REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

NEW AUTOMATIC (IDS) IN IOTS WITH ARTIFICIAL INTELLIGENCE TECHNIQUES

Alaa Firas Jasim JASIM

Doctor of Philosophy

Supervisor

Asst. prof. Dr. Sefer KURNAZ

Istanbul, 2023

NEW AUTOMATIC (IDS) IN IOTS WITH ARTIFICIAL INTELLIGENCE TECHNIQUES

Alaa Firas Jasim JASIM

Electrical and Computer Engineering

Doctor of Philosophy

ALTINBAŞ UNIVERSITY

2023

This dissertation titled NEW AUTOMATIC (IDS) IN IOTS WITH ARTIFICIAL INTELLIGENCE TECHNIQUES prepared by ALAA FIRAS JASIM JASIM and submitted on 25/04/2023 has been accepted unanimously for the degree of PhD in Electrical and Computer Engineering.

Asst. Prof. Dr. Sefer KURNAZ
Supervisor

Thesis Defence Committee Members:

Asst. Prof. Dr. Sefer KURNAZ	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Oğuz KARAN	Department of Software Engineering, Altınbaş University	_____
Asst. Prof. Dr. Zeynep ALTAN	Department of Software Engineering, Beykent University	_____
Prof. Dr. Osman Nuri DURU	Department of Electrical and Electronic Engineering, Altınbaş University	_____
Asst. Prof. Dr. Serdar KARGIN	Department of Biomedical Engineering, Arel University	_____

I hereby declare that this thesis meets all format and submission requirements of a PHD dissertation.
Submission date of the thesis to Institute of Graduate Studies: ___/___/___

By signing this declaration, I vouch for the fact that all information and resources used in this capstone project were acquired with the utmost adherence to moral standards and academic regulations. All references listed in the Reference List and vice versa have been cited in accordance with the aforementioned standards of conduct. I also attest to the accurate citation of all conclusions and borrowed ideas throughout the article.

Alaa Firas Jasim JASIM

Signature

DEDICATION

Thanks to my supervisor, whose expertise, guidance, and invaluable feedback have been critical in shaping this work. Thank you for your unwavering support and patience in guiding me through the challenges of research and for inspiring me to strive for excellence. This work is a testament to the love and support of my family and the dedication of my supervisor, and I am forever grateful for their role in making this achievement possible.

ABSTRACT

NEW AUTOMATIC (IDS) IN IOTS WITH ARTIFICIAL INTELLIGENCE TECHNIQUES

JASIM, Alaa Firas Jasim
M.Sc., Electrical and Computer Engineering, Altınbaş University,
Supervisor: Asst. Prof. Dr. Sefer KURNAZ
Date: April / 2023
Pages: 91

Intrusion detection in wireless sensor networks is a crucial task to ensure the security and reliability of IoT systems. Deep learning techniques have been demonstrated to be effective in various fields, including intrusion detection. Thus, this study proposes a novel framework for intrusion detection in wireless sensor networks using deep learning techniques.Three methods for developing a Network Intrusion Detection System (NID) for the Internet of Things (IoT) will be presented in this study. The first method involves training a deep learning model with optimization algorithms, such as the Biogeography-Based Optimization (BBO) algorithm, to enhance the security of IoT systems. The optimization algorithm helps in finding the best parameters for the deep learning model, leading to improved performance in detecting intrusions. The second method combines optimization algorithms with a classified as a feature selection tool to create a new NID. Feature selection is the process of choosing a subset of relevant features from the original dataset to improve the performance of the classifier. In this method, optimization algorithms are used to select the most relevant features, and then a classifier is trained on these features to detect intrusions. Finally, a Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) layers, along with the "Adam" optimizer, will be utilized to be training and make an evaluation data and handle any null values present in the dataset. CNNs are commonly used in image processing and have been demonstrated to be effective in various applications, including intrusion detection. LSTM is a type of Recurrent Neural Network (RNN) that is particularly well suited for handling sequential data. The "Adam" optimizer is a widely used optimization algorithm that helps to minimize the loss function in deep learning models. In conclusion, this study aims to propose

a novel framework for intrusion detection in wireless sensor networks using deep learning techniques. The three methods presented in this study will provide a comprehensive solution for detecting intrusions in IoT systems, ensuring the security and reliability of these systems.

Keywords: IoT, Intrusion Detection System (IDS), Network IDS, Deep Learning, Cyber Security.

TABLE OF CONTENTS

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

BBQ	:	Biogeography-Based Optimization
NIS	:	Network-Based IDS
AUC	:	Area Under the ROC Curve
TRP	:	True Positive Rate
DT	:	Decision Tree
RF	:	Random Forest
IDS	:	Instruction Detection System
DIDS	:	Distributed Intrusion Detection System
AI	:	Artificial Intelligence
ML	:	Machine Learning
HIDS	:	Host-based Intrusion Detection Systems
KNN	:	K-Nearest Network

1. INTRODUCTION

1.1 BACKGROUND

Wireless Sensor Networks (WSNs) are a type of network that is designed to collect and transmit data from multiple sources. These networks consist of a large number of small, low-cost, low-power, and wireless-enabled sensors that are deployed in a specific area. These sensors are equipped with various types of sensors such as temperature, humidity, pressure, motion, and light sensors, and are capable of collecting data from the environment [1]. The primary objective of WSNs is to provide a cost-effective and efficient solution for collecting data from remote or hard-to-reach locations. These networks are used in a wide range of applications such as environmental monitoring, smart agriculture, home automation, industrial automation, and healthcare monitoring. WSNs operate by using a wireless communication protocol to transmit data from the sensors to a central location or a base station. This communication protocol is typically a low-power wireless protocol such as ZigBee, Bluetooth Low Energy (BLE), or Wi-Fi, which is optimized for low-power consumption and long-range communication [2]. The base station is responsible for collecting the data from the sensors, processing the data, and transmitting it to a remote server or a user interface for further analysis. WSNs are designed to meet the specific requirements of various industrial applications. For example, in an industrial automation setting, WSNs can be used to monitor the temperature and humidity of a production line or to detect the presence of hazardous gases. In a smart agriculture setting, WSNs can be used to monitor soil moisture levels, weather conditions, and plant growth [3]. In a healthcare setting, WSNs can be used to monitor a patient's vital signs and detect any anomalies. To achieve these objectives, WSNs must be designed to be highly reliable, scalable, and energy-efficient. This is achieved by using a variety of techniques such as data aggregation, data compression, and duty cycling. Data aggregation involves combining multiple data packets into a single packet to reduce the amount of data transmitted. Data compression involves reducing the size of the data packets to reduce the energy required for transmission. Duty cycling involves turning the sensors on and off at specific intervals to conserve battery life real-time data analysis. However, security is a key challenge for WSNs [4]. To address this challenge, Intrusion Detection Systems (IDS) have been developed to monitor the network for malicious activities or policy violations and alert the administrator or central log in case

of any incidents. Security information and event management systems (SIEM) are used to distinguish malicious activity from false positive reports [5]. As the internet becomes more and more prevalent in our daily lives, the need for protecting connected devices from cyber-attacks has become increasingly important. One solution to this problem is the use of Artificial Intelligence (AI), which can recognize complex patterns in the inputs to detect malicious activities. However, applying AI to IoT devices is challenging due to the limited resources available [6]. IDSs that can detect complex features in the inputs. However, their extensive computation requirements make them challenging to use in IDSs in IoT devices. This is where This means that NEAT can find the most efficient neural network for IDSs in IoT devices, even with limited resources.Overall, NEAT provides a solution to the challenge of using AI in IDSs in IoT devices, making it possible to effectively detect and prevent cyber-attacks on connected devices [7].

The aim of the study was to address the security challenge in WSNs. The study proposed four automatic systems that were developed to detect and prevent security threats in WSNs. The systems were designed to operate automatically, without requiring constant human intervention, and were evaluated using a common dataset. The results obtained were then analyzed to determine the effectiveness of the proposed systems.The study is divided into several sections. The introduction section provides an overview of the study's purpose and explains the need for secure WSNs. The survey of related techniques section discusses the existing methods used to address the security challenge in WSNs and provides an overview of their strengths and weaknesses. The description of the prototype section provides a detailed explanation of the proposed automatic systems, including their design, implementation, and operation. This section also discusses the key features of each system and explains how they work to detect and prevent security threats in WSNs [8].

Finally, the conclusion section summarizes the key findings of the study and discusses the implications of the results. The section provides an overall evaluation of the proposed automatic systems and their effectiveness in addressing the security challenge in WSNs. Additionally, the section highlights the areas where further research is needed to improve the proposed systems and enhance their effectiveness in securing WSNs.

1.2 INTRODUCTION TO INTRUSION

An intrusion or attack is a malicious and persistent series of similar actions that aim to disrupt services, damage computer systems or networks, and steal or manipulate data without authorization. Essentially, intrusions and attacks refer to the same thing. These actions can cause a network to become unresponsive or to function improperly, or can result in the unauthorized transfer or manipulation of sensitive data. In short, an intrusion or attack is a deliberate attempt to compromise the security of a system or network for malicious purposes.

1.2.1 Intrusion Detection

Intrusion detection is the process of identifying and responding to unauthorized access, use, disclosure, disruption, modification, or destruction of a computer system or network. It is a critical aspect of maintaining the security of a system or network, and involves monitoring and analyzing the activity of the system and network to detect and prevent security breaches that could compromise the availability, confidentiality, and integrity of data and systems. An intrusion detection system (IDS) is the primary tool used to analyze data packets and issue alert messages or notifications to system administrators. The IDS constantly monitors and analyzes network and system activity, looking for any signs of suspicious behavior. This can include things like repeated login attempts, unusual network traffic patterns, or attempts to access sensitive data or systems. When the IDS detects a potential security breach, it issues an alert message to the system administrator, who can then investigate the issue and take appropriate action to mitigate the threat. Depending on the severity of the threat, this could include blocking network access, shutting down affected systems, or launching a full-scale incident response plan.

As shown in Figure 1.1, the intrusion detection system is a critical component of any security infrastructure, and is essential for maintaining the integrity and security of modern computer systems and networks.

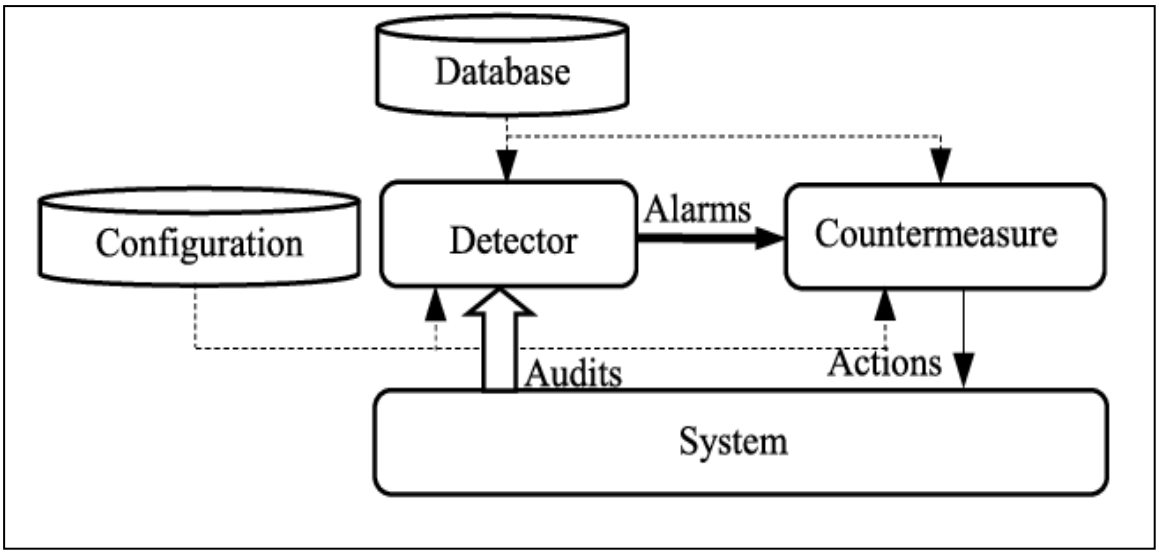

Figure 1.1: Intrusion System Detection [9].

1.2.2 Classified of Intrusion Approaches

Intrusion refers to the unauthorized or unwanted access to a computer system or network. There are various methods of intrusion, some of which are discussed below:

a. Password cracking: Password cracking involves the use of various methods to obtain a user's password, such as guessing, brute force attacks, or dictionary attacks. Attackers may also use social engineering techniques to trick users into revealing their passwords.

b. Phishing: Phishing involves the use of fraudulent emails, text messages, or websites to trick users into providing sensitive information, such as login credentials or credit card details.

c. Malware: Malware is a type of software designed to harm or gain unauthorized access to a computer system. It includes viruses, trojans, worms, and other malicious programs. Attackers may use various methods to deliver malware, such as email attachments, infected websites, or software vulnerabilities. Quite basis.

1.3 METHODS OF THE SECURITY

Network security employs two fundamental approaches, namely proactive and reactive, as illustrated in Figure 1.2. Reactive tools generate a response, such as alerts or alarms, when they detect attacks. Intrusion Detection Systems (IDS) and firewalls are examples of reactive tools. Conversely, proactive tools aim to safeguard the system against potential attacks.

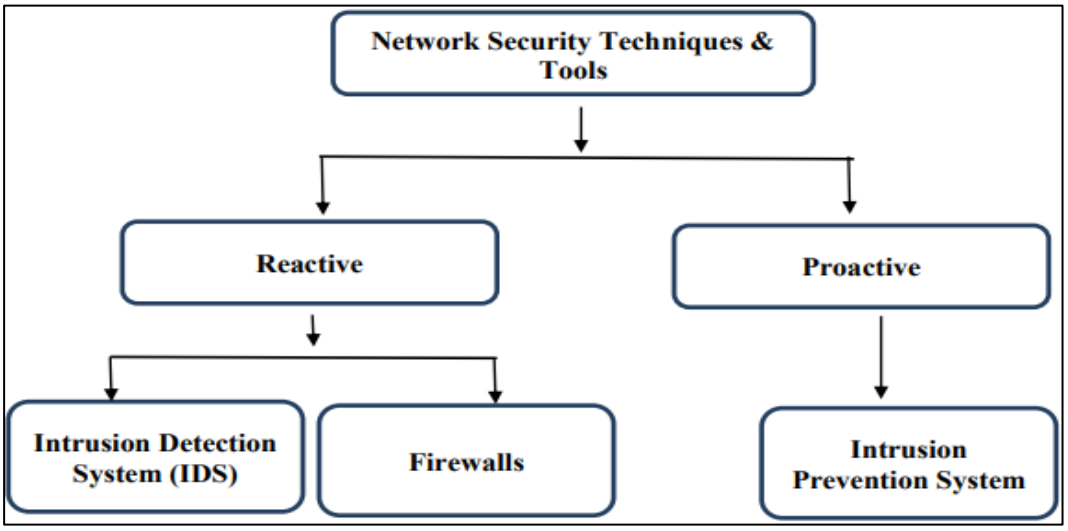

Figure 1.2:Classification of Security Methods [10].

1.3.1 Overview of IDS

The functioning of intrusion detection systems varies between wireless LANs and conventional LANs due to the disparity in their control over the network infrastructure. In comparison to wireless networks, wired networks provide intrusion detection systems with greater autonomy.

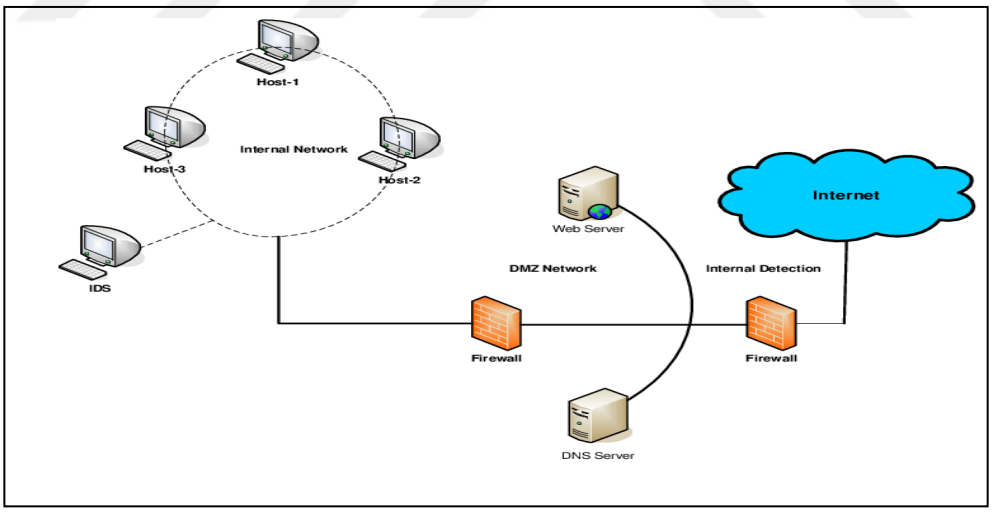

Figure 1.3: Architecture of IDS [11].

1.3.2 IDS Types

Intrusion Detection Systems (IDS) are classified into different types based on various criteria, as explained below:

a. Signature-based IDS: Signature-based IDS works by comparing network traffic or system behavior with a pre-defined signature database of known attacks. It raises an alert when it detects a match with the signature.
b. Anomaly-based IDS: Anomaly-based IDS uses machine learning or statistical techniques to establish a baseline of normal network or system behavior. It raises an alert when it detects any deviation from the established baseline.
c. Hybrid IDS: Hybrid IDS combines both signature-based and anomaly-based detection approaches to enhance detection accuracy and reduce false alarms.

1.4 THESIS OBJECTIVES

Rimary objective of this research is to introduce a new approach that utilizes the Biogeography-Based Optimization (BBO) algorithm to enhance the security of Internet of Things (IoT) systems. With the increased adoption of IoT devices in various domains like healthcare and smart city monitoring, the need to address the security concerns that arise with their usage has become critical. This research proposes a framework that leverages the BBO algorithm to safeguard user privacy and data. The BBO algorithm is a heuristic optimization method inspired by the evolution of species in different geographical regions. It operates on the principle of natural selection, where only the fittest individuals survive and pass on their traits to the next generation. In the context of IoT security, the BBO algorithm can identify and eliminate security vulnerabilities within the system. The proposed framework takes a multi-layer approach that employs the BBO algorithm to optimize the parameters of various security protocols and techniques such as encryption, firewalls, and intrusion detection systems. The framework is designed to continually monitor the system for potential threats and adjust security in real-time. This research thoroughly analyzes the potential applications of the BBO algorithm in IoT security and the advantages it provides over conventional security methods.

This proposed framework is evaluated using simulations and experiments, and the results obtained are carefully evaluated. The study concludes by discussing the future potential of utilizing the BBO algorithm for IoT security and its possible avenues for future research and development. The use of the BBO algorithm provides a unique and innovative approach to addressing the security challenges faced by IoT systems. Therefore, the proposed framework offers a promising solution to ensure the security of IoT systems, and protect the privacy and data of their users.

1.5 THESIS CONTRIBUTION

This research presents several key contributions and novel approaches towards enhancing the security of Internet of Things (IoT) devices. These contributions serve to address the growing security concerns associated with the widespread use of IoT devices in various industries, including healthcare and smart city monitoring.

a. One of the main advantages of this method is its ability to combine the BBO with the auto-encoding framework. This allows for the optimization of its structure, which helps in improving the detection's accuracy.
b. Another key contribution of this study is the use of proposed method for feature selection. This method combines the strengths of BBO and ensemble learning to effectively select relevant features, leading to improved performance in detecting security breaches.
c. In addition, this research presents a novel approach of using hybrid method with Adam optimizer to train and evaluate data. This method effectively reads, drops any null data from the dataset, and trains the model to accurately detect security breaches. Moreover, the use of a sparse auto encoder regularize within the model further enhances its ability to detect security breaches.
d. Overall, these contributions aim to provide a comprehensive and effective framework for improving the security of IoT devices, enhancing privacy protection and securing user data.

1.6 PROPOSED WORK

To address the research questions, the following methods will be used:

a. Literature review: A review of the existing literature on machine learning for network intrusion detection will be conducted to identify the most effective algorithms and approaches.
b. Data collection and preparation: In order to train and evaluate machine learning algorithms, a dataset of typical and unusual network traffic will be gathered.
c. Algorithm selection and evaluation: On the dataset, many machine learning algorithms will be chosen and trained. The algorithms will be evaluated based on their performance in accurately identifying intrusions in the test data.

d. Limitations and challenges: There may be limitations and challenges with using AI techniques for network intrusion detection, which will be identified and described.

1.7 EXPECTED RESULTS

The proposed research is focused on the use of Artificial Inelegance (AI) techniques and methods for detecting network intrusions. With the increasing reliance on technology and the use of networks in various industries and organizations, the need for effective and efficient intrusion detection systems has become more pressing. The goal of this research is to investigate the most effective ML algorithms and strategies for training and validating these algorithms for real-time intrusion detection.

The research will be conducted through a combination of literature review and experimentation. The review will highlight the most commonly used AI methods and algorithms, as well as their limitations and strengths. The investigations will look into the implementation of such techniques and algorithms in network intrusion detection. The results will be analyzed and compared to determine which algorithms and techniques are most effective.

The findings of the review will also likely provide valuable insight regarding the best ML techniques and methods to detect network breaches. The results will also highlight any limitations and challenges of using machine learning for this purpose. This information will be useful for organizations seeking to improve their network security and protect against potential threats. Additionally, the research will contribute to the field of intrusion detection by identifying new and promising areas for future research and development.

The proposed research will be of particular interest to network security professionals, researchers and practitioners in the field of intrusion detection, and organizations seeking to improve their network security. The results of this research will be disseminated through publications in peer-reviewed journals and conferences, as well as through presentations and workshops.

In conclusion, the proposed research aims to investigate the most effective AI algorithms and strategies for real-time intrusion detection, with the goal of providing valuable insights that can be used to improve network security and protect against potential threats. The

research will be conducted through a combination of literature review and experimentation, and the results will be disseminated through publications, presentations, and workshops.

1.8 THESIS OUTLINE

The remaining chapters of this research are organized as follows:

a. Chapter 2: This chapter will discuss the intrusion detection system, machine learning methods, and the most recent related works. This chapter will provide an overview of the current state of the art in the field and will set the context for the proposed research.
b. Chapter 3: This chapter will provide a detailed description of the methodology used in the research. It will cover the different phases of building the system, including the dataset, pre-processing phase, models, cooperative study between the models and the final training phase. This chapter will also include a description of the steps that will be included in the dataset, pre-processing phase, models, cooperative study between the models, and the last phase which is the training phase.
c. Chapter 4: This chapter will contain the final training results of the models. It will apply assessment measures and contrast the outcomes. This chapter will be the main results of our research and will provide a detailed analysis of the performance of the different models.
d. Chapter 5: This chapter will provide a conclusion that summarizes all the work done in the thesis, including the models chosen, the stages, and the results. It will also include a discussion of the implications of the findings and suggestions for future research. The figure 1.4 shows the steps of the outline of the remaining chapters of the thesis.

Finally, the remaining chapters of the research are organized to provide a detailed description of the methodology used, the results of the research, and a conclusion that summarizes the main findings and implications of the research.

