REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

A NEW SOFTWARE DEFINED NETWORKS (SDN) IN IOTS BASED DEEP LEARNING TECHNIQUES

Osamah Ridha Jawad AL-HWAIDI

Master's Thesis

Supervisor

Asst. Prof. Dr. Hasan Hüseyin BALIK

Istanbul, 2023

A NEW SOFTWARE DEFINED NETWORKS (SDN) IN IOTS BASED DEEP LEARNING TECHNIQUES

Osamah Ridha Jawad AL-HWAIDI

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis titled A NEW SOFTWAR DEFINED NETWORKS (SDN) IN IOTS BASED DEEP LEARNING TECHNIQUES DETECTION prepared by OSAMAH RIDHA JAWAD AL-HWAIDI and submitted on 17/04/2023 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering.

Asst. Prof. Dr. Hasan Hüseyin BALIK

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Hasan Hüseyin BALIK	Department of Computer Engineering, İstanbul Aydın University	————
Asst. Prof. Dr. Oğuz ATA	Department of Software Engineering, Altınbaş University	————
Asst. Prof. Dr. Aytuğ BOYACI	Department of Computer Engineering, Milli Savunma University	————

I hereby declare that this thesis meets all format and submission requirements for a Master's thesis.

Submission date of the thesis to Institute of Graduate Studies: ___/___/___

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Osamah Ridha Jawad AL-HWAIDI

Signature

DEDICATION

To my parents who stood by me throughout my education, To my wife and son who encouraged me, To my friends who have supported me all the time, To all of these people I dedicate this humble effort.

PREFACE

I want to thank my supervisor for everything, Asst. Prof. Dr. Hasan Hüseyin BALIK, for his support. Asst. Prof. Dr. Hasan Hüseyin BALIK and I talked about my progress and issues several times a week, and this was very helpful in making the research's logic clear to me. It made it easier for me to comprehend the technological specifications for this research endeavor.

ABSTRACT

A NEW SOFTWARE DEFINED NETWORKS (SDN) IN IOTS BASED DEEP LEARNING TECHNIQUES

AL-HWAIDI, Osamah Ridha Jawad

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Hasan Hüseyin BALIK

Date: 04 / 2023

Pages: 77

In this study, a new software defined networks (SDN) in IoTs based deep learning techniques was implemented using various types of classifiers such as DNN, CNN, GRU, LSTM, and RNN and SDN Ryu controller. The system was able to handle high-dimensional and complex data by using NSL-KDD dataset, and was able to detect unknown intrusions that traditional methods may miss. The effectiveness of the models was evaluated by accuracy, precision, recall, F-score, and confusion matrix. We used python 3.10 to implementation our system. The system was able to achieve good performance, however, the system's efficacy will be determined by the kind of the data we feed it and the scale of the issue we're attempting to address. This study highlights the potential of DL-based NIDS with SDN and IoT to detect network intrusions, but also highlights the need for continuous monitoring and updating to ensure that the system remains effective.

Keywords: Deep Learning (DL), Software Defined Network (SDN), Convolutional Neural Network (CNN), Internet of Things (IoT), Supervised Network Intrusion Detection System (SNIDS).

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

AI	:	Artificial Intelligence
SNIDS	:	Supervised Network Intrusion Detection System
IoT	:	Internet of Things
SDN	:	Software Defined Network
DL	:	Deep Learning
CNN	:	Convolutional Neural Network
DNN	:	Deep Neural Network
RNN	:	Recurrent Neural Network
GRU	:	Gated Recurrent Unit
LSTM	:	Long Short-Term Memory

1. INTRODUCTION

The traditional network model is being challenged by the emergence of Software-Defined Networking (SDN) with deep learning and Internet of Things (IoT) technology, SDN simplifies network management and provides greater flexibility and scalability, making it ideal for applications for example network intrusion detection systems [1]. Deep learning algorithms can be used to provide better security, detect malicious traffic more accurately, and identify new threats faster, with the help of IoT devices, SDN networks can also be monitored in real-time to detect any suspicious activities or anomalies [2]. This brings us to the question of how traditional networks compare to SDN networks with deep learning and IoT when it comes to network intrusion detection systems [3 , we can now create a network that is more secure, reliable, and cost-effective [6]. Furthermore, with the integration of IOT devices into this network, we can also detect any potential intrusions in real-time [7]. SDN networks with deep learning and IOT have revolutionized the way in which we create networks, these networks are much faster and more secure than traditional ones, This makes them ideal for organizations that require high levels of security [8]. Intrusion detection systems (IDS) are becoming increasingly important for businesses, as cyber-attacks and malicious activities become more sophisticated, by combining Software Defined Networking (SDN) and deep learning with Internet of Things (IoT), companies can create an intrusion detection system that is both reliable and secure [8]. SDN allows for the automation of network management, while deep learning enables the detection of complex patterns and anomalies in data traffic, this makes it possible to detect malicious activities quickly and accurately, Furthermore, by incorporating IoT devices into the system, companies can monitor a wide range of devices simultaneously [9]. Using an intrusion detection system with SDN and deep learning along with IOT can help businesses protect their networks from cyber-attacks and other malicious activities, companies can also benefit from improved efficiency and cost savings due to automated network management processes [10].

