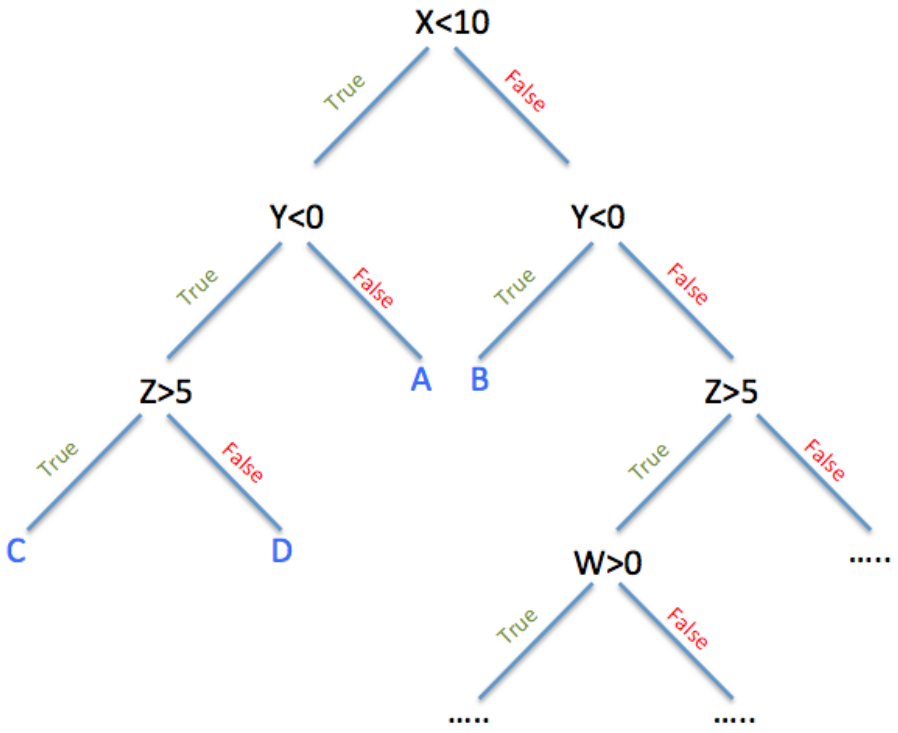

Figure 2-5: An example of a decision tree.

The construction of a Decision Tree is a form of supervised learning. It aims to understand the significance of each piece of data by treating the response as a series of bits. As the most informative feature is placed at the root of the tree, a decision tree uses a heuristic to determine which attribute is the most informative. The heuristic used to identify the most informative feature is based on information theory, which is a method to calculate the amount of information present in data, regardless of its meaning. Therefore, a one-bit response encapsulates one bit of information

about yes/no. Decision trees rely on the data's order, and the tree's optimality is susceptible to change when the order of the data is altered [22]. Nonetheless, the used heuristic is intended to minimize this difference.

2.2.4. Support Vector Machines

Support Vector Machine (SVM) is a technique for supervised machine learning that allows Regression and Classification [23]. SVM plots data points in n-dimensional space, where n is the number of features. The classification is completed by finding an appropriate hyperplane that distinguishes between two classes. In n-dimensional space, the dimensions of the hyperplane are (n-1).

$$\beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn = \beta 0 + \sum_{i=1}^{n} \beta i \backslash xi \quad (2\text{-}6)$$

- β0 is the bias term (also known as the intercept) in the SVM decision function.
- β1, β2, ..., βn are the coefficients corresponding to the features x1, x2, ..., xn, respectively, in the SVM decision function.
- x1, x2, ..., xn are the input features (also known as independent variables) of the data point to be classified.
- The symbol Σ (sigma) represents the summation notation, indicating that we are summing the products of each coefficient βi with its corresponding feature xi from i = 1 to n.

SVM assumes that classes are linearly distinguishable [24]. The equation's sign helps classify classes, while the magnitude aids in determining the distance between the observation and the hyperplane. The class assignment accuracy increases when the magnitude is high. Margin refers to the minimal distance of data points from the hyperplane to either class. A maximal margin is needed so that the magnitude will be high. This hyperplane is hence known as the Maximum Margin Classifier. Support Vectors are the observations that lie on or violate the hyperplane's edge.

Support Vectors assist the hyperplane. SVM has a maximum margin classifier with a soft margin (some observations can violate this margin) as shown in Figure (2-6).

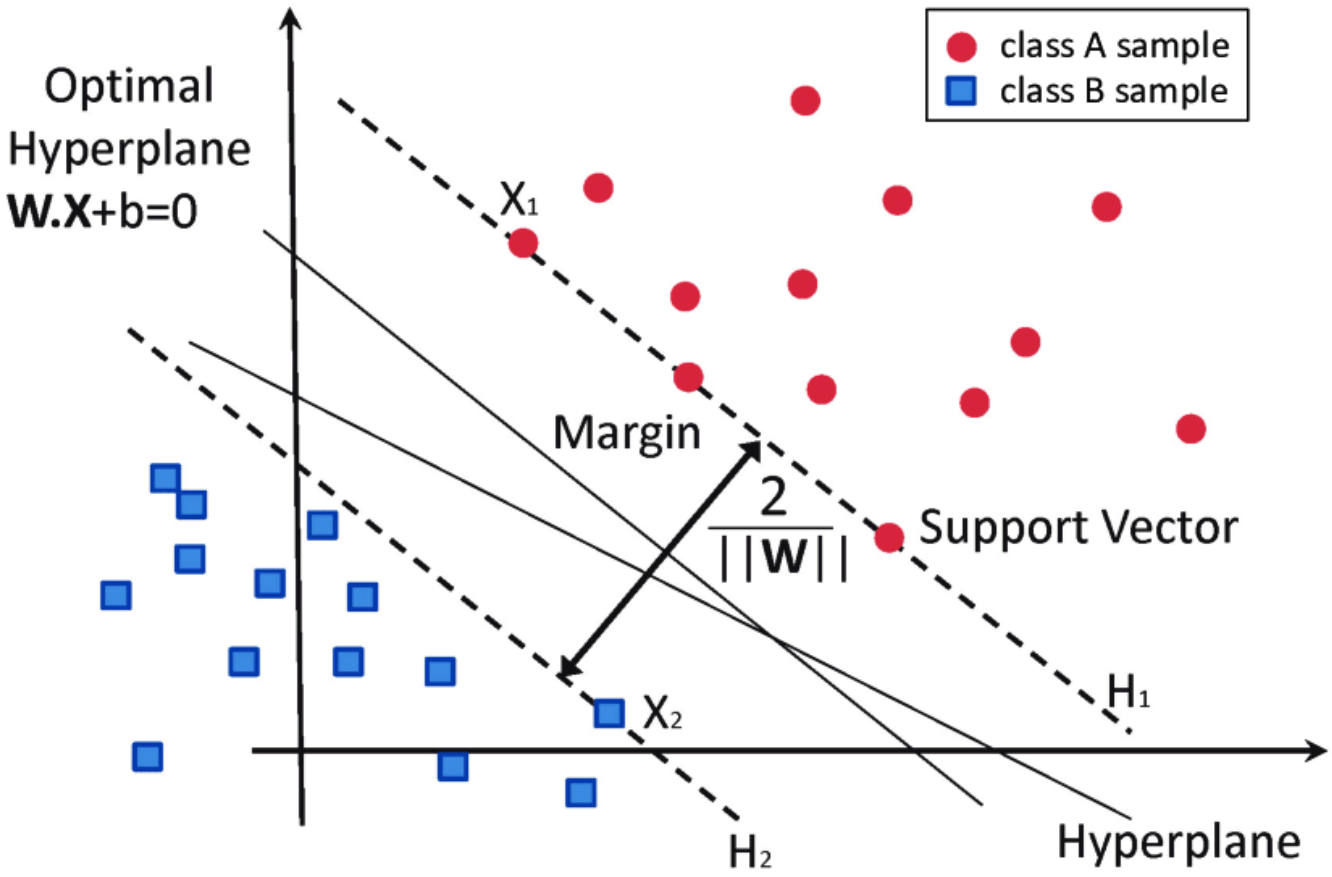

Figure 2-6: Support vector machine algorithm [197].

2.2.5. Gaussian Naïve Bayes

The Gaussian Naïve Bayes (NB) model is a Bayesian probability model that has been dramatically simplified [25]. This model evaluates the likelihood of an outcome given a variety of evidence factors. Given that the outcome happens, the model encodes both the likelihood of the result and the probability of the evidence variables. It is expected that the likelihood of one evidence variable, provided that the end result happens, is independent of the probabilities of other evidence variables, given that the final result occurs.

The Bayes theorem gives a method for computing the posterior probability [26], $P(c|x)$, using the prior probabilities, $P(c)$, $P(cx)$, and $P(x|c)$. A Naive Bayes classifier assumes that the influence of a predictor's value (x) on a given class (c) is independent of other predictor values. This hypothesis is known as class conditional independence.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{2-7}$$

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \ldots\ldots \times P(x_n|c) \times P(c) \tag{2-8}$$

- $P(c|x)$, represents posterior (the updated) probability of targeted class given predictor (features).
- $P(c)$ represents the current probability of targeted class.
- $P(x|c)$ quantifies the probability of a predictor (features) given a desired class, also known as the likelihood.
- $P(x)$, represents the current probability of predictor (features).

2.2.6. Deep Learning

Machine learning's deep learning subfield is inspired by how the brain works, particularly the neural networks that make up the brain [124]. It involves training complex artificial neural networks on large amounts of data, allowing the network to learn and make intelligent decisions. It has been the driving force behind many advancements in various fields, including voice recognition, computer vision, and natural language processing [125]. It has also been used to improve machine learning techniques and contributed to developing self-driving cars, intelligent personal assistants, and machine translation services.

One key difference between deep learning and traditional machine learning is the level of human involvement. In traditional machine learning, the feature engineering process (i.e., extracting useful features from raw data) is done manually by the data scientist. In deep learning, the neural network can automatically learn relevant features from the data [126]. This means that deep learning requires less human intervention and more data and computational power. Another difference is the type of tasks that each approach is suited for. Deep learning is particularly effective for tasks that involve unstructured data, such as image and speech recognition, while traditional machine learning algorithms are better suited for structured data [127].

There are several reasons why deep learning has become popular in recent years [128]:

- Performance: Deep learning models can achieve state-of-the-art performance on a wide range of tasks, such as image and speech recognition.
- Automated feature engineering: Deep learning models can learn useful features from raw data automatically without the need for manual feature engineering.
- Ability to learn from unstructured data: Deep learning models can learn from unstructured data such as images, text, and audio, which is difficult for traditional machine learning models.
- Scalability: Deep learning models can be trained on very large datasets, allowing them to learn complex patterns.

- Availability of hardware: The proliferation of GPUs has made it much easier to train deep learning models, as they can be trained much faster on GPUs than on CPUs.
- Improved performance: Deep learning algorithms have been shown to outperform traditional machine learning algorithms in a wide range of tasks, such as image recognition, speech recognition, natural language processing, and game playing.
- Big data: Deep learning algorithms are particularly effective for large and complex datasets, which are becoming increasingly common in many fields, such as healthcare, finance, and social media.
- Advances in computing power: The availability of powerful GPUs and distributed computing systems has made it possible to train large deep learning models efficiently, which was not possible just a few years ago.
- Open-source software: Many deep learning frameworks and tools, such as TensorFlow, PyTorch, and Keras, are open-source and freely available, making it easier for researchers and developers to experiment and innovate.

As shown in Figure (2-7), there are several deep learning models, such as:

- Convolutional neural networks (CNNs): These are often used for image and video recognition jobs. They are intended to handle data having a grid-like architecture, such as an image, and excel at discovering spatial hierarchies of characteristics.

- Recurrent neural networks (RNNs): These are used for sequential data-based tasks, such as language modelling and machine translation. They are able to analyse temporal data, such as a time series or a natural language phrase.

- Generative adversarial networks (GANs): These are used to generate synthetic data that is similar to a given training dataset. A generator and a discriminator are two neural networks that are used in the GAN and are taught to compete with one another. Both the generator and the discriminator work to generate synthetic data that is indistinguishable from the real

thing, while the discriminator attempts to determine whether or not a particular sample is genuine or synthetic.

- Autoencoders: These are used for dimensionality reduction and feature learning. They are trained to reconstruct their input data using a smaller number of dimensions, which allows them to learn a compact representation of the input data.

- Deep Reinforcement Learning (DRL): is a subfield of machine learning and artificial intelligence that combines deep learning techniques with reinforcement learning principles. It involves training agents to make decisions in an environment by interacting with it and receiving feedback in the form of rewards or penalties. The goal of the agent is to learn an optimal policy, which is a mapping from states to actions, that maximizes the cumulative rewards over time.

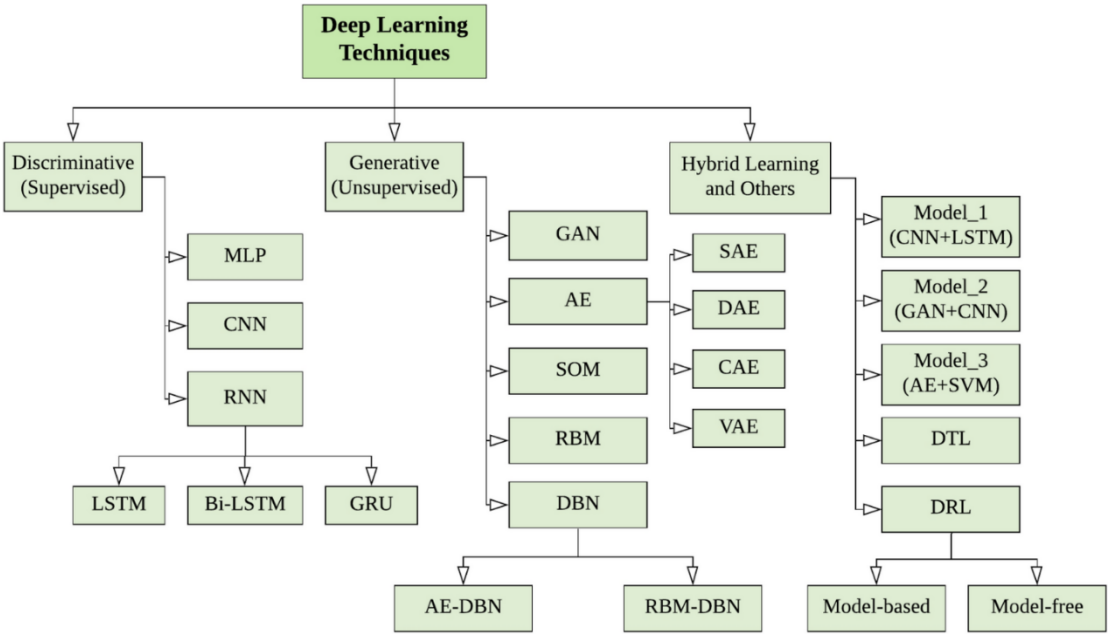

Figure 2-7: A taxonomy of DL techniques [123].

Deep learning can be effective for anomaly detection because it can learn and recognize patterns in data [129]. This can be particularly useful in cases where normal behaviour is complex and not easily captured by simple rules or thresholds. For example, in a network intrusion detection system, the normal behaviour of a network may be very complex. It may depend on many factors, such as the traffic type, the traffic's source and destination, and the time of day [130]. A deep learning model could learn to recognize the normal traffic patterns in a network and flag any deviations from those patterns as potential anomalies. Another advantage of deep learning for anomaly detection is that it can learn from unstructured data, such as images or time series data. This is useful in cases where the data is not easily represented in a structured format or where many features exist.

In network anomaly detection, both RNNs and CNNs could be helpful; however, CNN performs better than RNNs in anomaly detection in classification [122]. One reason a CNN might be a good choice for network traffic anomaly detection is that it can learn to recognize traffic data patterns indicative of normal behaviour. For example, a CNN could be trained on a large dataset of normal network traffic and learn to recognize the patterns and features that are characteristic of normal traffic. It could then be used to identify deviations from those patterns as potential anomalies. Another reason is that CNNs can be trained on very large datasets, which can be beneficial for learning complex patterns in the data. This is particularly useful in cases where normal behaviour is complex and not easily captured by simple rules or thresholds.

Overall, deep learning has the potential to revolutionize many fields by allowing machines to learn and make intelligent decisions on their own. Deep learning can be a powerful tool for anomaly detection, particularly in cases where normal behaviour is complex and cannot be easily captured by simple rules or thresholds. CNN has the potential to be the type of deep learning technique that suits anomaly detection; hence it is detailed further in the following sub-section.

A) Convolutional Neural Network

Convolutional neural networks (CNN) are used to capture high-level features in combination with local spatial features. CNN is often effective with structured and spatially correlated data [27]. When it comes to object detection in photos, they excel. CNN may also be used to analyse words as separate textual units and to do character recognition text analysis. CNN also works effectively with speech data. CNN is most recognised for image recognition [28]; however, nowadays, CNN

is utilised in various applications, including autonomous vehicles, robots, and drones. Figure (2-8) depicts the application of CNNs in computer vision.

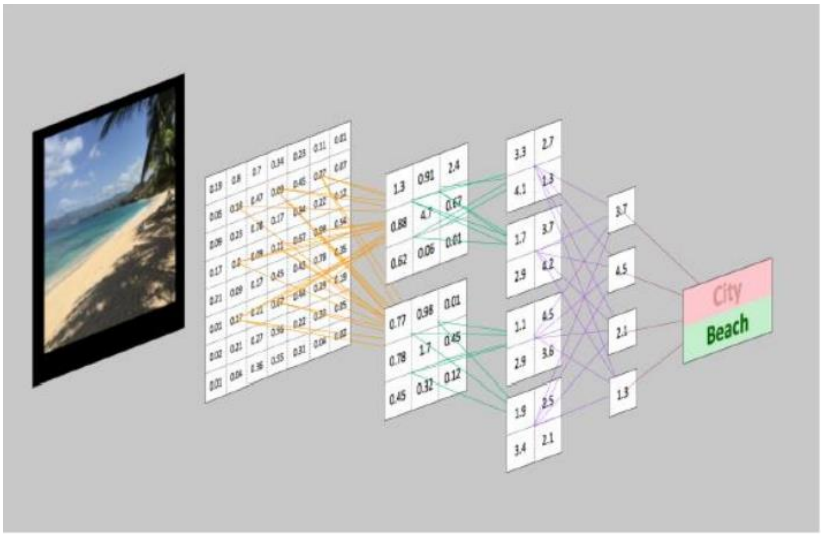

Figure 2-8: CNNs and computer vision [29]

In comparison with natural networks, CNN scales well with data extracted from images. CNN allows data of images to modify the architecture of its network. Using CNNs, neurons may thus be arranged in three dimensions utilising length, height, and depth. These dimensions can be translated to the image's width pixel, height pixel, and RGB channels. CNNs convert the input

picture via a series of interconnected layers and produce a set of class probabilities. As seen in Figure (2-9), all CNN designs have several common layers.

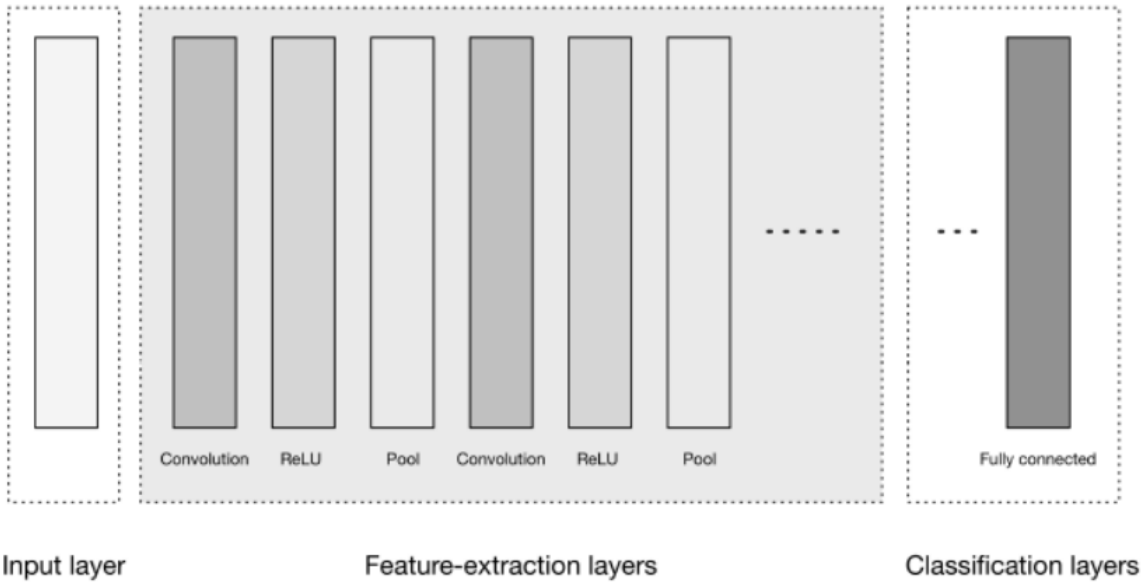

Figure 2-9: High-level CNN architecture [29]

Figure (2-10) shows data loading into the input layer in CNN. For example, the input layer accepts the following dimensions of image data (RGB channel, width, height)

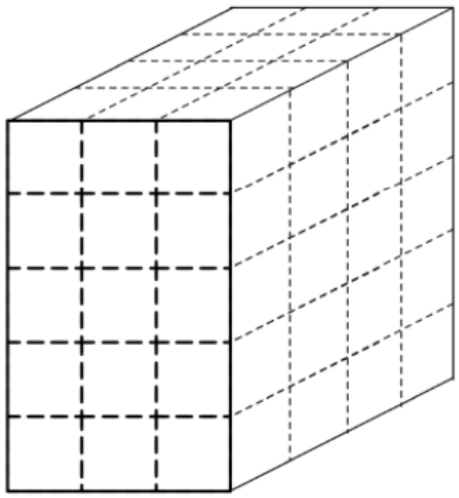

Figure 2-10: 3D data input

CNN consists of several layers. The Convolutional Layer, Rectified Linear Unit (ReLU) Layer, Pooling Layer, and Fully Connected Layer are examples of these layers [30]. Following a detailed explanation of these layers.

- **Convolutional Layers**

The convolutional layer is the main component of the CNN architecture. Convolution layers alter the picture by adding a filter or kernel to the input image. This layer generates the feature map by computing a dot product between the region of neurons in the input layer and the filters [31]. Figure (2-11) shows the Convolution layer's input and output volumes.

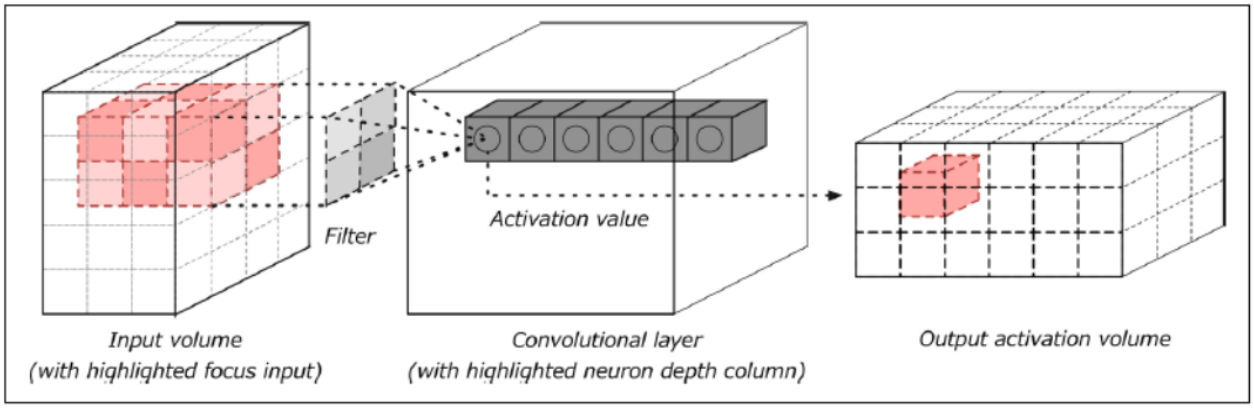

Figure 2-11: Convolution layer with input and output volume [29]

As seen in Figure (2-12), the convolutional layer output has the exact dimensions as the input.