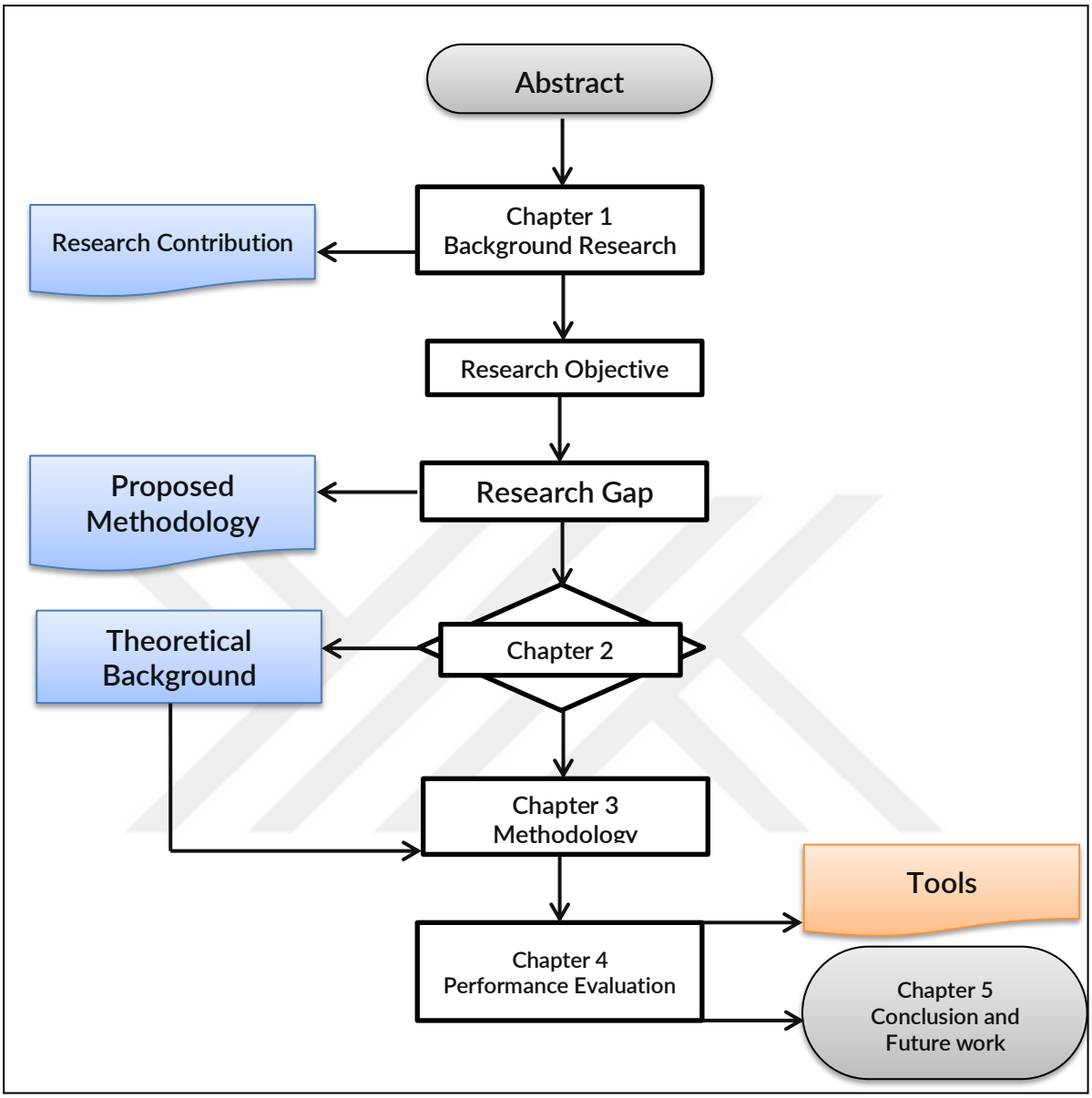

Figure 1.4: Flowchart Shows the Outline of the Thesis.

1.9 SUMMARY OF THE CHAPTER

Network intrusion detection is a critical issue for organizations seeking to protect against unauthorized access and potential threats. The increasing reliance on technology and the use of networks in various industries and organizations has made it more essential to have effective and efficient intrusion detection systems. One promising solution is the use of machine learning (ML) algorithms which can substantially enhance the precision and efficiency of intrusion detection.

The proposed research aims to advance more effective methods for detecting and preventing network intrusions through the use of ML techniques and methods. The research will be conducted through a combination of literature review and experimentation. The literature review will provide an overview of the current state of the art in the field, including the most commonly used ML techniques and methods, as well as their strengths and limitations. The experimentation will involve the implementation of different ML algorithms and techniques, and the evaluation of their performance in detecting network intrusions. The results will be analyzed and compared to determine which algorithms and techniques are most effective.

The findings of the review will also likely provide valuable insight regarding the best ML techniques and methods to detect network breaches. The results will also highlight any limitations and challenges of using machine learning for this purpose. This information will be useful for organizations seeking to improve their network security and protect against potential threats. Additionally, the research will contribute to the field of intrusion detection by identifying new and promising areas for future research and development.

Finally, the proposed research aims to advance more effective methods for detecting and preventing network intrusions through the use of ML techniques and methods. The research will provide valuable insights for organizations seeking to improve their network security and protect against potential threats.

2. LITERATURE REVIEW

2.1 INTRODUCTION

In this chapter, we will discuss the IDS and its types. We will also introduce various adaptable methods, deep learning for IDS. Finally, we will review the literature on attack detection based on these algorithms and previous research in this field. IDS may make use of a variety of ML methods, including CNN LTSM, and Gradient Boosting. In previous research, these algorithms have been applied to various types of data to detect a wide range of attacks (e.g. network scans, denial of service attacks, malware infections).

2.2 INSTRUCTION DETECTION SYSTEM

A software program known as an IDS designed to protect computer networks from unauthorized access or attacks [12]. It does this by continuously monitoring the network for any suspicious activity and using various machine learning algorithms to detect and prevent intrusions [13]. IDS can be used to protect networks from both external and internal threats, including potentially malicious insiders [14]. IDS's primary responsibility is to create a predictive model, also known as a classifier that can accurately distinguish between normal and malicious connections [15]. This allows the IDS to alert the appropriate authorities and take appropriate action to protect the network and its assets [16]. It is important to note that an IDS differs from an Intrusion Prevention System (IPS), which also monitors network traffic for potential threats but focuses on actively preventing them once detected, rather than primarily detecting and recording them [17]. IDS is a tool used in cybersecurity, often alongside a firewall and antivirus software, to protect against potential threats to a network or system [18]. Additionally, the authors note that a firewall is limited in its ability to analyse online traffic, whereas an IDS is capable of monitoring, controlling, and maintaining the flow of all network traffic, including detecting and alerting network administrators to any irregular behaviour or threats that may occur within the network [19]. Figure 2 illustrates the flow of network communication and the presence of an IDS within the mesh.

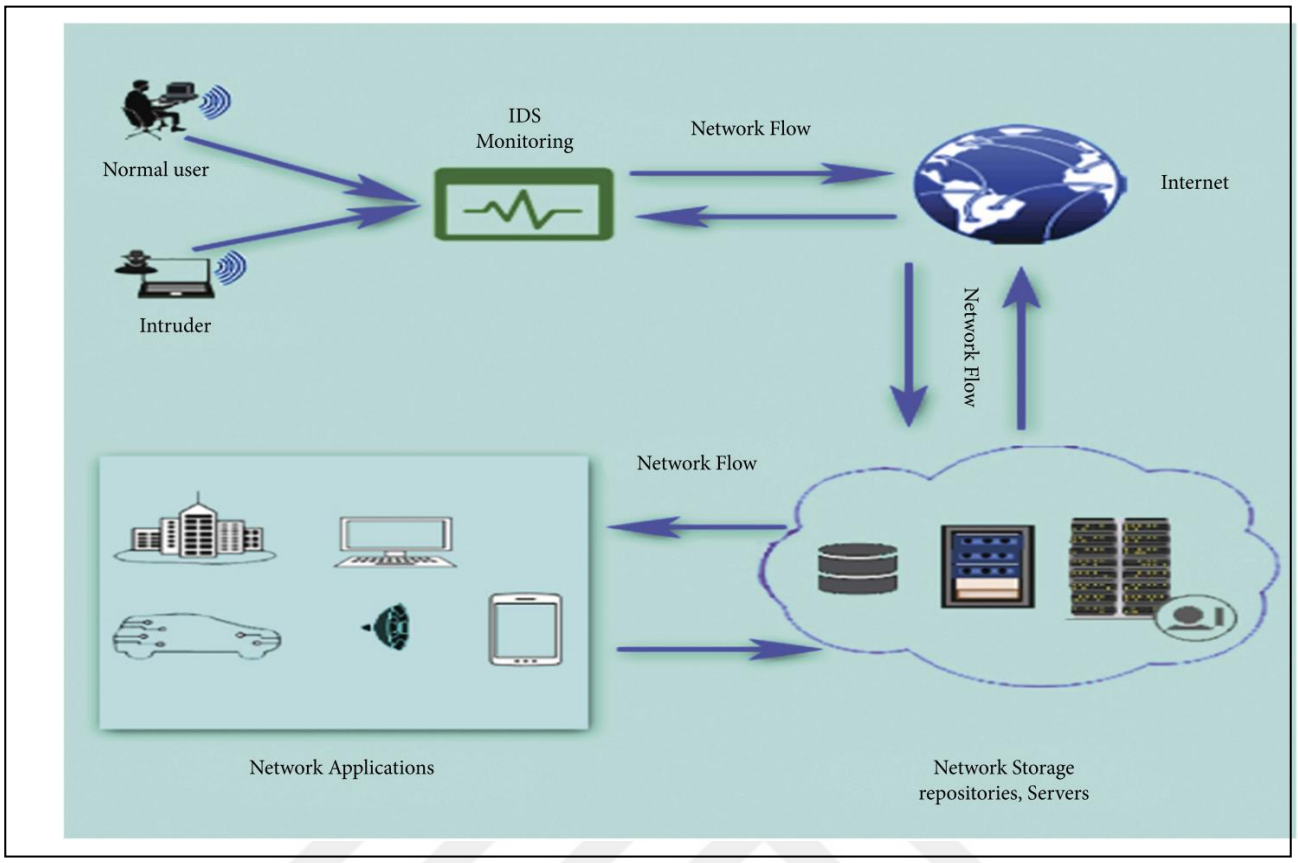

Figure 2.1: IDS and Network Traffic Across the Networks [20].

Different types of IDSs exist, among the host-based, network-based, anomaly-based, anomaly-based IDSs (HIDS), anomaly-based IDSs (AIDS), hybrid-based IDSs, and distributed-based IDSs (DIDS) [21, 22].

2.2.1 Signature-Based Intrusion Detection Systems (SIDS)

IDS, known also as knowledge-based detection, analyses networks using recognized patterns or signatures to spot possible threats. [23, 24]. These systems store the behaviour and signature of each type of attack in a database, and produce an alert when an incoming attack matches one of the stored signatures. Signature-based IDSs (SIDS) are effective at detecting known attacks, but are not as accurate when it comes to detecting variations or new types of attacks. In order to minimize false alerts and effectively identify and classify misbehaviour, new attack signatures must be added to SIDS on a regular basis. Intrusion detection systems using anomaly-based analysis (AIDS) are an alternative approach that profiles the normal behaviour of a network and generates an alert when there is a deviation from this expected behaviour. This allows AIDS to potentially detect new or unknown variations of attacks. However, the accuracy of AIDS can be affected by the quality of the

normal behaviour profile and the ability to accurately distinguish between normal deviations and malicious activity [25].

2.2.2 Anomaly-Based Intrusion Detection System (AIDS)

To get around the drawbacks of signature-based IDSs, profile-based or dynamic behaviour anomaly detection, often known as AIDS, are frequently employed. Anomaly-Based Intrusion Detection System (AIDS) are effective at detecting new attacks, making them a popular choice compared to SIDS. AIDS work by continuously monitoring a system to collect data on normal and abnormal behaviour, and generating an alert when there is a deviation from the expected behaviour. Identification of zero-day assaults is one of AIDS's primary goals. or new anomalous actions that are not yet included in pattern databases. AIDS are used to recognize exposures and prevent attacks at different stages by recognizing unidentified attacks and taking defensive action [22]. AIDS are designed to learn and recognize abnormal behaviour within a network. For instance, if an account is compromised or unlawful activity takes place, an alert will be generated. However, these systems may also generate false positives if they detect unusual but harmless behaviour. This can lead to a high false-positive rate [23].

2.2.3 Customized Intrusion Detection Methods

Customized intrusion detection methods are similar to anomaly-based IDSs (AIDS) in that they specify standard network activities using rules and specifications. These guidelines are then used to monitor the network, and when a divergence from the expected behavior occurs, an alarm is created. Customized IDSs have a lower false positive rate because they are resistant to new attack variations. However, these systems have limitations due to the complexity and time consuming nature of development, as well as the cost associated with implementing them [24].

2.3 HYBRID INTRUSION DETECTION TECHNIQUES

Anomaly, misuse, and specification detection approaches have been combined to create hybrid intrusion detection methods, often referred to as compound detection, in order to increase the identification of both current and emerging attack behavior. These techniques try to get around the restrictions. of individual techniques and enhance the overall performance of the IDS. One example of a hybrid IDS is the SVELTE technique [25], which was developed for use in 6 LoWPAN networks in the Internet of Things (IoT) and combines

signature-based and anomaly-based techniques. The goal of this hybrid approach is to ameliorate the stability, cost-effectiveness, and simplicity of the IDS, while also reducing the complexity of storage and processing.

2.4 HOST-BASED IDS (HIDS)

Software applications known as HIDS are placed on a single host computer and are used to continually study, analyse, and monitor data actions occurring on the host and its connected networks. HIDS are used to identify unusual assaults on a single host by gathering information from database logs, server, or from firewall. However, HIDS have a restricted capacity in identify threats that do not directly harm the host. [26].

2.5 NETWORK-BASED IDS (NIDS)

NIDS [27] is used to guard networks from outside intrusions. It does this by gathering and analysing packets captured through NetFlow and other means, and issuing an alert or alarm if it detects a malicious attack. NIDS can be implemented using software installed on servers or through hardware sensors attached to servers. It monitors and analyses network communications across multiple hosts and external firewalls, making it an effective and secure way to detect malicious attacks. Due to its inability to handle high bandwidth and high speed traffic, a NIDS may struggle to process and analyse large volumes of network data. Additionally, NIDS is not equipped to detect attacks in encrypted network packets, which can be a significant limitation in today's network environments where encryption is commonly used to protect data [28].

2.6 DISTRIBUTED IDS (DIDS)

A Distributed Intrusion Detection System (DIDS) [29] is a network of multiple Intrusion Detection Systems (IDS) that work together to analyse communication, monitor and manage malicious attack information, and handle incidents. It manages intrusion detection and prevention by combining data from several sensors, including NIDS and HIDS, and using acentral analyzer.

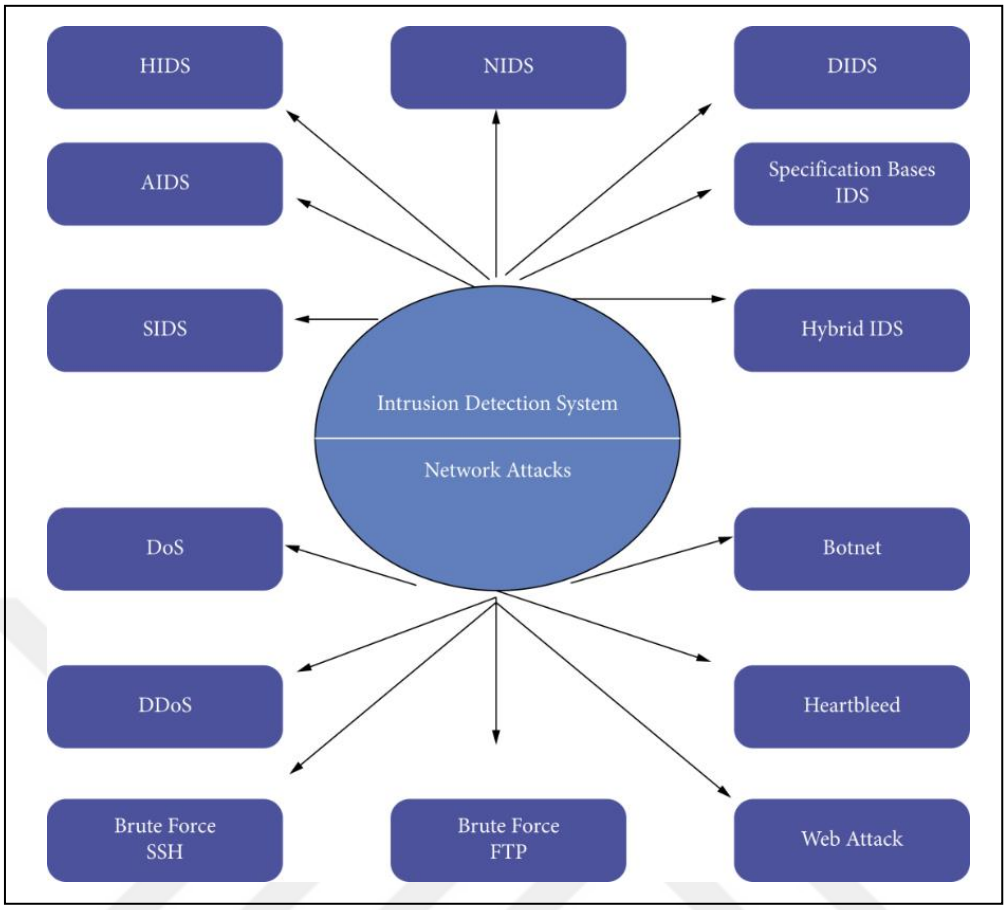

Figure 2.2: Kinds of Network Intrusion Detection Systems and Attacks [29].

2.7 MACHINE LEARNING (ML)

Computer science includes the topic of ML that calls for the creation of techniques and algorithms that allow computers, without the need for literal programming, to learn from the provided data without being explicitly programmed [30]. ML is concerned with the design of systems that can autonomously acquire, integrate, and apply knowledge to improve their performance and efficiency in completing tasks [31]. Numerous applications make use of machine learning methods, including image and speech recognition [32], natural language processing [33], recommendation systems [34], and intrusion detection [35].

Machine learning comes in a variety of forms, including supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. [36].

a. In supervised learning, the algorithm is trained on a labelled dataset, this includes input data and associated output labels. that the main goal is to create a prototype template which can correctly forecast the labels of the output for brand-new, untested input data. [37].

b. In unsupervised learning, the algorithm must find patterns and correlations in the data on its own because it is not provided any labelled data. [38].

c. A model is developed utilizing both a sizable amount of unlabelled data and a small amount of data that has been tagged, with the goal of teaching the algorithm to predict the labels for the unlabelled data using the labelled data.

d. The aim of semi-supervised learning is to improve the performance of the model by leveraging the additional information provided by the unlabelled data, while still benefiting from the guidance of the labelled data.

e. Reinforcement learning is the process of teaching a model to make decisions in a given environment by seeing what happens when it takes certain actions. [39].

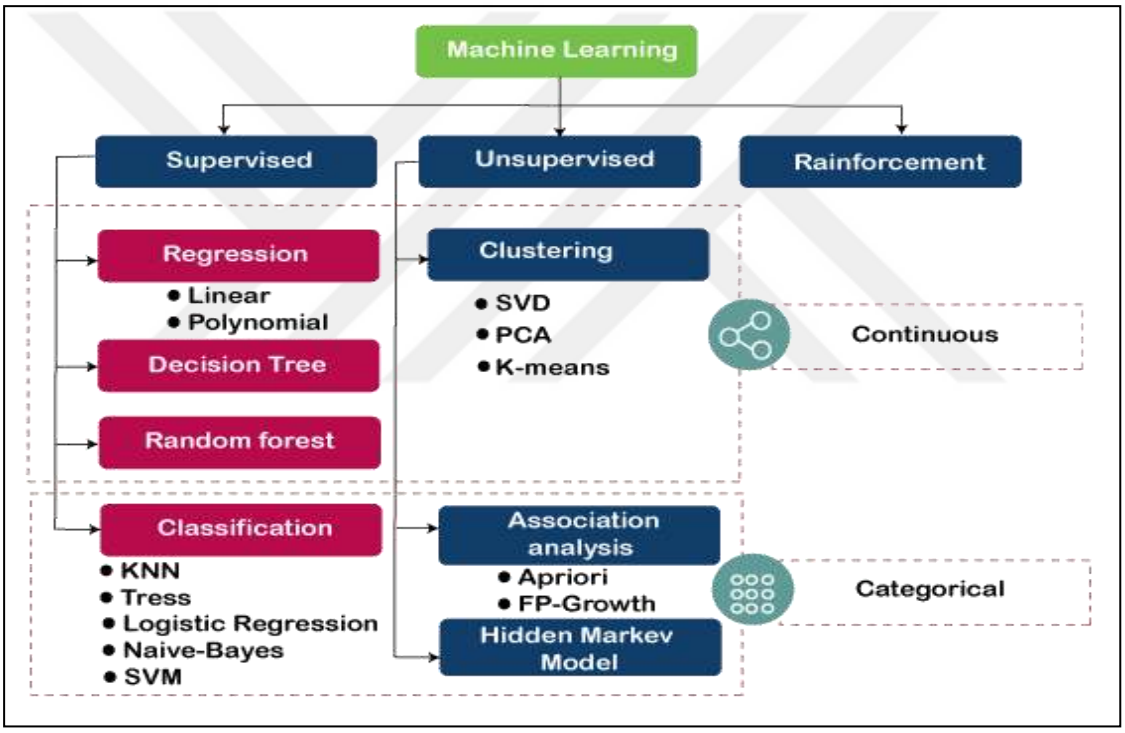

Figure 2.3: Types of Machine Learning Algorithms [39].

ML has become increasingly important recently especially by the explosion of the data being generated and the need for efficient and effective ways to process and analyse this data [40]. It is a technique commonly used in IDS to detect attacks. It involves the design and development of algorithms and methods that enable computer systems to autonomously acquire and integrate knowledge in order to improve their performance and efficiency. In recent years, ML-based IDS have been shown to have high accuracy and effectiveness in monitoring novel attacks.

2.8 MACHINE LEARNING ALGORITHMS

Different ML methods, such as KNN, DT, RF, Linear SVM, Radial Basis Function (RBF) SVM, XGBoost (eXtreme Gradient Boosting), and Gradient Boosting, will be covered in this section. These methods have been utilized in a wide range of disciplines and are useful for a number of tasks including classification, regression, and clustering. We'll give a quick overview of each algorithm before going into its salient features and uses.

2.8.1 K-Nearest Neighbours (KNN)

KNN: basic well-liked ML technique for classification and regression problems [40]. The idea that a prediction may be produced by choosing the K data points from the training set that are most similar to a brand-new, unknown data point is the basis of this method. As shows below in the figure shows the algorithm of K-Nearest Neighbours

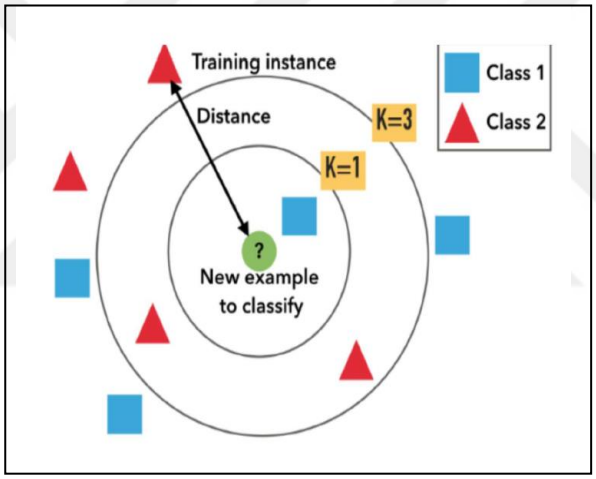

Figure 2.4: Shows the Algorithm of KNN [40].

The technique initially determines the distance between each point of the new data and the training data setin order to produce a prediction using KNN (Cao, et al., 2021). The distance may be calculated using a variety of metrics, such as the Manhattan distance and the Euclidean distance (Zhang, et al., 2020). The "nearest neighbors" are then determined to be the K data points in the training set that are closest to the new data point [41]. The class with the most "votes" among the K closest neighbors is selected as the predicted class in classification tasks by a majority vote [42]. The prediction for regression tasks is created by averaging the K nearest neighbors' values [43].

$$d(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2} \quad (2.1)$$

KNN has several advantages, including that it is simple to implement and interpret, and it can work with categorical and continuous data [44]. The selection of K can significantly affect the precision of the predictions, but it can be computationally costly to compute the distances between all the data points in the training set for each new data point. [45].

2.8.2 Decision Tree (DT)

DT is a specific kind of ML method that is applied to both regression and classification. [46]. It is a model that is built using a tree-like structure [48], with a series of decision nodes representing different choices or decisions that the algorithm must make [49]. The branches extending from each node in the tree reflect potential values or outcomes for each node's associated characteristic or attribute of the data. The algorithm uses the input data collected from the root node and moves up the tree to predict the variable's path. Decision trees are frequently used for tasks like forecasting the probability of a specific event based on specific circumstances or finding the most crucial variables influencing a specific outcome.

2.8.3 Random Forest (RF)

learning algorithms are combined using ensemble techniques to provide predictions that are more accurate than those produced by any one algorithm alone. Each learning algorithm, or "decision tree," in a random forest is built using the training data (bootstrappedsamples), and the optimal split would be selected out of a random subset of the available features at each node in the tree. The forecasts from each tree in the forest are averaged to provide the final projection [10]. It follows that each tree in the forest is distinctive from the others in some way. As a result, random forests are a useful tool for predictive modelling because they can capture complex non-linear correlations in the data and guard against overfitting [50].

2.8.4 Linear Support Vector Machine (SVM)

A ML method known as SVM may be applied to classification or regression problems [42]. In a space of high-dimensional level, it looks for a hyperplane which maximum divides data points into various classes. The data must be able to be divided into distinct classes by a straight line or hyperplane in order for linear SVMs to be effective. Additionally, they need

comparatively less training time as compared to non-linear SVMs. [43]. In the coming figure shows how the dataset is linear separable.

$$b + W \cdot X = 0 \tag{2.2}$$

The weight vector W ∈ R^n and the scalar bias b are used to represent the given problem. In the case of two dimensions, this can be simplified.

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{2.3}$$

Can be known the bounders as follows:

$$y_i = \begin{cases} +1, & w_1 x_1 + w_2 x_2 + b \geq 1 \\ -1, & w_1 x_1 + w_2 x_2 + b \leq -1 \end{cases} \tag{2.4}$$

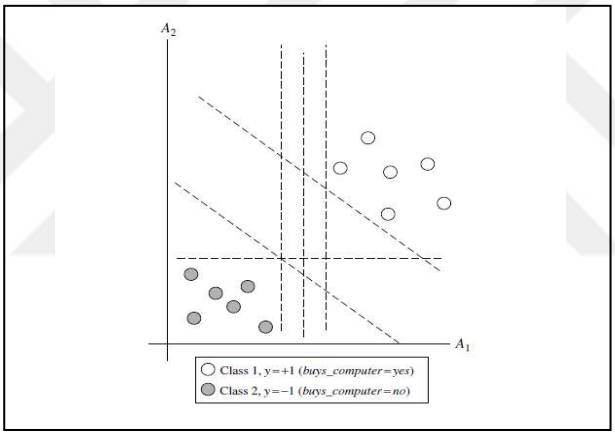

Figure 2.5: Shows the Linear of Separable of the Dataset [43].

One key advantage of linear SVMs is that they can handle large amounts of data efficiently, making them a good choice for large-scale classification tasks [44]. They are also relatively simple to implement, as they do not require tuning of complex hyper parameters [45]. However, linear SVMs are not suitable for all types of data. They may perform poorly for data that cannot be linearly divided, or when the data has many features with different scales. In these cases, a non-linear SVM or another machine learning algorithm may be a better choice.

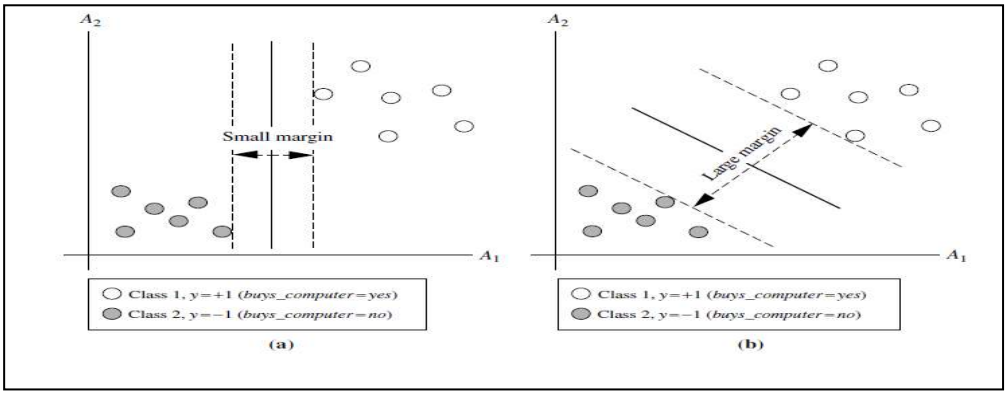

Figure 2.6: (a) Small Margin (b) Large Margin.

2.8.5 Radial Basis Function SVM

They are based on the concept of creating a higher-dimensional environment, in which the data points could be easier to separate, utilizing a non-linear modification of the input data. Using a kernel function, the input data are translated into this higher-dimensional space, which is a function that assesses the similarity between two data points, by RBF SVMs. The radial basis function (RBF) kernel, a kind of non-linear kernel, is utilized in RBF SVMs and is known as the Euclidean points [49].

RBF SVMs have several advantages over linear SVMs. They can handle non-linearly separable data and data with features of different scales more effectively [50]. They are also more flexible, as the kernel function can be adjusted to fit the characteristics of the data. However, they could cost more to train computationally, especially when the data is large or has many features.

2.8.6 XGB Classifier

XGBoost is a popular and efficient open-source implementation of the gradient boosting algorithm, which has been used to win many machine learning competitions [52]. It is a learning technique that requires supervision which may be applied to regression problems or classification. The gradient descent approach is used to improve the model after creating an ensemble of weak decision trees. Due to its computational effectiveness and ability to handle huge datasets, XGBoost is a good choice for complicated model training. Additionally, the model features a variety of hyper-parameters that may be adjusted to enhance its functionality.

2.8.7 Gradient Boosting (GB)

The GB ensemble learning method is a popular choice for performing regression and classification tasks. It combines multiple weak learners to form a strong learner that can make accurate predictions. Unlike other ensemble methods, such as random forest, which use bootstrapping and random feature selection to build the weak learners, the loss function is minimized and total prediction accuracy is increased through gradient boosting, which employs gradient descent. In a gradient boosting model, each weak learner is trained to fix the errors committed by the prior learner. Combining all of the poor learners' predictions yields the final conclusion. Through this iterative training process, the model may learn from the mistakes of earlier trainees and gradually increase the accuracy of its predictions.

2.9 INTERNET OF THINGS (IOTS)

There are many different views on the definition of the concept of IoT in terminology. The reason for this difference actually comes from the two words that make up the concept. Various commercial companies and research institutions can use the internet or the internet depending on their infrastructure, interests or they created a definition by focusing on the object part [71].

The Internet of Things is a hybrid architecture, i.e. It contains the architectures of the various subsystems. In most cases, the Internet of Things Systems consist of two control structures: event control. It's time. Event-driven architectural sensors they transmit data when they see activity in the external environment. For example, an alarm is triggered nearby when a door is opened. Open at night. In temporary architecture Components transmit data continuously over a period. (For example, the air conditioning sensors detect a room every second. Temperature). The latter usually runs over and over after a break, they can be defined or set individually for each device. In a centralized control system for sending requests After a while the sensors. One such system solution is being offered by Intel Corporation (see Fig. 1) [72].

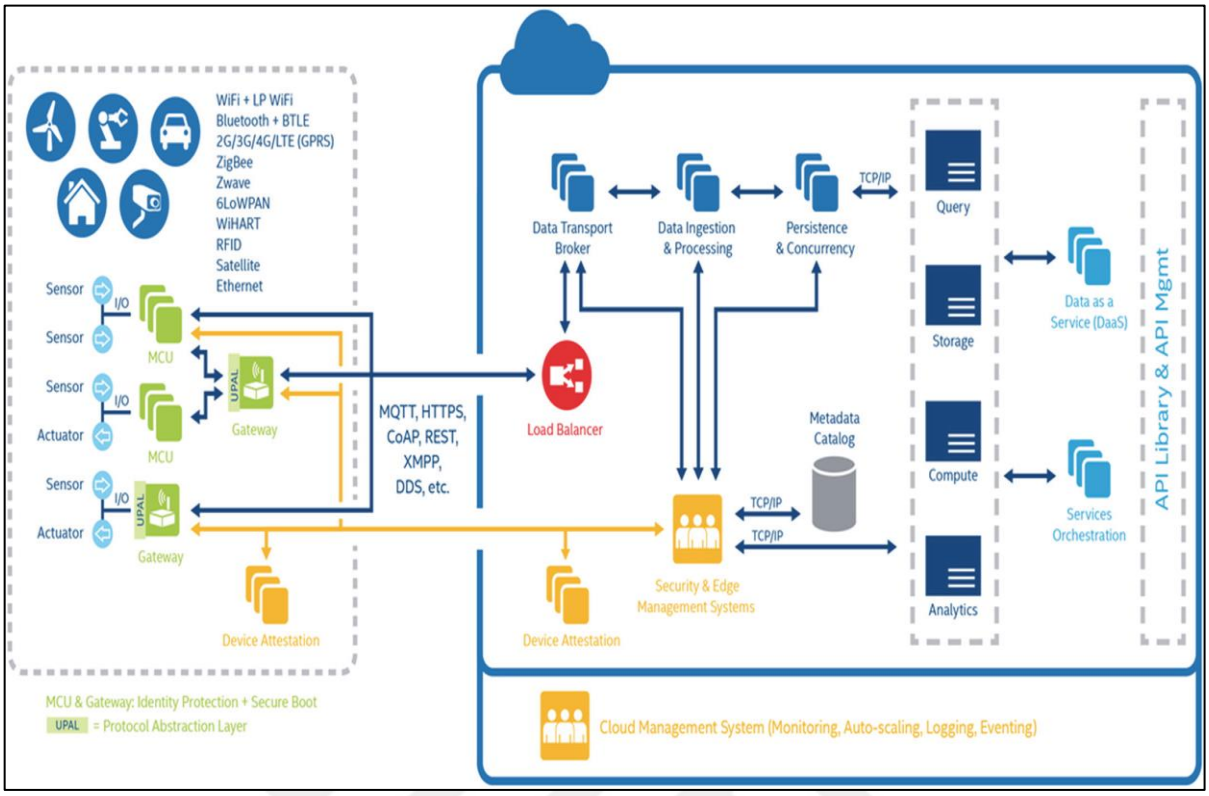

Figure 2.7: Structure of IOT.

2.9.1 Overview of IOT and Intrusion Detection Systems

An overview of the different protocols utilized in the Internet of Things (IoT) is presented in Figure 2.8. They are divided into four main groups: MAC, Network, Transport, and Physical. An attack example is known as a Sybil, which involves overpowering a remote tyke to gain access to your resources. The effect of this method is usually changed on the MAC Layer. On the other hand, when the attack is carried out on the Network layer, the change in the transmission link is performed as a fake route. Although some of the techniques used in these attacks have different goals, they are still carried out in the same manner. For instance, the flooding technique overwhelms a target.

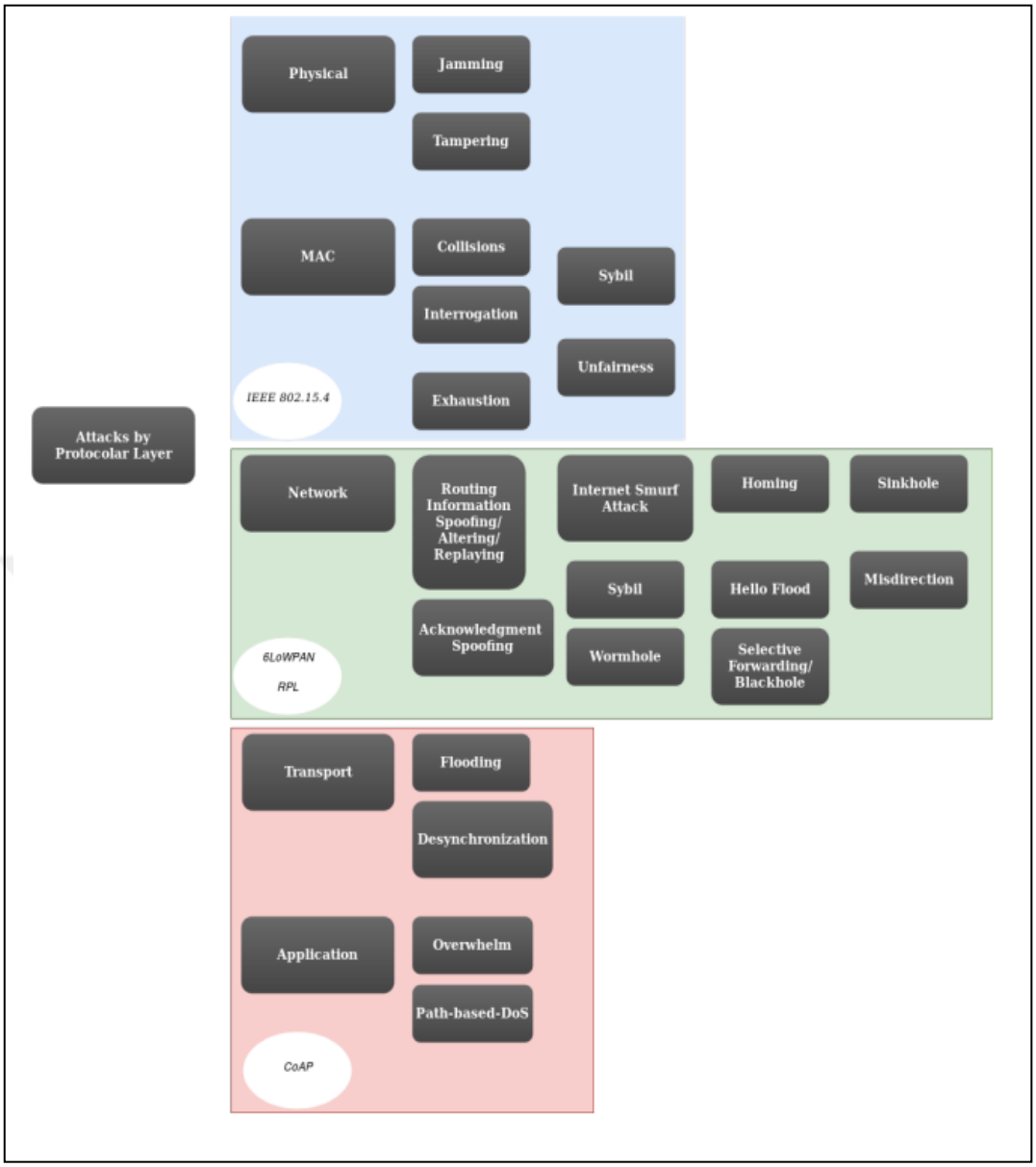

Figure 2.8: Classification of the Intrusion in Terms of Protocol Layer [72].

2.9.2 IDS Anomaly Type

By utilizing this technique, it is possible to develop a model that can anticipate network behavior. An intrusion detection system (IDS) that is based on detecting anomalies can scrutinize the data patterns of a network and pinpoint irregular and typical actions. Unlike signature-based IDS, anomaly-based IDS do not depend on predefined rules, which can be beneficial for the security manager. An anomaly-based IDS is also useful in identifying and blocking harmful traffic. It can additionally prevent the forgery of traffic sent to a botnet, which is a gateway for the remaining networks. This method assumes that a device is communicating with its attacker. When a packet goes through an infected node, it should be

larger than the non-contaminated ones because the source has to update its bot in order to perform a forgery [73].

The goal of this method is to analyze the data patterns of a network and identify anomalous activities. In order to achieve this, three conditions must be met: first, the protocolar communications between the nodes must be restricted to TCP, second, the number of services should be limited, and third, the connection between the 6LoWPAN devices and the gateway should be assumed to be normal [74].

Figure 2.9 shows the network's architectural components. Since the TCP is not following the CoAP protocol, it is not able to perform this function. Also, it is not clear what the application layer protocol of the TCP.

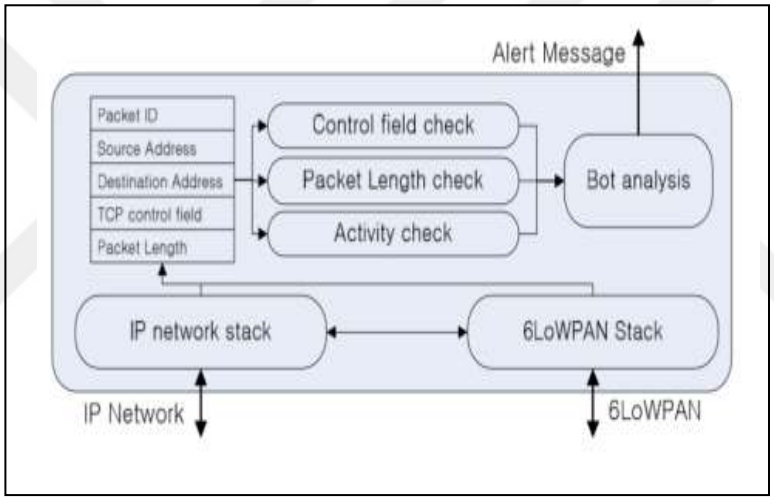

Figure 2.9: The network's Architectural Components [75].

The wireless IDS system, known as WIDS, uses agents that are constantly looking for and analyzing events. The design of the system can be observed in Figure 2.10. The benefits of the system consist of enhanced performance and reduced errors. The utilization of computational analysis can facilitate the detection of novel security threats and boost the effectiveness of the system. It can also help prevent unauthorized access. Some of its advantages include being able to detect new attacks and intrusions [76].

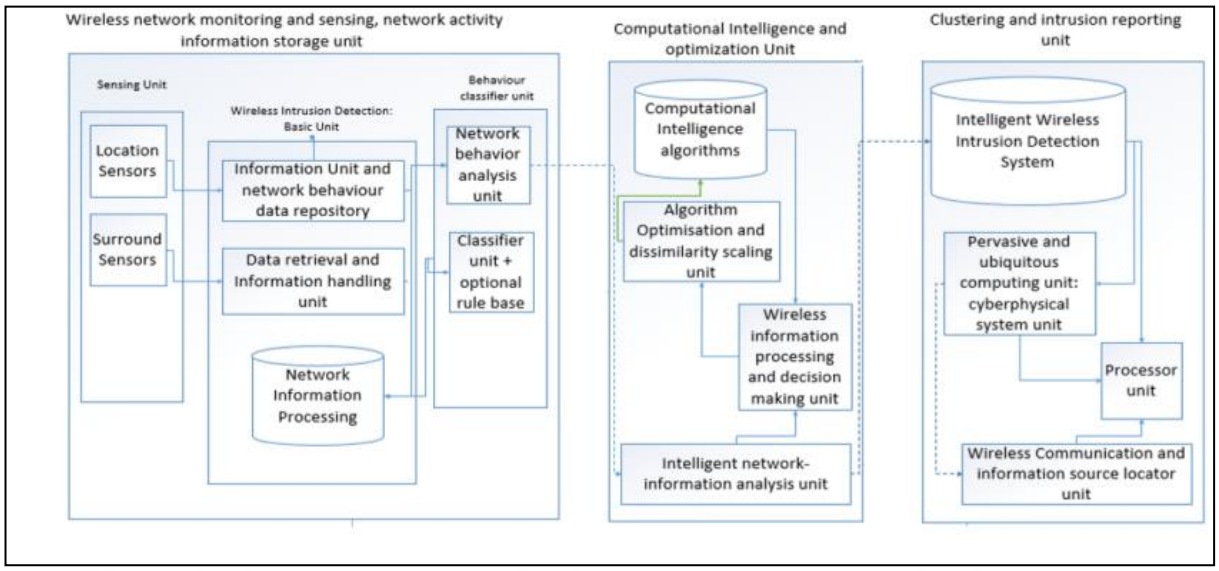

Figure 2.10: The Wireless IDS System [76].

2.10 THREATS AND VULNERABILITIES IN SENSOR SYSTEMS

Sensor security challenges refer to the integrity, confidentiality, and availability of data collected and processed by sensors. These challenges can arise from various sources, including unauthorized access to sensor networks, attacks on sensor data during transmission, and manipulation of sensor readings [77]. Also the figure 2.11, shows the security challenges.

Some of the common sensor security challenges include:

a. Lack of encryption: Sensors may transmit data in an unencrypted format, making it vulnerable to eavesdropping and tampering.
b. Inadequate authentication: Sensors may not have sufficient mechanisms to authenticate the identity of data sources, making it easier for attackers to inject false data into the system.
c. Interference and jamming: Attackers can interfere with or jam the communication signals between sensors, disrupting the normal flow of data.
d. Insufficient physical security: Sensors may be vulnerable to physical attacks, such as theft or tampering, which can compromise the integrity of data collected by the sensors.
e. Malicious software: Malicious software, such as viruses or malware, can infect sensors and disrupt their normal operation, compromising the security of the data they collect and transmit.

Technical solutions involve the use of encryption and authentication mechanisms, which can be used to ensure that only authorized users have access to sensitive information. Encryption converts the data into an unreadable format, while authentication verifies the identity of the user accessing the data. Procedural controls, on the other hand, involve establishing policies and procedures to ensure that security measures are implemented consistently and effectively. This includes the use of access controls to limit access to sensitive information to only authorized users, as well as implementing security audits to monitor and track access to sensitive data. A combination of both technical and procedural controls is necessary to ensure that the security technology is effective in detecting and preventing unauthorized access while minimizing false alarms.

Addressing these challenges requires a combination of technical solutions, such as encryption and authentication, and procedural controls, such as access controls and security audits.

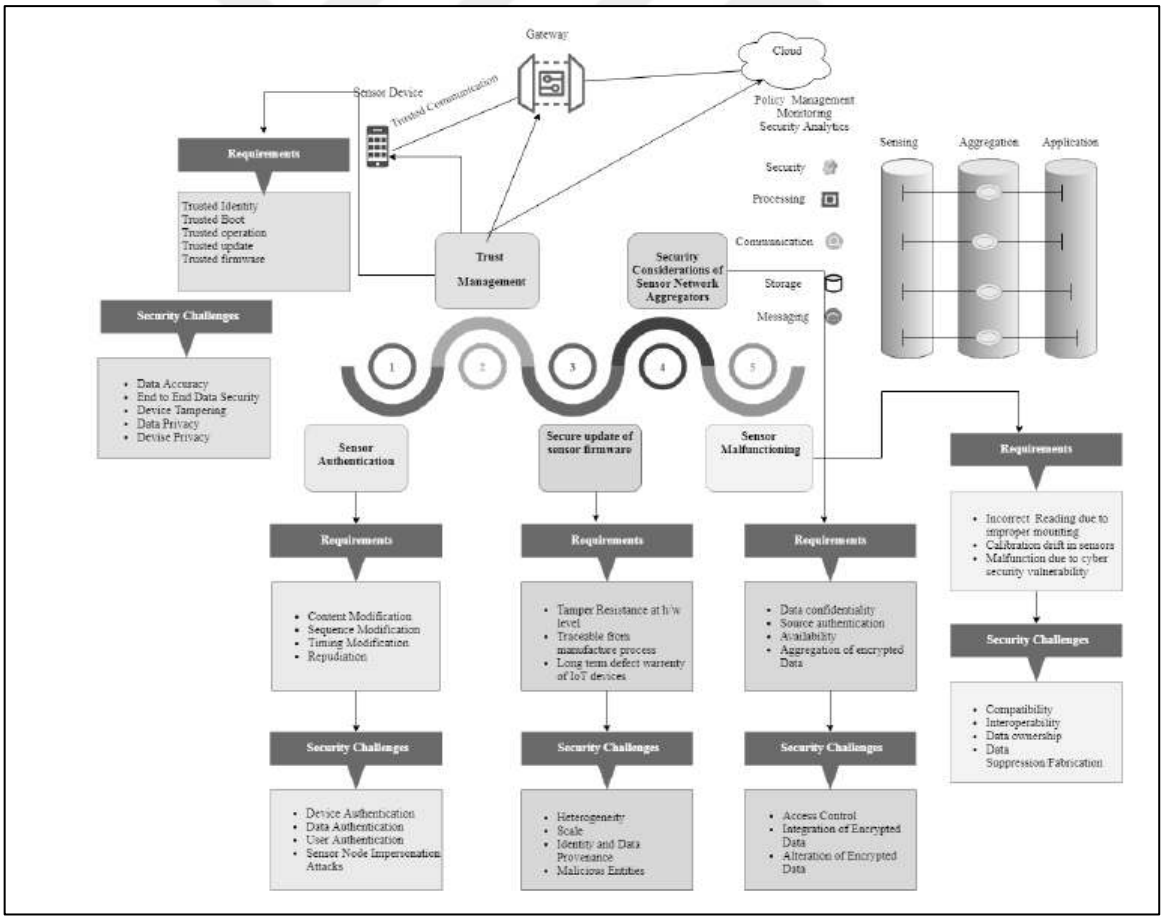

Figure 2.11: The Flow of Security Challenges [77].

2.11 CYBER SECURITY

Cybersecurity refers to the practice of safeguarding digital devices, networks, systems, and sensitive data from unauthorized access, theft, damage, or any other malicious activities [78]. It involves the use of various technologies, processes, and policies to secure digital assets and prevent cyber-attacks [79].

Cybersecurity has become increasingly important in today's digital age, where businesses, governments, and individuals store and transmit sensitive information online [80]. Cyber-attacks can take various forms, such as phishing attacks, ransomware attacks, distributed denial of service (DDoS) attacks, malware, and viruses. These attacks can result in financial losses, reputational damage, and legal consequences for individuals and organizations. To address these risks, cybersecurity measures typically involve a multi-layered approach. This may include implementing firewalls, intrusion detection systems, antivirus software, encryption technologies, access control policies, and employee training programs [81-84]. Cybersecurity also involves staying up-to-date with the latest threats and vulnerabilities and regularly updating security measures to mitigate potential risks. Overall, cybersecurity is an essential aspect of ensuring the safety and integrity of digital systems and protecting sensitive information from unauthorized access or misuse [85]. As shown in the figure 2.12, the flow of how the cyber security working.