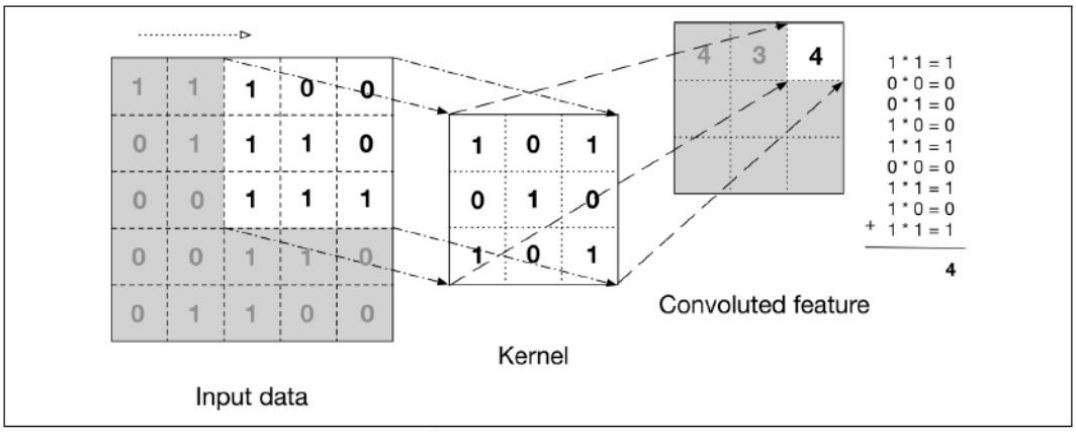

Figure 2-12: The convolution operation [29]

The filter or kernel has smaller sizer than the input size, as shown in Figure (2-12). It applies the supplied stride value to the input data in order to produce complex convoluted features. Feature detector is a common name for this technique. Along the depth dimension, the feature map of each filter (shown in Figure 2-13) is added to form the 3D output. Therefore, each filter learns to recognise a certain characteristic. This filter's two-dimensional activation map is generated by sliding the filter across the input.

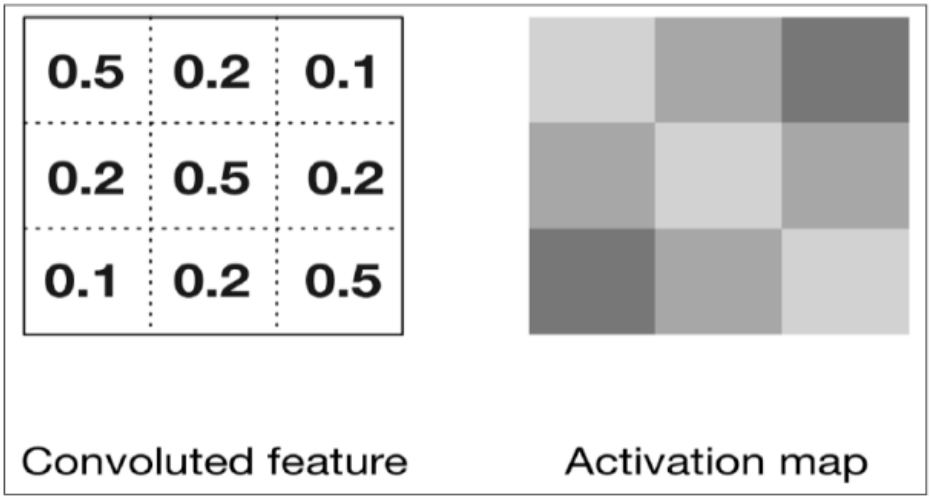

Figure 2-13: Convolution and activation maps [29]

Output volume is determined by the staked activation maps. The activation volume values represent neuronal outputs that encompass a tiny portion of the input volume as shown in Figure (2-14).

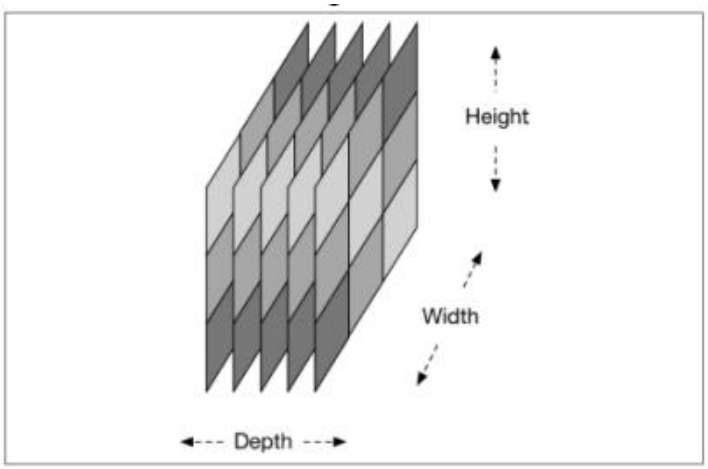

Figure 2-14: Activation volume output of convolutional layer [29]

The receptive field is the area on the neurons that activates them to send information to other neurons. It is used to specify the filter map size. For example, if the filter size is set to 6 x 6 x 4, then the number of weights coming for an output layer neuron is 6 x 6 x 4= 144. CNN uses parameter-sharing to restrict the number of parameters which reduces training time. Each filter learns a single specific feature. As seen in Figure 2-15, once a filter has learned a feature, such as a horizontal line, in one region of the input, it does not need to learn it again for another place in the picture, making CNN's position invariant.

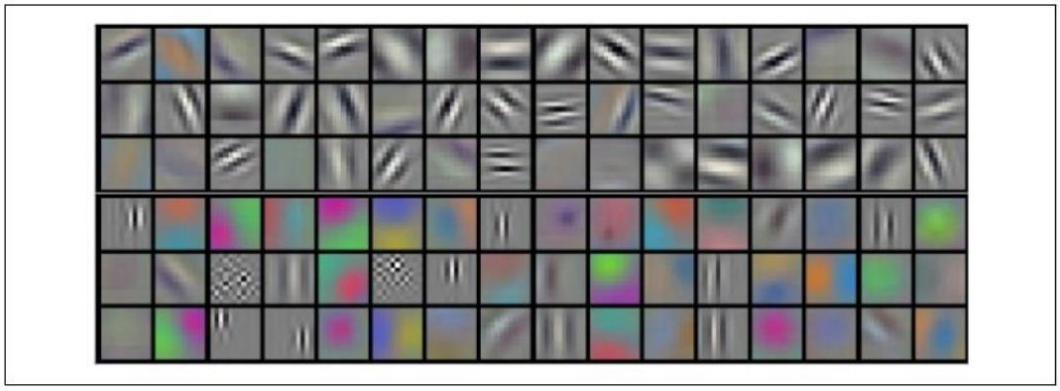

Figure 2-15: Example filters learned [32].

- **ReLU Layer**

This layer employs the ReLU function as a neuron output for the following input x, $f(x) = max\ (0, x)$. CNNs with ReLUs need less time to train than their counterparts with tanh function [33]. The equation for the tanh function is:

tanh(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x)) (2-9)

where:

- tanh(x) is the output value between -1 and 1,
- x is the input to the tanh function, which can be any real number,
- exp() is the exponential function, and
- (exp(x) - exp(-x)) / (exp(x) + exp(-x)) is the formula that maps x to the range (-1, 1).

ReLU is also advantageous since it does not need normalised inputs. Inputs with positive values indicate that these neurons are learning. Thus, local response normalisation aids in error rate reduction.

- **Pooling Layer**

After each convolutional layer, pooling layers are added. In order to minimise the spatial size of the feature maps, the pooling layer combine the output of neighbouring neurons within the same filter or map. Pooling layers reduce the dimensions of the feature maps and aid in preventing overfitting [34]. Each pooling provides a summary of the coverage area. Max pooling employs the max() function to spatially resize input. Using a 2 x 2 filter as an example, max() selects the biggest of the four results. The outcome is a condensed map of features.

- **Fully Connected Layer**

Using the input data, this layer calculates the probability of the data classes. A vector of N values is the output where each value reflects the likelihood of one of N output classifications.

2.3. Intrusion Detection

Every action taken to compromise the security, privacy, or availability of a resource is considered an intrusion [35]. The National Institute of Standards and Technology (NIST) offers a comprehensive document of standards for Intrusion Detection Systems. Effective intrusion detection is a challenging and elusive objective for system administrators and researchers in information security. The complexity of computer systems, the diversity of potential vulnerabilities, and the expertise of attackers combine to produce a problem domain that is exceedingly difficult to solve [36]. An intrusion occurs when a user gains unauthorised access, attempts to get such access, or makes malicious use of information resources. Anomaly intrusions and misuse intrusions are the two categories of intrusion.

Therefore, traditional intrusion detection has concentrated on anomaly or misuse detection. Anomaly detection seeks to identify user or group behaviours that differ from normal patterns. Typically, it involves the development of knowledge bases based on profiles of previously observed actions. Generally, one of the following methods is used to detect anomalies [37]:

- **Threshold detection**, which detects abnormal activity on the server or network, such as abnormal CPU usage on a single server or abnormal network congestion.
- **Statistical measures**, from statistical analysis of historical values.
- **Rule-based measures**, with expert system support
- **Non-linear algorithms**, include Neural Networks and Genetic Algorithms.

Misuse detection is the second method, and it works by contrasting a user's activities with the patterns associated with malicious actors trying to get into a system [38]. Misuse detection uses a rule-based methodology, whereas anomaly detection often uses threshold monitoring to identify instances. Typically, one of the following techniques is used to detect misuse [39]:

- Expert systems providing a set of attack-description rules.
- Signature verification in which attack scenarios are converted into audit event sequences.
- Petri nets, where pictorial Petri nets depict known attacks.
- State-transition diagrams, which express attacks as a sequence of objectives and transitions.

Expert systems, on the other hand, are notoriously rigid, with even little changes to an attack sequence having a major impact on the activity-rule comparison and therefore evading detection. Because of this weakness, particular solutions have been developed to further abstract the rule-based approach, hence decreasing the granularity of the intrusion detection process [40].

Using the log data generated by specialised software, such as firewalls or the operating system, is the most common method for detecting intrusions. A manual examination of these records may suffice to detect intruders. Even after an attack, it is simple to analyse the data to assess the level of damage. This investigation is also crucial for finding intruders and documenting their attack patterns for future detection. A well-designed IDS can be employed to investigate audit data for such insights and is a valuable information system tool. The process of anomaly detection involves the creation of a typical activity profile for each user and the identification of any deviations from that profile that may represent efforts at intrusion. Generation of signatures that include all possible

attacks in order to avoid false negatives and signatures that do not match nonintrusive actions in order to avoid false positives is a crucial component of the process of detecting misuse of a system. On the other hand, false negatives are sometimes seen as being a more significant issue. The determination of threshold values is very necessary in order to guarantee that none of the previously described challenges will be exaggerated bay an excessive amount of value [41].

A variety of IDS commercial tools are generally accessible to security professionals nowadays. The majority of them concern the abuse detection model. As previously stated, the key problem with such systems is their lack of adaptability. Therefore, they are incapable of identifying new, unknown, or unique behaviour and require frequent vendor updates. A complete network security system will include the following three core security components: prevention of attacks, detection of attacks, and reaction to attacks [42]. Table (2-1) displays these constituents.

Table 2-1: Components of a comprehensive intrusion detection system

Prevention	• Guarding against illegitimate users. • Removing application bugs. • Updating protocol implementation. • Stronger passwords. • Antivirus. • Firewalls
Detection	• Network-based IDS. • Anomaly based. • Host-based IDS. • Signature based.
Reaction	• Terminating active network connections. • Source traceback. • Filtration. • Reconfiguration.

	• Rate limiting.

- **Prevention**

The objective of the preventative phase is to strengthen the system's overall security by installing appropriate security devices, eliminating application defects, upgrading protocol implementation, and enhancing the security of all Internet-connected PCs. In prevention, the network administrator implements preventative measures to protect the system from unauthorised users. Although it is impossible to avoid all attacks, the objective is to make DoS attacks more difficult to execute [43].

- **Detection**

A proper phase of attack detection should involve a defence system's response. Every attack detection technique aims to identify intrusions before they cause considerable damage. An intrusion is any unauthorized attempt to access, falsify, modify, or remove information in order to make a system unreliable [44]. A good system can rapidly and with a low percentage of false positives detect attacks. Researchers are making more frequent attempts to develop intrusion detection systems in response to the rising number of attacks (IDSs). The sensor, the analyzer, and the user interface are the primary components of an IDS [45]. The capability of such a system to offer an overview of malicious activity as well as warnings that notify network managers and speed up the response time is the most essential aspect of such a system.

There are two primary types of intrusion detection systems (IDSs) known as signature-based detection and anomaly-based detection. If the monitored traffic matches the signatures in the IDS's database, the system will detect the attack. Signature-based methods are often more efficient and generate fewer false positives, but they require prior knowledge of breaches, leaving a network system vulnerable to new attacks until the signature database is updated. Anomaly-based IDSs detect intrusions by identifying abnormal patterns that deviate from normal behaviour, allowing them to detect new or modified attacks. Table (2-2) highlights the advantages and disadvantages of different detection methods [46]. Hybrid detection systems are a combination of both signature-based detection and anomaly-based detection, where the attack signature database is updated with information about anomalies found during the detection phase.

Table 2-2: Signature and anomaly-based intrusion detection systems: Pros and cons.

	Signature-based	**Anomaly-based**
Pros	• Identity known attacks effectively • Detailed analysis of contextual factors	• Identity new vulnerabilities effectively • Enhance privilege abuse detection
Cons	• Ineffective against new attacks and known attacks variations. • Difficult maintain updates of signatures. • Maintaining knowledge extraction is costly and time-consuming.	• Observable events are frequently changing which results in weak accuracy of normal profiles. • Ineffective when recreating behaviour profiles

There are several strategies and methodologies associated with each category of intrusion detection systems, such as rule-based approaches, statistical measurement, and threshold detection methods [47]. An example of such a methodology is threshold detection, wherein the user determines the amount of network traffic that exceeds a certain threshold. Any measurable deviation from these parameters is considered an intrusion., which therefore causes the alarm to be triggered by the system. In sensor networks, this kind of method is rather standard. The detection method raises an alert for an occurrence when the sensory inputs are greater than a previously determined threshold value [48].

Similarly, a statistical technique determines the typical traffic patterns of network traffic, and an alert is issued if these patterns change significantly. On the other hand, rule-based systems have predetermined sets of rules. A rule-based classifier is activated when detecting a match between an input record and a system rule. Ultimately, evolutionary computation techniques have become an essential tool in numerous research endeavours in this subject, but improvements are expected. These techniques are often used in conjunction with rule-based methods to learn normal and abnormal behaviour [49]. SVM, NB, DT, ANN and CNN are examples of artificial intelligence approaches used to address intrusion detection issues [50].

There are two distinct categories of IDSs, classified by the kind of data used for intrusion detection [51]. Two common types of intrusion detection systems are host-based (HIDS) and network-based (NIDS). A HIDS is a host-based intrusion detection system, therefore it only monitors activities on a single computer system at a time. As a result, HIDS protect essential computers that could hold confidential information. On the other hand, NIDS is not limited to packets that are intended for a particular host; rather, it protects all computers that are connected to the network. A network intrusion detection system (NIDS) is responsible for keeping an eye on and analysing all of the data coming from a network. Correlators are used to prioritise warnings from several detection systems in order to limit the amount of human intervention that is required and to combine the advantages of host-based and network-based IDSs. Correlators are able to do an analysis on warnings as well as comparable group alerts and assign priority depending on the degree of the danger posed to a key resource. This method may be helpful to analysts in spotting possible risks since it reduces the amount of data that they are needed to analyse [52].

- **Reaction**

When an attack is expected, it is necessary to develop reaction mechanisms. The first benefit of an intelligent reaction strategy is that attack traffic uses less bandwidth. The second benefit is that attack flow packets are kept separate from normal flow packets. During the filtering phase, it must make sure that only attack traffic is filtered and that normal traffic is not changed [53]. Combining any detection technique with one or more response mechanisms is viable to achieve improved results [54]. There are many different response techniques that have been reported in the literature. Some of these tactics include terminating active network connections, filtering, designing and implementing new rules, and following tracebacks. Both active and passive defences may be used in response to an attack. The reaction mechanism in an active system provides a response to attack traffic at the same instant that it is received. On the other hand, the attack traffic record is analysed in a passive manner in passive mode [55] to discover the attack sources.

2.4. Internet of Things

2.4.1. Overview of Internet of Things

The Internet of Things (IoT) is a significant technological advancement that is affecting our daily lives. According to sources [79] [80], the number of devices classified as IoT was approximately 27 billion in 2017, and this number is projected to increase to 75 billion by 2025 as shown in Figure (2-16). These devices will be collecting more than 180 zettabytes of data.

The Internet of Things (IoT) is a network of interconnected objects, such as devices, home appliances, and vehicles, that have embedded electronics, sensors, and software to enable them to connect and exchange data. According to [81], any "thing" that is part of the IoT can be reached and controlled through the cyber world. These capabilities make IoT devices more useful and convenient to use.

The Internet of Things (IoT) devices have a wide range of applications, including smart home technology and its associated gadgets such as Radio Frequency Identification (RFID), smart locks, smart meters, wireless sensors, wearable devices, security cameras, smart plugs, Machine-to-Machine (M2M), and Machine-to-Human (M2H) technology. It is believed that the usage of IoT devices will eventually touch every aspect of human life.

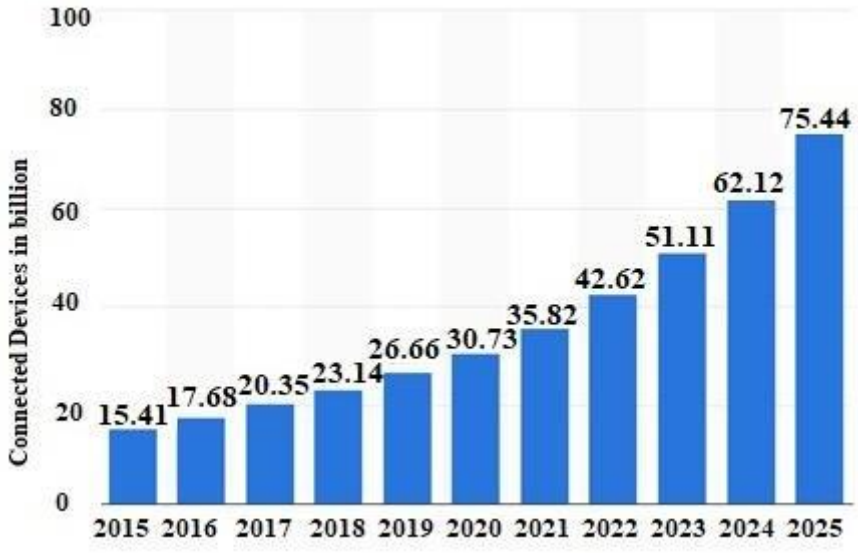

Figure 2-16: Growing number of IoT Devices [82].

2.4.2. Challenges in Securing IoT

Although the IoT incorporates aspects found in earlier computer networking paradigms, the IoT creates an entirely new situation and, thus, significant research difficulties, particularly in the security arena. The following points describe why unique and disruptive IoT security research should be pursued soon [83].

A) Network Size and Number of Devices

The existing strategies and technology for cybersecurity were never intended to scale up to the level of tens of billions of devices [84], which means that securing the networks that make up the IoT will be a huge problem. In addition, the strict spending restrictions imposed by IoT manufacturers impose a restricted amount of memory and computing capabilities in IoT devices, in addition to the utilization of affordable compact batteries. Most significantly, because it will be difficult or perhaps impossible to replace the batteries in these devices (for instance, when the sensors are installed on streetlight poles [85] or when they are implanted within the human body [86]), this procedure will be costly and critical. As a direct consequence of this, increasing energy efficiency is of the utmost importance. A large number of devices and the limited computing, memory, and energy capabilities of IoT devices [87] make it very important to come up with and use new scalable security mechanisms that can do their tasks without putting too much of a burden on IoT devices' capacity to compute or store information.

B) Human Factor

One of the most transformative aspects of the Internet of Things will be the seamless connectivity between humans and machines. Machine learning and artificial intelligence recent developments will make it possible for the IoT to learn and adapt to our individual tastes and ways of living in our homes, workplaces, and when we are traveling. Small sensors can flawlessly distribute drugs [88] and collect physiological data [89] remotely, giving physicians a comprehensive view of our health condition.

On the other hand, if criminals or other unauthorized third parties can get the information that we provide about ourselves, our houses, or our companies, then we may be putting ourselves in a position where we are vulnerable. As a result, protecting personal information and restricting unauthorized access have become essential features of the IoT. The fact that people are now the primary participants in the sensing process raises another issue about the Internet of Things. There is a lack of assurance that humans will produce valid information, for instance, if they are reluctant or unable to do so [90]. This is because there are many factors that may influence human behaviour In order to solve this significant issue, innovative trust and reputation systems that can be scaled to accommodate billions of individuals will be necessary.

C) Complexity

The IoT is a complex ecosystem that will connect humans, machines, handheld electronic gadgets, and other commonplace items into a massively interconnected network. Because there is such a wide variety of devices, there will be multiple protocols, algorithms, and standards for the IoT. This will be especially true in the realm of networking. The majority of the IoT is still dependent on legacy, proprietary technologies. This has led to the creation of an anti-paradigm that is known as the "Intranet of Things" [91]. While some companies are moving toward more open IoT protocols like MQTT and the Internet Engineering Task Force (IETF) protocol stack for limited IoT devices [92], the vast bulk of IoT is still built on incompatible, older infrastructure. In addition, most previous research has assumed that the link between the Internet of Things' resources and the real-world things surrounding it is unchanging. On the other hand, the Internet of Things environment is highly heterogeneous and dynamic due to the unpredictable mobility of IoT devices, which results in abrupt changes in communication capabilities and position over time [93]. This is because IoT devices can be located anywhere in the world. In this kind of environment, fixing accessibility issues with the Internet of Things devices might be challenging.

2.5. Research Gap Identification

Specific essential research difficulties in the realm of IoT security still need to be explored, despite the networking community's growing interest in the topic of IoT security. An overview of the issues and an investigation of them is provided below:

2.5.1. Signature/Event/Rule Based Intrusion Detection Systems

Stephen and Arockiam [94] proposed a hybrid lightweight and centralized method for detecting Sybil attacks and Hello Flood in IoT networks that use the Routing Protocol for Low-Power and Lossy. Their method utilizes an IDS agent, which calculates and monitors the intrusion ratio using detection metrics such as the sent and received number of packets. Raza et al. [95] created SVELTE, IoT real-time IDS. This system combines a firewall, an intrusion detection module and a 6LoWPAN Mapper. To detect the occurrence of network intrusions, it examines the mapped data. Its ability to detect a variety of attacks appears promising. However, only forged or altered data, selective forwarding attacks, and sinkhole attacks have been attempted. Santos et al. [96] and Shreenivas et al. [97] added an intrusion detection module to SVELTE to improve it. To identify suspicious network activity, the intrusion detection module compares the actual count of transmissions with the predicted one. Furthermore, they were able to hint at the location of the attacking nodes. Their results showed that when they merged the expected transmission count and rank-based methods, the total true positive (TP) rate increased.

Pongle and Chavan [98] proposed a hybrid IDS with a centralized and distributed design that they tested with simulated scenarios and networks. It is designed to detect wormholes and other routing attacks. Jun and Chi [99] suggested specification based IoT IDS based on event processing. This system utilizes several event processing techniques to identify attacks. In this system, IoT devices are gathered, events are extracted, and then the system tries to detect attacks by comparing events pattern repository stored rules. Despite being more efficient than traditional IDS, the system utilizes CPU intensively. A bit-pattern technique and a deep packet analysis approach to establish an IDS for IoT were developed by Summerville et al. [100]. Bit-pattern processing is used to

analyse incoming network packets, while n-grams are used to identify features from among a set of similar bits. A match occurs when all relevant bits in the bit-pattern and n-grams match. Four separate attacks are used to evaluate the system, with a low false positive rate.