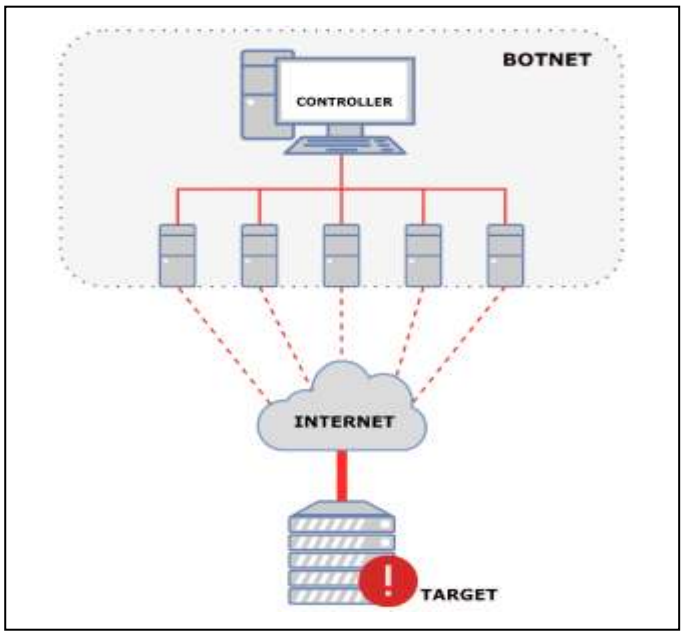

Figure 2.12: Indicts the DDOS Attacks [86].

2.12 RELATED WORK

In recent years, the number of IoT devices has increased dramatically, making IoT networks vulnerable to cyberattacks. As a result, developing effective IDS for IoT networks is essential to ensure their security. In the area of AI approaches, for instance ML approaches and in addition and DL approach, have emerged as promising tools for developing IDS in IoT networks.

This paper [89], presents a comprehensive survey of IDS techniques in IoT networks using AI techniques. The paper covers various aspects of IDS, including the types of attacks and the types of AI techniques used in IDS. The authors also provide a comparison of different AI-based IDS techniques and discuss their advantages and limitations. This paper [90] proposes an anomaly detection system for IoT networks using ML techniques. The proposed system uses unsupervised learning algorithms to detect anomalies in IoT network traffic. The authors evaluated the system using real-world datasets and demonstrated its effectiveness in detecting anomalies in IoT network traffic.

The authors [91], proposes a ML-based IDS for IoT networks that uses a combination of supervised and unsupervised learning algorithms. The authors used a publicly available dataset to evaluate the performance of the proposed IDS and demonstrated that it outperforms traditional rule-based IDS in terms of detection rate and false alarm rate.

M. R. Islam and M. A. Razzaque. [92], in the paper proposes a DL-based IDS for IoT networks that uses a convolutional neural network (CNN) to classify network traffic as normal or anomalous. The authors evaluated the proposed IDS using real-world datasets and demonstrated that it outperforms traditional rule-based IDS in terms of accuracy and false alarm rate.

In this study [93], suggested an intelligent IDS for IoT networks that combines ML and fuzzy logic techniques. The proposed system uses a ML algorithm to classify network traffic as normal or anomalous and a fuzzy logic algorithm to determine the severity of the detected anomaly. The authors evaluated the proposed IDS using real-world datasets and demonstrated its effectiveness in detecting and classifying network anomalies, an anomaly-based IDS for IoT networks that uses an unsupervised ML algorithm to detect anomalies in

network traffic. The authors evaluated the proposed IDS using real-world datasets and demonstrated its effectiveness in detecting various types of network anomalies.

Overall, the related works on new automatic IDS in IoT with AI techniques demonstrate that AI-based IDS can effectively detect and classify network anomalies in IoT networks. These works highlight the importance of using AI techniques to develop IDS in IoT networks to ensure their security.

3. METHODOLOGY

3.1 INTRODUCTION

The proposed network intrusion detection system presented in this study aims to address the challenges of identifying network intrusions. By utilizing four different methods, the system offers a comprehensive approach that can be tailored to specific data types and requirements. The first method, the native CNN with LSTM cells, is designed to predict future network traffic by training on current data. This method is effective in identifying patterns and anomalies in network traffic and can be utilized in real-time monitoring of network activity. The second method, the CNN-AE, combines CNN and Autoencoder to perform feature extraction and dimensionality reduction on the network traffic data. This method is especially useful when dealing with large amounts of data and can improve the efficiency of the network intrusion detection process. The third method, the RNN with LSTM cells, is designed to handle sequential data and is ideal for detecting network intrusions where the order of network packets is important. The RNN-LSTM model can capture patterns in the network traffic data over time and retain long-term memory, improving the accuracy of network intrusion detection. The fourth method, the CNN-RNN, combines both CNN and RNN for feature extraction and classification. The CNN extracts features from the network traffic data, while the RNN captures patterns in the data over time. The results from both networks are then combined to make the final classification decision, improving the overall accuracy of the network intrusion detection system.

Elaborate further, the proposed system aims to address the limitations of existing network intrusion detection methods by combining multiple approaches into one comprehensive system. The four methods that the proposed system utilizes - native CNN with LSTM, CNN-AE, RNN-LSTM, and CNN-RNN - offer different ways to detect and classify network intrusions, providing a more robust solution for different types of data and requirements. The CNN-AE and CNN-RNN methods will be evaluated and compared to the native CNN method to determine their effectiveness in detecting network intrusions. This evaluation will involve training and testing the models on a large dataset of network traffic data and comparing their accuracy. The results of the evaluation will provide insight into the performance of the different methods and help determine the most suitable method for the specific data and requirements of the network intrusion detection system.

Ultimately, the choice of method will depend on various factors, such as the amount and type of data available for training, the complexity of the network infrastructure, and the specific requirements of the network intrusion detection system. The proposed system offers a flexible and adaptable solution that can be customized to meet the specific needs of a particular network, making it a valuable tool for network security professionals.

3.2 INITIAL STEPS OF RESEARCH PROCESS

Here is a flowchart that illustrates the steps involved in an anomaly-based intrusion detection system (IDS) using machine learning algorithms:

a. Collect and Preprocess Data: The first step is to collect network traffic data from various sources, such as log files, packet captures, or network devices. The collected data is then preprocessed to clean, normalize, and filter the data to remove irrelevant or noisy information.
b. The next step involves extracting the features from the pre-processed data. This process can be done through various techniques such as feature engineering, feature scaling, and dimensionality reduction.
c. Split Data: After the features have been extracted, the next step is to divide the data into two sets.
d. Model Training: The system will then train an anomaly detection model using different machine learning techniques such as supervised or unsupervised learning. The model will learn the normal patterns of network activity by analyzing the features of the data.
e. Deployment and Monitoring: Once the model is trained, it is then deployed in the network and used to monitor live traffic in real-time. The model will analyze the network traffic and detect any patterns or behaviors that deviate from normal or expected behavior.
f. Anomaly Detection and Alert Generation: The system will generate a warning notification whenever an anomaly is sensed and could be sent to a security analyst for further investigation.
g. Deployment: Once the model is trained and evaluated, it is then deployed in the network to monitor live traffic in real-time. One can use different libraries such as scikit-learn, Tensorflow, or Keras to implement the model

h. Adaptation: The system will continuously learn from new data, and update its model to improve its performance over time. This will help to detect unknown or emerging intrusions, which are not represented in the initial training dataset.

i. Evaluation: Regular evaluation is important for the system's performance, based on metrics. By conducting such a comparison, it is possible to evaluate the proposed system in relation to other existing methods and gain a deeper understanding of its advantages and limitations.

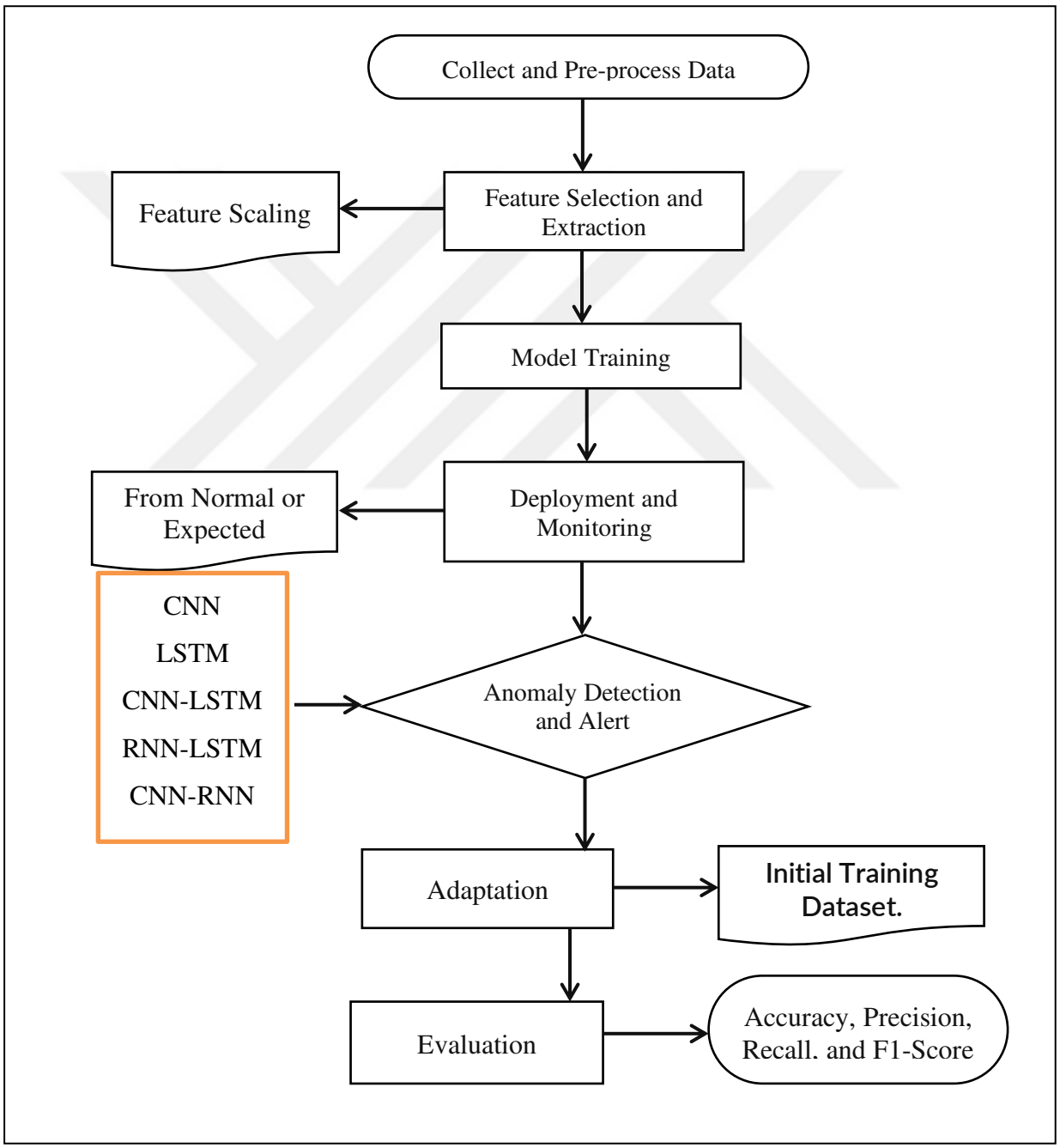

Figure 3.1: Proposal Work Flowchart.

3.3 THESIS APPROACH

The proposed Network Intrusion Detection approach involves the following steps as showing the Figure:

a. Gathering the Network Intrusion Detection dataset.
b. Pre-processing the input data using various methods to ensure accuracy.
c. Implementing the four machine learning models.
d. Training the models using the training subset.
e. Predicting the Network intrusion class using the testing subset.

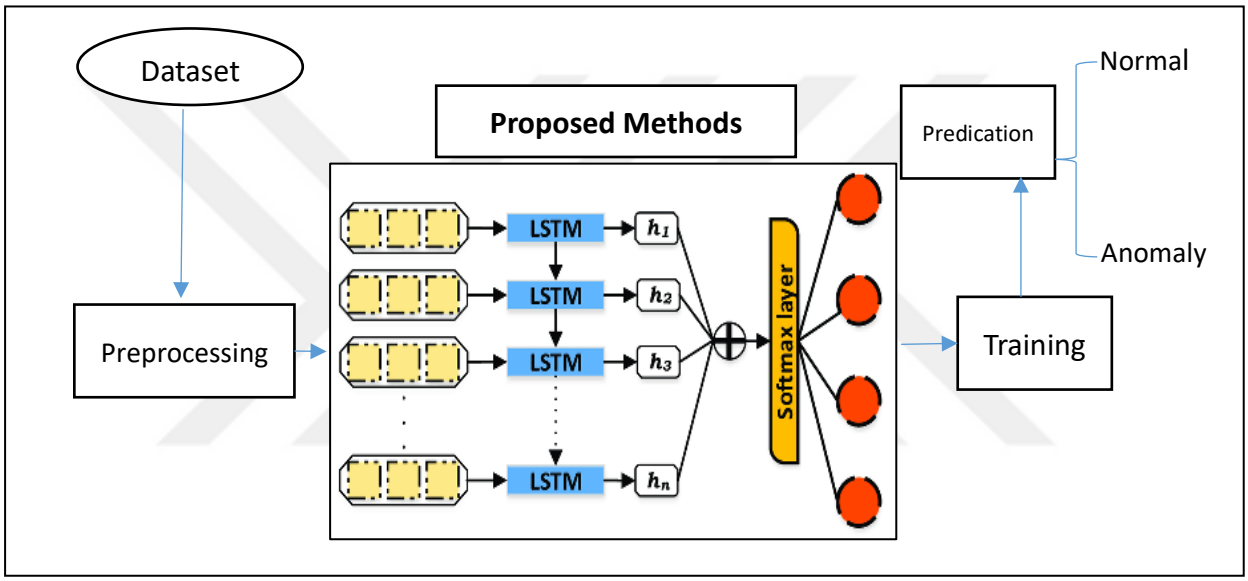

Figure 3.2: Proposed Network Instruction Classification Approach.

3.3.1 Dataset

The dataset for network intrusion detection offered comprises a number of intrusions that were simulated in a network environment for use by the military. It contains raw TCP/IP data for a network that was simulated to seem like a normal LAN used by the US Air Force and was attacked several times. Each connection in the dataset consists of TCP packets consisting a series flowing between two endpoints in the form of source and destination IP address and is either classified as regular or an attack with a particular attack type. There are a total of 41 variables for each connection, including both quantitative and qualitative information like duration, protocol type, service, and flag. The class variable is divided into two categories: typical and abnormal.

Table 3.1: Dataset Features Description.

Feature Name	Description
Duration	This is the length of time that a connection was active, measured in seconds.
Protocol type	This refers to the type of protocol used for the connection, such as TCP, UDP, or ICMP.
Service	This refers to the type of service being provided by the connection, such as HTTP, FTP, or SSH.
Flag	This is a binary feature that indicates whether a connection was "normal" or "abnormal."
Src bytes	Bytes transmitted from the originating IP address are represented by this number.
Dst bytes	Bytes delivered to the target IP address total this amount.
Land	This is a binary feature that indicates whether the source and destination IP addresses are the same.
Wrong fragment	This is a binary feature that indicates whether there was a problem with the fragment offset in the connection.
Urgent	This is a binary feature that indicates whether the urgent pointer field was used in the connection.
Hot	This is a binary feature that indicates whether the connection is considered "hot," meaning that it may be part of an attack.

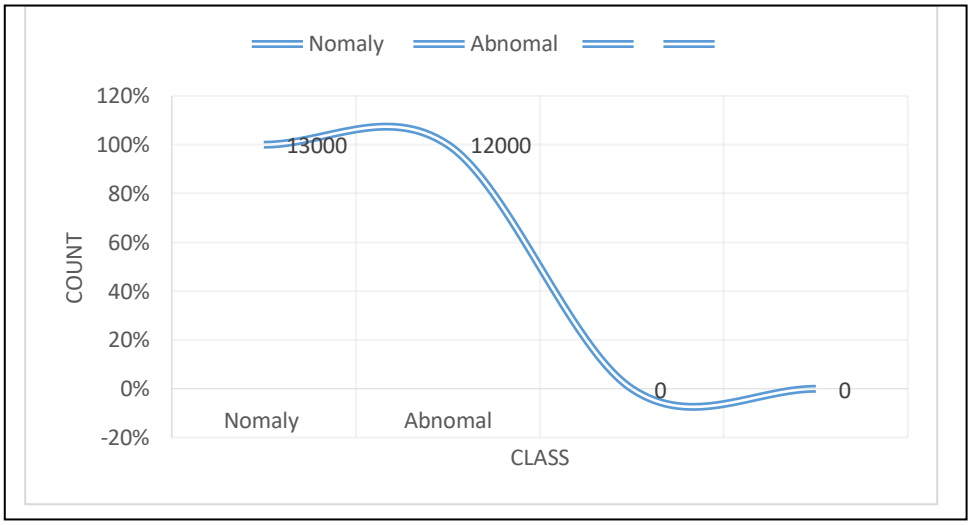

Figure 3.3: Dataset Class Labels Distribution.

3.3.2 Pre-Processing

The pre-processing stage plays a crucial role in preparing real-world data for machine learning (ML) and deep learning (DL) methods. Raw data is often plagued with inconsistencies, errors, and incompleteness, making it challenging for machines to handle. Pre-processing helps to transform the data into a format that is appropriate for these algorithms and can also decrease the processing power and training time required to achieve reliable and precise outcomes.

In this particular case, the data is tabular and stored in a Data Frame. The pre-processing steps include checking for null or missing values and duplicates, extracting the labels of the data samples to identify the class categories of Network Instruction types (normal, anomaly), applying label encoding and data normalization techniques to standardize the dataset, and using the RF approach to extract accurate features (as shown in Figure 3.4).

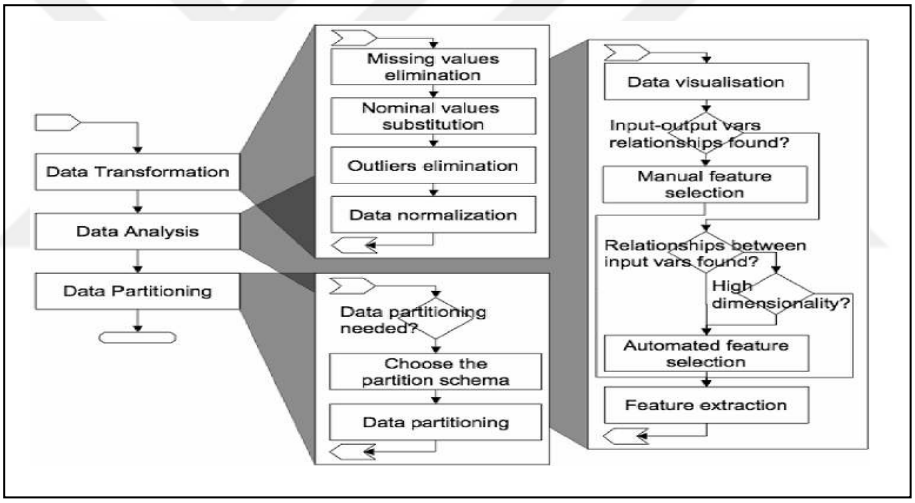

Figure 3.4: Pre-Processing Steps.

3.4 HANDLE NULL/MISSING VALUES AND DUPLICATE DATA

There are no empty or missing values in our dataset and does not have any redundant data, as shown in Figure 3.5. However, additional preprocessing is still required, such as feature extraction, label encoding, and data normalization.

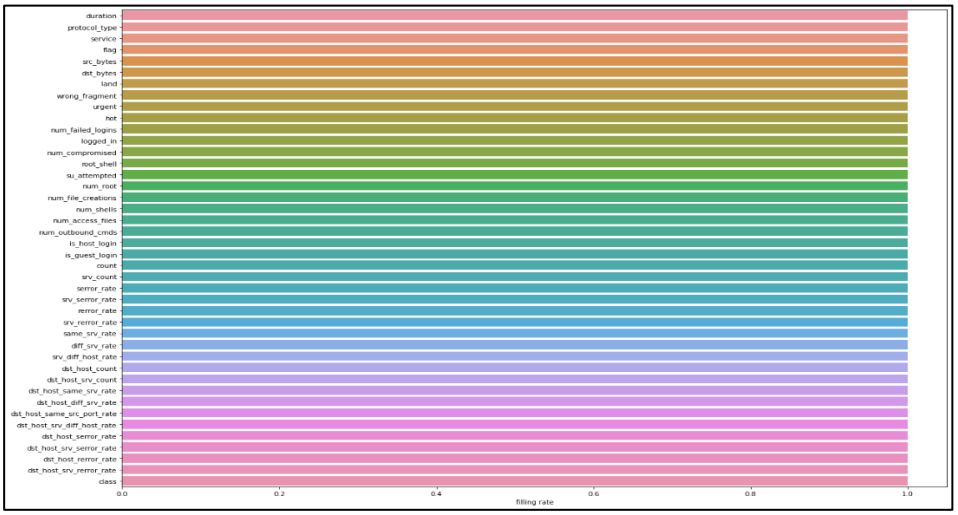

Figure 3.5: Shows the Null and Missing Value.

3.5 LABEL ENCODING

Categorical data, or data that can be divided into categories, cannot be processed directly by machine learning algorithms. These algorithms require all input and output variables to be numeric, so it is necessary to transform categorical data into a numerical format. One way to do this is through label encoding, which assigns a unique numerical value to each category. In this case, the dataset has two labels that need to be encoded in order to be fed into the model. The label encoding method will be used to convert these labels from their original categorical form (normal and anomaly) into numeric values (0 and 1, respectively).

protocol_type	service	flag	src_bytes	dst_bytes	]
tcp	ftp_data	SF	491	0	
udp	other	SF	146	0	
tcp	private	S0	0	0	
tcp	http	SF	232	8153	
tcp	http	SF	199	420	

protocol_type	service	flag	src_bytes	dst_bytes
1	19	9	491	0
2	41	9	146	0
1	46	5	0	0
1	22	9	232	8153
1	22	9	199	420

Figure 3.6: Features Encoding.

class
normal
normal
anomaly
normal
normal

class
1
1
0
1
1

Figure 3.7: Class Labels Encoding.

3.5.1 Data Standardization

Data standardization is a process that transforms the values of a dataset into a standard format, typically with a mean of 0 and a variance of 1. Due to the fact that many machine learning algorithms demand uniform input data, this is frequently done to enhance their performance. The Python scikit-learn library provides a StandardScaler() function that can be used to standardize the values in a dataset. This function scales the features of the dataset so that they are all on the same unit scale (mean = 0 and variance = 1).

3.5.2 Feature Extraction

It relies on modeling the remaining characteristics after repeatedly deleting the undesirable ones. It uses the model provided (in this case, a RF Classifier) to assign weights to each feature, and then selects the features with the highest weights.

In the code in Figure, the RFE object is first initialized with the for exampe Random Forest Classifier ‘rfc’ and the number of features to select ‘n_features_to_select=10’. Then, the RFE object is fit to the training data and labels.

After fitting the RFE object, the list of selected features can be obtained by mapping the boolean values of the ‘get_support()’ method to the feature names in ‘train.columns’. The resulting list is stored in the variable ‘feature_map’, and the selected features are extracted and stored in the list ‘selected_features’. Then we will use the ‘selected_features’ list to train the proposed models using only the selected features. The list of selected features can be seen in Figure 3.8.

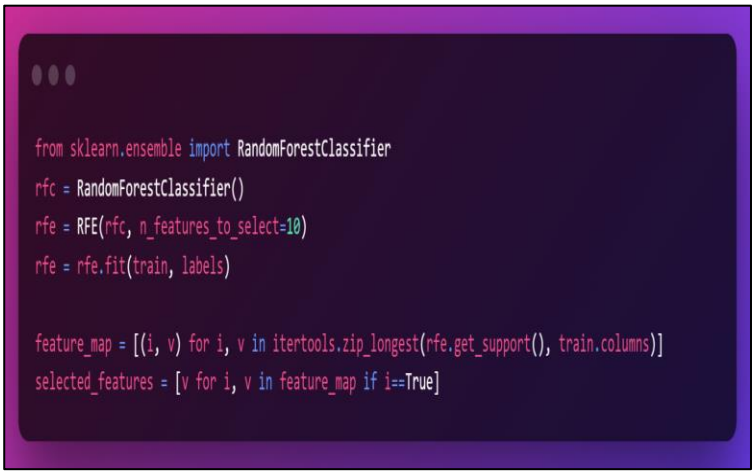

Figure 3.8: Feature Selection.

	0
0	protocol_type
1	flag
2	src_bytes
3	dst_bytes
4	logged_in
5	count
6	same_srv_rate
7	diff_srv_rate
8	dst_host_srv_count
9	dst_host_same_srv_rate

Figure 3.9: The List of Selected Features.