A lightweight, adaptive and knowledge-driven IDS was proposed by Midi et al. [101]. It captures data of the monitored network and utilizes it to create a compelling collection of detection procedures dynamically. It can be expanded to accept new protocols while also serving as a knowledge exchange platform for incident detection collaboration. According to the findings, the system accurately identified the routing and DoS attacks. Thanigaivelan et al. [102] demonstrated a hybrid Internet of Things IDS. In this approach, each network node keeps an eye on its neighbour. At the data-link layer, the packets coming from the attacking node are blocked by the monitoring node and abnormal behaviour is reported to the parent node. Oh et al. [103] built an IoT lightweight distributed IDS based on a matching mechanism for packet payloads and threat signatures. They put the IDS to the test by using standard attacks and signature-based attacks from older IDSs like SNORT. The results showed that the performance of this IDS is promising. Ioulianou et al. [104] developed a lightweight and hybrid signature-based IDS for dealing with two types of DoS attacks: version number manipulation and "Hello" flood. Despite the positive results, their system has only been using Cooja based simulated environment.

The IoT systems are notably dynamic and diverse in composition, as discussed in section (2.4.2). This feature, in addition to the nearly impossible predictability of the actions of criminal organizations, presents a substantial barrier to the design and implementation of efficient signature-based intrusion detection systems (IDS) for IoT. This study uses machine learning to overcome these challenges and create self-learning and adaptive detection techniques for securing the IoT.

2.5.2. Modelling of Inputs and Attacks on IoT Networks

For machine learning algorithms to work correctly, they need to have a consistent and explicit formalization regarding their input data, states such as (attack/normal), and outputs. To be more precise, the idea of dimensionality reduction, which is used to define the process of selecting and extracting features [105], requires a specific input characterization to be applied. Methods for selecting features look for a subset of the initial features to use in their analyses. Three strategies are used for selecting features, including, filtering (information gain), wrapping (search led by accuracy), and embedding strategy (selecting features based on the number of predicted errors). Sometimes, data analysis tasks like regression and classification may be executed with more precision in the smaller space as compared to the original space in which they were conducted. Defining what an attack is and how to appropriately describe it is another key difficulty that has to be addressed. To put it differently, are we able to formalize and characterize:

- the "regular" state of an Internet of Things network (that is, normal functioning) and

- the "bad" state of an Internet of Things network (that is, an attack is occurring)?

This effort was made in [106], where the authors linked the likelihood of an attack and its related data using Bayesian learning. On the other hand, further study is needed to describe attack classes and the effect that these classes have on the status of the network. In addition, an attacker may utilize this information to fine-tune an attack if they have access to data used in training or an understanding of how machine learning algorithms are trained prior to the creation of the IoT nodes, and unsupervised machine learning algorithms rely on attributes stored on the IoT devices' hardware chip; this information should not be accessible. However, further research has to be done on the effects of attacks of this kind.

2.5.3. Machine Learning Intrusion Detection Systems

There are two types of research on anomaly detection using the machine learning which are binary classification and multiclass classification. The binary classification aims to detect normal/attack states of the traffic without classification the attack. The multi-class classification aims at categorizing the attack into different types such as DDoS, APR Spoofing etc. Here are the studies related to binary classification and multi classification since 2019:

A) Binary Classification

As shown in Table (2-3), 21 studies utilized the KDDCUP99 and its refined version NSLKDD. 12 of the studies achieved more than 99% binary classification accuracy. None of these studies were repeated on an IoT dataset. None of the reviewed studies aimed to or tried to reduce the input size.

Table 2-3: Summary of the reviewed binary classification studies related to the use of Machine Learning in traffic anomaly detection

Article	Year	Model	Dataset	Accuracy
Li et al. [115]	2019	RF	KDDCUP99	96
Yao et al. [112]	2019	MSML	KDDCUP99	96.6
Anthi et al. [114]	2019	DT	Testbed	99.97
Al Hakami et al. [116]	2019	NB	KDDCUP99	84.06
Jan et al. [117]	2019	SVM	CICIDS2017	98
Hwang et. al. [107]	2019	LTSM	ISCX2012	99.99
Arivud. et. al [108]	2019	CNN	NSLKDD	99.67
Vinayakumar et. [109]	2019	ANN	KDDCUP99	93
Faker et. al.[110]	2019	ANN	CICIDS-2017	97.73
Anani et. al.[111]	2019	LTSM	NSLKDD	99.43

Li et. al.[113]	2019	GRU	NSLKDD	82.87
Chen et. al [145]	2019	SVM	KDDCUP99	99.5
Sapre et. al [147]	2019	NB	KDDCUP99	99.37
Kawelah et. al [148]	2019	DT	KDDCUP99	96.47
Wasi et. al [151]	2019	ANN	KDDCUP99	99.23
Kim et. al.[118]	2020	CNN-LTSM	CICIDS-2017	93
Roopak et. al.[119]	2020	CNN-LTSM	CICIDS-2017	99.03
Jiang et. al.[120]	2020	LTSM	KDDCUP99	98.94
Susilo et. al.[121]	2020	CNN	BoT-IoT	91
Ferrag et. al.[184]	2020	RNN	BoT-IoT	98.31
Hai et. al.[132]	2021	LTSM	CICIDS-2017	99.55
Pooja et. al.[133]	2021	BiLTSM	KDDCUP99	99.7
Biswas et. al.[134]	2021	LTSM-GRU	NSLKDD	99.14
Laghrissi et. al.[135]	2021	LTSM	KDDCUP99	98.88
Imrana et. al.[136]	2021	BiLTSM	NSLKDD	94.26
ElSayed et. al.[137]	2021	CNN	InSDN	97.5
Joshi et. al.[138]	2021	ANN	CTU-13	99.94
Alyasiri et. al.[139]	2021	GE	MQTTset	97.94
Hussain et. al.[140]	2021	DT	MQTTset	99.47
Vaccari et. al.[141]	2021	RF	MQTTset	99.68
Christiana et. al [146]	2021	SVM	KDDCUP99	100
Maithem et. al [150]	2021	ANN	KDDCUP99	99.978
Ullah et. al.[142]	2022	BiLTSM	NSLKDD	99.92

Mohammed et. al [143]	2022	SVM	KDDCUP99	99.99
Reddy et. al [149]	2022	ANN	KDDCUP99	99.2

B) Multiclass Classification

The Table (2-4) shows that the most used dataset for the multi-classification in the recent years is UNSW-NB15. Hence this dataset will be used in the multi classification test in Chapter5 also it is detailed further in Section 2.6.2. The Table (2-4) also show that the highest accuracy achieved on the UNSW-NB15 is 89.08%.

Table 2-4: Summary of the reviewed Multi classification studies related to the use of Machine Learning in traffic anomaly detection.

Article	**Year**	**Model**	**Dataset**	**Accuracy**
Ge et al. [182]	2019	DNN	BoT-IoT	98.09
Vinayakumar et al. [109]	2019	DNN	KDD99	92.9
Chouhan et al. [183]	2019	CNN	NSLKDD	89.41
Ferrag et al. [184]	2019	RNN	BoT-IoT	98.2
Li et al. [186]	2019	GRU	Testbed	80.3
Nguyen et al. [187]	2019	GRU	Testbed	95.6
Moreton et al. [188]	2019	LTSM, GRU	Testbed	96.06
Dong et al. [193]	2019	MCA-LSTM	UNSW-NB15	77.74
Yang et al. [195]	2019	ICVAE-DNN	UNSW-NB15	89.08
Aldhaheri et al. [178]	2020	BoT-IoT	SNN	98.73
Ferrag et al. [179]	2020	RNN	BoT-IoT	98.37
Ge et al. [180]	2020	DNN	BoT-IoT	99.79
Malik et al. [181]	2020	LTSDM-CNN	CIC-IDS2017	98.6
Kasongo et al. [189]	2020	CNN-BiLTSM	UNSW-NB15	77.16

Mebawondu et al. [194]	2020	ANN-MLP	UNSW-NB15	76.96
Sethi et al. [174]	2021	Reinforcement	NSLKDD	96.5
ElSayed et al. [137]	2021	CNN	InSDN	97.5
Imrana et al. [136]	2021	BiLTSM	NSLKDD	91.36
Jia et al. [175]	2021	IE-DBN	NSLKDD	98.79
Borisenko et al. [176]	2021	LTSM	CIC-IDS2018	94
Liu et al. [177]	2021	DSSTE-LTSM	NSLKDD	81.78
Acharya et al. [190]	2021	Bagging	UNSW-NB15	87.4192
Aleesa et al. [191]	2021	RNN-LTSM	UNSW-NB15	85.38
Lin et al. [196]	2021	RF-SMOT-C4.5	UNSW-NB15	87.35
Wu et al. [185]	2021	GBDT-SMOTE	NSLKDD	78.47
Ullah et al. [142]	2022	BilTSM	NSLKDD	99.92
Kasongo et al. [192]	2023	XGBoost-RNN	UNSW-NB15	87.07

2.6. Datasets

Many public datasets are commonly used for anomaly detection testing and evaluation, such as KDDCUP99 and UNSW-NB15 [56]. According to [57][58] and as found from the literature, the most used dataset for anomaly detection is KDDCUP99 which is used for testing and evaluation in Chapter 4, Chapter 5, and Chapter 6. According to [59][60], the UNSW-NB15 has more modern-day network attacks. As found in the literature, it is the most used dataset for multi-classification in recent studies; hence it is included in Chapter 5 to evaluate the CNNwGFC model. These datasets are described in the below sub-sections:

2.6.1. KDDCUP99 Dataset

KDDCUP99 is a modified version of the dataset generated by an IDS software at Lincoln Laboratory, MIT, and evaluated in 1998 and 1999. The DARPA-funded program generated the dataset often known as DARPA98. Afterwards, the DARPA98 was selected for usage in the International Knowledge Discovery and Data Mining Tools Competition [62], which led to the creation of the KDDCUP99 dataset [63]. Here are the characteristics of the KDDCUP99 dataset:

A) Genesis

The KDDCUP99 dataset is built from TCPdump data generated from simulated network traffic that was collected in 1998 at Lincoln Labs through the use of the TCPdump program [64]. Following seven weeks of traffic, five million connection data were collected and used as a training set. The test dataset was generated from a two further weeks of simulated network traffic yielded two million test cases. The complete programme is available on the MIT website. The KDDCUP99 data format is a formatted version of the TCPdump data.

B) Target classes

Probing Attack, User to Root (U2R), Denial Of Service (DoS), Remote to Local (R2L), and Normal are the five classes of KDDCUP99 dataset [65]. Each class is further subclassified based on the attack method used. Table (2-5) list the distribution of patterns across class labels.

C) Size and redundancy

The KDDCUP99 training dataset has 4,898,431 data points, but because of a high amount of duplication (78%), only 1,074,992 unique data points were found [66]. The testing dataset has also a lot of redundancy, with 89.5% of it being repetitive, reducing it from 2,984,154 to 311,029 patterns. Decrease in datasets is examined considered in this research.

D) Features

Each pattern consists of 41 features that are assigned a category: Traffic, Basic, or Content. In contrast, an investigation of the Mean Decrease Impurity measure determined that the value 17 was unimportant [67]. Consequently, these 24 characteristics were examined during the evaluation.

E) Skewedness

Table (2-5) reveals the skewed nature of the dataset: 98.61% of the data falls into the Normal or DoS classes. This hinders the performance of classifiers for the remaining classes, as seen in the following sections.

F) Non-stationary

The KDDCUP99 dataset has a non-stationary character whi ch can be seen in the distributions of the training and testing datasets shown in Table (2-5) and Table (2-6). The training set has 23% of DoS cases and 75.61% of Normal cases, whereas the test set has 73.9% of DoS cases and 19.48%

of Normal cases. This kind of disparity has been found to negatively affect performance as per [68].

Table 2-5: Distribution of the KDDCUP99 training dataset.

Class	Percentage	Training Set Count
Normal	75.61%	812,814
DoS	23.00%	247,267
Probe	1.29%	13,860
R2L	0.09%	999
U2R	0.01%	52
Total	100%	1,074,992

Table 2-6: Distribution of the KDDCUP99 testing dataset.

Class	Percentage	Test Set Count
Normal	19.48%	60,593
DoS	73.90%	229,853
Probe	1.34%	4,166
R2L	5.21%	16,189
U2R	0.07%	228
Total	100%	311,029

2.6.2. UNSW-NB15 Dataset

The UNSW-NB15 dataset is sufficiently close to KDDCUP99 in terms of generating process and functionality to be a suitable replacement [61]. It avoids several shortcomings that make KDDCUP99 unsuitable for testing a modern NIDS. Here are the characteristics of the UNSW-NB15 dataset:

G) Genesis

This dataset was simulated over two days at the Australian Centre for Cyber Security (ACCS) using the IXIA PerfectStorm software in 16-hour and 15-hour sessions. 45 unique IP addresses were used over three networks, as opposed to 11 IP addresses utilized across two networks by the KDDCUP99. While usual behaviour was not replicated, attacks were picked from a CVE site that is routinely updated. TCPdump was used to capture communications at the packet level. There are a total of 2,540,044 records on the ADFA website [69]. For the UNSW-NB15 training and testing datasets, a much more compact split was used.

H) Target classes

UNSW-NB15 was created as an enhancement to the outdated KDDCUP99 dataset, it includes 10 different categories: one Normal and nine atypical, including Worms, Analysis, Backdoors, Fuzzers, Exploits, DoS, Reconnaissance, Generic, and Shell Code. The UNSW-NB15 dataset has more target classes (10) than the KDDCUP99 (5), which increases the complexity of the classification task. Additionally, the higher Null Error Rate (55.056% in UNSW-NB15 compared to 26.1% in KDDCUP99) makes the classification task more challenging, which in turn reduces the overall accuracy of the classifier.

I) Size and redundancy

The training set of UNSW-NB15 has 175,341 data points, and the test set has 82,331. Although these numbers are relatively small compared to KDDCUP99, it was found that the size is sufficient for training high-variance classifiers for intrusion detection [131]. There are no duplicate data points in the dataset.

J) Features

49 features were extracted and grouped into five categories - Flow, Basic, Content, Time, and Additional Generated - using Argus and Bro-IDS. The Mean Decrease Impurity method was applied to eliminate unimportant characteristics.

K) Skewedness

Compared to the KDDCUP99 dataset, the skewness of UNSW-NB15 is substantially lower than KDDCUP99 [70].

L) Non-Stationarity

Table (2-7) and Table (2-8) demonstrates that data stationarity is preserved across the training and test sets in UNSW-NB15 since both have comparable distributions [70].

Table 2-7: Distribution of the UNSW-NB15 training dataset.

Class	Percentage	Training Set
Normal	31.94%	56,000
Generic	22.81%	40,000
Exploits	19.05%	33,393
Fuzzers	10.37%	18,184
DoS	6.99%	12,264
Reconnaissance	5.98%	10,491
Analysis	1.14%	2,000
Backdoor	1.00%	1,746
Shell Code	0.65%	1,133
Worms	0.07%	130
Total	100%	175,341

Table 2-8: Distribution of the UNSW-NB15 testing dataset.

Class	Percentage	Test Set
Normal	44.94%	37,000
Generic	22.92%	18,871
Exploits	13.52%	11,132
Fuzzers	7.36%	6,062
DoS	4.97%	4,089
Reconnaissance	4.25%	3,496
Analysis	0.82%	677
Backdoor	0.71%	583
Shell Code	0.46%	378
Worms	0.05%	44
Total	100%	82,332

2.6.3. IoT Dataset

A) Attack Types

This dataset was released in 2019 in the forms of pcap traces (unnormalized) [78]. It contains two types of attacks simulated on IoT devices: direct and reflective. The direct attacks included Ping of Death, Fraggle (UDP flooding), TCP SYN flooding, ARP spoofing, and, while the reflective attacks included TCP SYN, Smurf, SSDP, and SNMP. The researchers used devices such as the WeMo switch and WeMo motion lack substantial computing resources and are prone to malfunctioning when subjected to high levels of traffic. The threshold for what constitutes a "high" traffic rate varies between different devices.

In the reflective attacks simulated in this dataset [78], the traffic rate was kept low in order to keep the targeted device operational while still reflecting the attack. For instance, the WeMo switch would remain working under low attack traffic levels but become non-functional under higher levels, making the attack unsuccessful. The researchers launched various attacks at different levels (low: 1 packet per second, medium: 10 packets per second, high: 100 packets per second) and from different locations (either the internet or a local network). All of the attacks lasted for 10 minutes, and a total of 200 attacks were conducted. The researchers wanted to see how these attacks would affect traffic on various protocols, including ICMP, UDP, TCP, and ARP as well as application layer attacks like SMTP, HTTP, DNS, and HTTPS. The researchers aimed to conduct these attacks without being detected by a classic intrusion detection system.

This dataset included simulations of attacks that were carried out both on the local network and the internet. For attacks originating from the Internet, the researchers used port forwarding to mimic malware behaviour on the gateway. For attacks launched from within the local network, they used IP and port spoofing.

B) Tool:

The researchers developed a Python-based tool that can detect vulnerabilities in consumer IoT devices by conducting tests on the devices within the local network. These vulnerabilities may include weak encryption, SNMP, exposed ports, or unencrypted communication and SSDP. Once vulnerabilities are identified, the tool launches relevant attacks and generates annotations containing information about the targeted IP, the intruders' host information, the start and end time of the attack, the bitrate, the attack protocol, and the attack port number.

C) Testbed:

Figure 2-18 shows the testbed used to capture the pcap traces, The researchers set up a testbed with a TPLink router running OpenWrt firmware and a variety of IoT devices: Amazon Echo, Samsung smart camera, Phillips Hue bulb, Netatmo camera, WeMo switch, Chromecast Ultra, LiFX bulb, WeMo motion sensor, and iHome smart plug, TPLink smart plug. They included two attackers in the testbed, one local to the LAN and one remote on the internet. The researchers used an external hard drive connected to the router and the tcpdump tool to record all network traffic, both local and remote. They TCPdumps of normal and attack traffic over a 16-day period, annotating the attack traffic in the dataset. They also discovered that other attacks, known as wild attacks, were launched from the internet as a result of enabling port forwarding. They recorded all possible user interactions with each device. For example, the researchers used a tool installed on a Samsung galaxy tab to capture the normal behaviour touches. They replayed these touches at random intervals to simulate real user's activity. For the Amazon Echo, they used a program that randomly selected statements from a predetermined text-to-speech list. The researchers collected pcap traces covering a one-month period of normal and attack traffic for ten IoT devices, which were released with annotations for the attacks. The published dataset includes 30 pcap files, each corresponding to a single day trace.

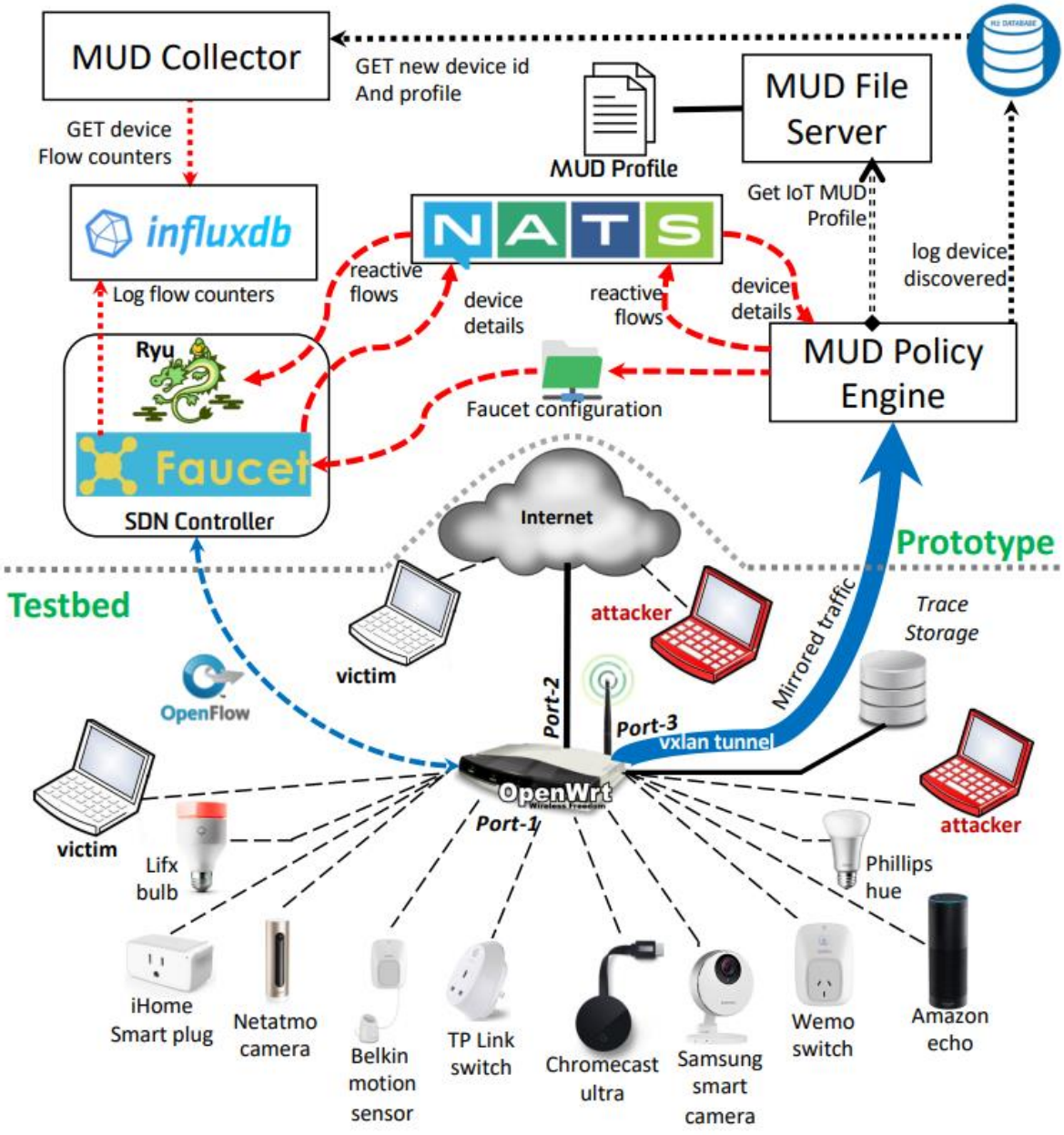

Figure 2-17: System prototype and testbed [78].

2.7. Time Series

The values or occurrences in a time series dataset are typically recorded at predetermined intervals [71]. Insights may be gleaned from the time series data produced by things like real-time surveillance systems, web traffic, network sensors, and data collection tools found online. There are several uses for time series datasets, including quality control, financial analysis, scientific research, and medicinal therapies.

Large volumes of data may be created in dynamic situations and incorporate several data sources. This creates an extra difficulty when analysing time series data. In addition to many data formats, a rapid rate of change, and enormous volumes of gathered data, time may be recorded inconsistently, or data may include noise that obscures the "truth" inside the data. A thorough picture of the sequence of events may be obtained by correlating occurrences from numerous sources.

Models of windows may be categorised as a landmark, sliding, or decaying [72]. Depending on the context, a window might be measured in either time or count. The exponentially decaying window (or damped window) is a sliding window variation in which previous events are given less weight than more recent ones. In landmark windows, aggregated values between a landmark period and the current are included.

In order to handle event streams effectively, sliding windows are often used [73]. Instead of making decisions based on a representative sample or a thorough analysis of all the data, just the most recent data are used. Once the sums of the preceding N values have been computed, they are stored in the respective window as shown in Figure (2-19). As time goes on, new things are introduced while others are removed. Typically, the window has a defined size. Restricting the use of more current data helps ensure that irrelevant data does not affect statistical results.

Figure 2-18: Sliding Window Model

Time series analysis aims to explain how past events might influence future occurrences, how two-time series interact, and to predict future values [74][75]. The following procedures are utilised to achieve these goals:

- Trend analysis entails the identification of a trend, cyclic movement, seasonal changes, or irregular movement. A trend line is used to represent trends over an extended period. The weighted average and least squares techniques are typical approaches for determining long-term tendencies.

- Cyclical movements are long-term swings or oscillations around a trend line or curve.

- Seasonal variations are calendar-based and often recurring changes, such as holidays. Unregular motions are the result of random chance.

- Similarity search identifies sequences with minor differences from a given sequence. In addition, similarity search is capable of matching either partial or whole sequences. Find a comparable-performing stock as an example.

- Clustering classifies time series data into subsets that share some defining feature, such as a high degree of similarity or a small average distance. Classification builds a model from the time series to predict the label of an unlabelled time series.

2.8. Summary

In summary, the Internet of Things (IoT) enables us to link practically everything, including people, devices, and physical items, to the internet, which significantly impacts our society. In the following years, networked devices will be considerably increased, which means that the Internet of Things will offer significant problems to the information security industry. There are several challenges in securing IoT devices, including network size, the number of devices, the human factor as well the complexity of the IoT networks.

There are several gaps in the research related to anomaly detection. The signature-based Intrusion detection systems are not dynamic; hence they lack the ability to detect anomalies. No study currently defines the parameters that could be used for anomaly detection. Most of the models reviewed in the related work are based on a high number of features which results in high usage of resources on the IoT devices which already suffers from resource constraints. Most used standard datasets, such as KDDCUP99 and UNSW-NB15, without testing on real IoT datasets. The highest multi-classification accuracy achieved from the literature on the UNSW-NB15 is 89.08%,

The gaps mentioned above are covered in the following chapters using the reviewed machine learning models: Artificial Neural Networks, Decision Trees, Support Vector Machines, Naïve Byes and Convolutional Neural Networks. Chapter 3 compares the performance of several machine learning algorithms in traffic anomaly detection on KDDCUP99 and IoT datasets. Chapter 4 focuses on optimising the input parameters to reduce the time and resources required to train the models. Chapter 5 aims to develop a model that archives an improved multi-classification accuracy on the UNSW-NB15 than the highest found from the literature (89.08%)

3. Chapter Three: Comparing the binary classification Performance of Machine Learning Algorithms in Anomaly Detection on KDDCUP99 and Created IoT datasets

3.1. Overview

This chapter tests the binary classification performance of the following Machine Learning (ML) algorithms in anomaly detection in the KDDCUP99 and created IoT datasets: Neural Networks (NN), Gaussian Naive Bayes (NB), Decision Trees (DT), and Support Vector Machine (SVM). It shows that Decision Trees and Neural Networks perform similarly in anomaly detection on both datasets. It also shows that the Gaussian Naive Bayes and Support Vector Machines model performance is lower than other models on the IoT dataset.

This chapter includes four other sections: Methodology, Experimentation, Findings and Conclusion. The methodology details the proposed system architecture, the algorithm used, the environment specification, the data sample, and the evaluation performance metrics. The Experimentation section includes the results of applying the above-mentioned machine learning models on the datasets. The Findings section compares the performance of applying the machine learning algorithms on the KDDCUP99 and the IoT datasets. The conclusion section includes a summary of the findings of this chapter.

3.2. Methodology

3.2.1. System Overview

Various assumptions regarding consumer IoT networks are made in the threat model (Figure 3-1). It is assumed that the network includes a middle box device such as a home gateway router, that links the IoT network to other networks and analyses traffic between IoT devices on the local area network and the Internet. This device will analyse, store, alter, and block any network communication that passes through it. This middlebox handles all communication between LAN Wi-Fi devices and Internet-connected devices.

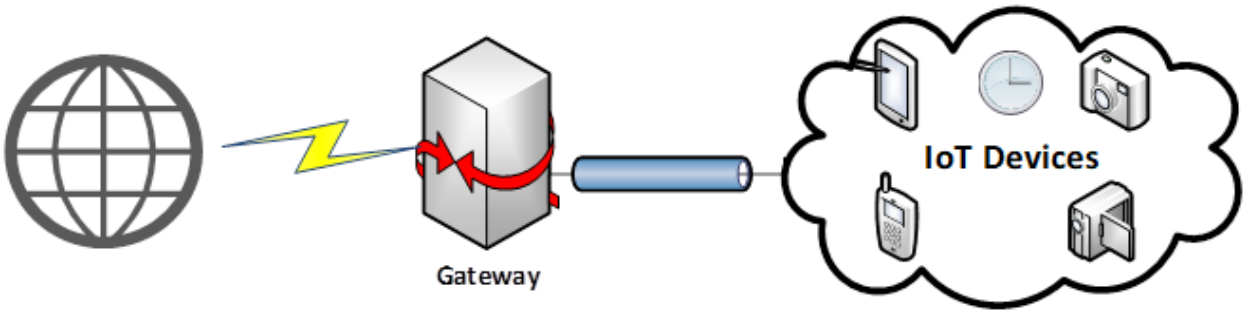

Figure 3-1: Middlebox Approach for Capturing IoT traffic.

The aim is to protect IoT devices from DoS attack traffic; hence they are connected to the middlebox which enables them to send and receive network traffic, including attack traffic. In addition, each device may counter DoS attacks, and the duration of consecutive attacks may vary. Traffic is analysed in time series of 1 second which is shorter than typical DoS attacks to avoid detection [76][77].

The programming logic of the trained model is mentioned in Algorithm 1. Algorithm 1 takes the data captured by the middlebox as well as the instructed ML model such as NN, SVM, NB or DT. After that, Algorithm 1 starts to train the model using the captured data. Once trained model is available, Algorithm 1 starts to analyse the traffic, if anomaly/attack is detected then traffic is blocked if not then traffic is allowed.

Algorithm 1 Machine learning based Anomaly Detection programming

INPUTS: Datasets, machine learning models

OUTPUT: machine learning based Anomaly Detector

PROCEDURE:

```
Read traffic going through the middle box
Apply machine learning model
Train the machine learning model to detect normal/abnormal traffic trend
if Trained machine learning model is available then
    Test the traffic
    if traffic trend is abnormal then
        Block traffic
    else
        Allow traffic
    end if
else
    Wait for creating a trained machine learning model
end if
```

Python is the language chosen to implement the model. Google Colab is the execution environment chosen to implement, train and test the models. At the time of the experiment, the Google Colab allowed the use of 25.6 GB of RAM, Disk space of 225.89 GB and offered Intel(R) Xeon(R) CPU @ 2.20GHz.

3.2.2. Data Sample

Figure (3-2) shows the flow followed to prepare the data sample for training and testing the models. The first dataset used is KDDCUP99 dataset which is detailed in section 2.6.1. The second sample is a subset of an open real IoT dataset which is detailed in section 2.6.3 [58]. The real IoT dataset was collected from an instrumented living lab with 10 IoT devices emulating a smart environment. The real IoT dataset used include several types of IoT devices, including motion sensors, cameras, plugs, lights, and appliances. The real IoT dataset is in the form of Packet Capture (PCAP) traces. The real IoT dataset contains the following type of attacks Address Resolution Protocol (ARP) Spoofing, TCP Sync, Ping of Death, UDP Device, TCP Sync Reflection, SMURF, Simple Network Management Protocol (SNMP), Simple Service Discovery Protocol (SSDP).

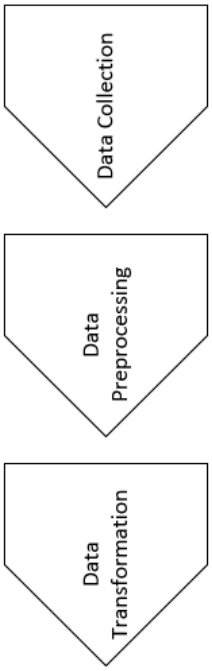

Figure 3-2: Data preparation process [198].

Table (3-1) shows that UDP and TCP protocols represent more than 80% of the data. The number of attack cases represent 1.7% of the dataset as shown in Table (3-2). Table (3-3) shows that the highest used services are TCP, NTP and UDP.

Table 3-1: Percentage of each Transport/Network protocol type in sample dataset.

Protocol Type	**Percent**
ICMP	2.8
IGMP	0.1
TCP	32.3
UDP	48.8
NULL	16.0

Table 3-2: Percentage of attack cases in sample dataset.

Attack	**Percent**
0	98.3
1	1.7

Table 3-3: Percentage of each application in sample dataset.

service	Percent
TCP	21.9
NTP	10.8
UDP	9.4
GQUIC	5.4
ARP	4.8
ICMP	2.8
TLSv1.2	2.8
SSHv2	2.7
TLSv1	2.6
DNS	2.4
STUN	1.0
HTTP	0.6
HTTP/XML	0.6
ICMPv6	0.3
MDNS	0.2
IGMPv2	0.1
MQTT	0.1
Others	31.4

3.2.3. Performance Evaluation

The following metrics were calculated for each model:

Accuracy: It measures the percentage of correct predictions made by the model out of the total number of samples in the dataset. Here is the equation for accuracy:

$$\frac{TP+TN}{TP+FP+FN+TN} \tag{3-1}$$

Precision: It evaluates the ability of a model to correctly identify positive samples (true positives) from the total number of samples. Here is the equation for Precision:

$$\frac{TP}{TP+FP} \tag{3-2}$$

Recall: It evaluates the ability of a model to correctly identify positive samples out of all the actual positive samples in the dataset. Here is the equation for Recall:

$$\frac{TP}{TP+FN} \tag{3-3}$$

F1 Score: It Is used to assess the balance between precision and recall. Here is the equation for F1 Score:

$$\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3-4}$$

where

- TP = The Number of True Positives
- TN = The Number of True Negatives
- FP = The Number of False Positives
- FN = The Number of False Negatives

Accuracy, precision, recall and F1 are used to evaluate the four Machine Learning algorithms chosen for this study which will be detailed in the next section.

3.3. Experimentation

The four machine learning algorithms were tested to classify normal and abnormal traffic. Here are the results for each machine learning algorithm:

3.3.1. Gaussian Naive Bayes Model

Gaussian Naive Bayes Model was implemented using Equation (3-1). The equation assumes that the variables $(a_1, a_2, \dots, a_n|c)$ are independent. The class to be predicted "c" is category of the traffic which is "Attack" or "Benign".

$$P(E|c) = P(a_1, a_2, \dots, a_n|c) = \prod_{i=1}^{n} P(a_i|c) \quad (3\text{-}1)$$

The results are shown in Table (3-4). On KDDCUP99 dataset, it achieved 0.9614 accuracy in detection of attacks. It achieved a 0.9604 F1-Score. It achieved a precision of 0.9628. It achieved a recall of 0.9614.

On the IoT dataset, it achieved 0.89914 accuracy in detection of attacks. It achieved 0.89808 F1-Score. It achieved a precision of 0.91601. It achieved a recall of 0.89914.

Table 3-4: Gaussian Naive Bayes Model Classification Results.

Metric	KDDCUP99 Dataset	IoT Dataset
Accuracy	0.9614	0.89914
F1-Score	0.9604	0.89808
Precision	0.9628	0.91601
Recall	0.9614	0.89914

3.3.2. Decision Tree Model

The C4.5 algorithm was used in implementing the decision tree model. The C4.5 algorithm builds a decision tree from a set of training data, where each internal node represents a decision based on one of the input features, each branch represents the outcome of that decision, and each leaf node represents the class label of the data instances falling into that leaf's region [199]. It is represented by Equation (3-2):

$$Info(S) = -\sum_{i=1}^{k}\left(\left(\frac{freq(C_i,S)}{|S|}\right).log_2\left(\frac{freq(C_i,S)}{|S|}\right)\right) \quad (3\text{-}2)$$

Where:

- Info(S): Represents the information or uncertainty in the set of data instances S with respect to their class labels. It measures how much randomness or disorder exists in the class distribution within S.
- k: Represents the number of distinct classes in the dataset. In other words, if there are k unique class labels, the sum in the formula will have k terms.
- freq(C_i, S): Represents the frequency (count) of instances in S that belong to class C_i. In other words, it counts how many data points in S have the class label C_i.
- |S|: Represents the total number of data instances in set S.
- log_2(x): Represents the base-2 logarithm of x.

As shown in Table (3-5), using the KDDCUP99 dataset, the Decision Tree model achieved 0.9997 in all metrics: accuracy, F1-score, precession, and recall. However, on IoT dataset, it achieved 0.98244 accuracy in detection of attacks. It achieved 0.98243 F1-Score. It achieved a precision of 0.98303. It achieved a recall of 0.98244. These results are shown in Table (3-5).

Table 3-5: Decision Tree Classification Results.

Metric	KDDCUP99 Dataset	IoT Dataset
Accuracy	0.9997	0.98244
F1-Score	0.9997	0.98243
Precision	0.9997	0.98303
Recall	0.9997	0.98244

3.3.3. Support Vector Machine Model

The supervised SVM model was implemented using the "Radial Basis Function" RBF kernel which is represented by the Equation (3-3):

$$K(x, x') = exp(-\gamma \|x - x'\|^2) \quad (3\text{-}3)$$

- K(x, x'): Represents the RBF kernel function applied to two data points x and x', where x and x' are vectors of input features.
- γ (gamma): Is a hyperparameter of the RBF kernel that controls the spread of the kernel function. It determines the influence of each training data point on the decision boundary. Smaller values of γ lead to a broader influence, while larger values of γ result in a narrower influence.
- ||x - x'||^2: Represents the Euclidean distance (L2 distance) between the two data points x and x'. It measures the similarity between the two data points in the feature space.
- exp(): Is the exponential function, which is used to scale the distance term. The exponentiation in the RBF kernel ensures that the similarity decreases as the distance between the data points increases.

As shown in Table (3-6), using the KDDCUP99 dataset, the SVM as shown in Table 3-6, achieved 0.9943 accuracy in detection of attacks. It achieved a 0.9942 F1-Score. It achieved a precision of 0.9942. It achieved a recall of 0.9943.

On the other hand, using the IoT dataset, it achieved 0.89963 accuracy in detection of attacks. It achieved a 0.89860 F1-Score. It achieved a precision of 0.91638. It achieved a recall of 0.89963.

Table 3-6: SVM Classification Results.

Metric	KDDCUP99 Dataset	IoT Dataset
Accuracy	0.9943	0.89963
F1-Score	0.9942	0.89860
Precision	0.9942	0.91638
Recall	0.9943	0.89963

3.3.4. Neural Network Model

The topology of the ANN consisted of three hidden layers of size 8 nodes, 4 nodes and 2 nodes. The output layer is of a single node. Each node is using the following equation to calculate the weight:

$$output = f(\sum_i w_i x_i) \quad (3\text{-}4)$$

The activation function used is the ReLU function which is defined as:

$$ReLU(x) = \max(0, x) \quad (3\text{-}5)$$

Results are shown in Table (3-7). it achieved 0.9987 in all metrics: accuracy, F1-score, precession, and recall on the KDDCUP99 dataset. On the IoT Dataset, it achieved 0.98103 accuracy and 0.98172 precision.

Table 3-7: ANN Classification Results.

Metric	KDDCUP99 Dataset	IoT Dataset
Accuracy	0.9987	0.98103
F1-Score	0.9987	0.98102
Precision	0.9987	0.98172
Recall	0.9987	0.98103

3.4. Findings

When comparing intrusion detection in both KDDCUP99 and IoT sample data set, neural network and DT maintained similar performance. NB and SVM failed to maintain the similar performance in intrusion detection in both KDDCUP99 and IoT sample data set.

Figure (3-3) shows that using Decision Trees and Neural Networks, there is a difference of less than 1.8% in accuracy on both KDDCUP99 and IoT datasets. It also shows there is a difference of more than 8% for SVM and NB.

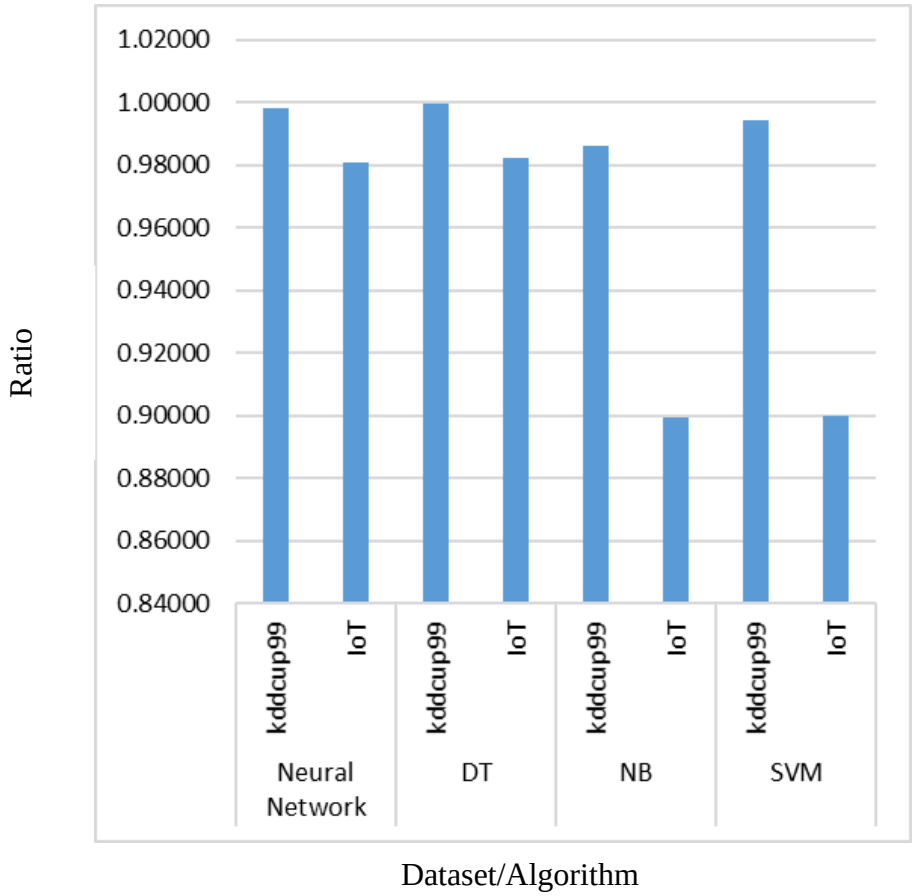

Figure 3-3: Comparing the accuracy result of the IoT dataset with KDDCUP99.

Figure (3-4) shows that using Decision Trees and Neural Networks, there is a difference of less than 1.8% in f1-score score on both KDDCUP99 and IoT datasets. It also shows there is a difference of more than 8.5% for SVM and NB.

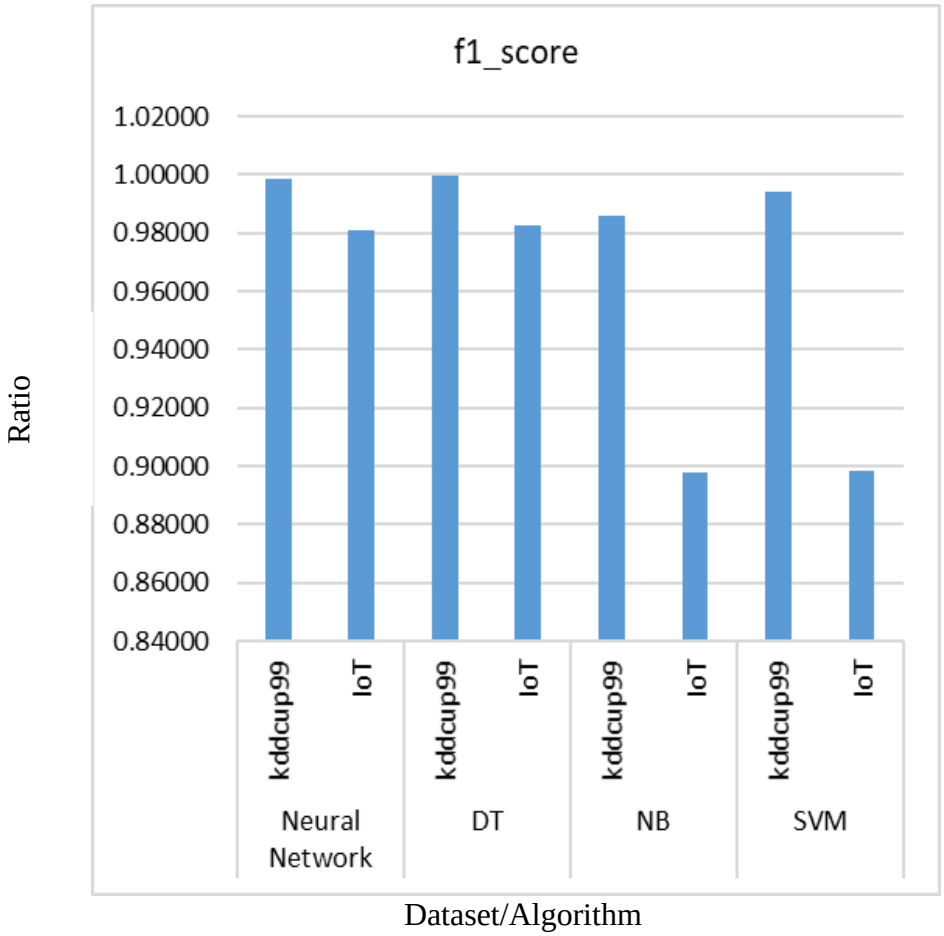

Figure 3-4: Comparing the f1_score result of the IoT dataset with KDDCUP99 dataset.

Figure (3-5) shows that using Decision Trees and Neural Networks, there is a difference of less than 1.7% in precision score on both KDDCUP99 and IoT datasets. It also shows there is a difference of more than 7% for SVM and NB.

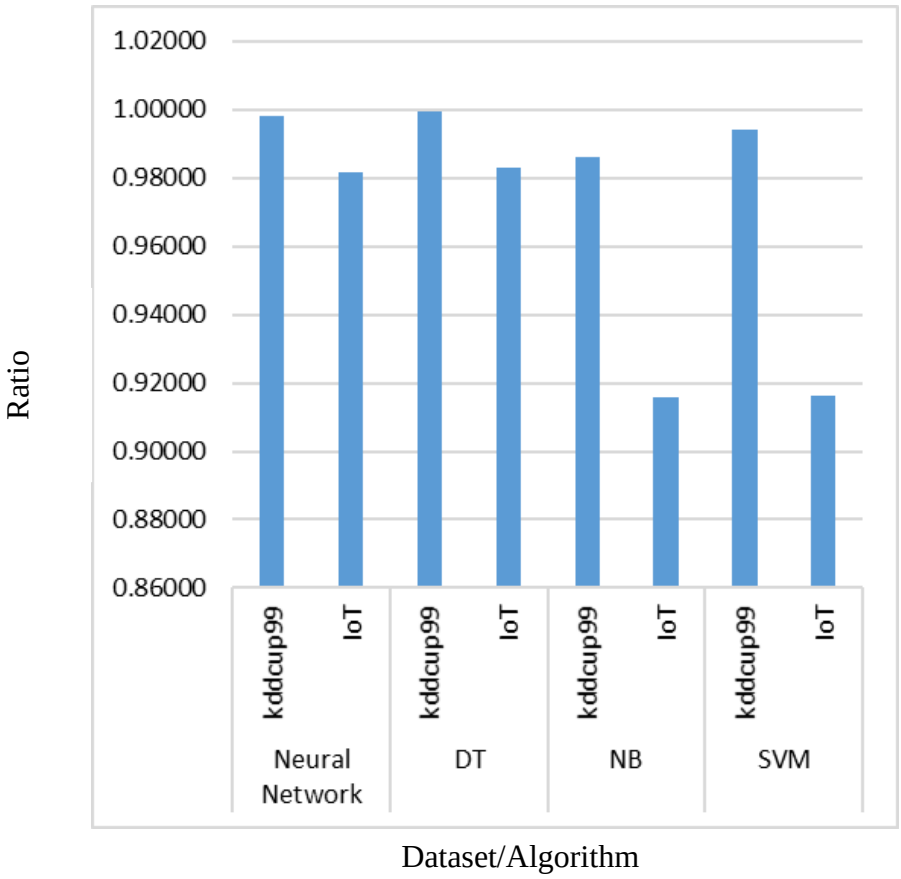

Figure 3-5: Comparing the precision score result of the IoT dataset with KDDCUP99 dataset.

These findings show the following:

- While the SVM and NB machine learning models work well on the KDDCUP99 dataset, as found by [143][145][146][147], they do not perform well on the IoT dataset.

- SVM performance dropped because it works better with a high number of features while it performs lower on a low number of features [152][153]. The KDDCUP99 has 41 features, while the IoT dataset has only 5 features. SVMs are a type of linear classifier that seeks to find the hyperplane in a high-dimensional space that maximally separates the data points of different classes. When the number of dimensions is low, it may be more difficult for an SVM to find a good separation boundary.