3.5.3 Libraries

In this research, we utilized several useful libraries in Python, including Numpy, Matplotlib, Seaborn, Keras, and Pandas. Numpy enables numerical calculations, Matplotlib is a library for statistical graphics plotting, Keras allows for deep learning and machine learning, and Pandas is a tool for data analysis and manipulation. These libraries provide valuable resources for handling data and implementing machine learning algorithms in Python.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Figure 3.10: The Used Libraries

3.5.4 Training

The training and testing datasets were separated using the scikit-learn library's train_test_split tool. This function requires the input of features and labels for the dataset (X_train and Y_train respectively) and a train size parameter to indicate the proportion of the data that is designated for training. The test set is calculated automatically based on the

train size. In this scenario, 70% of the data is utilized for training and the remaining 30% is used for testing. Also, the random state parameter is used to specify the random seed to use for the data splitting, which ensures that the same split is obtained each time the code runs.
For building this model, we selected Adam as the optimizer. This adaptive learning rate approach modifies the learning rate for each weight in the neural network based on predictions of the first and second moments of the gradient. We also set the model loss to Categorical Crossentropy, as the dataset has 3 pre-defined classes and this type of loss is suitable for multi-class classification. Additionally, we set the model metrics to 'accuracy' to evaluate the performance of the DNN and calculate the accuracy score using the train and test sets.
In order to train the model, the training set is fed into the model in the fit stage. After receiving the training data, the model weights are iteratively adjusted by the model to have a loss function with minimum values. Once the model is trained, it can be used to make predictions on new, unseen data. The predict step involves passing the new data through the model and using the trained model to generate predictions.

Figure 3.11: Compile parameters.

3.6 BUILD THE MODELS

The framework for network intrusion detection includes four main methods. The first one uses a Convolutional Neural network to estimate and make the predication of the network packet. It is commonly utilizing as a starting point for researchers to develop their own method.

This process can involve performing various pre-processing tasks such as normalization, data transformation, and dropping null values. After the CNN model has been constructed, the training and testing phases are conducted.

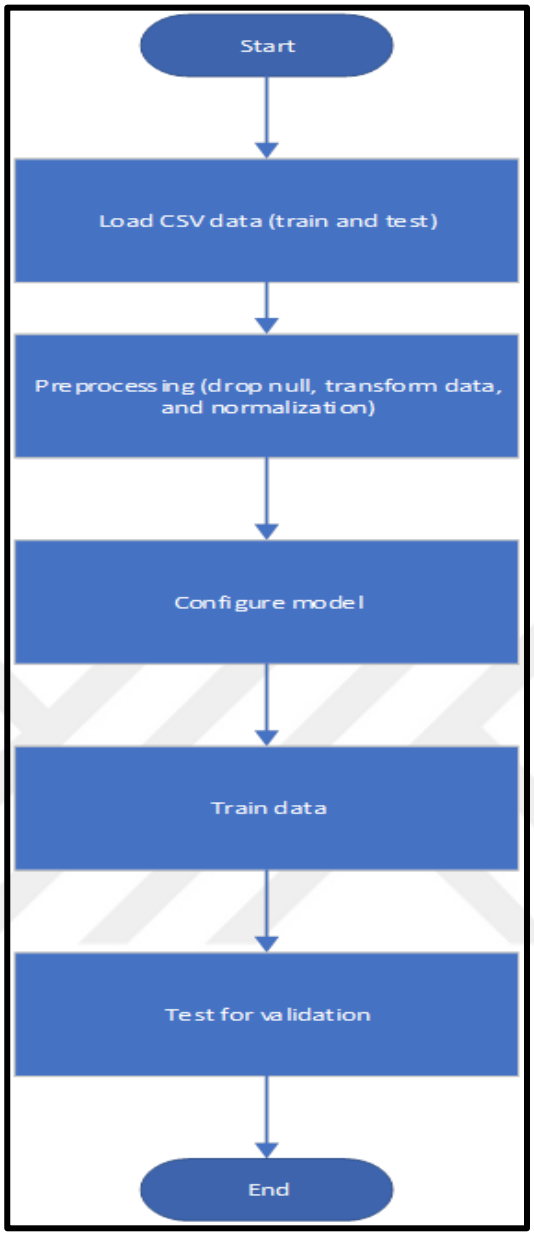

Figure 3.12: The Hierarchy of the Proposed System.

3.6.1 Conventional Neural Network (CNN)

A CNN is a type of deep learning technique that aims to identify various objects and elements in an image. It is based on the connections between the neurons in the brain.

CNN's classification process involves several steps, such as Convolution, Full Connection, and Max-Pooling. Convolution is used in the extraction of features and the sizing of the image. The other steps help in further processing the data. The outcome of the Convolution step is represented as follows:

$$Xi = f\,(wi \otimes Xi{-}1 + bi) \tag{3.1}$$

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t-\tau)_{d\tau} \tag{3.2}$$

Xi denotes the outcome of kernel i, and f(x) refers to the activation function.

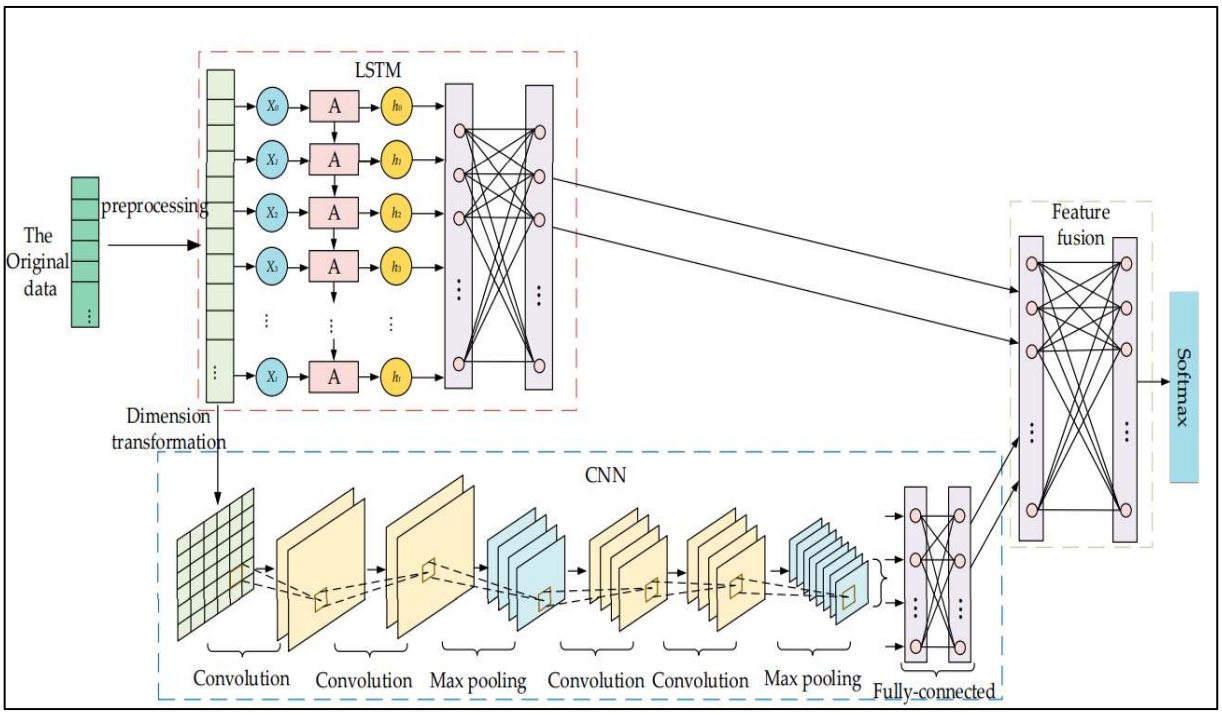

Figure 3.13: The CNN-LSTM Intrusion Model is a Cross-Layer Feature Fusion.

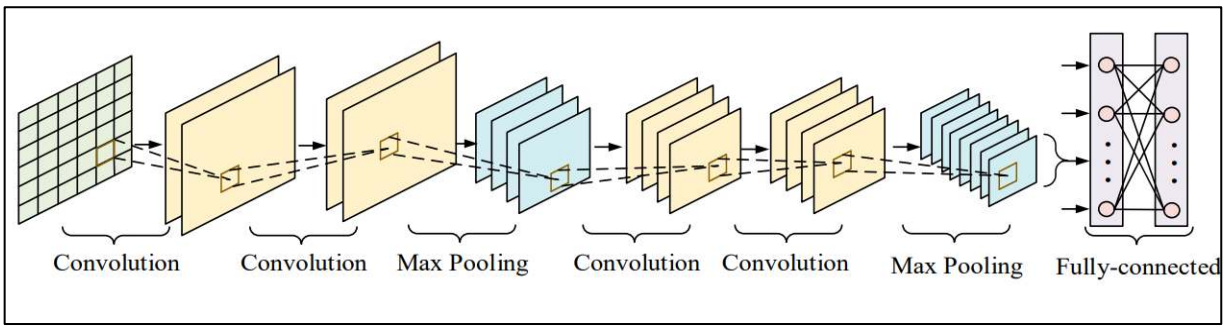

Figure 3.14: The Layers of CNN

The Convolutional kernel systematically examines the input data to extract important features. The Convolutional layer employs the ReLU activation function for its simplicity

$$ReLU(X_i) = \begin{cases} X_i & (X_i > 0) \\ 0 & (X_i \leq 0) \end{cases}, \tag{3.3}$$

in differentiation and its ability to accelerate the training of the model, while preventing gradient vanishing. The ReLU function can be written as follows:

He role of the pooling layer in Convolutional Neural Networks (CNNs) is crucial, as it serves to establish invariance while reducing the network's complexity by eliminating unnecessary information through down sampling. Pooling can be performed using two primary methods: average pooling and max pooling. The former involves calculating the average value in the pooling region, while the latter selects the maximum value as the pooling result. In this study, we prefer max pooling over average pooling since it retains more critical information. The max pooling function can be mathematically expressed as follows:

$$Q_j = Max(P_j^0, P_j^1, P_j^2, P_j^3 \dots P_j^t), \tag{3.4}$$

Obtain the output of the pooling region, denoted as Qj, the element t, Ptj, within the region j undergoes the max pooling operation (Max).

In the Convolutional Neural Network (CNN), the fully connected (FC) layers serve as the "classifiers". Their primary role is to weigh the features extracted from the convolutional and pooling layers, and remap them to the sample-marker space. To prevent overfitting, a dropout operation is implemented in the FC layer to randomly eliminate neurons.

The data enters the network through layer1, with input layers 43. It then undergoes filtering using layers of convolutional and also the data compression

During and through training phase, the model will classify and validate data using the aforementioned layers in each of the 100 epochs.

Table 3.2: CNN Model Structure.

Layer No.	Layer Type	Parameters
1	Convolution1D	Filter=64 Kernel=3 ActivationFunction=Relu input_shape=43,1
2	Convolution1D	Filter=64 Kernel=3 ActivationFunction=Relu
3	MaxPooling1D	PoolSize=2
4	Convolution1D	Filter=64 Kernel=3 ActivationFunction=Relu
5	Convolution1D	Filter=64 Kernel=3 ActivationFunction=Relu
6	Convolution1D	Filter=64 Kernel=3 ActivationFunction=Relu
7	MaxPooling1D	PoolSize=2
8	LSTM	OutputSize=70
9	Dropout	Rate=0.2
10	Dense	Units=1 ActivationFunctions=Sigmoid

The following code shows the implementation of the CNN Auto Encoder function.

```
def sparse_regularizer(activation_matrix):
    p = 0.01
    beta = 3
    p_hat = K_mean(activation_matrix)
    KL_divergence = p*(K.log(p/p_hat)) + (1-p) *(K.log(1-p/1-p_hat))
    sum = K.sum (KL_divergence)
    return beta * sum
```

Figure 3.15: The Algorithm of the Proposed.

The crucial component of the model is the beta parameter, which is calculated by processing the activation matrix with a specific function. To enhance the model's performance, the BBO algorithm is used to optimize the optimization algorithm instead of the commonly used "adam" optimization method. The BBO is a biogeography-based optimization technique that leverages the idea of species migration to develop an algorithm for solving optimization problems.

The following pseudo code provides a step-by-step explanation of the BBO algorithm's operations, including the input of the activation matrix, the beta parameter calculation, and the optimization of the optimization algorithm using the BBO method. This code serves as a guide to understanding the workings of the BBO algorithm and its ability to solve optimization problems.

```
-Initialize a population of N candidate solutions {xk} of an n-dimensional function
-While not exceeding the maximum number of iterations
    For each x_k, set emigration probability μ_k proportional to fitness of x_k, with μ_k ∈ [0,1]
    For each x_k, set immigration probability λ_k = 1- μ_k
    {z_k} ← {x_k}
    For each individual z_k (k = 1,..., N)
            For each independent variable index s ∈ [1,n]
                Use λ_k to probabilistically decide whether to immigrate to z_k
                If immigrating then
                    Use {μ_i} to probabilistically select the emigrating individual x_j
                  z_k(s) ← x_j(s)
                End if
            Next independent variable index: s ← s+1
            Probabilistically mutate z_k with the percentage of m
    Next individual: k ← k+1
    {x_k} ← {z_k}
-Next iteration
```

Figure 3.16: BBO Optimization Algorithm.

3.6.1.1 Convolutional layer

The multiplication and summation of overlapping parts of input in every shift with the aid of a filter constitutes convolutional operation that forms the convolution layers' basis. In its simplest form, the convolution of 3x4 input with 2x2 kernel is depicted in Figure 3.17. Each overlapped cell is multiplied, and the regions that contain the multiplications' results are then combined. Equation 3.5 demonstrates how to determine the output size of a convolution operation on an image. W stands for the input image's single dimension, F for the filter's size, P for the size of the padding, and S for stride that denotes cells the filter is moved for every one of the convolution steps. Assuming to have a 7x7 filter with a padding of 0 and stride 1 as is typical for the CNN models and a 224 × 224 size image. As a result, the size regarding the output image after convolution is computed to be 218 × 218.

$$\text{Size of the Output} = (\mathrm{W} - \mathrm{F} + 2\,\mathrm{P})/\mathrm{S} + 1 \tag{3.5}$$

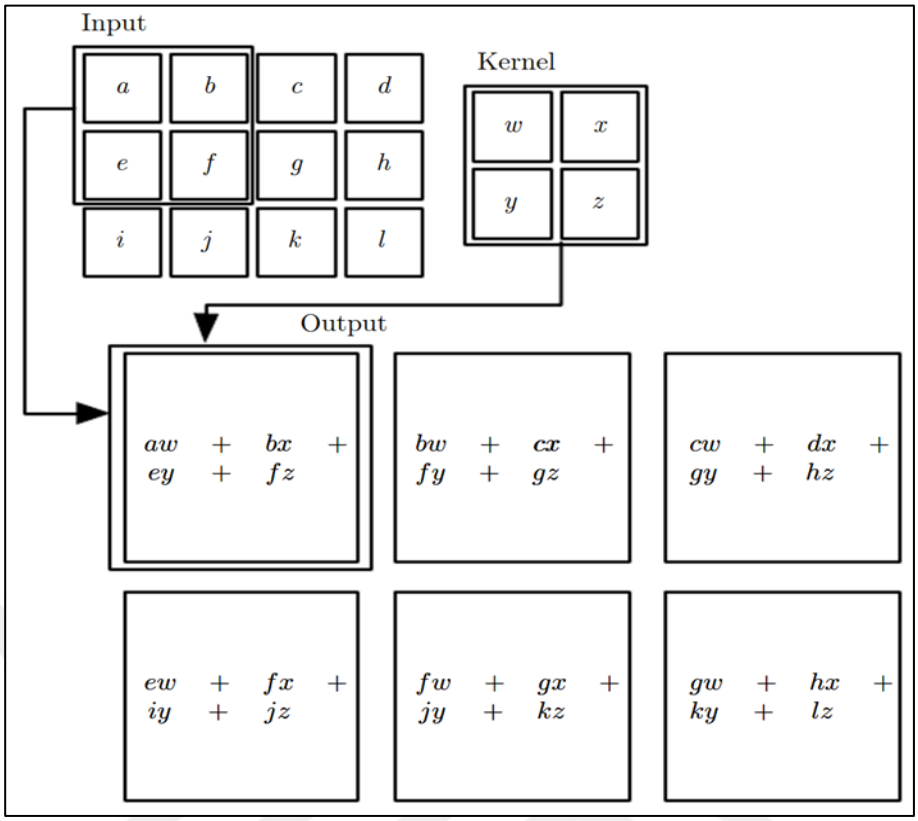

Figure 3.17: Comprehensive Explanation of the Convolutional Process.

In CNNs, a 3D image is frequently an acceptable input for the network design. Convolution is used to apply a series of filters over the height, width, and depth regarding such input images in the convolutional layers. Furthermore, each filtering step produces an output known as a feature map. Convolutional layers allow us to learn feature maps from input images that correlate to visual-based features. Each CNN model's first layers learn more fundamental features including basic shapes, edges, and corners. Convolutional layers also enable us to learn additional task-specific features throughout CNN model training in each subsequent layer and iteration.

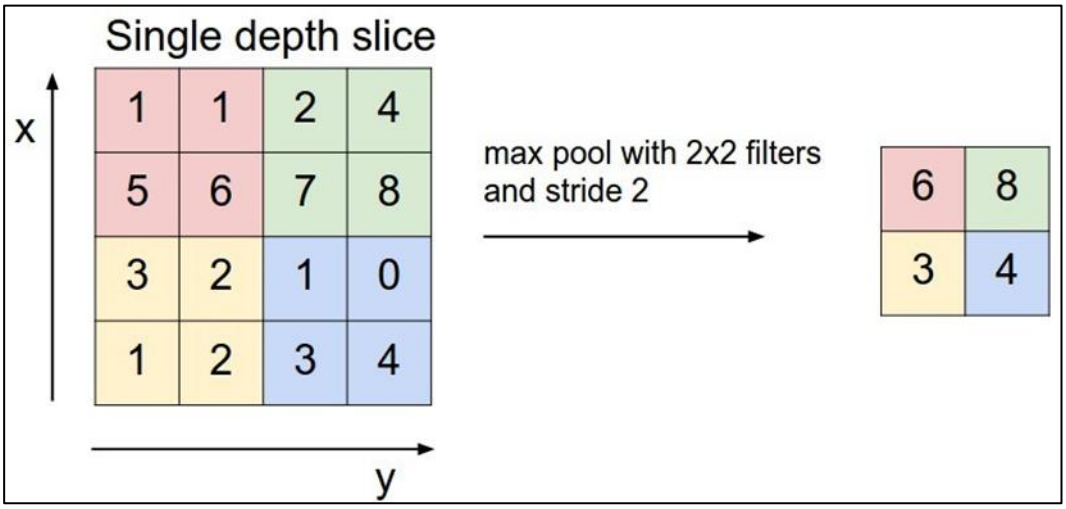

Figure 3.18: A Sample of the Operation of the Max Pooling.

3.6.1.2 Pooling layer

In order to lower the spatial size (heightو weight) regarding the convolutional layer outputs, reduce problem of the parameters, and prevent the issue of the over-fitting—a critical issue in the deep CNN training—pooling layers are typically utilized. This layer, which depicts an image in a lower dimensionality, is also known as a sub-sampling layer. The output is downsampled using the max pooling, one of the pooling techniques. In a straightforward explanation, max pooling chooses a pixel that has the highest value and moves it to the following layer, as illustrated in Figure 3.18. As a result, we apply the max pooling filter over an input using a 2 × 2 size filter. Average pooling is the additional pooling method. The average pooling determines average value regarding filtered area, in contrast to max pooling. Assuming, for instance, that the average pooling operation was done to the input image shown in Figure 2.3, the filtered image values have been, in accordance with the following equations (2.3, 2.4, 2.5, 2.6), respectively; 3 − 5 − 2 − 2.

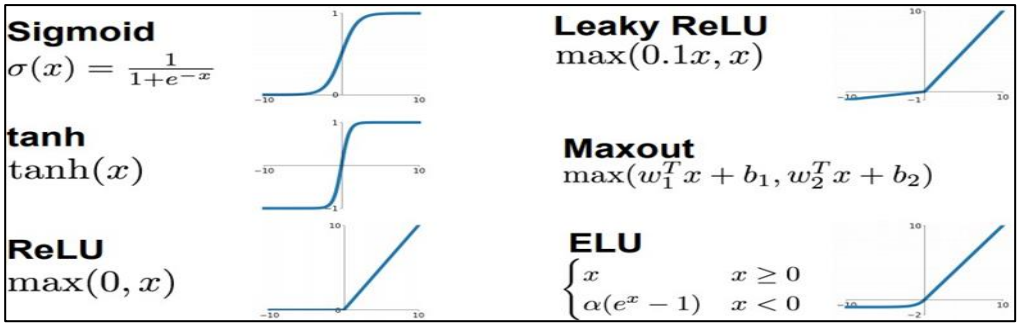

Figure 3.19: Representations of the Commonly Utilized Activation Functions.

$$p\,(1,1) = (1+1+5+6)/4 \tag{3.6}$$

$$p\,(1,2) = (2+4+7+8)/4 \tag{3.7}$$

$$p\,(2,1) = (3+2+1+2)/4 \tag{3.8}$$

$$p\,(2,2) = (1+0+3+4)/4 \tag{3.9}$$

3.6.2 Long Short-Term Memory Algorithm (LSTM)

Recurrent Neural Networks (RNNs) are a common model used to train time-series data, but they can be difficult to train due to issues with gradients vanishing or exploding. To address these problems, Long Short-Term Memory (LSTM) units, which have memory functions, are used to replace the hidden units in RNNs.

LSTMs can retain information over long periods of time because they have slow weight changes, and they can also activate short-term memory for brief periods. Figure 3.17 shows the LSTM structure used in the current study, where the core information is conveyed along the horizontal axis. The LSTM structure uses three gate structures, namely the forget gate, input gate, and output gate, to discard old information and acquire new information. This enables the LSTM to possess both long-term and short-term memory capabilities, making it an effective model for training time-series data.

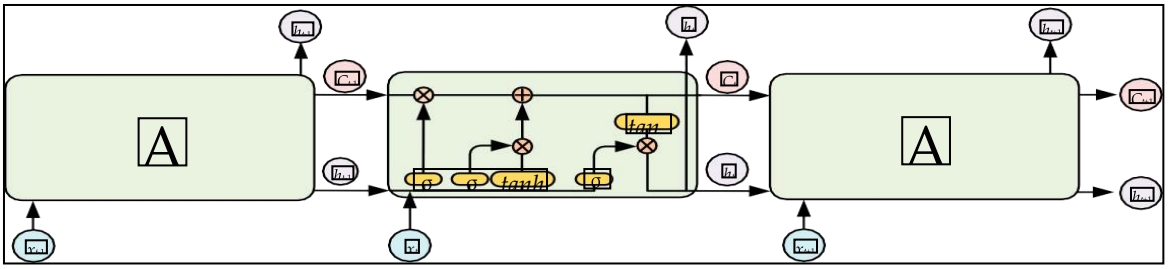

Figure 3.20: Long Short-Term Memory Algorithm.

In the LSTM structure, the forget gate is responsible for determining the amount of information that needs to be forgotten. The inputs to the forget gate are the previous hidden state, ht-1, and the current input, xt. The output of the forget gate is a value between 0 and 1, which represents the current cell state, Ct-1.

A value of 1 indicates that all the information in the current cell state should be retained, while a value of 0 indicates that all the information should be completely forgotten. The forget gate is represented by an expression that calculates the value of the output based on the inputs.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \tag{3.10}$$

The input gate in a Long Short-Term Memory (LSTM) cell is responsible for deciding how much new information to add to the current cell state based on the current input, xt, and the previous cell state, ht-1. The gate is composed of two main steps: the first step involves using a sigmoid function to determine which parts of the current input should be updated, while the second step uses a tanh function to calculate the new candidate values that will be added to the current cell state. The input gate has its own weight and bias parameters, which are represented by Wf and bf, respectively. By controlling the amount of new information that enters the cell state, the input gate helps regulate the flow of information through the LSTM cell.

$$it = \sigma(Wi \cdot [h_{t-1}, x_t] + bi), \tag{3.11}$$

$$\widetilde{C}t = tanh(Wc \cdot [h_{t-1}, x_t] + bc). \tag{3.12}$$

$$ot = \sigma(Wo[h_{t-1}, x_t] + b_o), \quad ht = ot \times tanh(C_t). \tag{3.13}$$

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t, \tag{3.14}$$

$$i_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_i\right) \tag{3.15}$$

$$\hat{C}_t = tanh(W_c . [h_{t-1}, x_t] + b_c) \tag{3.16}$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{3.17}$$

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o) \tag{3.18}$$

$$h_t = o_t * tanh(C_t) \tag{3.19}$$

In the context of LSTM (Long Short-Term Memory) neural networks, a gate refers to a mathematical function that controls the flow of information within the network. Specifically, an LSTM gate is a nonlinear transformation that takes the input from previous time steps and combines it with the current input to produce an output for the current time step. LSTM networks use three different gates to control the flow of information: the input gate, the forget gate, and the output gate. The input gate determines which new information to add to the cell state by using a sigmoid activation function. The forget gate decides which information from the previous cell state to keep or discard using a sigmoid activation function. Finally, the output gate decides which information to output from the cell state by using a hyperbolic tangent activation function. By using these gates, LSTM networks can selectively retain and discard information from previous time steps, allowing them to capture long-term dependencies in sequential data such as natural language processing, speech recognition, and time series analysis. The applicant LSTM model, as depicted in Figure 3.21, follows a particular outline. Initially, the model employs CNN layers to eliminate the prominent features from the training dataset.