- The reason the performance dropped for the NB model is that it assumes that features are independent; however, in anomaly, there is higher collinearity between different features [25][154][155]. In the IoT dataset, the is higher collinearity hence models that detect collinearity will perform better. The Naive Bayes classifier makes the assumption of independence between features because it simplifies the calculation of the probability of a feature occurring given a particular class. Without the independence assumption, the probability would need to take into account the interactions between all of the features, which can be computationally expensive and may not always be possible.

- The NN and DT machine learning models maintained the same performance on both datasets, which shows that they have the potential to be used as anomaly detectors for IoT networks.

- Machine learning algorithms need to be tested on different datasets to verify their performance as they differ from one dataset to another.

3.5. Summary

This chapter explored applying Gaussian Naive Bayes, Support Vector Machine, Decision Trees and Neural Networks machine learning models in anomaly detection in the KDDCUP99 and IoT datasets. When applying the machine learning algorithms on the KDDCUP99 dataset, it is found that Decision Trees, Support Vector Machines and Neural Networks perform the best in binary classification anomaly detection. The Gaussian Naive Bayes model performance is lower than other models. The accuracy of the Gaussian Naive Bayes model is lower by 0.0361 than the average of the other models. The precision of the Gaussian Naive Bayes model is lower by 0.0347 than the average of the other models.

This chapter showed that when applying the machine learning algorithms on a real IoT dataset, the binary classification performance results differ from the performance on the KDDCUP99. The results show that NN and DT perform similarly on KDDCUP and IoT datasets. On the other hand, SVM and NB have a difference of 8% in accuracy. It also shows that SVM and NB have a difference of more than 7% in precision.

When using all the features of the KDDCUP99 or IoT datasets, the findings show potential in using Decision Trees, Support Vector Machines in anomaly detection. However, in reality, only some of the features in the KDDCUP99 dataset are available when capturing network traffic. In addition, using all the features increases the resources and execution time required to generate the results. Hence the next chapter reduces the input size from 41 in the KDDCUPP99 dataset to 5 features only and compares the performance results with the findings of this chapter.

4. Chapter Four: Reducing Anomaly Detection Time in the KDDCUP99 data via Dimensionality Reduction

4.1. Overview

This chapter aims to reduce the input size required by Neural Network, Support Vector Machine, Decision Tree, and Naïve Bayes machine learning models to detect binary classification anomalies in the IoT traffic. It measures the performance of the machine learning algorithms when using five features only as input. It compares the performance of applying the machine learning algorithms using five and all dataset features. The results will show that using only five features, the tested machine learning models can distinguish between normal and abnormal traffic on the KDDCUP99 dataset in a reduced time of 58% on average.

This chapter includes four other sections: Methodology, Experimentation, Findings and Conclusion. The methodology details the proposed system architecture, the algorithm used, the environment specification, the data sample, and the evaluation performance metrics. The Experimentation section includes the results of applying the above-mentioned machine learning models on the datasets using only five features. The Findings section compares the performance of applying the machine learning algorithms using five and all features as input. The conclusion section includes a summary of the findings of this chapter.

4.2. Methodology

4.2.1. System Overview

In order to compare the results between experiments, the same architecture, algorithm and system specifications are used. It is assumed that the network includes a middle box that links the IoT network to other networks and analyses traffic between IoT devices on the local area network and the Internet. This device will analyse, store, alter, and block any network communication that passes through it. It is illustrated in Figure (4-1). It is explained in further details in section 3.2.1.

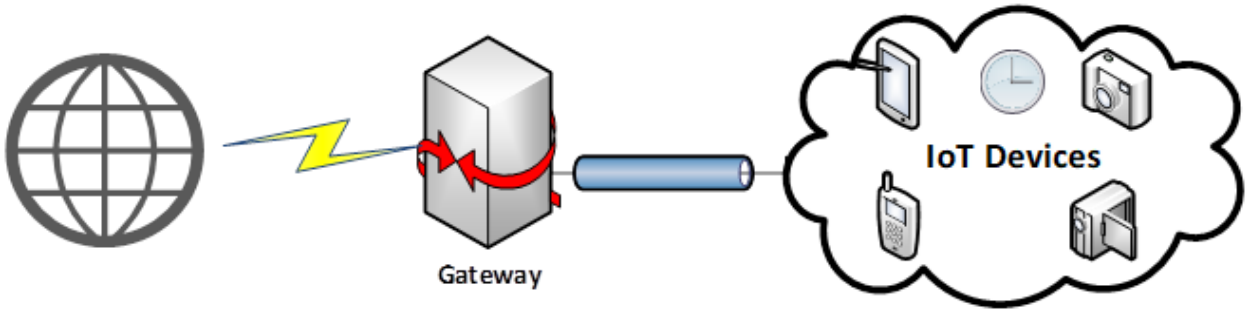

Figure 4-1: Middlebox Approach for Capturing IoT traffic.

Algorithm 1 which is explained in section 3.2.1 in details is used. It takes the data captured by the middlebox as well as the instructed ML model. After that, the Algorithm starts to train the model using the captured data. Once trained model is available, the Algorithm starts to analyse the traffic, if anomaly/attack is detected then traffic is blocked if not then traffic is allowed. System specifications were detailed in section 3.2.1. Python is the language chosen to implement the model.

4.2.2. Selected Features

A network monitor is a tool that records traffic data packets or log results when a network failure or attack occurs. Several types of network monitors are available, each with its own set of features and capabilities [166][167]. For example, Wireshark is a popular open-source network protocol analyser that can be used to capture and analyse network traffic in real time. NetFlow is a Cisco-developed protocol that is used to collect and analyse IP traffic data and can be used for network traffic accounting, monitoring, and analysis.

Most network monitors work by collecting data from network devices, such as switches, routers, or dedicated network collection devices [168]. The collected data is then analysed and reported on, providing network administrators with insights into network activity, performance, and security. Network monitoring is typically performed using a flow-based approach, where a traffic flow is defined using a five-tuple (source IP, source port, destination IP, destination port, and protocol) [169][170]. This information is used as a baseline to create a minimal feature list that can be captured by most network monitors. The following are the features in the minimal feature list:

- **Attack:** The data in benign state were tagged with 0 and during attack with 1.
- **protocol_type:** protocol type of the connection i.e. TCP, UDP and ICMP
- **Application:** for example, http, ftp, smtp, telnet... and other
- **length:** total bytes sent or received in one connection.
- **count:** sum of connections to the same destination IP address occurred in the past 2 seconds.
- **srv_count:** sum of connections to the same destination port number occurred in the past 2 seconds.

Table (4-1) shows sample values for each of the selected parameters.

Table 4-1: Sample values for the selected features

Parameter	Sample Value
Attack	0
protocol_type	ICMP
service	MQTT
length	466
count	13
srv_count	599

Table (4-2) shows the mapping of the traffic flow attributes to the minimal feature list:

Table 4-2: Sample values for the selected features

Feature	Traffic Flow
protocol_type	Protocol
service	Source/Destination Port
count	Source/Destination IP
srv_count	Source/Destination Port

The “length” feature is added to the minimal feature list because it decides the amount of traffic generated between two devices which is a non-trivial information for anomaly detection [171][172]. The “length” can be detected and analysed by network monitors. Most network monitors, including Wireshark, and NetFlow, are able to capture and analyse packet size information as part of the network traffic data [173].

4.2.3. Performance Evaluation

The same performance metrics used in the previous experiment were used in this experiment. Here are the metrics:

Accuracy: It measures the percentage of correct predictions made by the model out of the total number of samples in the dataset. Here is the equation for accuracy:

$$\frac{TP+TN}{TP+FP+FN+TN} \tag{3-1}$$

Precision: It evaluates the ability of a model to correctly identify positive samples (true positives) from the total number of samples. Here is the equation for Precision:

$$\frac{TP}{TP+FP} \tag{3-2}$$

Recall: It evaluates the ability of a model to correctly identify positive samples out of all the actual positive samples in the dataset. Here is the equation for Recall:

$$\frac{TP}{TP+FN} \tag{3-3}$$

F1 Score: It Is used to assess the balance between precision and recall. Here is the equation for F1 Score:

$$\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3-4}$$

where

- TP = The Number of True Positives
- TN = The Number of True Negatives
- FP = The Number of False Positives
- FN = The Number of False Negatives

4.3. Experimentation

The four machine learning algorithms were tested to classify normal and abnormal traffic on the trimmed KDDCUP99 dataset. Here are the results for each machine learning algorithm:

4.3.1. Gaussian Naive Bayes Model

The Gaussian Naive Bayes Model was tested using the five selected features as input. The results are shown in Table (4-3). The results shows that the Gaussian Naive Bayes Model achieved better accuracy and precision after reducing the input size.

Table 4-3: Gaussian Naive Bayes Model Classification Results.

Metric	Before	After
Accuracy	0.9614	0.9862
F1-Score	0.9604	0.9858
Precision	0.9628	0.9862
Recall	0.9614	0.9862

4.3.2. Decision Tree Model

Decision Tree Model was tested using the five selected features as input. The results are shown in Table (4-4). The results shows that the Decision Tree Model achieved a similar accuracy and precision before and after the input size reduction.

Table 4-4: Decision Tree Classification Results.

Metric	Before	After
Accuracy	0.9997	0.9994
F1-Score	0.9997	0.9994
Precision	0.9997	0.9994
Recall	0.9997	0.9994

4.3.3. Support Vector Machine Model

The SVM model was tested using the five selected features as input and tested on IoT dataset. The results are shown in Table (4-5). The results shows that the SVM Model achieved similar accuracy and precision before and after the input size reduction.

Table 4-5: SVM Classification Results.

Metric	Before	After
Accuracy	0.9943	0.9944
F1-Score	0.9942	0.9944
Precision	0.9942	0.9944
Recall	0.9943	0.9944

4.3.4. Neural Network Model

The Neural Network Model was tested using the five selected features as input. The results are shown in Table (4-6.) The results shows that the NN achieved similar accuracy and precision before and after the input size reduction.

Table 4-6: ANN Classification Results.

Metric	Before	After
Accuracy	0.9987	0.9983
F1-Score	0.9987	0.9983
Precision	0.9987	0.9983
Recall	0.9987	0.9983

4.4. Findings

When comparing the results of the experiment done in Chapter 3 which used all features as input and the experiment done in this Chapter which uses only five features: protocol_type, Application, length, count and srv_count. It is found that the performance of the Neural Network, SVM and Decision Trees machine learning models is similar. However, for Gaussian Naive Bayes Model, there is an improvement in the performance.

Figure (4-2) shows that there is less than 0.04% in accuracy between using all variables and the five selected features for Decision Trees, SVM and Neural Networks. However, there is an increase of 2.5796 % for Gaussian Naive Bayes Model.

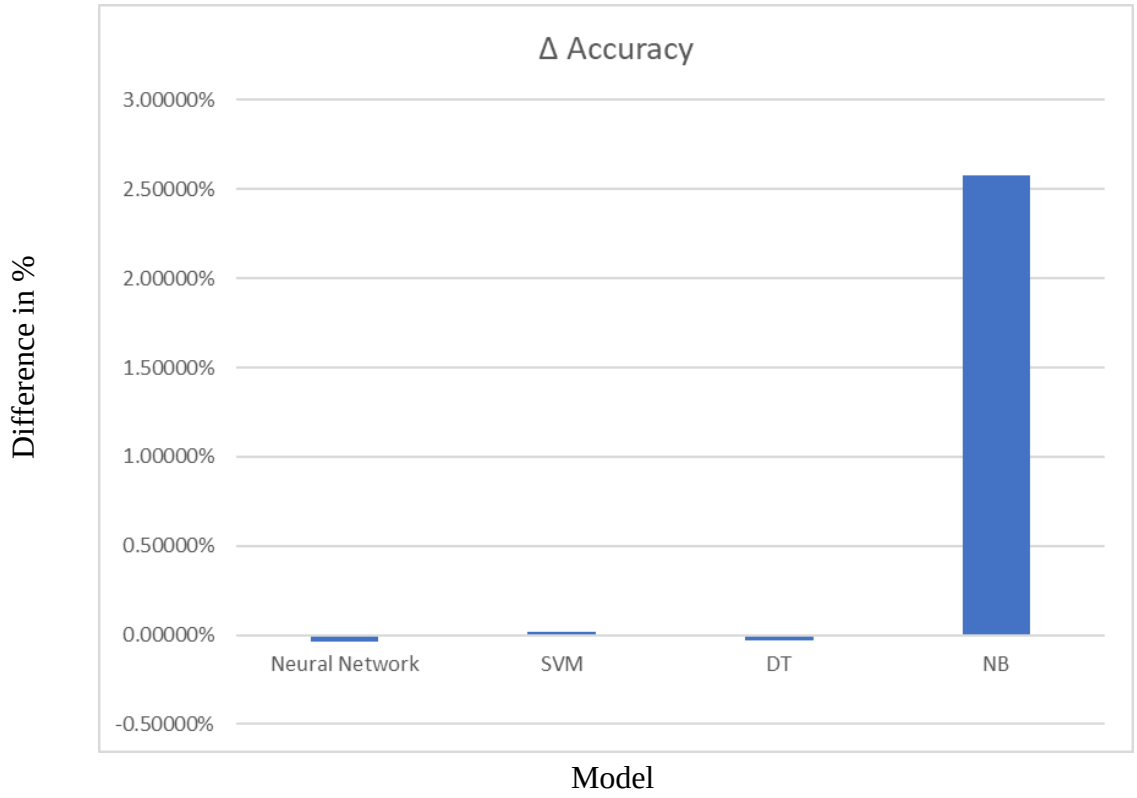

Figure 4-2: The difference in accuracy between using all features and the five chosen features.

Figure (4-3) shows that there is less than 0.04% in F1-score between using all variables and the five selected features for Decision Trees, SVM and Neural Networks. However, there is a difference of 2.6490 % for Gaussian Naive Bayes Model.

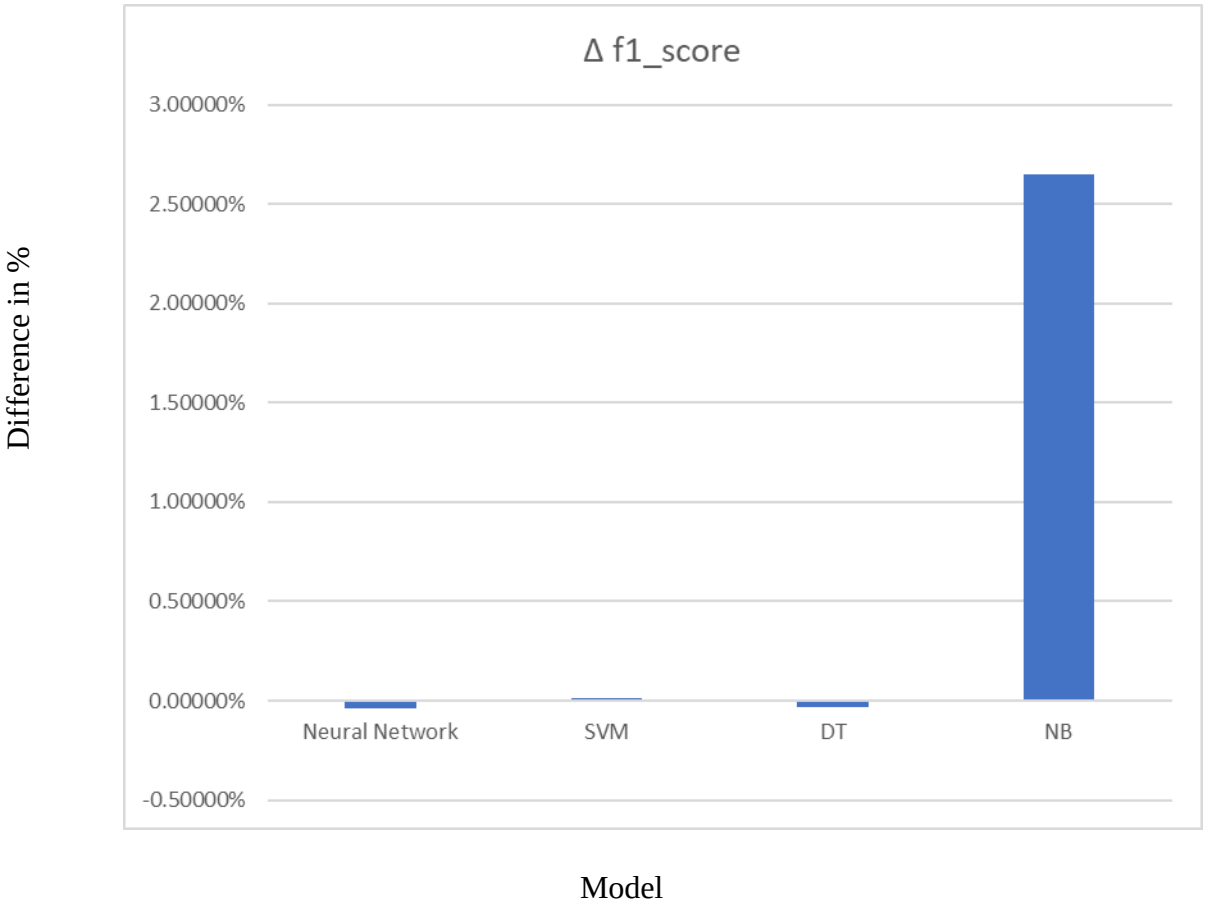

Figure 4-3: The difference in F1-score between using all features and the five chosen features.

Figure (4-4) shows that there is less than 0.04% in accuracy between using all variables and the five selected features for Decision Trees, SVM and Neural Networks. However, there is a difference of 2.24250 % for Gaussian Naive Bayes Model.

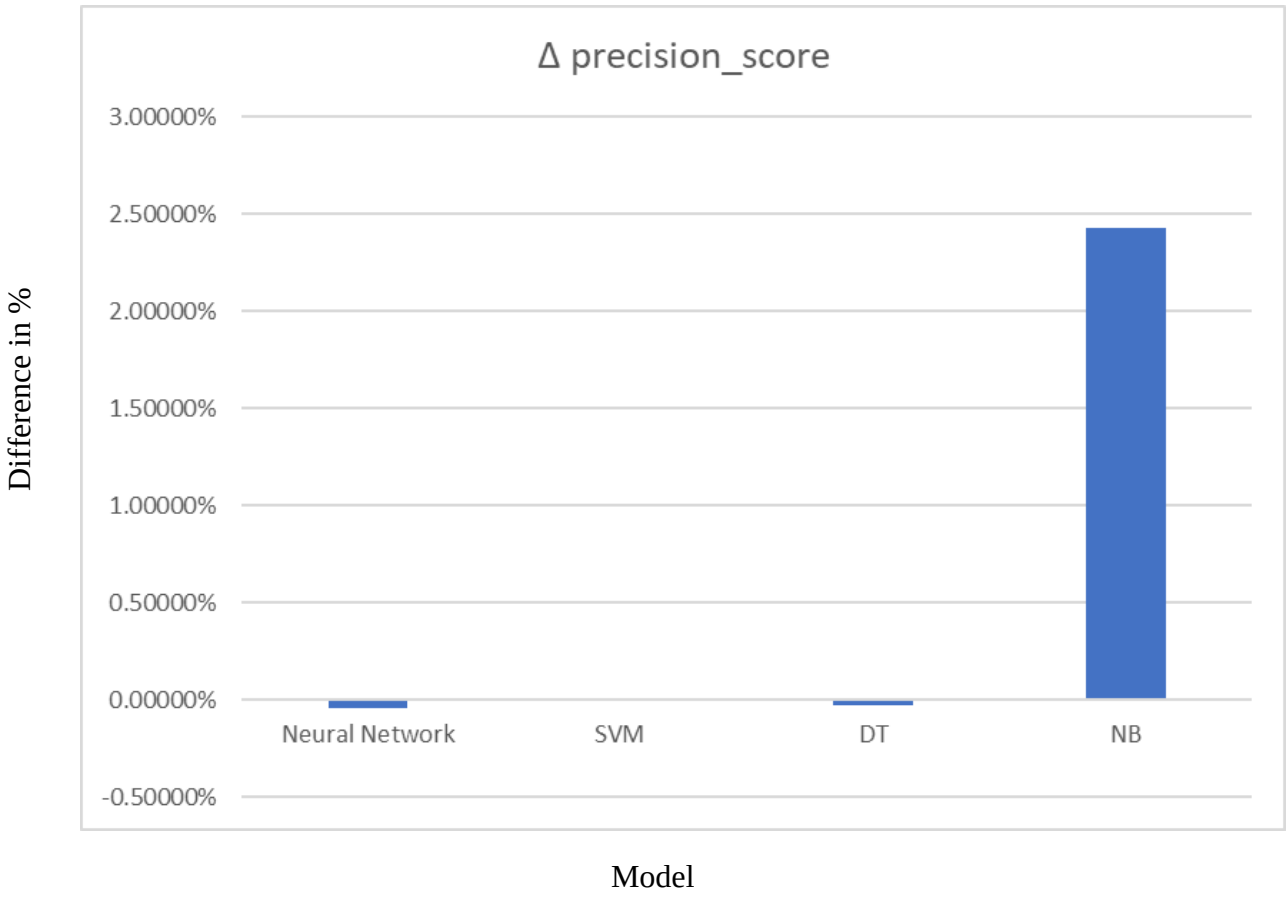

Figure 4-4: The difference in F1-score between using all features and the five chosen features.

Figure (4-5) shows that there is less than 0.04% in accuracy between using all variables and the five selected features for Decision Trees, SVM and Neural Networks. However, there is a difference of 2.57963 % for Gaussian Naive Bayes Model.

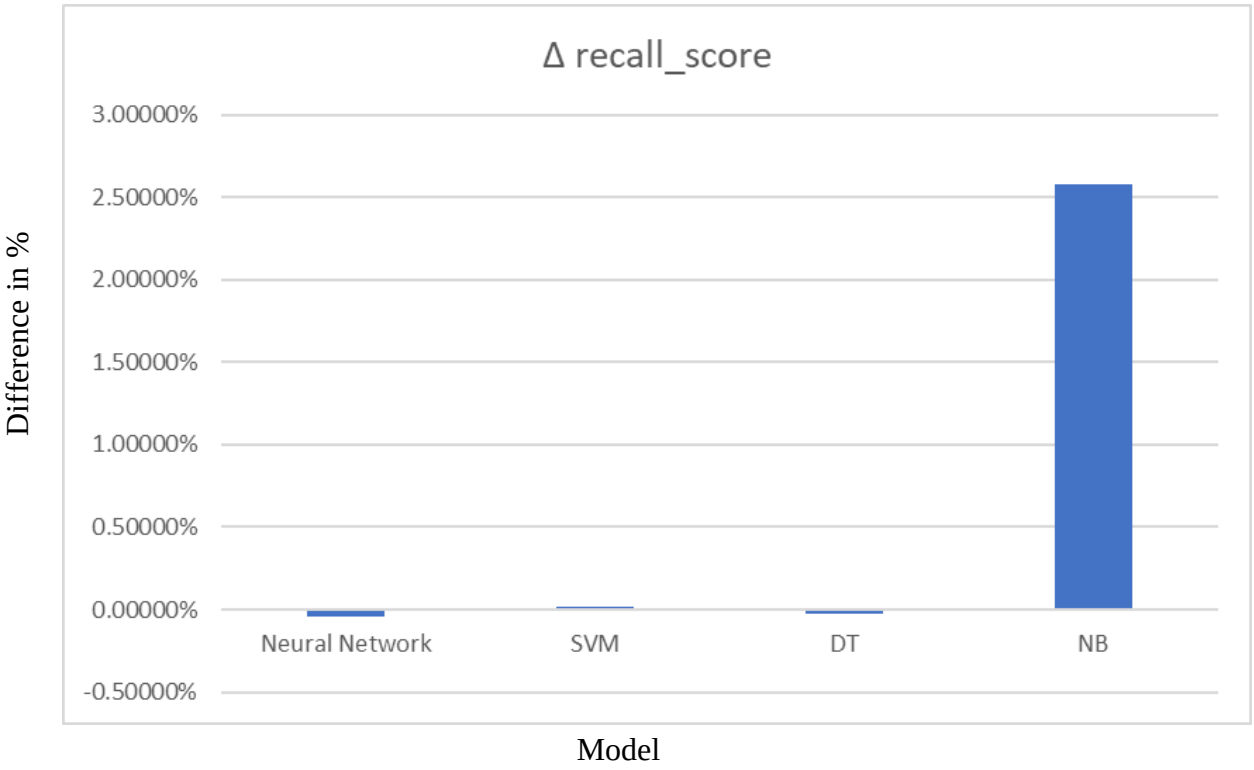

Figure 4-5: The difference in recall between using all features and the five chosen features.