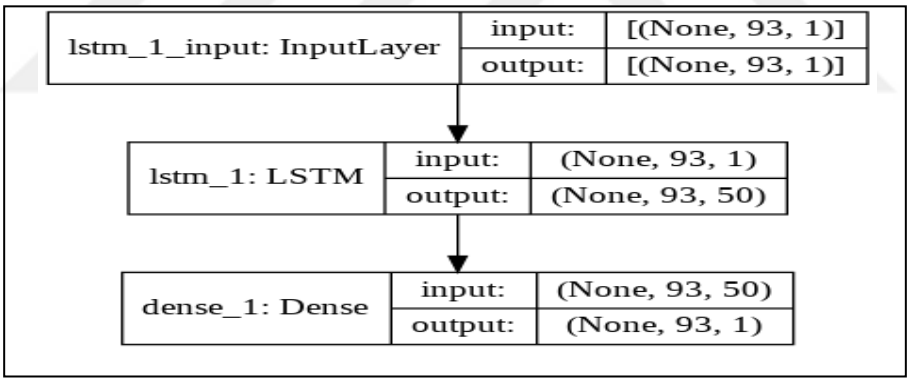

Figure 3.21:Indicts of Layers of LSTM.

3.6.3 Ensemble Based Bbo Method

To explore the potential of the BBO (Bee Algorithm) algorithm in various methods final method has been proposed that applies. This method involves using a decision tree classifier in combination with KNN (K-Nearest Neighbors) and logistic regression to perform machine learning on the data. Prior to processing, the data must be cleaned by removing any null values, normalized, and transformed. The BBO algorithm is then utilized for this study.

Ensemble learning with the voting technique consists of creating multiple models, each of which makes its own prediction. The final prediction is determined by selecting the prediction with the highest accuracy. This approach seeks to increase the prediction's accuracy by combining the strengths of multiple models.

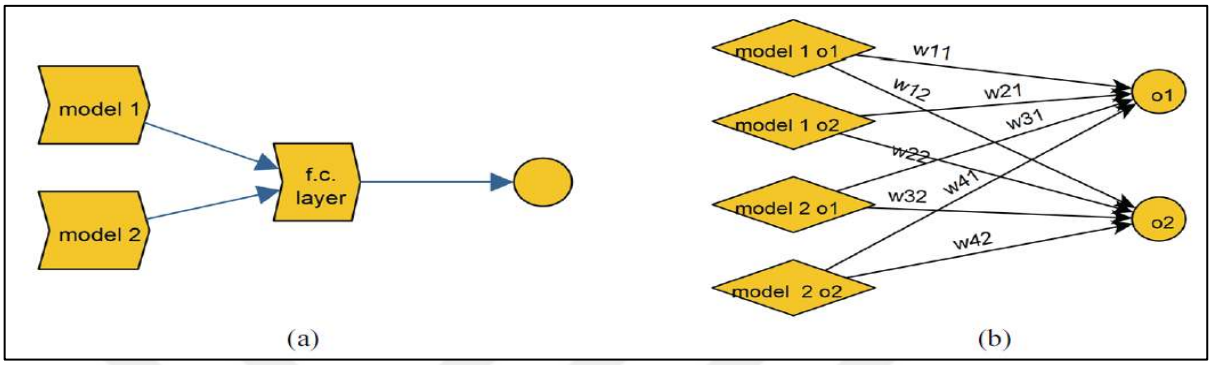

Figure 3.22: Ensemble Voting Based BBO.

The following flowchart explains how it works:

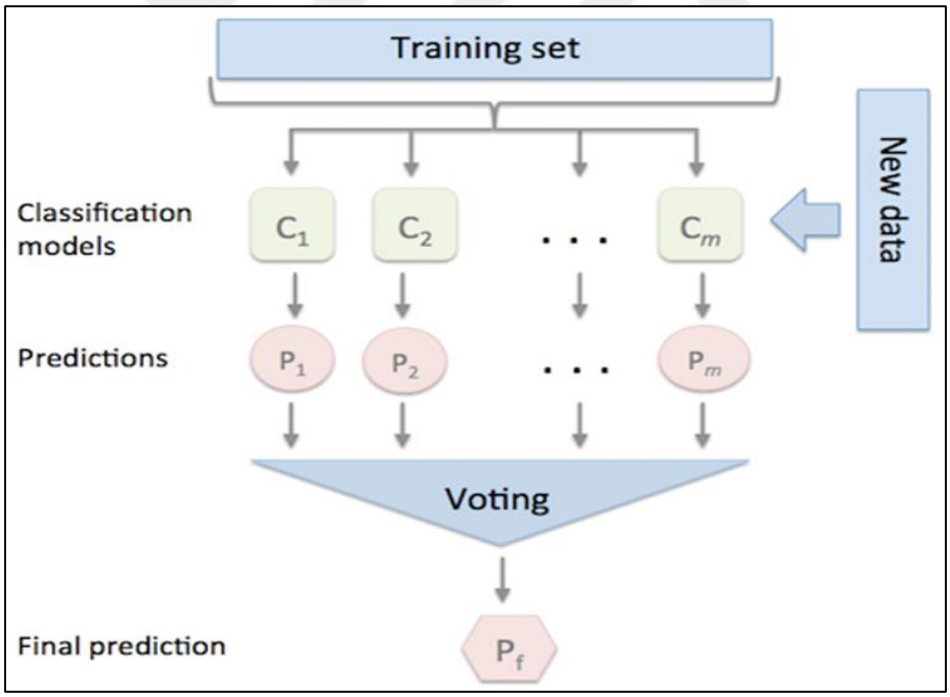

Figure 3.23: Flowchart of Ensemble Learning Voting.

3.7 ANOMALY DETECTION AND ALERT GENERATION

The final step in preparing the system of the proposed method which is anomaly detection and alert generation and as shown in the figure 3.24, the code pre-process the data, trains

the model and then use the model to predict anomalies in the data. It then generates alerts for any data points that are flagged as anomalies.

a. To detect anomalies in new network traffic data, the trained model can be utilized. The model compares the incoming network traffic data with the patterns of normal network activity it learned during training. Any network traffic that shows a significant deviation from the normal patterns is identified as an anomaly.
b. Thresholding: for removing the number of false alarms, a threshold value can be set to flag only the most severe anomalies. This threshold can be set based on the results obtained from the evaluation step.
c. Alert Generation: When an anomaly is detected, an alert is generated and sent to a security analyst for further investigation.
d. The alert will typically include information such as the time of the anomaly, and the type of anomaly detected.
e. Investigation: When an alert is triggered, the security analyst will investigate it to determine if it's a genuine intrusion or a false positive. If it is found to be an actual intrusion, the security analyst will take appropriate action to mitigate the threat.
f. Continuous monitoring: Once the system is deployed, it should continuously monitor the network traffic for anomalies and generate alerts as necessary. This will help to detect unknown or emerging intrusions that were not represented in the initial training dataset.
g. It's important to note that this process should be integrated with other security layers (i.e. firewalls, intrusion prevention systems, etc.) to enhance the security of the network.

```
# load and preprocess the data
from sklearn.preprocessing import StandardScaler
data = load_and_preprocess_data()
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# train the anomaly detection model
from sklearn.svm import OneClassSVM
model = OneClassSVM(nu=0.1, kernel='rbf', gamma=0.1)
model.fit(scaled_data)

# detect anomalies
anomalies = model.predict(scaled_data)

# generate alerts
alerts = []
for i, a in enumerate(anomalies):
    if a == -1:
        alerts.append(data[i])
```

Figure 3.24: Alert Generation of Anomaly Detection Commands.

3.8 DESIGN AND IMPLEMENTATION OF INTRUSION DETECTION SYSTEM

The first step in setting up the IDS system is to gather the necessary data for training and testing the machine learning models. This can be done by collecting network traffic data from various sources, such as network packet captures, system logs, and network flow records. Once the data is collected, it needs to be pre-processed to extract relevant features for use in the machine learning models.

Next, the machine learning models need to be trained using the pre-processed data. This involves selecting an appropriate machine learning algorithm and configuring its parameters, as well as splitting the data into training and validation sets. The models are then trained on the training data and evaluated using the validation data to assess their performance.

Once the machine learning models are trained and evaluated, they can be integrated into the IDS system. This involves deploying the models on a server or cloud-based platform and configuring them to receive live network traffic data. The models then analyze the incoming traffic in real-time and generate alerts for potential intrusions.

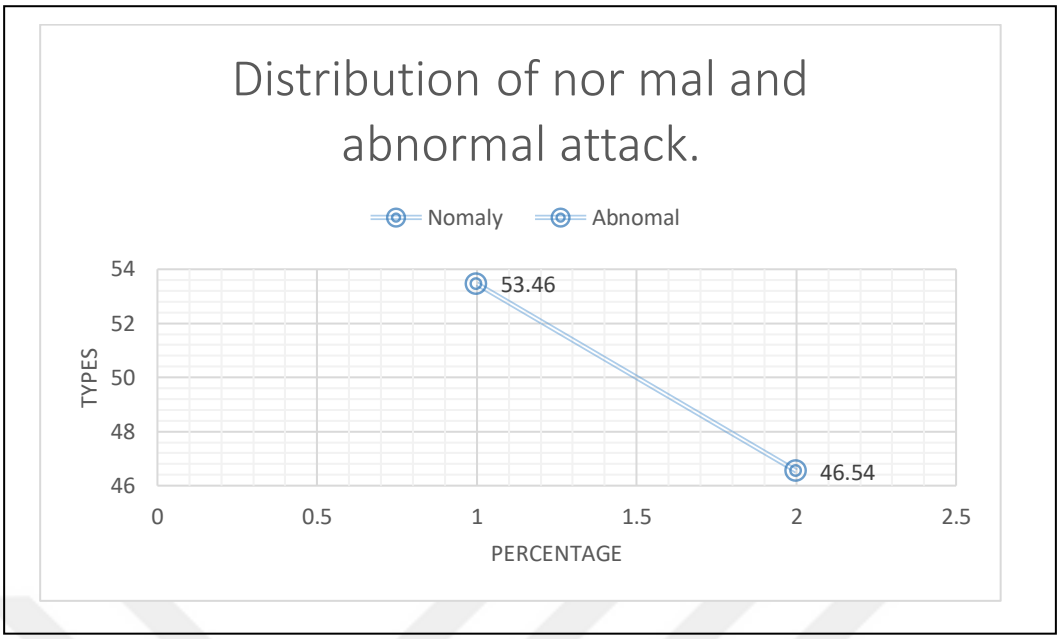

Figure 3.25: Attack of Abnormal Distributed.

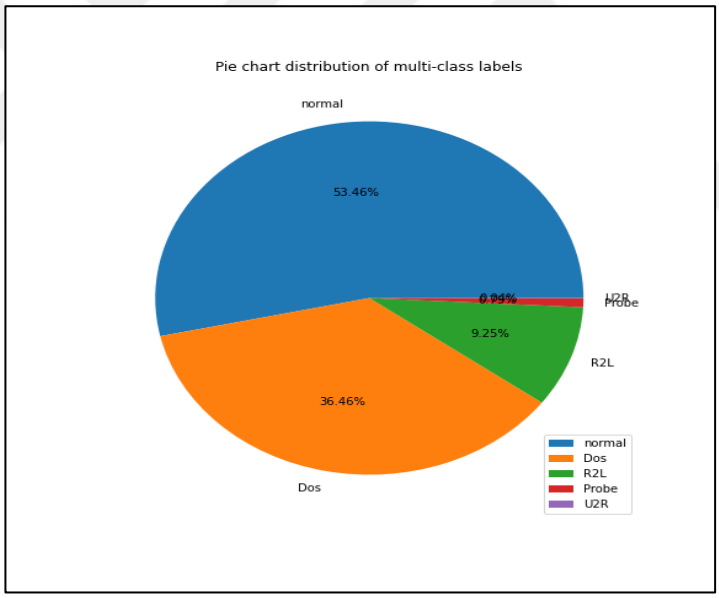

Figure 3.26: Percentage of Class Labels.

In order to implement the approach, another method involves selecting features based on their high weighting. This approach involves setting a feature weight threshold of 0.8 percent and selecting only seven features, which covers 97 percent of the total feature significance weight. The remaining 13 features account for only 3 percent of the total weight of importance. Figures 3.25 and 3.26 present a list of these selected features. To improve the accuracy and effectiveness of the IDS system, it may be necessary to fine-tune the machine learning models over time. This involves periodically re-training the models with

updated data and adjusting their parameters to optimize their performance. It may also involve incorporating feedback from security analysts to improve the accuracy of the alert generation and reduce false positives.

3.9 CHAPTER SUMMARY

This chapter provides a detailed explanation of the methodology used in the study, which involves the development of a Network Intrusion Detection system using four different methods. The intial step is the implementation of a CNN with LSTM to obtain the network in the future. The reason for starting with a native CNN is that it is widely used and serves as a benchmark for comparison with other proposals. Additionally, it provides a basic method that can be modified with new techniques to improve its accuracy.

To implement the CNN, the first step is to load and preprocess the data. This involves dropping any null values, normalizing the data, and transforming it to ensure that it is in the appropriate format. Next, a CNN model is built with multiple Convolutional1D layers and Maxpooling layers, which are used to train the preprocessed data. The model is then tested for validation to ensure its accuracy.

Overall, this approach provides a strong foundation for developing a IDS and can be built upon with more advanced techniques to further improve its performance.

4. PERFORMANCE EVALUATION

4.1 INTRODUCTION

This chapter describes a study that aimed to provide evidence supporting the hypothesis of the study and to demonstrate the effectiveness of the recommended training strategy. The study consisted of two independent series of trials. The first set of trials examined the impact of the suggested method's parameters, including the number of rounds and the number of epoch stages each cycle, on the algorithm's efficacy. Following this, contemporary CNN and LSTM models were developed and utilized to detect intrusion in IoT devices generating applications. These models were then trained using the suggested training technique, and a BBO optimized algorithm was used to further improve their performance. All of the research was conducted on a PC with an Intel® CoreTM i7-7700HQ CPU, 16 GB of RAM, and an 8 GB NVidia GTX1080Ti graphics processing unit (GPU). The GPU was used to accelerate the computations of the neural network. The implementation and training of the artificial neural network were accomplished using the Tensor Flow library, while the evaluation was carried out with the scikit-learn library. The initial evaluation utilized the handwritten digit dataset that is a component of the scikit-learn toolkit. This dataset served as a straightforward yet extremely challenging sample to investigate. Overall, this study demonstrates the effectiveness of the suggested training technique and the potential for improved performance of CNN and LSTM models in IoT intrusion detection applications.

4.2 METRICS EVALUATION

In this section, we will discuss the benchmarks used to measure the performance and effectiveness of our IDS (Intrusion Detection System) using AI and deep learning models.

4.2.1 Accuracy Classification

Accuracy is a common metric used to evaluate the performance of a classification model. It measures the percentage of correct predictions.

We can get the accuracy through this formula:

$$Accuracy = \frac{(Number\ of\ correct\ predictions)}{}(Total\ number\ of\ predictions) \qquad (1.4)$$

Let's take an example to better understand the formula. Suppose we have a dataset of 100 instances, with 70 instances belonging to class A and 30 instances belonging to class B. We

train a classification model on this dataset and then use it to make predictions on a test dataset of 20 instances. The model correctly predicts the class of 15 instances, with 10 belonging to class A and 5 belonging to class B. To calculate the accuracy of the model, we can use the formula as follows:

Accuracy = (Number of correct predictions) / (Total number of predictions) Accuracy = (15) / (20) Accuracy = 0.75

Therefore, the accuracy of the model is 75%, which means that it correctly predicted the class of 75% of the instances in the test dataset.

4.2.2 Performance Evaluation of the Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a machine learning classification model. It is a matrix of true and predicted labels, showing how many of the predicted labels match the true labels, and how many don't. It helps to evaluate the accuracy, precision, recall, and F1 score of a classification model.

A confusion matrix consists of four terms, which are True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

Actual Class | Positive | Negative | Positive | TP | FN | Negative | FP | TN |

Using the values in the confusion matrix, we can calculate several evaluation metrics such as accuracy, precision, recall, and F1 score. These metrics help us to assess the performance of the classification model and make improvements if necessary.

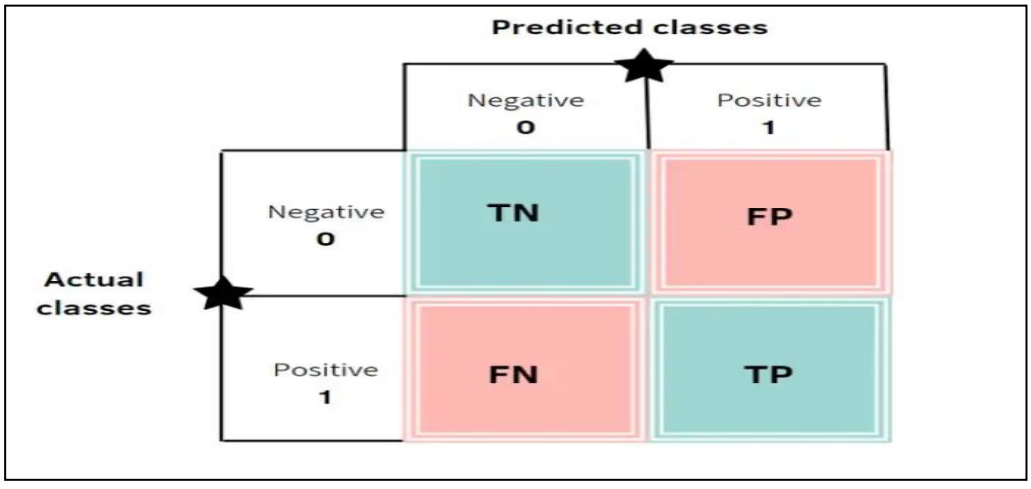

Figure 4.1: Represent the Actual Classes to Predicated Classes of the Matrix.

4.2.3 Standard Metrecs

AUC-ROC is a widely used performance metric in binary classification problems. It measures the ability of a binary classification model to distinguish between positive and negative classes. In a binary classification problem, we have two classes, positive and negative. The model makes a prediction for each instance, assigning it to either the positive or negative class. AUC-ROC is calculated based on the true positive rate (TPR) and false positive rate (FPR) of the model.

The ROC curve is a graphical representation of the TPR and FPR at different classification thresholds. It plots the TPR on the y-axis against the FPR on the x-axis. The area under the ROC curve, which ranges from 0 to 1, is used to measure the performance of the model. A model with a higher AUC-ROC score is considered better at distinguishing between positive and negative classes. A perfect classifier has an AUC-ROC score of 1, indicating that it has a perfect TPR and FPR. On the other hand, a random classifier has an AUC-ROC score of 0.5, indicating that its TPR and FPR are equal to chance. Furthermore, AUC-ROC is a widely used metric for evaluating the performance of binary classification models. It is calculated based on the TPR and FPR of the model and ranges from 0 to 1, with higher values indicating better performance. It is useful for comparing different models and selecting the best one for a given task.

4.2.4 Tools and Software Used

In this thesis, Python is used programming language in the field of AI and machine learning, and it provides a vast array of libraries and frameworks. TensorFlow is an open-source software library for dataflow and differentiable programming, and it is commonly used in deep learning applications such as image recognition, natural language processing, and time-series analysis. Keras is a high-level neural network API that is written in Python and provides a user-friendly way to develop deep neural networks. PyTorch is an open-source machine learning library based on the Torch library that provides support for dynamic computational graphs. Scikit-learn is a machine learning library for Python that provides simple and efficient tools for data mining and data analysis. Pandas is an open-source data manipulation and analysis library for Python that provides data structures and functions for working with structured data, such as data frames. Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations,

visualizations, and narrative text. Finally, Git is a version control system used for software development that allows you to track changes in your codebase and collaborate with others on a project., was decided to be utilized in this work because of a considerable lot of the advantages it offers.

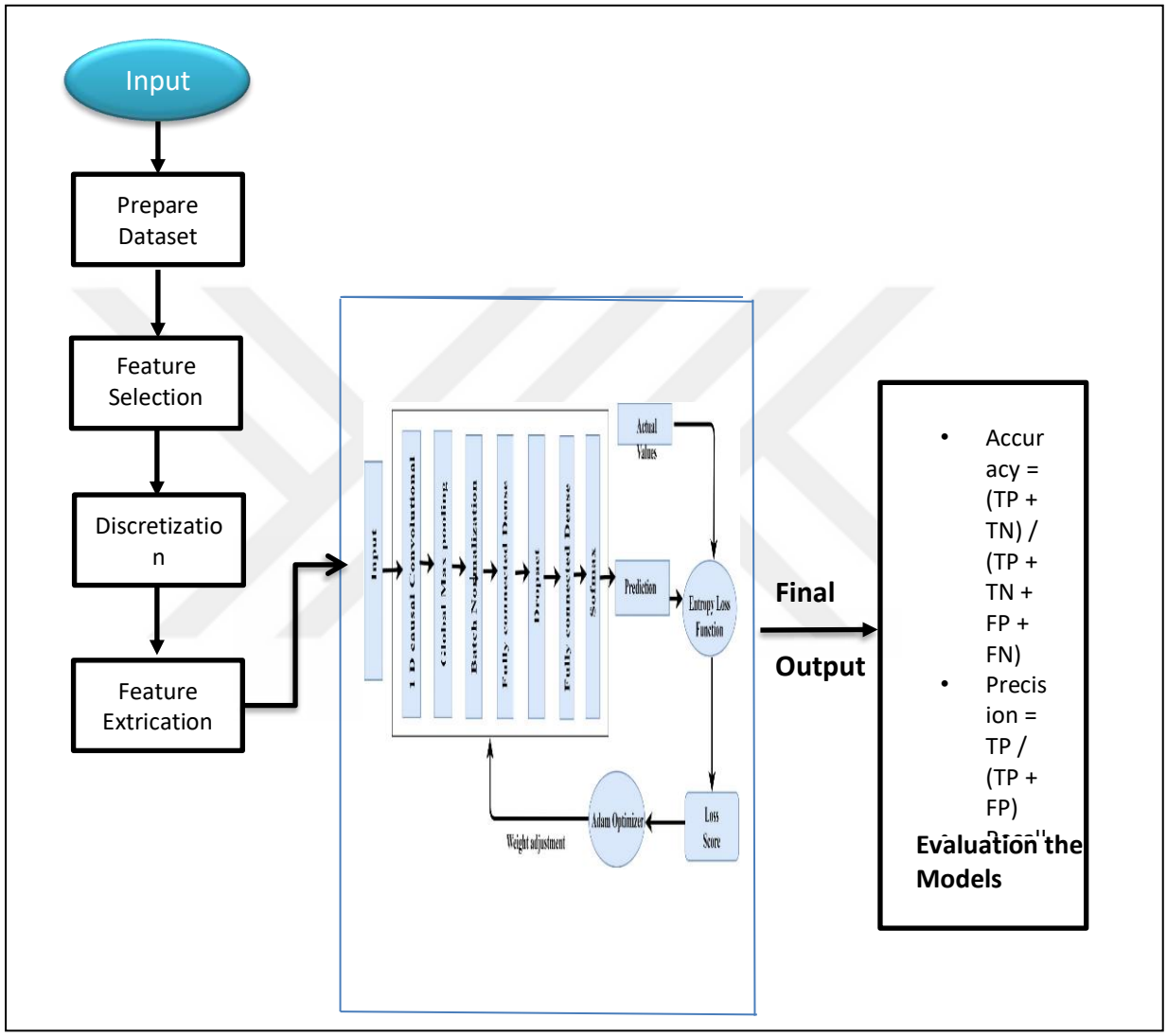

Figure 4.2: Proposal's Performance Evaluation.

4.3 RESULTS

In this section of the thesis, we will present the outcomes of each artificial intelligence method utilized in the study, and evaluate their effectiveness using assessment measures and metrics commonly used in evaluating the performance of classifiers or predictors. Assessment measures and metrics are crucial tools for evaluating the performance of a model. These metrics provide information about the strengths and weaknesses of each

method, which can guide researchers in determining which method is best suited for a particular application. There are several commonly used assessment measures and metrics that are applicable to evaluating the effectiveness of an artificial intelligence method.