Figure (4-6) shows the time of execution before and after the input size reduction. Figure (4-7) shows that the time for identifying the test sample traffic type is reduced by more 70% for NN, 56% for SVM, 75% for DT and 30% for NB. This result in average of 58%-time reduction for the four models.

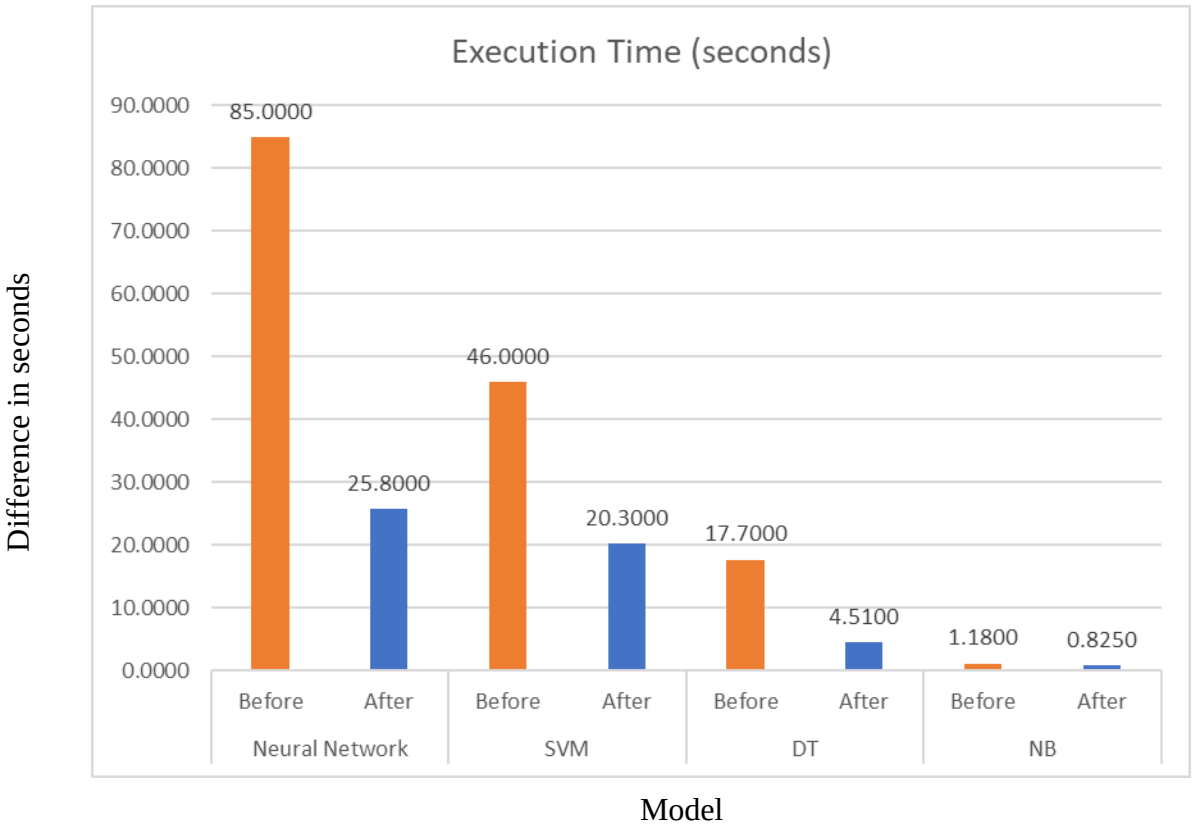

Figure 4-6: Execution time using all features and the five chosen features.

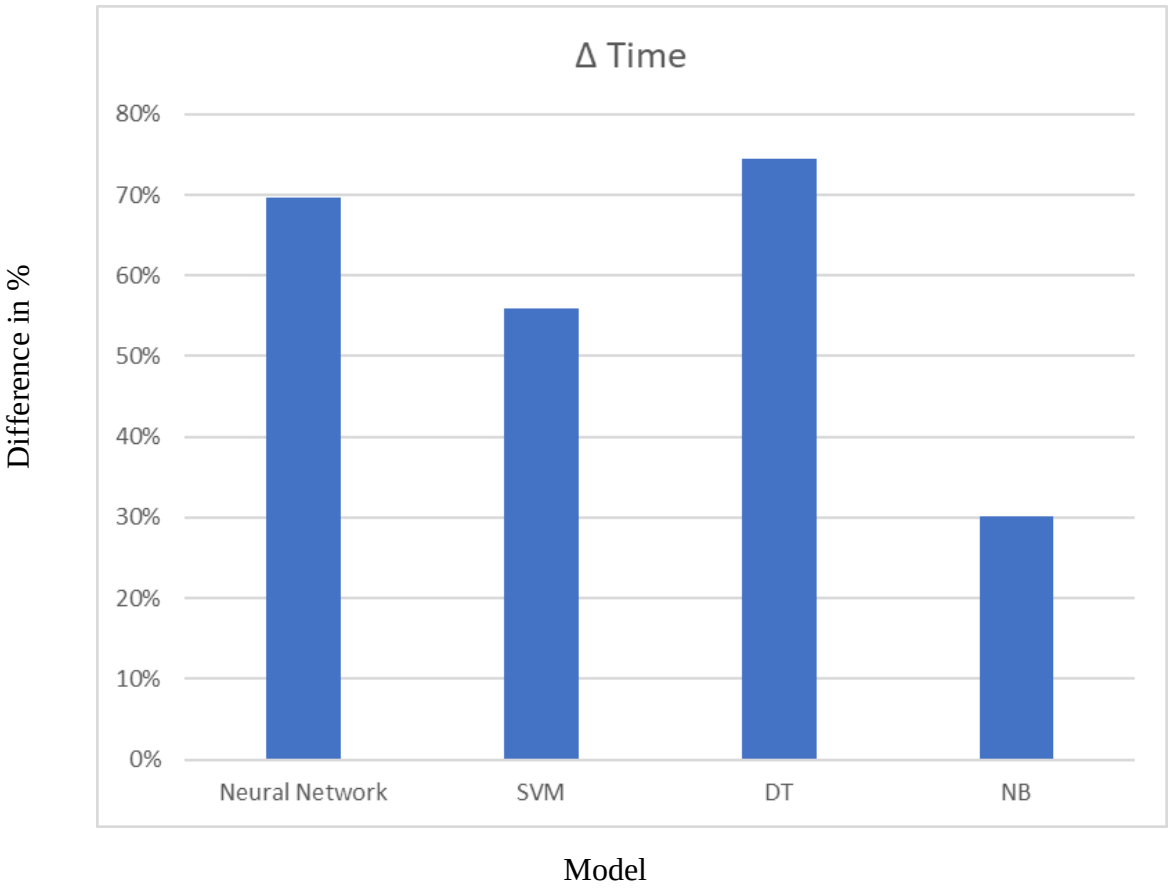

Figure 4-7: The difference in execution time between using all features and the five chosen features.

The findings show that:

- The accuracy, precision, and F1-Score for the Neural Network, SVM and Decision Trees is similar when using five or all the features in the KDDCUP99 dataset.

- The Neural Network achieved similar performance as it can reduce the influence of the irrelevant features (noise) while increasing the influence of the relevant features during learning [155][156][157]. Hence removing the irrelevant features did not impact the performance of the Neural Network.

- Usually, SVM works better with high dimensional data as seen in Chapter3 however when the irrelevant features (noise) are reduced then SVM performance is not affected and, in some cases, it improves when it finds a good separation boundary [158][159].

- The DT achieved similar performance due to its nature in constructing a tree-like model of decisions based on the data features. Hence removing the noise, will not impact the DT much unless an important root leaf feature is removed [160][161].

- The reason for the NB model performance to be better unlike the drop seen in Chapter3 is that KDDCUP99 dataset five features are not highly correlated. The NB works better when the features are independent [25][155].

- There is high difference in execution time for the Neural Network, SVM and Decision Trees.

- Using fewer input features in a neural network generally means that there is less data for the network to process and learn from, which can make training the network faster[162][163]. This is because the network has fewer parameters to update and fewer computations to perform during training.

- Using fewer input features in a support vector machine (SVM) can also make training faster because there is less data for the model to process [164]. SVMs are a type of supervised machine learning model that can be used for classification or regression tasks. They work by finding the hyperplane in a high-dimensional space that maximally separates different classes or values. When there are fewer input features, there are fewer dimensions in this space, which means that the SVM has less work to do in order to find the hyperplane.

- Using fewer input features in a decision tree can also make training faster because there is less data for the model to process [160]. Decision trees are a type of supervised machine learning model that can be used for classification or regression tasks. They work by learning a series of rules that can be used to make predictions based on the values of the input features. When there are fewer input features, there are fewer rules that the decision tree has to learn, which means that it has less work to do.

- Naive Bayes classifiers are generally fast to train comparing to other models as seen in previsions studies [144][165] because they are based on simple probability estimates hence, they achieved lower difference in execution time than other models.

- Using fewer input features in a Naive Bayes classifier can make training faster because there is less data for the model to process. Naive Bayes classifiers are a type of supervised machine learning model that can be used for classification tasks. They are based on the idea of using Bayes' theorem to predict the probability of a given class label, based on the values of the input features. When there are fewer input features, there are fewer probabilities that the classifier needs to estimate, which means that it has less work to do.

4.5. Summary

In summary, this chapter compared the performance of the Neural Network, Support Vector Machine, Decision Tree, and Naïve Bayes machine learning models in binary classification anomaly detection when using all KDDCUP99 dataset features and five selected features as input. The selected input features are protocol type, service type, total length, count of connections from the same IP and number of connections from the same destination port. The results show that the applied machine learning modes can distinguish between normal and abnormal attacks using all or five features only. The Neural Network, Support Vector Machine, and Decision Tree maintained the same performance. The Naïve Bayes achieved better performance when using only five features. The benefit of using only five features is the reduced time required to train and process data by the machine learning model. The results show a reduction in time required by 58% on average. Hence, it is concluded that using only five features; the tested machine learning models can distinguish between normal and abnormal traffic on the KDDCUP99 dataset in a reduced time of 58% on average.

5. Chapter Five: Anomaly Detection and Classification using CNNwGFC

5.1. Overview

This chapter tests the Convolutional Neural Networks with Global Feature Correlation (CNNwGFC) model, which is an enhanced Convolutional Neural Networks model, in detecting and classifying anomalies in network traffic data. In this chapter, the CNNwGFC model is tested on two datasets: KDDCUP99 and UNSW-MB15 datasets. The performance results are compared with Neural Network Model and the classic Convolutional Neural Network model. The results will show the global feature correlation structure can be added to the CNN model in order to improve the convolutional network's ability to achieve better network traffic anomaly classification results.

This chapter includes three other sections: Methodology, Findings and Summary. The Methodology section includes a description of the CNNwGFC model; the datasets used for testing, the experiment environment specifications, and the performance metrics used to evaluate the CNNwGFC model. The Findings section shows the performance results of applying the CNNwGFC on the datasets and compares its performance with Neural Networks and classic Convolutional Neural Networks. The conclusion section includes a summary of the findings of this chapter.

5.2. Methodology

The CNNwGFC detection model presented in this thesis makes use of CNN in order to extract both global and local characteristics of the target data set. After going through some preliminary processing, the raw data is then split into a training set and a testing set (70:30), as shown in Figure (5-1). Through the use of the training set, the model is able to learn the traffic characteristics of a variety of attacks. The data from the test set is then used to validate the accuracy of the CNNwGFC detection model. Figure (5-1) presents a representation of the model's iterative learning process.

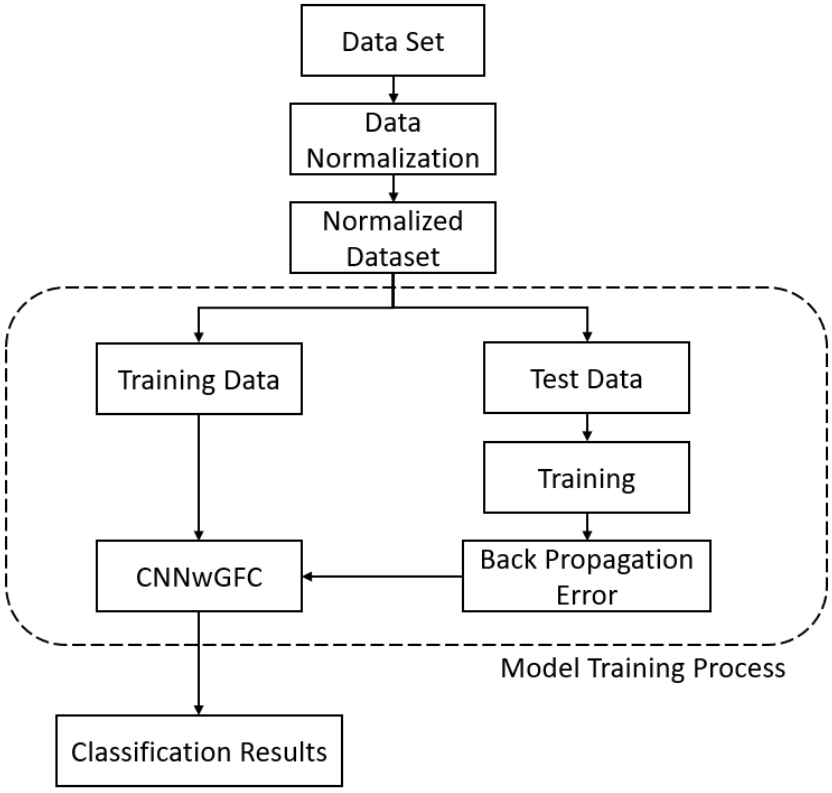

Figure 5-1: CNNwGFC model Training Process

5.2.1. CNNwGFC Model

CNN is a model of an artificial neural network that can be trained on a global scale and has many stages. It can create particular network topologies for certain tasks, and after pre-processing the data, it can learn general, abstract features and concepts from the pre-processed data. When dealing with network traffic data, the traffic characteristics of each type of attack are converted to images of two-dimensions. After that, a convolutional neural network is used to extract the local as well as global features of the images to identify relevant traffic features indicating various types of attacks.

Figure (5-2) shows the enhanced network structure, consisting of three convolutional layers, three pooling layers, Global Feature Correlation layer, fully connected layer, and SoftMax classifier. The input layer receives a two-dimensional matrix that is made of the characteristics of the network's traffic, and the size of this matrix is what defines the size of the input vector. The function of the convolution layer is to extract and map data from one plane to the next. At the same time, the pooling layer functions as a fuzzy filter and is responsible for the extraction of secondary features. Because the data on network traffic goes via the GFC structure in the final feature layer, the feature mapping has been adjusted to extract the global characteristics that are relevant to the situation. The fully connected layer is responsible for connecting all the features and pass the results to the SoftMax function.

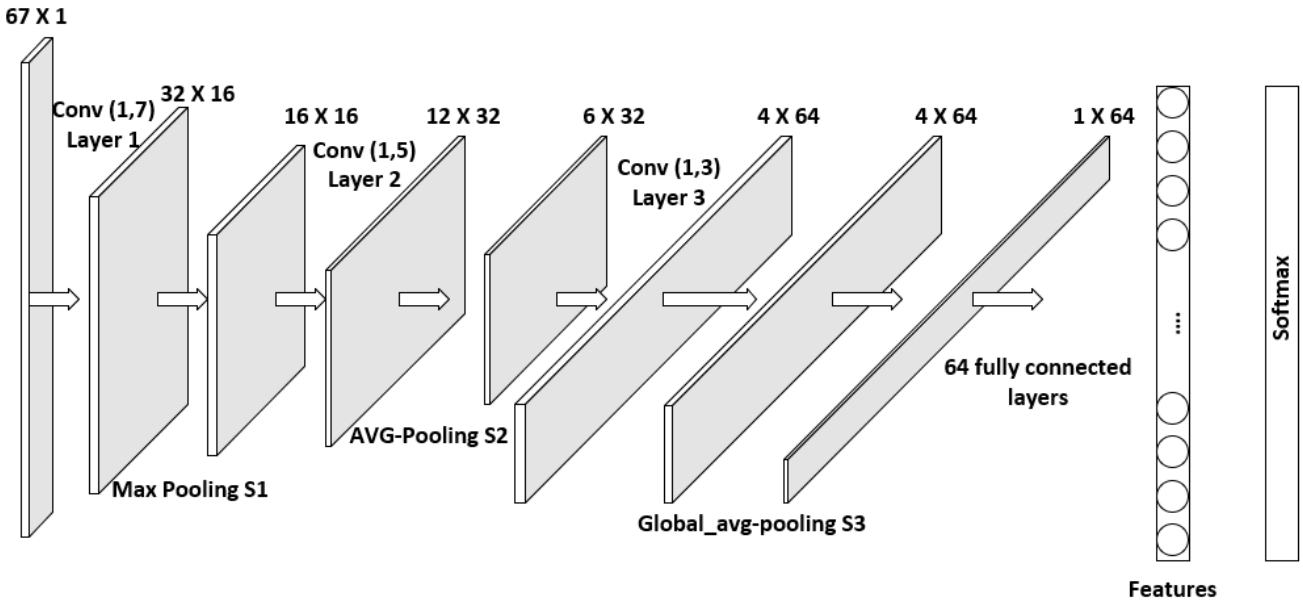

Figure 5-2: CNNwGFC Model

5.2.2. GFC Structure

This section details structure of the Global Feature Correlation (GFC) structure. Figure (5-3) illustrates the local features of the GFC structure. In this thesis, the data flow is processed into images of two-dimensions. The local characteristics of benign and abnormal traffic are markedly different. However, the partial differences in features between different types of abnormal traffic are not obvious and require macroscopic observation. The last feature layer of the model is added to further optimize the mapping of features. The network structure design is shown in Figure (5-3). The structure includes, three convolution operations, two matrix multiplications, residual skip connection to prevent the vanishing gradient problem. The convolution with the RELU activation function is used to produce normalized feature map. The matrix size is constant.

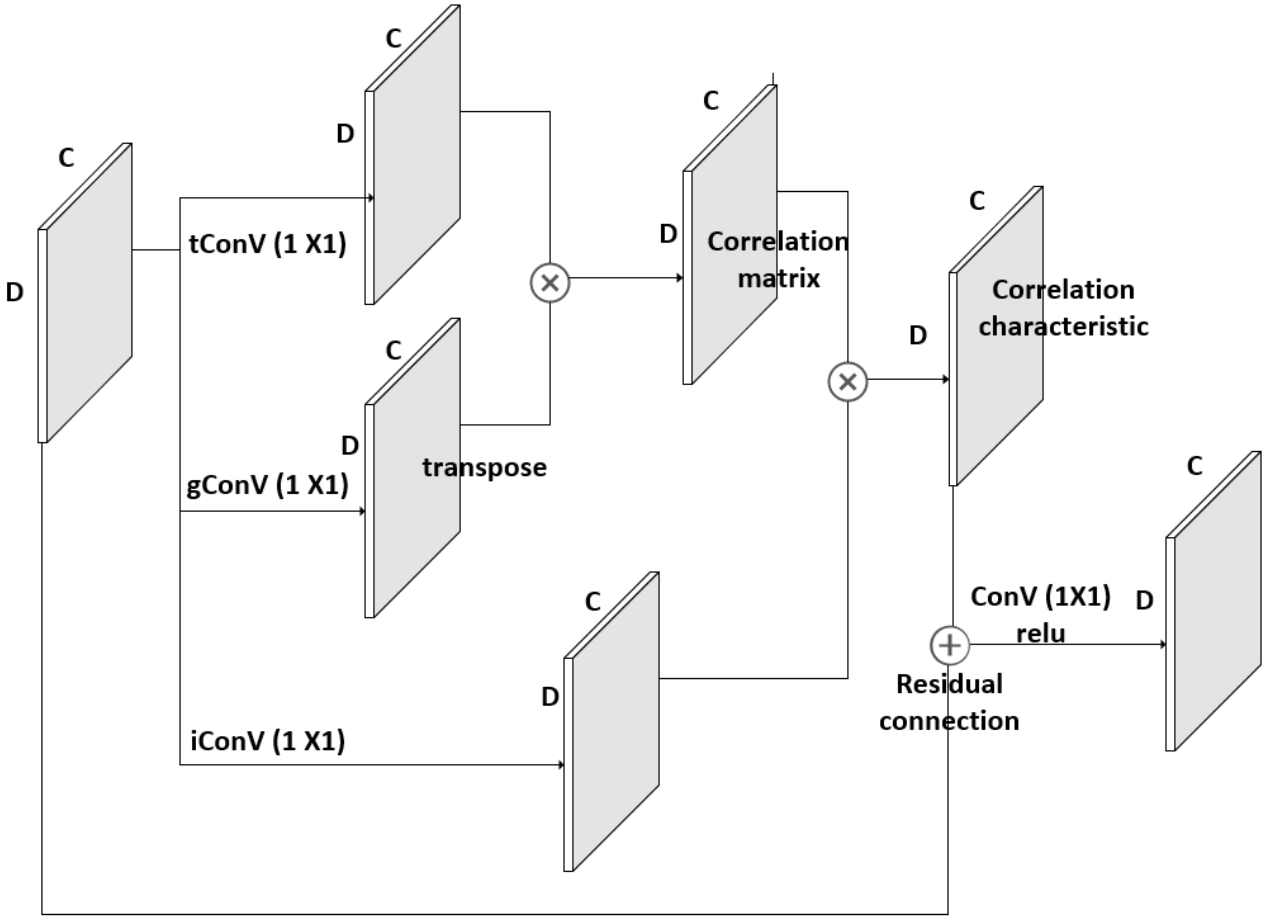

Figure 5-3: Global Feature Correlation Structure

Dimension of the feature map is generated from the CNN which is ($H \times W$), where H represents height and W represents width. This will result in $x \in R^{H \times W}$, hence each x_i has:

$$f(x_i, y_j) = e^{t(x_i, y_j) g(x_i, y_j)^T} \quad (5\text{-}1)$$

- $f(x_i, y_j)$: Represents the similarity or influence function between two vectors x_i and y_j.
- $e^{t(x_i, y_j) g(x_i, y_j)^T}$: Is an expression involving two vectors x_i, y_j and two functions t() and g() used to compute the similarity value.

$$y_i = \frac{1}{\sum_i e^{x_i}} \sum_{Aj} f(x_i, y_j) \,.\, i(x_i) \quad (5\text{-}2)$$

- y_i: Represents the output value for a specific index i.
- $\sum_i e^{x_i}$: This is the sum of exponential terms for all $x_i$values in the dataset.
- $\sum_{Aj} f(x_i, y_j)$: This is the sum of terms involving a function $f(x_i, y_j)$ for all possible indices y_j.

$$Z_i = Relu(W_z(x_i + y_i)) \quad (5\text{-}3)$$

- Z_i: Represents the output value for a specific index i.
- $Relu$: The Rectified Linear Unit activation function.
- W_z: Represents a weight matrix for the linear transformation.
- $x_i and\ y_i$: These are input vectors for index i.

5.2.3. Datasets

The datasets used in this experiment are the KDDCUP99 and the UNSW-NB15 datasets. The following procedures were applied on both datasets separately:

A) Data transformation

The symbolic values of "state", "proto", "service," as well as "attack_cat" were translated to numeric values. For instance, "proto" attribute has three significant values which are "TCP," "UDP," and "ICMP". These values were mapped to 1, 2, and 3, respectively, other values (IGMP, SCTP, RDP) were mapped to 4. The processed features are merged with the unprocessed features.

B) Data normalization

Each attribute has a very different range of possible values. The dataset requires normalization in order to be used. Because the data with high values have a large weight, the data with low values have very little effect on the outcomes. As a consequence, part of the information in the initial data

set could be lost. For this, the values are normalized by the use of the linear transformation represented by the following formula:

$$f(x) = \begin{cases} \frac{x - x_{min}}{x_{max} - x_{min}}, x_{max} \neq x_{min} \\ 0, \; x_{max} \neq x_{min} \end{cases} \quad (5\text{-}4)$$

The transformation has two cases depending on whether x_max is equal to x_min or not:

If x_max is not equal to x_min (x_max ≠ x_min):

In this case, the function returns the normalized value of x within the range [x_min, x_max]. It scales the input x to a value between 0 and 1 based on its position within the specified range. When x is equal to x_min, the output is 0, and when x is equal to x_max, the output is 1.

If x_max is equal to x_min (x_max = x_min):

In this case, the function returns 0 for any input x. This means that the transformation is constant and outputs 0 regardless of the input value.

5.2.4. Experiment Environment

The experiment was conducted on a server with Ubuntu 20.04 LTS. Python 3.10 was used as the programming language, CUDA 10.2 and Pytorch1.10.1 were used for backend processing, and an Nvidia GTX 1080 GPU with 11 Gigabytes of video RAM.

5.2.5. Evaluation Metrics

The metrics used to evaluate the model in this study are:

Accuracy: It measures the percentage of correct predictions made by the model out of the total number of samples in the dataset. Here is the equation for accuracy:

$$\frac{TP+TN}{TP+FP+FN+TN} \tag{5-5}$$

Precision: It evaluates the ability of a model to correctly identify positive samples (true positives) from the total number of samples. Here is the equation for Precision:

$$\frac{TP}{TP+FP} \tag{5-6}$$

Recall: It evaluates the ability of a model to correctly identify positive samples out of all the actual positive samples in the dataset. Here is the equation for Recall:

$$\frac{TP}{TP+FN} \tag{5-7}$$

where

- TP = The Number of True Positives
- TN = The Number of True Negatives
- FP = The Number of False Positives
- FN = The Number of False Negatives

5.3. Findings

The CNNwGFC model, Neural Network, and regular CNN were applied on the two data sets. In the beginning, the experiment was performed on KDDCUP99, which is the most used dataset in field of network intrusion and considered to be the benchmark. The evaluation metrics for the three models were compared. The results of the experiments are presented in Figure (5-4) and Figure (5-5). The CNNwGFC model presented in this study achieved an accuracy of 99.9 percent, which is close to the accuracy achieved by the NN and classic CNN. Figure (5-4) shows a comparison of three different models using the KDDCUP99 dataset.

Then, the same comparison experiment was performed on the UNSW-NB15 dataset, and the experiment outcomes are presented in Figure (5-5) below. The tested CNNwGFC model achieved better results when measured against a variety of assessment measures. CNNwGFC achieved 15.34 % higher accuracy than classic CNN. It achieved 15.39 % higher recall than classic CNN. It achieved 15.59% higher procession than classic CNN. The CNNwGFC accuracy (96.24%) is higher by 7.16 than the highest from the literature [195].

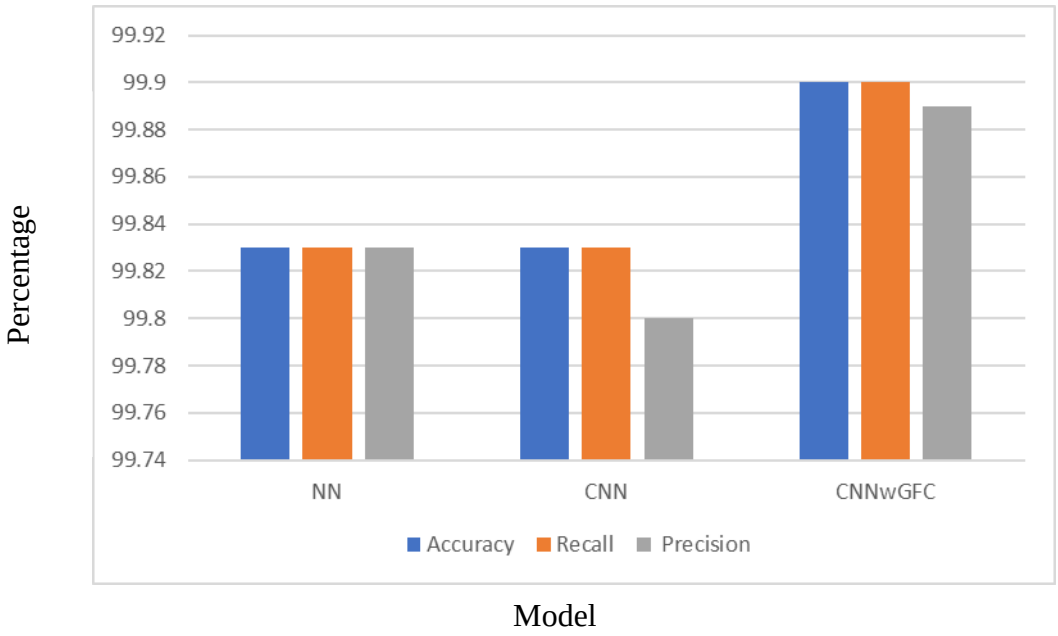

Figure 5-4: Evaluation Metric Results on the KDDCUP99 dataset.

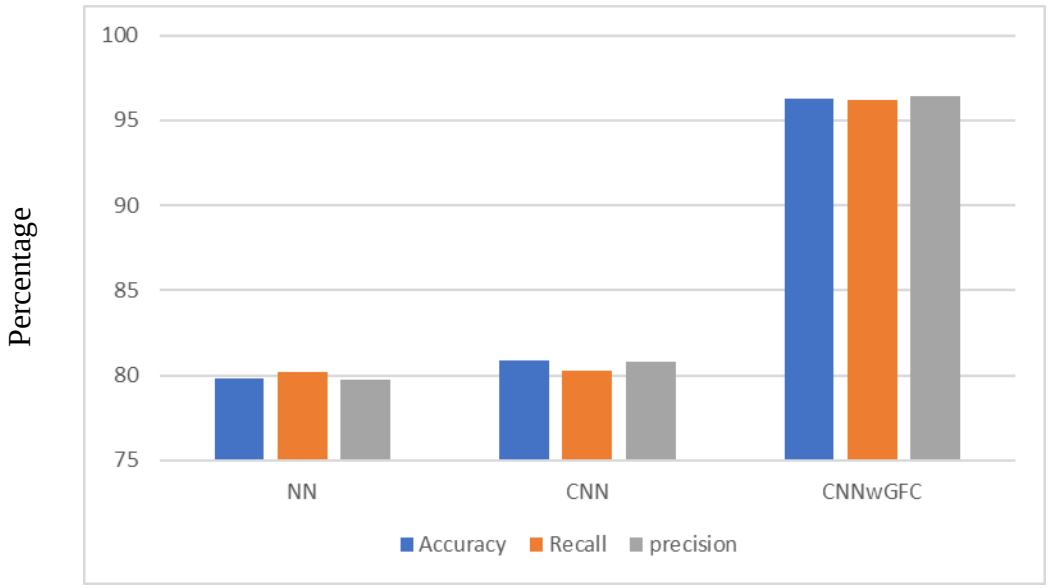

Figure 5-5: Evaluation Metric Results on the UNSW-NB15 dataset.

5.4. Summary

This chapter tested the CNNwGFC model, which is an enhanced CNN model to detect and classify anomalies in network traffic data. The CNNwGFC model was tested on two datasets: KDDCUP99 and UNSW-NB15 datasets. The performance results were compared with Neural Network Model and the classic Convolutional Neural Network model. The results show that the CNNwGFC model achieved on the KDDCUP99 dataset an accuracy of 99.9%, which is close to the accuracy achieved by the NN and classic CNN. However, the CNNwGFC model achieved better results when measured against various performance metrics on the UNSW-NB15 dataset. CNNwGFC achieved 15.34% higher accuracy than classic CNN. It achieved 15.39% higher recall than classic CNN. It achieved 15.59% higher procession than classic CNN. The CNNwGFC accuracy (96.24%) is higher by 7.16 than the highest from the literature [195]. This shows that the global feature correlation structure can be added to the CNN model in order to improve the convolutional network's ability to achieve better network traffic anomaly classification results.

6. Chapter Six: Conclusions and Future Work

6.1. Conclusions

In conclusion, the Internet of Things (IoT) enables us to link practically everything, including people, devices, and physical items, to the internet, which significantly impacts our society. In the following years, networked devices will be considerably increased, which means that the Internet of Things will offer significant problems to the information security industry. This thesis identified the challenges and gaps in securing the Internet of Things networks. The challenges are network size, the number of devices, the human factor, and the complexity of IoT networks. The gaps identified include the lack of research on the following:

- Signature-based intrusion detection systems use for anomaly detection.
- Modelling input parameters required for anomaly detection in IoT networks.
- Comparison of the performance of machine learning algorithms on standard and real IoT datasets.
- High performance machine learning model to classify anomalies in the IoT networks.

This thesis explored applying Gaussian Naive Bayes, Support Vector Machine, Decision Trees and Neural Networks machine learning models in anomaly binary classification in the KDDCUP99 and IoT datasets. This thesis showed that when applying the machine learning algorithms on a real IoT dataset, the performance results differ from the performance on the KDDCUP99. The results show that NN and DT perform similarly on KDDCUP and IoT datasets. On the other hand, SVM and NB have a difference of 8% in accuracy. It also shows that SVM and NB have more than a 7% difference in precision.

When using all the features of the KDDCUP99 or IoT datasets, the findings show potential for using Decision Trees, Support Vector Machines in anomaly detection. However, in reality, only some of the features in the KDDCUP99 dataset are available when capturing network traffic. In addition, using all the features increases the resources and execution time required to generate the results.

This thesis compared the binary classification performance of the Neural Network, Support Vector Machine, Decision Tree, and Naïve Bayes machine learning models in anomaly detection when using all KDDCUP99 dataset features and five selected features as input. The selected input features are protocol type, service type, total length, count of connections from the same IP and number of connections from the same destination port. The results show that the applied machine learning modes can distinguish between normal and abnormal attacks using all or five features only. The Neural Network, Support Vector Machine, and Decision Tree maintained the same performance. The Naïve Bayes achieved better performance when using only five features. The benefit of using only five features is the reduced time required to train and process data by the machine learning model. The results show a reduction in time required by 58% on average. Hence, it is concluded that using only five features; the tested machine learning models can distinguish between normal and abnormal traffic on the KDDCUP99 dataset in a reduced time of 58% on average.

This thesis tested the CNNwGFC model, which is an enhanced CNN model, in detecting and classifying anomalies in network traffic data. The CNNwGFC model was tested on two datasets: KDDCUP99 and UNSW-NB15 datasets. The performance results were compared with Neural Network Model and the classic Convolutional Neural Network model. The results show that the CNNwGFC model achieved on the KDDCUP99 dataset an accuracy of 99.9%, which is close to the accuracy achieved by the NN and classic CNN. The CNNwGFC model achieved better results when measured against various performance metrics on the UNSW-NB15. CNNwGFC achieved 15.34% higher accuracy than classic CNN. It achieved 15.93% higher recall than classic CNN. It achieved 15.59% higher procession than classic CNN. The CNNwGFC accuracy (96.24%) is higher by 7.16 than the highest from the literature. This shows that the global feature correlation

structure can be added to the CNN model to improve the convolutional network's ability to achieve better network traffic anomaly classification results.

6.2. Future Work

Throughout the work from each chapter, further work can be done to achieve more information. The future investigations recommended are as follows:

- The testing was done on the popular machine learning algorithms; however, further testing needs to be done on other classification machine learning models such as Linear Classifiers, Logistic Regression, Perceptron, Quadratic Classifiers, K-Means Clustering, Boosting, Random Forest (RF) machine learning algorithms.

- Add more datasets to the testing, such as NSL-KDD and UKM-IDS20 and other real network IoT datasets. These datasets will be added to confirm further the performance of the machine learning models, dimensionality reduction results, and the performance of the CNNwGFC model.

- Test and compare the performance of the CNNwGFC before and after the dataset input size reduction. This will help determine whether dimensionality reduction benefits the CNNwGFC machine learning model.

References

[1] S. Thudumu, pp. Branch, J. Jin, and J. J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data", *Journal of Big Data*, vol. 7, no. 1, pp. 1–30, 2020.

[2] R. Hu, C. C. Aggarwal, S. Ma, and J. Huai, "An embedding approach to anomaly detection", in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 385–396.

[3] A. K. Ray and A. Bagwari, "IoT based Smart home: Security Aspects and security architecture", in *2020 IEEE 9th international conference on communication systems and network technologies (CSNT)*, 2020, pp. 218–222.

[4] H. Salehi and R. Burgueño, "Emerging artificial intelligence methods in structural engineering", *Engineering structures*, vol. 171, pp. 170–189, 2018.

[5] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead", *Computer networks*, vol. 76, pp. 146–164, 2015.

[6] D. Goodin and A. Technica, "9 baby monitors wide open to hacks that expose users' most private moments", Ars Technica. https://arstechnica.com/information-technology/2015/09/9-baby-monitors-wide-open-to-hacks-that-expose-users-most-private-moments/ (accessed Dec. 28, 2022).

[7] J. Hirsch, "Hackers can now hitch a ride on car computers", Los Angeles Times. http://www.latimes.com/business/autos/la-fi-hy-car-hacking-20150914-story.html (accessed Dec. 28, 2022).

[8] K. Atherton, "Hackers can tap into hospital drug pumps to serve lethal doses to patients", Popular Science. http://tinyurl.com/qfscthv (accessed Dec. 28, 2022).

[9] D. Pauli, “Hacked terminals capable of causing pacemaker deaths”, iTnews. http://tinyurl.com/ycl4z9xf (accessed Dec. 28, 2022).

[10] J. T. Senders, M. M. Zaki, A. V. Karhade, B. Chang, W. B. Gormley, M. L. Broekman, T. R. Smith and O. Arnaout, “An introduction and overview of machine learning in neurosurgical care”, *Acta neurochirurgica*, vol. 160, no. 1, pp. 29–38, 2018.

[11] S. Kotsiantis, I. Zaharakis and P. Pintelas, “*Machine Learning: A Review of Classification and Combining Techniques,” Artificial Intelligence Review*, vol. 26, no. 3, pp. 159–190.

[12] B. Mahesh, “Machine learning algorithms-a review”, *International Journal of Science and Research (IJSR)*, vol. 9, pp. 381–386, 2020.

[13] D. P. Kumar, T. Amgoth, and C. S. R. Annavarapu, “Machine learning algorithms for wireless sensor networks: A survey”, *Information Fusion*, vol. 49, pp. 1–25, 2019.

[14] S. Ray, “A quick review of machine learning algorithms”, in *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*, 2019, pp. 35–39.

[15] F. Y. Osisanwo, J. E. T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi, and J. Akinjobi, “Supervised machine learning algorithms: classification and comparison”, *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

[16] E.-A. Minastireanu and G. Mesnita, “An Analysis of the Most Used Machine Learning Algorithms for Online Fraud Detection”, *Informatica Economica*, vol. 23, no. 1, 2019.

[17] S. Eltanbouly, M. Bashendy, N. AlNaimi, Z. Chkirbene, and A. Erbad, “Machine learning techniques for network anomaly detection: A survey”, in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 2020, pp. 156–162.

[18] I. Arel, D. C. Rose, and T. P. Karnowski, “Research frontier: deep machine learning--a new frontier in artificial intelligence research”, *IEEE computational intelligence magazine*, vol. 5, no. 4, pp. 13–18, 2010.

[19] W. Yamany, M. Fawzy, A. Tharwat, and A. E. Hassanien, “Moth-flame optimization for training multi-layer perceptrons”, in *2015 11th International computer engineering Conference (ICENCO)*, 2015, pp. 267–272.

[20] K. Janocha and W. M. Czarnecki, “On loss functions for deep neural networks in classification”, *arXiv*, 2017.

[21] A. Segatori, F. Marcelloni, and W. Pedrycz, “On Distributed Fuzzy Decision Trees for Big Data”, *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 1, pp. 174–192, 2018.

[22] W.-Y. Loh, “Classification and regression trees”, *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[23] A. E. Mohamed, “Comparative study of four supervised machine learning techniques for classification”, *International Journal of Applied*, vol. 7, no. 2, pp. 1–15, 2017.

[24] T. Rumpf, A.-K. Mahlein, U. Steiner, E.-C. Oerke, H.-W. Dehne, and L. Plümer, “Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance”, *Computers and electronics in agriculture*, vol. 74, no. 1, pp. 91–99, 2010.

[25] I. Rish and Others, “An empirical study of the naive Bayes classifier”, in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 2001, vol. 3, pp. 41–46.

[26] J. M. McNamara, R. F. Green, and O. Olsson, “Bayes’ theorem and its applications in animal behaviour”, *Oikos*, vol. 112, no. 2, pp. 243–251, 2006.

[27] R. Zhang, F. Zhu, J. Liu, and G. Liu, “Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-based steganalysis”, *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1138–1150, 2019.

[28] L. Chen, S. Wang, W. Fan, J. Sun, and S. Naoi, “Beyond human recognition: A CNN-based framework for handwritten character recognition”, in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, pp. 695–699.

[29] J. Patterson and A. Gibson, *Deep learning: A practitioner’s approach*. Sebastopol, CA: O'Reilly, 2017.

[30] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network”, in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[31] M. Salomon, R. Couturier, C. Guyeux, J.-F. Couchot, and J. M. Bahi, “Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key: A deep learning approach for telemedicine”, *European Research in Telemedicine/La Recherche Européenne en Télémédecine*, vol. 6, no. 2, pp. 79–92, 2017.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, in *Advances in Neural Information Processing Systems*, 2012, vol. 25.

[34] D. Yu, H. Wang, pp. Chen, and Z. Wei, “Mixed pooling for convolutional neural networks”, in *International conference on rough sets and knowledge technology*, 2014, pp. 364–375.

[35] U. Ravale, N. Marathe, and P. Padiya, “Feature selection based hybrid anomaly intrusion detection system using K means and RBF kernel function”, *Procedia Computer Science*, vol. 45, pp. 428–435, 2015.

[36] B. C. Rhodes, J. A. Mahaffey, and J. D. Cannady, “Multiple self-organizing maps for intrusion detection”, in *Proceedings of the 23rd national information systems security conference*, 2000, pp. 16–19.

[37] Y. Sani, A. Mohamedou, K. Ali, A. Farjamfar, M. Azman, and S. Shamsuddin, “An overview of neural networks use in anomaly intrusion detection systems”, in *2009 IEEE Student Conference on Research and Development (SCOReD)*, 2009, pp. 89–92.

[38] H. Mazzawi, G. Dalal, D. Rozenblatz, L. Ein-Dorx, M. Niniox, and O. Lavi, “Anomaly detection in large databases using behavioral patterning”, in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017, pp. 1140–1149.

[39] T. Peng, C. Leckie, and K. Ramamohanarao, “Survey of network-based defense mechanisms countering the DoS and DDoS problems”, *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, pp. 3-es, 2007.

[40] L.-C. Chen, T. A. Longstaff, and K. M. Carley, “Characterization of defense mechanisms against distributed denial of service attacks”, *Computers & Security*, vol. 23, no. 8, pp. 665–678, 2004.

[41] J. Haines, D. K. Ryder, L. Tinnel, and S. Taylor, “Validation of sensor alert correlators”, *IEEE Security & Privacy*, vol. 1, no. 1, pp. 46–56, 2003.

[42] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, “Attacks against Process Control Systems: Risk Assessment, Detection, and Response”, in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, Hong Kong, China, 2011, pp. 355–366.

[43] Z. A. Baig, S. Khan, S. Ahmed, and M. H. Sqalli, “A selective parameter-based evolutionary technique for network intrusion detection”, in *2011 11th International Conference on Intelligent Systems Design and Applications*, 2011, pp. 65–71.

[44] S. X. Wu and W. Banzhaf, “The use of computational intelligence in intrusion detection systems: A review”, *Applied soft computing*, vol. 10, no. 1, pp. 1–35, 2010.

[45] S. Roschke, F. Cheng, and C. Meinel, “An advanced IDS management architecture”, *Journal of Information Assurance and Security*, vol. 5, pp. 246–255, 2010.

[46] M. Baqer and A. I. Khan, “Energy-efficient pattern recognition approach for wireless sensor networks”, in *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, 2007, pp. 509–514.

[47] C. R. Haag, G. B. Lamont, pp. D. Williams, and G. L. Peterson, “An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions”, in *International Conference on Artificial Immune Systems*, 2007, pp. 420–435.

[48] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: A comprehensive review”, *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[49] J. Allen, A. Christie, W. Fithen, J. McHugh, and J. Pickel, “State of the practice of intrusion detection technologies”, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2000.

[50] E. H. Spafford and D. Zamboni, “Intrusion detection using autonomous agents”, *Computer networks*, vol. 34, no. 4, pp. 547–570, 2000.

[51] A. Householder, A. Manion, L. Pesante, G. M. Weaver, and R. Thomas, “Managing the threat of denial-of-service attacks”, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep, 2001.

[52] N. Hubballi and V. Suryanarayanan, “False alarm minimization techniques in signature-based intrusion detection systems: A survey”, *Computer Communications*, vol. 49, pp. 1–17, 2014.

[53] S. Mukkamala, G. Janoski, and A. Sung, “Intrusion detection using neural networks and support vector machines”, in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN’02 (Cat. No. 02CH37290)*, 2002, vol. 2, pp. 1702–1707.

[54] D. Novikov, R. V. Yampolskiy, and L. Reznik, “Anomaly detection based intrusion detection”, in *Third international conference on information technology: new generations (ITNG’06)*, 2006, pp. 420–425.

[55] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, M. Embrechts, and Others, “Network-based intrusion detection using neural networks”, *Intelligent Engineering Systems through Artificial Neural Networks*, vol. 12, no. 1, pp. 579–584, 2002.

[56] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, “Toward credible evaluation of anomaly-based intrusion-detection methods”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 516–524, 2010.

[57] S. Choudhary and N. Kesswani, “Analysis of KDD-Cup’99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT”, *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2020.

[58] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, “Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives”, in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, pp. 1–8.

[59] D. Jing and H.-B. Chen, “SVM based network intrusion detection for the UNSW-NB15 dataset”, in *2019 IEEE 13th international conference on ASIC (ASICON)*, 2019, pp. 1–4.

[60] A. Husain, A. Salem, C. Jim, and G. Dimitoglou, “Development of an efficient network intrusion detection model using extreme gradient boosting (xgboost) on the unsw-nb15 dataset”, in *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2019, pp. 1–7.

[61] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”, in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[62] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, “Cost-based modeling for fraud and intrusion detection: Results from the JAM project”, in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX’00*, 2000, vol. 2, pp. 130–144.

[63] K. D. D. Cup, “http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html”, *The UCI KDD Archive*, 1999.

[64] N. N. M. Yusof and N. S. Sulaiman, “Cyber attack detection dataset: A review”, in *Journal of Physics: Conference Series*, 2022, vol. 2319, p. 012029.

[65] B. S. Bhati, C. S. Rai, B. Balamurugan, and F. Al-Turjman, “An intrusion detection scheme based on the ensemble of discriminant classifiers”, *Computers & Electrical Engineering*, vol. 86, p. 106742, 2020.

[66] R. D. Ravipati and M. Abualkibash, “Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets-a review paper”, *International Journal of Computer Science & Information Technology (IJCSIT) Vol*, vol. 11, 2019.

[67] R. Magán-Carrión, D. Urda, I. Díaz-Cano, and B. Dorronsoro, “Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches”, *Applied Sciences*, vol. 10, no. 5, p. 1775, 2020.

[68] D. A. Cieslak and N. V. Chawla, “A framework for monitoring classifiers’ performance: when and why failure occurs?”, *Knowledge and Information Systems*, vol. 18, no. 1, pp. 83–108, 2009.

[69] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”, in *2015 military communications and information systems conference (MilCIS)*, 2015, pp. 1–6.

[70] N. Moustafa and J. Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set”, *Information Security Journal: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016.

[71] M. Lion and Y. Shahar, “Implementation and evaluation of a multivariate abstraction-based, interval-based dynamic time-warping method as a similarity measure for longitudinal medical records”, *Journal of Biomedical Informatics*, vol. 123, p. 103919, 2021.