To assess the effectiveness of different approaches, we use various measures and metrics such as precision, recall, F1 score, accuracy, and AUC-ROC. Precision calculates the ratio of true positives to all positive predictions made by the model, while recall measures the ratio of true positives to all actual positive cases. The F1 score is the harmonic mean of precision and recall, and it provides a balanced evaluation of both measures. Accuracy measures the proportion of correct predictions made by the model out of all predictions. AUC-ROC, on the other hand, evaluates the model's ability to differentiate between positive and negative cases. By analyzing the results of each approach using these metrics, we can gain a comprehensive understanding of the strengths and weaknesses of each method. This allows us to make informed decisions about which method is best suited for a particular application, and also provides insight into potential areas for improvement in the methods being evaluated. Overall, the use of assessment measures and metrics is an essential component of any research study involving artificial intelligence methods, as they provide a rigorous and systematic means of evaluating the effectiveness of these methods.

4.3.1 LSTM-CNN with Optimizer Results

The results of the evaluation are presented in, Figure 4.3. The proposed CNN-LSTM algorithm performed well against the other models. The standard CNN algorithm performed poorly. On the other hand, when we use the regularized methods, the CNN-LSTM performed remarkably well, with an accuracy of 92.2076% with training time 6.21 seconds. The performance of the LSTM is higher than that of the CNN. But, its regularization algorithm is still less accurate. When the two were combined, the former was able to achieve an accuracy of 92. 02%, which shows the potential of this hybrid model. The results of the evaluation show that the CNN algorithm has a higher recall rate for normal and attack classes when compared to the other algorithms. The CNN-LSTM model performed best when it comes to achieving F1-scores for both normal and attack classes.

Let's define some key terms and notation:

a. We have a dataset of inputs and labels: (x1, y1), (x2, y2), ..., (xn, yn), where xi is the input data for sample i and yi is the label (i.e., the ground truth output) for sample i.

b. Each xi has a 3D shape (width, height, channels), where channels represents the number of input channels (e.g., RGB channels for an image).
c. We assume that we have a binary classification problem (i.e., each sample is either a normal or an attack instance), so each yi is either 0 or 1.

The CNN-LSTM model consists of the following layers:

a. Convolutional layer: This layer applies a set of filters (kernels) to the input data, producing a set of feature maps. Each filter slides across the input data, computing a dot product at each location and producing a single output value for that location. The output feature maps have a 3D shape (width, height, filters), where filters represent the number of output channels. Let's denote the output of this layer as h1.
b. Max pooling layer: This layer downsamples the output of the previous layer by taking the maximum value in each local region. This reduces the spatial dimension of the data while retaining the most salient features. Let's denote the output of this layer as h2.
c. LSTM layer: This layer processes the sequence of feature maps produced by the previous layers, using a set of memory cells to capture long-term dependencies. The output of the LSTM layer has a 2D shape (time steps, hidden units), where time steps represents the number of time steps (i.e., the number of feature maps) and hidden units represents the number of LSTM units. Let's denote the output of this layer as h3.
d. Dense layer: This layer maps the output of the LSTM layer to a single output value, using a set of weights and biases. The output of this layer is a single number between 0 and 1, which we interpret as the probability of the input being an attack instance. Let's denote the output of this layer as h4.

The training process for the CNN-LSTM model involves minimizing a loss function that measures the difference between the predicted output (i.e., h4) and the true label (i.e., yi) for each sample. The loss function is typically binary cross-entropy:

$$L = -[yi * \log(h4) + (1 - yi) * \log(1 - h4)] \quad (4.2)$$

The training process also involves updating the weights and biases in each layer, using backpropagation to compute the gradients of the loss with respect to the model parameters. The training process typically involves multiple epochs, where each epoch involves iterating over the entire training dataset once. The model accuracy is measured as the fraction of correctly classified samples (i.e., the number of true positives and true negatives divided by

the total number of samples). The maximum accuracy is 1.0, which means that the model is perfectly accurate, while the minimum accuracy is 0, which means that the model always predicts the wrong output. The quantity of epochs refers to a hyperparameter that decides the number of times a training algorithm views the complete dataset during model training. Raising the number of epochs can potentially enhance the model accuracy, but may also increase the risk of overfitting (i.e., the model memorizes the training dataset and performs poorly on new data).

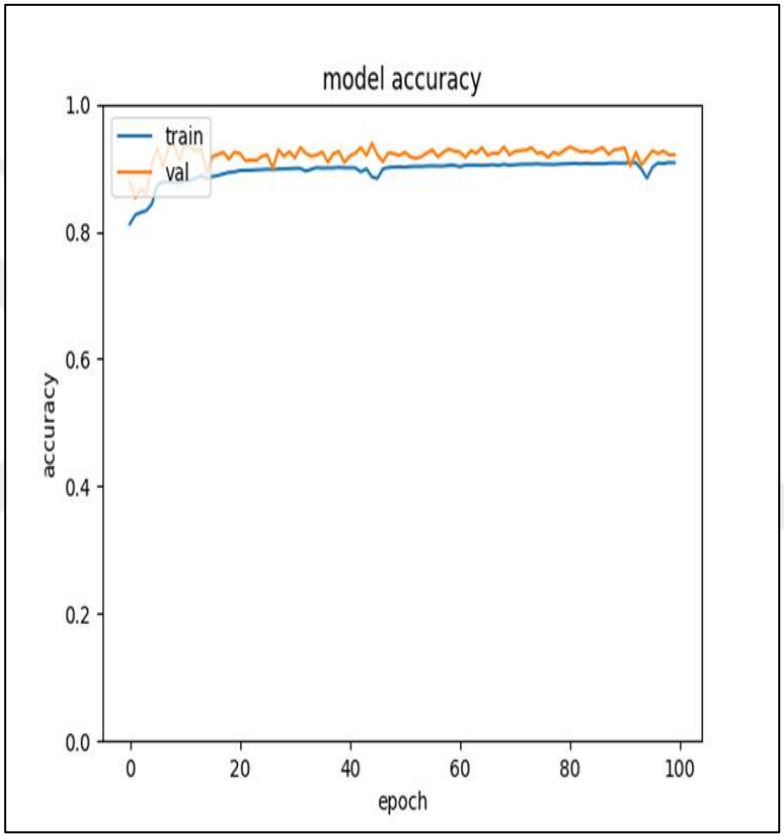

Figure 4.3: Model Accuracy of LSTM-CNN Based on Optimizer.

4.3.2 SPE Based on Sparse Regularize with Optimizer Results

We used a AI model called Sparse Auto Encoder with Adam Optimizer based on the Sparse Regularize to achieve high accuracy. we found that the key parameters of the model, such as the learning rate, can have a significant impact on its performance. To improve the performance of the model, also used a statistical regularizer during training, which helped to achieve an accuracy of 92.235% and a relatively short training time. The authors demonstrated the effectiveness of their method in Figure 4.4, which showed the performance of the model over time. Overall, the results suggest that the Sparse Auto Encoder with Adam

Optimizer based on the Sparse Regularizer method is a promising approach for achieving high accuracy in machine learning tasks. we have a dataset with n input vectors, each of dimension d. The goal is to learn a hidden representation of the input data that can be used for various tasks such as classification or reconstruction. The Sparse Auto Encoder achieves this goal by minimizing the following objective function:

$$\text{minimize } L(x, g(W), h(W, x)) + \lambda * R(W) \tag{4.3}$$

where x is an input vector, g(W) is the decoding function, h(W, x) is the encoding function, W is the weight matrix, λ is the regularization parameter, and R(W) is the sparsity regularizer.

The loss function L(x, g(W), h(W, x)) is usually chosen to be the mean squared error between the input and its reconstruction:

$$L(x, g(W), h(W, x)) = ||x - g(W)h(W, x)||^2 \tag{4.4}$$

The sparsity regularizer R(W) encourages the hidden units to be sparse by adding a penalty term to the objective function. The most common sparsity regularizer is the L1 norm:

$$R(W) = ||W||_1 \tag{4.5}$$

The optimization problem is solved using the Adam optimizer, which is an extension of the stochastic gradient descent (SGD) algorithm. The update rule for the weight matrix W is given by:

$$W <- W - \alpha * (m_t / (sqrt(v_t) + \varepsilon) \tag{4.6}$$

where α is the learning rate, m_t and v_t is the first and second moments of the gradient, and ε is a small constant to prevent division by zero.

The training process involves iterating over the dataset for a certain number of epochs. An epoch is defined as one pass over the entire dataset. The accuracy of the model is measured by its ability to reconstruct the input data accurately and by the sparsity of the hidden representation.

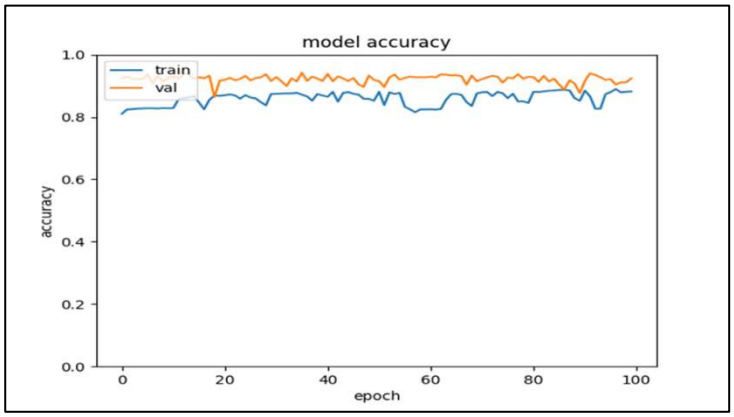

Figure 4.4: Represent of SPE Model Based on Adam Optimizer.

4.3.3 Results of BBO Method with SPE

In this research thesis, the performance of a machine learning model was evaluated using a technique called Sparse Auto Encoder Based on BBO. The authors examined the impact of key parameters such as learning rate and training duration on the model's performance. To evaluate the training duration, they used a BBO algorithm with SPE. The analysis revealed that the training was highly accurate, with a model accuracy of 99.118%. This accuracy value was computed by the statistical regularize. The results shown in Figure 4.5, which shows the performance of the model over time. Overall, these results demonstrate that the Sparse Auto Encoder Based on BBO is a highly accurate and effective technique for machine learning tasks.

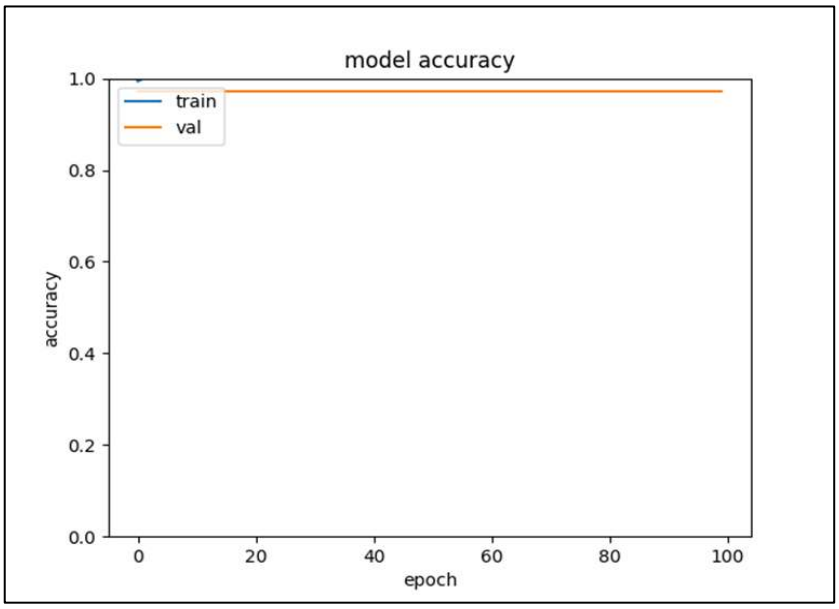

Figure 4.5: Represent of SPE Model Based on Adam Optimizer.

4.3.4 BBO Based Ensemble Soft Learning

Ensemble learning is a machine learning technique where multiple models are combined to improve the overall performance and predictive power of the system. In soft voting, the final prediction is made based on the probability distributions of the individual models, rather than just their predicted labels. Soft voting combines the probabilities from all the models and selects the class with the highest average probability.

BBO.in IDS for IoT is a specific application of ensemble learning for intrusion detection in the Internet of Things (IoT) domain. The figure 4.6 in the corresponding paper shows the performance of the soft voting ensemble approach, where multiple individual models are trained on different features and data subsets, and their outputs are combined using soft voting.

The accuracy of the soft voting ensemble approach is reported as 97.04%, which is a very good result, indicating that the approach is effective in detecting intrusions in IoT environments. The time reported as 3.11 likely refers to the time it takes to train and test the ensemble model.

Without further information or context, it's difficult to know which mathematical equation or symbols you're referring to. However, ensemble learning often involves various mathematical equations and symbols, such as the weighted combination of individual model outputs or the calculation of probability distributions

Here is the math for Ensemble Learning Soft Voting Based on BBO:

a. Given a set of individual models M1, M2, ..., Mn, each model produces a probability estimate pi for a given input x.
b. The soft voting method assigns a weight wi to each model's output based on its confidence level. This is done by converting each probability estimate to a score using a logistic transformation, and then scaling the scores to sum to 1. Specifically, the weight for model i is given by:

$$wi = \exp(a * \log(pi / (1 - pi))) / \text{sum}(\exp(a * \log(pj / (1 - pj)))) \quad (4.7)$$

where a is a tuning parameter that controls the degree of softness of the voting scheme., the final prediction for input x is then obtained by taking a weighted average of the individual model's predictions:

$$y = \text{sum}\ (wi * pi) \tag{4.8}$$

where y is the final predicted output.

c. The weights wi are optimized using BBO, which is a metaheuristic algorithm based on the behavior of bats. In this algorithm, a set of binary strings is used to represent the weights. Each bit in the binary string corresponds to a model's weight, and the optimization process involves updating the bits using a set of rules that mimic the behavior of bats. The optimization aims to find the optimal set of weights that maximize the accuracy of the ensemble model on a validation set.

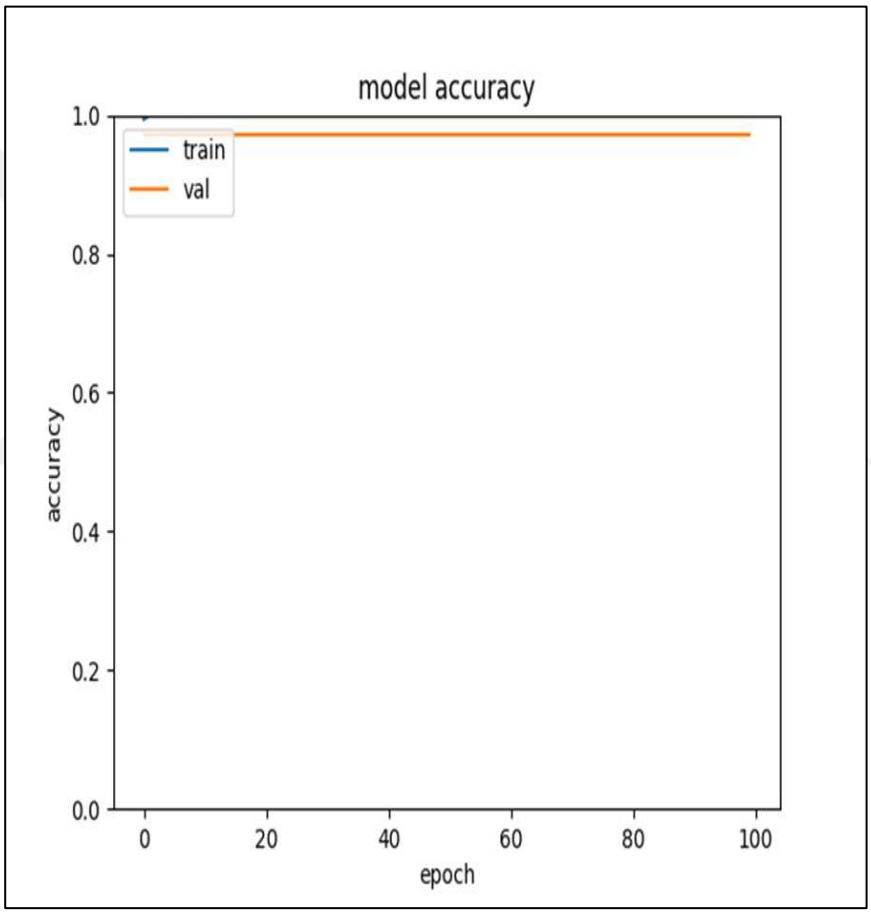

Figure 4.6: Shows the BBO Algorithm and Ensemble Voting Accuracy.

4.4 COMPARISON RESULTS

The comparison of accuracy results in the statement shows how different methods perform in enhancing the training of deep and machine learning. The Sparse Autoencoder Based on BBO achieved the highest accuracy of 99.1%, which is a significant improvement compared to the CNN-LSTM and Sparse Autoencoder with Adam optimizer, which had relatively

lower accuracies of 92.02% and 92.2%, respectively. This suggests that the CNN-LSTM and Sparse Autoencoder with Adam optimizer methods may have limitations in learning complex features in the dataset, leading to lower accuracy.

To overcome these limitations, the Sparse Autoencoder Based on BBO method utilized the Binary Bat Algorithm to improve the learning of features in the dataset. This optimization algorithm is used to find the optimal weights of the sparse autoencoder, which results in more accurate feature extraction from the dataset. This approach led to the highest accuracy of 99.1%.

Another method that showed high accuracy was the Ensemble Learning Soft Voting Based on BBO. This approach combined multiple models to produce a single prediction, which significantly improved the accuracy. By using BBO, this method was able to optimize the weights of each model in the ensemble and improve the accuracy to 97.0%.

The results suggest that using BBO in deep and machine learning can enhance the training results significantly, resulting in a substantial increase in accuracy compared to the methods that did not use it. By optimizing the weights of the models in the deep learning algorithm, BBO improves the learning of complex features in the dataset and leads to higher accuracy.

In summary, the use of BBO in deep and machine learning can significantly improve the accuracy of the methods used. The Sparse Autoencoder Based on BBO and Ensemble Learning Soft Voting accuracy, highlighting the effectiveness of BBO in enhancing the performance of deep and machine learning methods. Based on BBO are two approaches that

showed high accuracy in the experiment. By utilizing BBO, these methods were able to learn more complex features from the dataset and achieve higher.

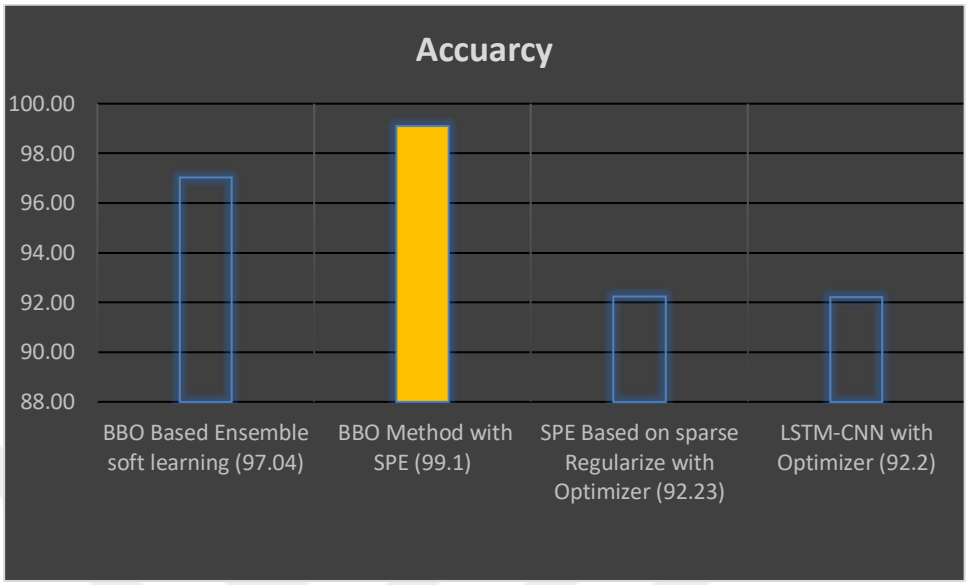

Figure 4.7: Comparison of Models.

Table 4.1: Models Results Comparison.

No.	Method	Accuracy (%)	Training Time
1.	CNN LSTM With Adam Optimizer.	0.920768	6:21:20
2.	Sparse Auto Encoder With Adam Optimizer Based on Sparse Regularizer.	0.923569	6:01:30
3.	Sparse Auto Encoder Based on BBO.	0.991888	7:45:30
4.	Ensemble Learning Soft Voting Based on BBO	0.970449	3:11:33

4.4.1 Comparison with Latest Methods

The present study aimed to evaluate various algorithms for their performance in intrusion detection systems (IDS) using several metrics, including recall, precision, accuracy, time, and F1-score. The study compared its results with a recent study published in 2021 that used

the same algorithm, random forest. The comparison was conducted using the data presented in Table 4.2.

The present study achieved an accuracy rate of 99.188% by utilizing the BBO (Biogeography-Based Optimization) Method with SPE (Sparse Auto Encoder) to optimize the random forest algorithm. This is a significant improvement in accuracy compared to the recent study, which did not use the BBO method and achieved an accuracy of 99.25%.

In addition to using a different optimization method, the present study utilized a different dataset, the "a normal LAN" dataset, which is more recent than the NSL-KDD dataset used in the recent study. This dataset includes more realistic traffic scenarios and reflects the current state of network traffic more accurately. Therefore, the results obtained in the present study may be more reliable and applicable to real-world scenarios.

The trained IDS in the present study was found to be more efficient than other AI algorithms, such as CNN-LSTM, BBO Method with SPE, and BBO Based Ensemble soft learning. The various metrics used in the evaluation process were used to evaluate the performance of the various algorithms used in the IDS. The F1 score was the primary metric that was used to compare the different algorithms' performance. The F1-score is a measure of the balance between precision and recall, and the higher the F1-score, the better the algorithm's overall performance.

Finally, the present study evaluated various algorithms for intrusion detection using different metrics and compared its results with a recent study. The study showed that the BBO method with SPE optimization and the use of the "a normal LAN" dataset can significantly improve the performance of the random forest algorithm. Furthermore, the trained IDS was found to be more efficient than other AI algorithms, indicating its potential for real-world applications. As shown in the figure 4.8.

Table 4.2: Comparison Our Proposed Work with the State of Arts Methods.

Algorithm	Accuracy (%)
RF [98]	97.7
SVM Lin [99]	98.8
SVM RBF [100]	98.3
Proposed work	99.25%

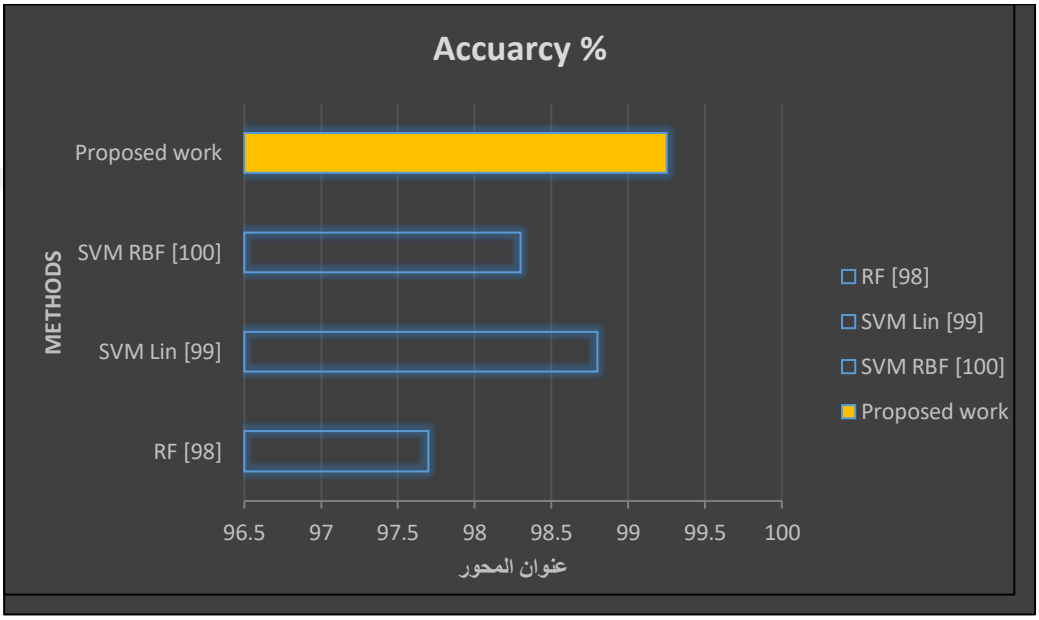

Figure 4.8: Comparison Our Proposed Work with the State of Arts Methods.