[72] S. Wu, H. Lin, Y. Gao, D. Lu, and Others, “Finding frequent items in time decayed data streams”, in *Asia-Pacific Web Conference*, 2016, pp. 17–29.

[73] Y. Yang, Y. Huang, J. Cao, X. Ma, and J. Lu, “Design of a sliding window over distributed and asynchronous event streams”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2551–2560, 2013.

[74] M. Aljubran, J. Ramasamy, M. Albassam, and A. Magana-Mora, “Deep learning and time-series analysis for the early detection of lost circulation incidents during drilling operations”, *IEEE Access*, vol. 9, pp. 76833–76846, 2021.

[75] A. Saas, A. Guitart, and A. Periánez, “Discovering playing patterns: Time series clustering of free-to-play game data”, in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016, pp. 1–8.

[76] E. Kabir, J. Hu, H. Wang, and G. Zhuo, “A novel statistical technique for intrusion detection systems”, *Future Generation Computer Systems*, vol. 79, pp. 303–318, 2018.

[77] X. Ma and Y. Chen, “DDoS detection method based on chaos analysis of network traffic entropy”, *IEEE Communications Letters*, vol. 18, no. 1, pp. 114–117, 2013.

[78] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, “Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity”, in *Proceedings of the 2019 ACM Symposium on SDN Research*, 2019, pp. 36–48.

[79] M. A. Jamshed, A. Nauman, M. A. B. Abbasi, and S. W. Kim, “Antenna Selection and Designing for THz Applications: Suitability and Performance Evaluation: A Survey”, *IEEE Access*, vol. 8, pp. 113246–113261, 2020.

[80] F. Zhou and Y. Chai, “Near-sensor and in-sensor computing”, *Nature Electronics*, vol. 3, no. 11, pp. 664–671, 2020.

[81] J. C. Talwana and H. J. Hua, “Smart World of Internet of Things (IoT) and Its Security Concerns”, in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016, pp. 240–245.

[82] T. Alam, “A Reliable Communication Framework and Its Use in Internet of Things (IoT)”, vol. 3, 05 2018.

[83] L.-D. Radu, “Disruptive technologies in smart cities: a survey on current trends and challenges”, *Smart Cities*, vol. 3, no. 3, pp. 1022–1038, 2020.

[84] J. Čapek, “Cybersecurity and internet of things”, *IDIMT-2018 Strategic Modeling in Management, Economy and Society*, 2018.

[85] M. Shirvanimoghaddam *et al.*, “Towards a Green and Self-Powered Internet of Things Using Piezoelectric Energy Harvesting”, *IEEE Access*, vol. 7, pp. 94533–94556, 2019.

[86] Y. Wang, H. Wang, J. Xuan, and D. Y. C. Leung, “Powering future body sensor network systems: A review of power sources”, *Biosensors and Bioelectronics*, vol. 166, p. 112410, 2020.

[87] K. J. Singh and D. S. Kapoor, “Create your own Internet of things: A survey of IoT platforms”, *IEEE Consumer Electronics Magazine*, vol. 6, no. 2, pp. 57–68, 2017.

[88] H. Teymourian *et al.*, “Wearable electrochemical sensors for the monitoring and screening of drugs”, *ACS sensors*, vol. 5, no. 9, pp. 2679–2700, 2020.

[89] G. B. Rehm *et al.*, “Leveraging IoTs and machine learning for patient diagnosis and ventilation management in the intensive care unit”, *IEEE Pervasive Computing*, vol. 19, no. 3, pp. 68–78, 2020.

[90] F. Restuccia, S. K. Das, and J. Payton, “Incentive mechanisms for participatory sensing: Survey and research challenges”, *ACM Transactions on Sensor Networks (TOSN)*, vol. 12, no. 2, pp. 1–40, 2016.

[91] A. P. Ingle and S. D. Ghode, “Internet of Things (IoT): Vision, Review, Drivers of IoT, Sensors Nodes, Communication Technologies and Architecture”, 2017.

[92] B. B. Gupta and M. Quamara, “An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols”, *Concurrency and Computation: Practice and Experience*, vol. 32, no. 21, p. e4946, 2020.

[93] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, “Middleware for internet of things: a survey”, *IEEE Internet of things journal*, vol. 3, no. 1, pp. 70–95, 2015.

[94] R. Stephen and L. Arockiam, “Intrusion detection system to detect sinkhole attack on RPL protocol in Internet of Things”, *International Journal of Electrical Electronics and Computer Science*, vol. 4, no. 4, pp. 16–20, 2017.

[95] S. Raza, L. Wallgren, and T. Voigt, “SVELTE: Real-time intrusion detection in the Internet of Things”, *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[96] L. Santos, C. Rabadao, and R. Gonçalves, “Intrusion detection systems in Internet of Things: A literature review”, in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, 2018, pp. 1–7.

[97] D. Shreenivas, S. Raza, and T. Voigt, “Intrusion detection in the RPL-connected 6LoWPAN networks”, in *Proceedings of the 3rd ACM international workshop on IoT privacy, trust, and security*, 2017, pp. 31–38.

[98] P. Pongle and G. Chavan, “Real time intrusion and wormhole attack detection in internet of things”, *International Journal of Computer Applications*, vol. 121, no. 9, 2015.

[99] C. Jun and C. Chi, “Design of complex event-processing IDS in internet of things”, in *2014 sixth international conference on measuring technology and mechatronics automation*, 2014, pp. 226–229.

[100] D. H. Summerville, K. M. Zach, and Y. Chen, “Ultra-lightweight deep packet anomaly detection for Internet of Things devices”, in *2015 IEEE 34th international performance computing and communications conference (IPCCC)*, 2015, pp. 1–8.

[101] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, “Kalis—A system for knowledge-driven adaptable intrusion detection for the Internet of Things”, in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 656–666.

[102] N. K. Thanigaivelan, E. Nigussie, R. K. Kanth, S. Virtanen, and J. Isoaho, "Distributed internal anomaly detection system for Internet-of-Things", in *2016 13th IEEE annual consumer communications & networking conference (CCNC)*, 2016, pp. 319–320.

[103] D. Oh, D. Kim, and W. W. Ro, "A malicious pattern detection engine for embedded security systems in the Internet of Things", *Sensors*, vol. 14, no. 12, pp. 24188–24211, 2014.

[104] P. Ioulianou, V. Vasilakis, I. Moscholios, and M. Logothetis, "A signature-based intrusion detection system for the internet of things", *Information and Communication Technology Form*, 2018.

[105] P. Pudil and J. Novovičová, "Novel methods for feature subset selection with respect to problem knowledge", in *Feature extraction, construction and selection*, Springer, 1998, pp. 101–116.

[106] L. Zhang, F. Restuccia, T. Melodia, and S. M. Pudlewski, "Learning to detect and mitigate cross-layer attacks in wireless networks: framework and applications", in *2017 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9.

[107] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level", *Applied Sciences*, vol. 9, no. 16, p. 3414, 2019.

[108] D. Arivudainambi, V. K. Ka, and S. Sibi Chakkaravarthy, "LION IDS: A meta-heuristics approach to detect DDoS attacks against Software-Defined Networks", *Neural Computing and Applications*, vol. 31, no. 5, pp. 1491–1501, 2019.

[109] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system", IEEE *Access*, vol. 7, pp. 41525–41550, 2019.

[110] O. Faker and E. Dogdu, “Intrusion detection using big data and deep learning techniques”, in *Proceedings of the 2019 ACM Southeast Conference*, 2019, pp. 86–93.

[111] W. Anani and J. Samarabandu, “Comparison of recurrent neural network algorithms for intrusion detection based on predicting packet sequences”, in *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, 2018, pp. 1–4.

[112] 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.

[113] Z. Li, A. L. G. Rios, G. Xu, and L. Trajković, “Machine learning techniques for classifying network anomalies and intrusions”, in *2019 IEEE international symposium on circuits and systems (ISCAS)*, 2019, pp. 1–5.

[114] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, “A supervised intrusion detection system for smart home IoT devices. IEEE Internet Things J. 6, 9042--9053 (2019)’. 2019.

[115] J. Li, Z. Zhao, R. Li, and H. Zhang, “Ai-based two-stage intrusion detection for software defined iot networks”, *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2093–2102, 2018.

[116] W. Alhakami, A. ALharbi, S. Bourouis, R. Alroobaea, and N. Bouguila, “Network anomaly intrusion detection using a nonparametric Bayesian approach and feature selection”, *IEEE Access*, vol. 7, pp. 52181–52190, 2019.

[117] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, “Toward a lightweight intrusion detection system for the internet of things”, *IEEE Access*, vol. 7, pp. 42450–42471, 2019.

[118] A. Kim, M. Park, and D. H. Lee, “AI-IDS: Application of deep learning to real-time Web intrusion detection”, *IEEE Access*, vol. 8, pp. 70245–70261, 2020.

[119] M. Roopak, G. Y. Tian, and J. Chambers, “An intrusion detection system against ddos attacks in iot networks”, in *2020 10th annual computing and communication workshop and conference (CCWC)*, 2020, pp. 0562–0567.

[120] F. Jiang *et al.*, “Deep learning based multi-channel intelligent attack detection for data security”, *IEEE transactions on Sustainable Computing*, vol. 5, no. 2, pp. 204–212, 2018.

[121] B. Susilo and R. F. Sari, “Intrusion detection in IoT networks using deep learning algorithm”, *Information*, vol. 11, no. 5, p. 279, 2020.

[122] J. Cui, J. Long, E. Min, Q. Liu, and Q. Li, “Comparative Study of CNN and RNN for Deep Learning Based Intrusion Detection System”, in *Cloud Computing and Security*, 2018, pp. 159–170.

[123] I. H. Sarker, “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions”, *SN Computer Science*, vol. 2, no. 6, pp. 1–20, 2021.

[124] O. Voican, “Credit Card Fraud Detection using Deep Learning Techniques”, *Informatica Economica*, vol. 25, no. 1, 2021.

[125] L. Liu *et al.*, “Deep learning for generic object detection: A survey”, *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[126] C. Janiesch, pp. Zschech, and K. Heinrich, “Machine learning and deep learning”, *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[127] M. Tayefi *et al.*, “Challenges and opportunities beyond structured data in analysis of electronic health records”, *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 13, no. 6, p. e1549, 2021.

[128] S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence”, *Information*, vol. 11, no. 4, pp. 193, 2020.

[129] S. Naseer, R. Faizan Ali, pp. D. D. Dominic, and Y. Saleem, “Learning representations of network traffic using deep neural networks for network anomaly detection: A perspective towards oil and gas IT infrastructures”, *Symmetry*, vol. 12, no. 11, p. 1882, 2020.

[130] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, “A survey on application of machine learning for Internet of Things”, *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399–1417, 2018.

[131] V. U. J, S. Roy, and P. B. Honnavalli, ‘Correlative Analysis of Combined Machine Learning Classifiers on Anomaly-based Intrusion Detection Systems’, in *2021 IEEE 2nd International Conference on Technology, Engineering, Management for Societal impact using Marketing, Entrepreneurship and Talent (TEMSMET)*, 2021, pp. 1–6.

[132] T. H. Hai and L. H. Nam, “A Practical Comparison of Deep Learning Methods for Network Intrusion Detection”, in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2021, pp. 1–6.

[133] T. S. Pooja and P. Shrinivasacharya, “Evaluating neural networks using Bi-Directional LSTM for network IDS (intrusion detection systems) in cyber security”, *Global Transitions Proceedings*, vol. 2, no. 2, pp. 448–454, 2021.

[134] R. Biswas and S. Roy, “Botnet traffic identification using neural networks”, *Multimedia Tools and Applications*, vol. 80, no. 16, pp. 24147–24171, 2021.

[135] F. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, “Intrusion detection systems using long short-term memory (LSTM)”, *Journal of Big Data*, vol. 8, no. 1, pp. 1–16, 2021.

[136] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, “A bidirectional LSTM deep learning approach for intrusion detection”, *Expert Systems with Applications*, vol. 185, p. 115524, 2021.

[137] M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, “A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique”, *Journal of Network and Computer Applications*, vol. 191, p. 103160, 2021.

[138] C. Joshi, R. K. Ranjan, and V. Bharti, “A Fuzzy Logic based feature engineering approach for Botnet detection using ANN”, *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 9, pp. 6872–6882, 2022.

[139] H. Alyasiri, J. A. Clark, A. Malik, and R. de Fréin, “Grammatical Evolution for Detecting Cyberattacks in Internet of Things Environments”, in *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–6.

[140] F. Hussain *et al.*, “A framework for malicious traffic detection in IoT healthcare environment”, *Sensors*, vol. 21, no. 9, p. 3025, 2021.

[141] I. Vaccari, S. Narteni, M. Aiello, M. Mongelli, and E. Cambiaso, “Exploiting Internet of Things protocols for malicious data exfiltration activities”, *IEEE Access*, vol. 9, pp. 104261–104280, 2021.

[142] I. Ullah and Q. H. Mahmoud, “Design and development of RNN anomaly detection model for IoT networks”, *IEEE Access*, vol. 10, pp. 62722–62750, 2022.

[143] M. A. Almaiah *et al.*, “Performance Investigation of Principal Component Analysis for Intrusion Detection System Using Different Support Vector Machine Kernels”, *Electronics*, vol. 11, no. 21, p. 3571, 2022.

[144] M. Belouch, S. El Hadaj, and M. Idhammad, “Performance evaluation of intrusion detection based on machine learning using Apache Spark”, *Procedia Computer Science*, vol. 127, pp. 1–6, 2018.

[145] H.-H. Chen and Y.-J. Lee, “Distributed consensus reduced support vector machine”, in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 5718–5727.

[146] C. Ioannou and V. Vassiliou, “Network Attack Classification in IoT Using Support Vector Machines”, *Journal of Sensor and Actuator Networks*, vol. 10, no. 3, p. 58, 2021.

[147] S. Sapre, pp. Ahmadi, and K. Islam, “A robust comparison of the KDDCup99 and NSL-KDD IoT network intrusion detection datasets through various machine learning algorithms”, *arXiv*, 2019.

[148] W. Kawelah and A. Abdala, “A Comparative Study on Machine Learning Tools Using WEKA and Rapid Miner with Classifier Algorithms C4. 5 and Decision Stump for Network Intrusion Detection”, *European Academic Research*, vol. 7, pp. 852–861, 2019.

[149] D. A. Reddy, V. Puneet, S. S. R. Krishna, and S. Kranthi, “Network Attack Detection And Classification using ANN Algorithm”, in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, 2022, pp. 66–71.

[150] M. Maithem and G. A. Al-sultany, “Network intrusion detection system using deep neural networks”, in *Journal of Physics: Conference Series*, 2021, vol. 1804, p. 012138.

[151] S. Wasi, S. Shams, S. Nasim, and A. Shafiq, “Intrusion Detection Using Deep Learning and Statistical Data Analysis”, in *2019 4th International Conference on Emerging Trends in Engineering, Sciences and Technology (ICEEST)*, 2019, pp. 1–5.

[152] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features”, in *European conference on machine learning*, 1998, pp. 137–142.

[153] D.-Y. Chiu and P.-J. Chen, “Dynamically exploring internal mechanism of stock market by fuzzy-based support vector machines with high dimension input space and genetic algorithm”, *Expert Systems with Applications*, vol. 36, no. 2, pp. 1240–1248, 2009.

[154] Z. Zhang, “Naïve Bayes classification in R”, *Annals of translational medicine*, vol. 4, no. 12, 2016.

[155] N. Kriegeskorte and T. Golan, “Neural network models and deep learning”, Current Biology, vol. 29, no. 7, pp. R231–R236, 2019.

[156] A. D. Dongare, R. R. Kharde, A. D. Kachare, and Others, “Introduction to artificial neural network”, International Journal of Engineering and Innovative Technology (IJEIT), vol. 2, no. 1, pp. 189–194, 2012.

[157] N. Gupta and Others, “Artificial neural network”, Network and Complex Systems, vol. 3, no. 1, pp. 24–28, 2013.

[158] R. C. Williamson, A. J. Smola, and B. Scholkopf, “Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators”, IEEE Transactions on Information Theory, vol. 47, no. 6, pp. 2516–2532, 2001.

[159] V. N. Vapnik, “An overview of statistical learning theory”, IEEE transactions on neural networks, vol. 10, no. 5, pp. 988–999, 1999.

[160] J. H. Cho and P. U. Kurup, “Decision tree approach for classification and dimensionality reduction of electronic nose data”, Sensors and Actuators B: Chemical, vol. 160, no. 1, pp. 542–548, 2011.

[161] G. T. Reddy et al., “Analysis of Dimensionality Reduction Techniques on Big Data”, IEEE Access, vol. 8, pp. 54776–54788, 2020.

[162] S. Velliangiri, S. Alagumuthukrishnan, and Others, “A review of dimensionality reduction techniques for efficient computation”, Procedia Computer Science, vol. 165, pp. 104–111, 2019.

[163] L. S. Moulin, A. P. A. da Silva, M. A. El-Sharkawi, and R. J. Marks, “Support vector and multilayer perceptron neural networks applied to power systems transient stability analysis with input dimensionality reduction”, in IEEE Power Engineering Society Summer Meeting, 2002, vol. 3, pp. 1308–1313.

[164] N. B. Amor, S. Benferhat, and Z. Elouedi, “Naive Bayes vs Decision Trees in Intrusion Detection Systems”, in Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 2004, pp. 420–424.

[165] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, “A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction”, Journal of Applied Science and Technology Trends, vol. 1, no. 2, pp. 56–70, 2020.

[166] K. M. Majidha Fathima, “A Survey of the Exemplary Practices in Network Operations and Management”, in Data Intelligence and Cognitive Informatics, 2021, pp. 181–194.

[167] D. Mistry, P. Modi, K. Deokule, A. Patel, H. Patki, and O. Abuzaghleh, “Network traffic measurement and analysis”, in 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2016, pp. 1–7.

[168] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, “A survey on network data collection”, Journal of Network and Computer Applications, vol. 116, pp. 9–23, 2018.

[169] L. Han, Z. Guo, X. Huang, and X. Zeng, “A Multifunctional Full-Packet Capture and Network Measurement System Supporting Nanosecond Timestamp and Real-Time Analysis”, IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1–12, 2021.

[170] M. Felix, C. Safitri, and R. Mandala, “Framework for Analyzing Intruder Behavior of IoT Cyber Attacks Based on Network Forensics by Deploying Honeypot Technology”, in 2022 5th International Conference on Information and Communications Technology (ICOIACT), 2022, pp. 423–428.

[171] A. Bhardwaj, V. Mangat, R. Vig, S. Halder, and M. Conti, “Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions”, Computer Science Review, vol. 39, p. 100332, 2021.

[172] J. David and C. Thomas, “Discriminating flash crowds from DDoS attacks using efficient thresholding algorithm”, Journal of Parallel and Distributed Computing, vol. 152, pp. 79–87, 2021.

[173] A. Callado et al., “A Survey on Internet Traffic Identification”, IEEE Communications Surveys & Tutorials, vol. 11, no. 3, pp. 37–52, 2009.

[174] K. Sethi, Y. V. Madhav, R. Kumar, and P. Bera, “Attention based multi-agent intrusion detection systems using reinforcement learning”, Journal of Information Security and Applications, vol. 61, p. 102923, 2021.

[175] H. Jia, J. Liu, M. Zhang, X. He, and W. Sun, “Network intrusion detection based on IE-DBN model”, Computer Communications, vol. 178, pp. 131–140, 2021.

[176] B. B. Borisenko, S. D. Erokhin, A. S. Fadeev, and I. D. Martishin, “Intrusion detection using multilayer perceptron and neural networks with long short-term memory”, in 2021 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO, 2021, pp. 1–6.

[177] L. Liu, P. Wang, J. Lin, and L. Liu, “Intrusion detection of imbalanced network traffic based on machine learning and deep learning”, Ieee Access, vol. 9, pp. 7550–7563, 2020.

[178] S. Aldhaheri, D. Alghazzawi, L. Cheng, B. Alzahrani, and A. Al-Barakati, “Deepdca: novel network-based detection of iot attacks using artificial immune system”, Applied Sciences, vol. 10, no. 6, p. 1909, 2020.

[179] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study”, Journal of Information Security and Applications, vol. 50, p. 102419, 2020.

[180] M. Ge, N. F. Syed, X. Fu, Z. Baig, and A. Robles-Kelly, “Towards a deep learning-driven intrusion detection approach for Internet of Things”, Computer Networks, vol. 186, p. 107784, 2021.

[181] J. Malik, A. Akhunzada, I. Bibi, M. Imran, A. Musaddiq, and S. W. Kim, “Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in SDN”, IEEE Access, vol. 8, pp. 134695–134706, 2020.

[182] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, “Deep learning-based intrusion detection for IoT networks”, in 2019 IEEE 24th pacific rim international symposium on dependable computing (PRDC), 2019, pp. 256–25609.

[183] N. Chouhan, A. Khan, and Others, “Network anomaly detection using channel boosted and residual learning based deep convolutional neural network”, Applied Soft Computing, vol. 83, p. 105612, 2019.

[184] M. A. Ferrag and L. Maglaras, “DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids”, IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1285–1297, 2019.

[185] T. Wu, H. Fan, H. Zhu, C. You, H. Zhou, and X. Huang, “Intrusion detection system combined enhanced random forest with SMOTE algorithm”, EURASIP Journal on Advances in Signal Processing, vol. 2022, no. 1, pp. 1–20, 2022.

[186] F. Li, A. Shinde, Y. Shi, J. Ye, X.-Y. Li, and W. Song, “System statistics learning-based IoT security: Feasibility and suitability”, IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6396–6403, 2019.

[187] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “DÏoT: A federated self-learning anomaly detection system for IoT”, in 2019 IEEE 39th International conference on distributed computing systems (ICDCS), 2019, pp. 756–767.

[188] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. García, and C. Benavides, “Multiclass classification procedure for detecting attacks on MQTT-IoT protocol”, Complexity, vol. 2019, 2019.

[189] S. M. Kasongo and Y. Sun, “Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset”, Journal of Big Data, vol. 7, no. 1, pp. 1–20, 2020.

[190] T. Acharya, I. Khatri, A. Annamalai, and M. F. Chouikha, “Efficacy of Machine Learning-Based Classifiers for Binary and Multi-Class Network Intrusion Detection”, in 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), 2021, pp. 402–407.

[191] A. M. Aleesa, M. Younis, A. A. Mohammed, and N. M. Sahar, “Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques”, Journal of Engineering Science and Technology, vol. 16, no. 1, pp. 711–727, 2021.

[192] S. M. Kasongo and Y. Sun, “A deep learning method with wrapper based feature extraction for wireless intrusion detection system”, Computers & Security, vol. 92, p. 101752, 2020.

[193] R.-H. Dong, Y.-L. Shui, and Q.-Y. Zhang, “Intrusion Detection Model Based on Feature Selection and Random Forest”, International Journal of Network Security, vol. 23, no. 6, pp. 985–996, 2021.

[194] J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, “Network intrusion detection system using supervised learning paradigm”, Scientific African, vol. 9, p. e00497, 2020.

[195] Y. Yang, K. Zheng, C. Wu, and Y. Yang, “Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network”, Sensors, vol. 19, no. 11, p. 2528, 2019.

[196] H.-C. Lin, P. Wang, K.-M. Chao, W.-H. Lin, and Z.-Y. Yang, “Ensemble Learning for Threat Classification in Network Intrusion Detection on a Security Monitoring System for Renewable Energy”, Applied Sciences, vol. 11, no. 23, p. 11283, 2021.

[197] E. García-Gonzalo, Z. Fernández-Muñiz, P. J. Garcia Nieto, A. Bernardo Sánchez, and M. Menéndez Fernández, “Hard-rock stability analysis for span design in entry-type excavations with learning classifiers”, Materials, vol. 9, no. 7, p. 531, 2016.

[198] R. M. Reffat, J. Gero, and W. Peng, “Using data mining on building maintenance during the building life cycle”, in Proceedings of the 38th Australian & New Zealand Architectural Science Association (ANZASCA) Conference, 2004, pp. 91–97.

[199] B. Gupta, A. Rawat, A. Jain, A. Arora, and N. Dhami, “Analysis of various decision tree algorithms for classification in data mining”, International Journal of Computer Applications, vol. 163, no. 8, pp. 15–19, 2017.