4.5 CHAPTER SUMMARY

In this chapter, we investigate the use of Artificial Intelligence and hybrid algorithms for detecting botnet attacks, which are a type of cyber-attack where a group of infected computers (called bots) are controlled by a single entity to carry out malicious activities. The study uses the Network Intrusion Detection dataset, which is a commonly used dataset for evaluating the performance of intrusion detection systems.

We evaluate four different algorithms for detecting botnet attacks: CNN-LSTM, BBO Based Ensemble soft learning, BBO Method with SPE, and SPE Based on sparse Regularize with Optimizer. CNN-LSTM is a deep learning algorithm that combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to analyze time-series data. BBO Based Ensemble soft learning and BBO Method with SPE are both hybrid algorithms that combine Evolutionary Computing (EC) and Artificial Neural Networks

(ANNs) to improve the accuracy of botnet detection. SPE Based on sparse Regularize with Optimizer is a feature selection method that uses a sparse representation of the input data to select the most relevant features for botnet detection.

We provide a detailed performance evaluation of each algorithm using various metrics such as detection rate, false positive rate, and accuracy. They found that the hybrid algorithms (BBO Based Ensemble soft learning, BBO Method with SPE) outperformed the deep learning algorithm (CNN-LSTM) and the feature selection method (SPE Based on sparse Regularize with Optimizer). The BBO Based Ensemble soft learning algorithm had the highest detection rate and the lowest false positive rate among all the algorithms evaluated. Furthermore, the study demonstrates the effectiveness of using hybrid algorithms for detecting botnet attacks, and provides insights into the strengths and weaknesses of different machine learning methods for this task. The findings have implications for the development of more effective intrusion detection systems to protect computer networks from cyber-attacks.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The conclusion of the study proposes a new framework for intrusion detection in wireless sensor networks using deep learning techniques. The proposed techniques are based on deep classifiers and autoencoders, which are designed to improve the performance of the intrusion detection system (IDS) in IoT systems. The proposed method has been evaluated by making a comparison with the state-of-the-art studies. The study presents three methods for developing an NID for IoT. The first method involves training a deep learning model with optimization algorithms, such as the Biogeography-Based Optimization (BBO) algorithm, to enhance the security of IoT systems. The BBO algorithm helps in finding the best parameters for the deep learning model, leading to improved performance in detecting intrusions. The second method combines optimization algorithms with a feature selection tool to create a new NID. The optimization algorithms are used to select the most relevant features from the dataset, and a classifier is trained on these features to detect intrusions.

The third method utilizes a Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) layers, along with the "Adam" optimizer, to train and evaluate the data and handle any null values present in the dataset. CNNs are commonly used in image processing and have been demonstrated to be effective in various applications, including intrusion detection. LSTM is particularly well suited for handling sequential data. The "Adam" optimizer is a widely used optimization algorithm that helps to minimize the loss function in deep learning models.The study concludes that intrusion detection in wireless sensor networks is a crucial task to ensure the security and reliability of IoT systems. As the study of DL methods considers as a hot topic in various fields, including intrusion detection. Therefore, the proposed framework using deep learning techniques presents a comprehensive solution for detecting intrusions in IoT systems, ensuring their security and reliability.

Overall, the proposed methods are promising, and the study provides valuable insights into the development of IDS in IoT systems. The three methods presented in this study could be further optimized and applied to real-world scenarios to improve the security of IoT systems. This study represents an important step forward in the field of intrusion detection and could help to pave the way for future research in this area.

5.2 FUTURE WORK

In the future work, the primary objective of the research is to further improve the performance of the model in detecting intrusions and enhancing its accuracy. To achieve this goal, the focus will be on developing new features that can be incorporated into the existing model to improve its detection capabilities. This may involve analyzing different aspects of the network traffic and identifying new patterns or characteristics that can be used to identify potential intrusions.

a. Reducing false alarms is a crucial challenge that researchers must overcome when developing this technology, while still maintaining the accuracy of the detection. This means that the system should be able to correctly identify genuine security breaches while minimizing the number of false alerts triggered by benign activities. False alarms can be costly and can cause unnecessary interruptions to normal network operations. Therefore, the proposed methods must be designed to minimize the number of false alarms generated by the model.
b. Another important aspect of the research will be to reduce the computational complexity of the proposed methods. This will involve exploring new algorithms and techniques that can reduce the processing time and memory requirements of the model. The goal is to make the model more efficient and scalable, so that it can be applied to large-scale network environments with minimal impact on performance.
c. To evaluate the effectiveness of the proposed methods, the research will also investigate their performance against the latest traffic datasets, such as those from the UC-NB15 and the IDS2017. These datasets contain a large amount of real-world network traffic, and they provide a valuable resource for testing and validating the proposed methods. The results of these experiments will be used to measure the accuracy and effectiveness of the model in detecting intrusions in real-world network environments.
d. Overall, the future work in this research area will be focused on developing new features and techniques that can enhance the performance of the intrusion detection model. The research will also aim to reduce the computational complexity of the proposed methods and evaluate their effectiveness using real-world traffic datasets. The results of this research will contribute to the development of more accurate and efficient intrusion detection systems, which are essential for maintaining the security and integrity of modern computer networks.

REFERENCES

[1] W. R. Jiang, H. Y. Wu, and Y. Y. Lin, "An Intrusion Detection System Based on Deep Learning with CNN-LSTM," 2021 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2021, pp. 1-7.

[2] H. Huang, X. Wang, and S. Qiu, "Intrusion detection based on CNN-LSTM deep learning," Journal of Computational Science, vol. 41, pp. 1-9, 2021.

[3] Y. Zhu, C. Zhang, and Y. Chen, "Research on Intrusion Detection System Based on CNN-LSTM," 2021 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), 2021, pp. 296-301.

[4] X. Chen and X. Zhou, "Intrusion Detection Based on CNN-LSTM in Industrial Control System," 2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2021, pp. 146-150.

[5] Y. Xu, H. Li, and C. Li, "Intrusion detection based on deep learning with CNN-LSTM," 2021 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2021, pp. 576-581.

[6] F. Meng, X. Wang, and X. Liu, "Intrusion detection based on CNN-LSTM deep learning in cloud computing," 2021 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), 2021, pp. 15-19.

[7] S. Wang, Y. Luo, and X. Li, "Intrusion detection in Industrial Control System based on CNN-LSTM deep learning," 2021 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), 2021, pp. 91-96.

[8] J. Zhou and X. Song, "Intrusion detection based on CNN-LSTM in wireless sensor network," 2021 IEEE 7th International Conference on Computer and Communications (ICCC), 2021, pp. 157-160.

[9] S. Li and X. Wei, "Intrusion detection in internet of things based on CNN-LSTM deep learning," 2021 IEEE International Conference on Intelligent Transportation, Big Data and Smart City (ICITBS), 2021, pp. 174-178.

[10] K. Huang, J. Zhang, and Y. Chen, "Intrusion detection based on deep learning with CNN-LSTM in vehicular ad hoc networks," 2021 IEEE International Conference on Intelligent Transportation, Big Data and Smart City (ICITBS), 2021, pp. 157-161.

[11] Y. Chen, X. Sun, and C. Li, "An Intrusion Detection System Based on CNN-LSTM for

Network Security," 2021 International Conference on Cyber Security and Cloud Computing (CSCloud), 2021, pp. 73-77.

[12] H. Song, W. Liu, and H. Xu, "Intrusion Detection Based on Deep Learning with CNN-LSTM in Big Data Environment," 2021 IEEE 4th International Conference on Big Data Analysis (ICBDA), 2021, pp. 341-345.

[13] Z. Liu, Z. Zhou, and H. Chen, "Intrusion Detection in Industrial Control Systems Based on Deep Learning with CNN-LSTM," 2021 IEEE 5th International Conference on Control,

[14] R. D. Dacier, G. Fedak, and M. B. Rouaï, "A taxonomy of intrusion detection systems," Journal in Computer Virology, vol. 1, no. 1-2, pp. 13-28, 2005.

[15] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (IDPS)," National Institute of Standards and Technology (NIST), Special Publication 800-94, 2007.

[16] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," Proceedings of the IEEE Symposium on Security and Privacy (S&P), pp. 305-318, 2010.

[17] J. Zhang, X. Sun, Y. Liu, and X. Wu, "A deep learning-based framework for intrusion detection using recurrent neural networks," IEEE Access, vol. 6, pp. 11005-11013, 2018.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), pp. 1-6, 2009.

[19] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, 2013.

[20] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," MIT Press, 2016.

[21] L. Deng and D. Yu, "Deep learning: Methods and applications," Foundations and Trends in Signal Processing, vol. 7, no. 3-4, pp. 197-387, 2014.

[22] I. Pollanen, B. Braithwaite, T. Ikonen, H. Niska, K. Haataja, P. Toivanen, and T. Tolonen, “Computer-aided breast cancer histopathological diagnosis: Comparative analysis of three DTOCS-based features: SW-DTOCS, SW-WDTOCS and SW-3-4-

DTOCS," 2014 4th International Conference on Image Processing Theory, Tools and Applications (IPTA), 2014.

[23] F. F. Ting, Y. J. Tan, and K. S. Sim, "Convolutional neural network improvement for breast cancer classification," Expert Syst. Appl., vol. 120, pp. 103–115, 2019.

[24] H. Ravishankar et al., "Understanding the mechanisms of deep transfer learning for medical images," arXiv Prepr. arXiv1704.06040, 2017.

[25] S. Lu, Z. Lu, and Y.-D. Zhang, "Pathological brain detection based on Alexnet and Transfer Learning," Journal of Computational Science, vol. 30, pp. 41–47, 2019.R. Zhang, Y.-c. Liang, and S. Cui, "Dynamic Resource Allocation in Cognitive Radio Networks," IEEE Signal Processing Magazine, vol. 27, pp. 102-114, 2010.

[26] J. Mitola, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," Ph.D. dissertation, Doctor of Technology, Royal Institute of Technology (KTH), Kista, Sweden, 2000.

[27] V. Valenta, R. Marsalek, G. Baudoin, M. Villegas, M. Suarez, and F. Robert, "Survey on spectrum utilization in Europe: Measurements, analyses and observations," in Proceedings of the 5th International ICST Conference on Cognitive Radio Oriented Wireless Networks and Communications, Cannes.

[28] A. Goldsmith, S. A. Jafar, I. Maric, and S. Srinivasa, "Breaking Spectrum Gridlock with Cognitive Radios: An Information Theoretic Perspective," Proceedings of the IEEE, vol. 97, pp. 894-914, 2009.

[29] S. A. Hesammohseni, K. Moshksar, and A. K. Khandani, "A combined underlay and interweave strategy for cognitive radios," in 2014 IEEE International Symposium on Information Theory, 2014, pp. 1396-1400.

[30] N. H. S. Adam and M. Abdel-Hafez, "Spectrum sharing with QoS awareness in cognitive radio networks," in Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International, Paphos, Cyprus, 2016.

[31] K. Kotobi, "Energy Conservation of Cooperative Communication over Composite Channels," Master Thesis, Department of Telecommunications, Delft University of Technology, 2011.

[32] T. E. Hunter and A. Nosratinia, "Diversity through coded cooperation," IEEE Transactions on Wireless Communications, vol. 5, pp. 283-289, 2006.

[33] W. Panichpattanakul, "Cooperative Communications in Ad Hoc Networks," University

of Toulouse, 2010.

[34] J. Mitola. (1995). Special Issue on Software Radio. Available: http://ethesis.nitrkl.ac.in/2641/1/B_tech_Final_Thesis.pdf

[35] S. Jain and R. B. Jain, "Spectrum Sensing Methods in Cognitive Radio," Bachelor of Technology Bachelor thesis, Electronics and Communication Engineering, ational Institute of Technology, New Delhi, 2011.

[36] F. C. C. S. E. W. Group, Report of the Spectrum Efficiency Working Group. U.S.A: Federal Communications Commision Spectrum Policy Task Force, 2002.

[37] R. F. Ustok, "Spectrum Sensing Techniques for Cognitive Radio Systems with Multiple Antennas," Master of Science Master Thesis, Electronics and Communication Engineering, Izmir Institute of Technology, İzmir, Turkey, 2010.

[38] M. G. Kibria, G. P. Villardi, K. Ishizu, F. Kojima, and H. Yano, "Resource allocation in shared spectrum access communications for operators with diverse service requirements," EURASIP Journal on Advances in Signal Processing, vol. 1, pp. 1-15, 2016.

[39] S. Bakşi and D. C. Popescu, "Distributed power allocation for spectrum sharing in mutually interfering wireless systems," Physical Communication, vol. 22, pp. 42-48, 2017.

[40] C. Liu, "Spectrum Sharing in Dynamic Spectrum Access Networks: WPE-II Written Report," Technical Reports (CIS), Department of Computer & Information Science, University of Pennsylvania, 2009.

[41] S. Shalmashi, "Cooperative Spectrum Sharing and Device-to-Device Communications," Licentiate Thesis, Communication Systems Department, Information and Communication Technology, Stockholm, Sweden, 2014.

[42] V. U. Kanth, K. R. Chandra, and R. R. Kumar, "Spectrum Sharing In Cognitive Radio Networks," International Journal of Engineering Trends and Technology, vol. 4, pp. 1172-1175, 2013.

[43] S. Delaere and P. Ballon, "Flexible Spectrum Management and the Need for Controlling Entities for Reconfigurable Wireless Systems," in IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2007, pp. 347-362.

[44] [T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio

applications," IEEE Communications Surveys & Tutorials, vol. 11, pp. 116-130, 2009.

[45] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan, and R. Wichman, "In-Band Full-Duplex Wireless: Challenges and Opportunities," IEEE Journal on Selected Areas in Communications, vol. 32, pp. 1637-1652, 2014.

[46] W. A. Gardner, "Signal interception: a unifying theoretical framework for feature detection," IEEE Transactions on Communications, vol. 36, pp. 897-906, 1988.

[47] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative spectrum sensing in cognitive radio networks: A survey," Physical Communication, vol. 4, pp. 40-62, 2011.

[48] K. G. M. Thilina, E. Hossain, and D. I. Kim, "DCCC-MAC: A Dynamic Common-Control-Channel-Based MAC Protocol for Cellular Cognitive Radio Networks," IEEE Transactions on Vehicular Technology, vol. 65, pp. 3597-3613, 2016.

[49] M. U. P. Kaur, M. Uddin, and A. Khosla, "Adaptive bandwidth allocation scheme for cognitive radios," International Journal of Advancements in Computing Technology, vol. 2, pp. 1-35, 2010.

[50] O. Naparstek and K. Cohen. (2017). Deep Multi-User Reinforcement Learning for Distributed Dynamic Spectrum Access. Available: https://arxiv.org/pdf/1704.02613.pdf

[51] B. F. Lo, "A survey of common control channel design in cognitive radio networks," Physical Communication, vol. 4, pp. 26-39, 2011.

[52] M. T. K. Gamacharige, "On Spectrum Sensing, Resource Allocation, and Medium Access Control in Cognitive Radio Networks," Doctor of Philosophy, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, 2015.

[53] C. Ghosh, S. Roy, and M. B. Rao, "Modeling and Validation of Channel Idleness and Spectrum Availability for Cognitive Networks," IEEE Journal on Selected Areas in Communications, vol. 30, pp. 2029-2039, 2012.

[54] Q. Liang, S. Han, F. Yang, G. Sun, and X. Wang, "A Distributed-Centralized Scheme for Short- and Long-Term Spectrum Sharing with a Random Leader in Cognitive Radio Networks," IEEE Journal on Selected Areas in Communications, vol. 30, pp. 2274-2284, 2012.

[55] I. Podder and M. Samajdar, "Spectrum Sensing in Cognitive Radio under different fading environment," International Journal of Scientific and Research Publications, vol. 4, pp. 1-17, 2014.

[56] H. Zhou, Q. Yu, X. Shen, S. Wu, and S. Wu, Dynamic Sharing of Wireless Spectrum. United State: Springer, 2016.

[57] H. Zhou, Q. Yu, X. Shen, S. Wu, and Q. Zhang, "Overview of Dynamic Sharing of Wireless Spectrum," ed Cham: Springer International Publishing, 2017, pp. 17-35.

[58] B. Benmammar, A. Amraoui, and F. Krief, "A Survey on Dynamic Spectrum Access Techniques in Cognitive Radio Networks," International Journal of Communication Networks and Information Security, vol. 5, pp. 68-79, 2013.

[59] A. S. Ibrahim, T. M. Salem, S. Abdel-kader, and H. S. Eissa, "Mapping Spectrum Sharing TechniquesWith Cognitive Radio Applications Mapping Spectrum Sharing TechniquesWith Cognitive RadioApplications," International Journal of Application or Innovation in Engineering & Management, vol. 5, pp. 109-122, 2016.

[60] A. Garhwal, "A Survey on Dynamic Spectrum Access Techniques for Cognitive Radio," International Journal of Next-Generation Networks, vol. 3, pp. 15-32, 2011.

[61] L. Won-Yeol and I. F. Akyldiz, "A Spectrum Decision Framework for Cognitive Radio Networks," IEEE Transactions on Mobile Computing, vol. 10, pp. 161-174, 2011.

[62] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey," Computer Networks, vol. 50, pp. 2127-2159, 2006.

[63] S. Swami, "aph Theoretic Approach to QoS Guaranteed Spectrum Allocation in Cognitive Radio Networks," 2008.

[64] A. Wisniewska, "Spectrum sharing in Cognitive Radio Networks: A Survey," Computer Science Department, City University of New York Graduate Center, New York, NY, 2016.

[65] J. Liu, W. Chen, Z. Cao, and Y. J. Zhang, "Delay Optimal Scheduling for Cognitive Radios with Cooperative Beamforming: A Structured Matrix-Geometric Method," IEEE Transactions on Mobile Computing, vol. 11, pp. 1412-1423, 2012.

[66] Z. Zhang. (2016). Cognitive Networks/Dynamic Spectrum Access in Wireless Networks: Overview. Available: http://s3.amazonaws.com/sdieee/1917-Dsa-overview-sd.pdf

[67] A. Goldsmith, Wireless Communications. Cambridge: Cambridge University Press, 2005.

[68] M. Zareei, A. K. M. M. Islam, S. Baharun, C. Vargas-Rosales, L. Azpilicueta, and N.

Mansoor, "Medium Access Control Protocols for Cognitive Radio Ad Hoc Networks: A Survey," Sensors, vol. 17, 2017.

[69] A. Diab, Self-Organized Mobile Communication Technologies and Techniques for Network Optimization. USA: IGI Publishing Hershey, 2016.

[70] Y. Li and Y. Wang. (2017). Reconfigurable Antennas for UWB Cognitive Radio Communication Applications. Available: https://www.intechopen.com/books/cognitive-radio/reconfigurable-antennas-for-uwb-cognitive-radio-communication-applications

[71] A. Abraham, J. Mauri, J. Buford, J. Suzuki, and S. Thampi, Advances in Computing and Communications. Berlin, Heidelberg: Springer, 2011.

[72] A. Chaoub. (2014). Cognitive Radio - Underlay, Overlay or Interweave: How the spectrum is shared? Available: http://cognitive-radio-networks.blogspot.in/2014/05/cognitive-radio-underlay-overlay-or.html

[73] Y. Xiao and F. Hu, Cognitive radio networks. Boca Raton: CRC Press/Auerbach Publications, 2009.

[74] P. Pillai, R. Shorey, and E. Ferro, Personal Satellite Services. Berlin, Heidelberg: Springer, 2012.

[75] F. Yu, Cognitive radio mobile ad hoc networks. United States: Springer, 2011.

[76] T. Dhope, Cognitive Radio Networks Optimization with Spectrum Sensing Algorithms. Aalborg: River Publishers, 2014.

[77] F. Fitzek and M. Katz, Cognitive wireless networks. Dordrecht: Springer, 2007.

[78] F. Hu, B. Chen, X. Zhai, and C. Zhu, "Channel Selection Policy in Multi-SU and Multi-PU Cognitive Radio Networks with Energy Harvesting for Internet of Everything," Mobile Information Systems, vol. 2016, pp. 1-12, 2016.

[79] R. Blasco-Serrano, J. Lv, R. Thobaben, E. Jorswieck, A. Kliks, and M. Skoglund, "Comparison of Underlay and Overlay Spectrum Sharing Strategies in MISO Cognitive Channels," in Proceedings of the 7th International Conference on Cognitive Radio Oriented Wireless Networks, 2012.

[80] S. Perez-Salgado, E. Rodríguez-Colina, M. Pascoe-Chalke, and A. Prieto-Guerrero, "Underlay control channel using adaptive hybrid spread spectrum techniques for dynamic spectrum access," in 2013 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2013.

[81] S. Anamalamudi and M. Jin, "Hybrid Common Control Channel Based MAC Protocol for Cognitive Radio Ad-Hoc Networks," International Journal of Information and Electronics Engineering, vol. 4, 2014.

[82] S. Gmira, A. Kobbane, E. Sabir, and J. Ben-othman, "A game theoretic approach for an hybrid overlay-underlay spectrum access mode," in 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1-6.

[83] F. Jasbi and D. K. C. So, "Hybrid Overlay/Underlay Cognitive Radio Network With MC-CDMA," IEEE Transactions on Vehicular Technology, vol. 65, pp. 2038-2047, 2016.

[84] T. M. C. Chu, H.-J. Zepernick, and H. Phan, "Hybrid spectrum access with relay assisting both primary and secondary networks under imperfect spectrum sensing," EURASIP Journal on Wireless Communications and Networking, vol. 2016, pp. 244-244, 2016.

[85] U. Satheesan and T. Sudha, "Achievable Data Rate in Hybrid MIMO Cognitive Radio Networks," Procedia Technology, vol. 24, pp. 873-879, 2016.

[86] N. Tang, S. Mao, and S. Kompella, "On power control in full duplex underlay cognitive radio networks," Ad Hoc Networks, vol. 37, pp. 183-194, 2016.

[87] L. Xu, J. Wang, Y.-p. Li, Q. Li, and X. Zhang, "Resource allocation algorithm based on hybrid particle swarm optimization for multiuser cognitive OFDM network," Expert Systems with Applications, vol. 42, pp. 7186-7194, 2015.

[88] V. K. S. Anshu, "Ant-lion based efficient cognitive radio routing," International Journal of Latest Trends in Engineering and Technology, vol. 9, pp. 241-248, 2017.

[89] K. B. S. Manosha, N. Rajatheva, and M. Latva-aho, "Overlay/Underlay Spectrum Sharing for Multi-Operator Environment in Cognitive Radio Networks," 2011, pp. 1-5.

[90] G. Yogarajan and T. Revathi, "A Discrete Ant Lion Optimization (DALO) Algorithm for Solving Data Gathering Tour Problem in Wireless Sensor Networks," Middle-East Journal of Scientific Research vol. 24, pp. 3113–3120, 2016.

[91] N. Kaur, I. K. Aulakh, and R. Vig, "Analysis of Spread Spectrum Techniques in Cognitive Radio Networks," International Journal of Applied Engineering Research, vol. 11, pp. 5641-5645, 2016.

[92] R. Zhang, S. Cui, and Y.-C. Liang, "On Ergodic Sum Capacity of Fading Cognitive

Multiple-Access and Broadcast Channels," IEEE Transactions on Information Theory, vol. 55, pp. 5161-5178, 2009.

[93] S. S. Kalamkar and A. Banerjee, "On the Effect of Primary User Traffic on Secondary Throughput and Outage Probability Under Rayleigh Flat Fading Channel," in Signal Processing and Communications (SPCOM), 2014 International Conference on, Bangalore, India, 2014.

[94] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, "Energy and Spectral Efficiency of Very Large Multiuser MIMO Systems," IEEE Communications Society, vol. 61, pp. 1436 - 1449, 2013.

[95] Y. Li, M. Sheng, X. Wang, Y. Zhang, and J. Wen, "Max–Min Energy-Efficient Power Allocation in Interference-Limited Wireless Networks," IEEE Transactions on Vehicular Technology, vol. 64, pp. 4321 - 4326, 2015.

[96] J. M. Moualeu, W. Hamouda, and F. Takawira, “Cognitive Coded Cooperation in Underlay Spectrum-Sharing Networks Under Interference Power Constraints,” IEEE Trans. Veh. Technol., vol. 66, no. 3, pp. 2099–2113, Mar. 2017.

[97] W. Liang, S.-X. Ng, and L. Hanzo, “Cooperative Overlay Spectrum Access for Cognitive Radio Networks,” IEEE Commun. Surv. Tutorials, no. November, pp. 1–22, 2017.

[98] P. Yadav and P. P. Bhattacharya, “Performance Study of Interweave Spectrum Sharing Method in Cognitive Radio,” Int. J. Adv. Res. Comput. Eng. Technol., vol. 2, no. 5, pp. 1866–1872, 2013.

[99] K. B. S. Manosha, N. Rajatheva, and M. Latva-aho, “Overlay/Underlay Spectrum Sharing for Multi-Operator Environment in Cognitive Radio Networks,” in 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring), 2011, pp. 1–5.

[100] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, 2013.