1.1 RESEARCH PROBLEM

This research issue is motivated by the rising need of protecting IoT networks from attack. We can no longer imagine our lives without the Internet of Things, and the numeral of linked hardwares continues headed for increase. The growth of IoT devices, however, has increased the risk for hacking and network penetration. However, traditional NIDS may not be able to identify new or developing threats in IoT settings since they depend on preset signatures [11]. The suggested system intends to use deep learning approaches to deal with high-dimensional and complicated data, both of which are prevalent in IoT settings. The suggested NIDS uses deep learning and Software Defined Networking (SDN) to detect attacks that more conventional approaches could miss. The suggested NIDS also incorporates software-defined networking (SDN) in an effort to supply a more flexible and malleable solution for protecting IoT settings. Since SDN allows the partitioning of the data plane to the control plane, network assets may be managed in a more dynamic and centralized manner. The suggested research challenge also incorporates the implementation of the system in a real-world setting, as well as the assessment of its performance utilizing a variety range of metrics, accuracy, precision, recall, and F-score [12, 14]. This will aid in determining the efficacy of the planned NIDS in discovering new and undiscovered network intrusions in IoT settings using SDN.

1.2 QUESTIONS OF THE RESEARCH

Aims of this study include answering the following question:

i. What are some best practices for designing and deploying an (NIDS) for (IoT) networks with SDN based on DL that can quickly and reliably identify cyberwarfare attacks?

ii. When it comes to IDS in IoT networks, what is the state-of-the-art approach have proven to be the most effective?

iii. What features should be included in NIDS for IoT with SDN based on DL?

iv. How can the specified NIDS framework be operationalized and implemented?

v. How can this NIDS model be evaluated using methodologies and metrics?

1.3 GOAL OF THE RESEARCH

The reason behind this study is to examine and put into practice the DL method that is most suited for the NIDS based on SDN with IOT.

1.4 AIMS OF THE STUDY

These are the specific, realizable goals that make up the overall aim :

i. To create highly developed procedures for NIDS and best applicable approaches.

ii. Second goal is to use NIDS datasets to test and compare different Deep learning strategies.

iii. Create and implement an intrusion detection system (NIDS) prototype for IoT networks.

iv. Assessing the efficiency of the suggested NIDS.

1.5 METHODOLOGY OF RESEARCH

The main strategy adopted in this research is the Design Science (DS) method. This approach was chosen as it is deemed suitable for achieving the study's objectives. The DS method encompasses several steps, such as creating and evaluating technological tools aimed at solving a specific organizational problem, implementing a structured approach to designing effective solutions for practical challenges, conducting related research, analyzing the designs and publishing the conclusions to the relevant parties [15]. The research ultimate goal was to develop a method for spotting malicious activity in IoT systems. According to [15], the DS approach was therefore decomposed into the DS research steps (Figure 1.1):

i. The first step is to identify the issue or problem analysis.

ii. Establishment or development of state-of-the-art.

iii. Planning for implementation (Solution design).

iv. Evaluation of the Solution.

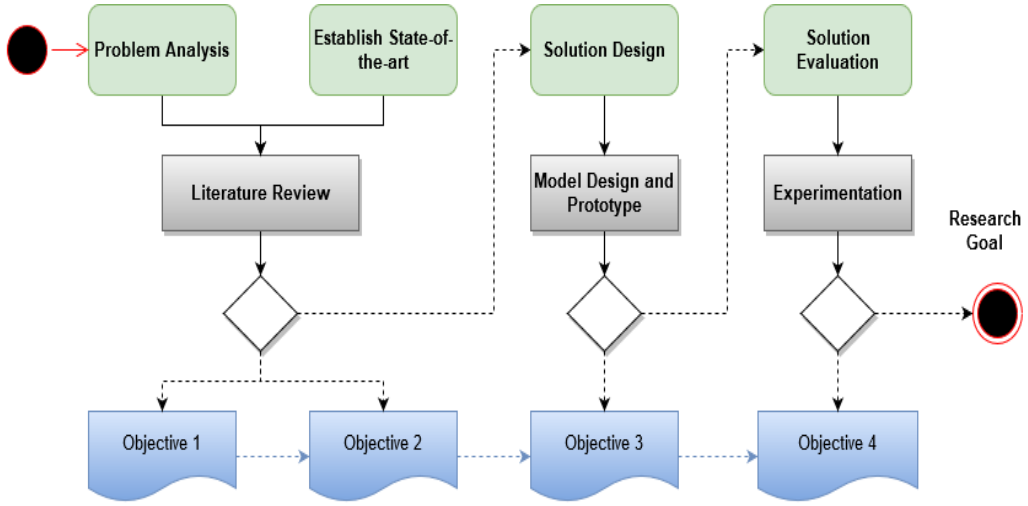


Figure 1.1: Methodology Overview [15].

The next sections (1.6.1–1.6.4) provide an overview of these epochs.

1.5.1 Identify the Issue or Problem Analysis

According to [16], researchers in the field of Design Science are obligated to come up with a technological answer to a pressing business issue. Since more devices are being linked and networked, security is becoming a challenge in IOT applications. That's why it's important to keep an eye on security and address problems as they arise. This research therefore addresses a pressing, real-world issue that might help businesses strengthen their network safety and incident response procedures.

1.5.2 Establishment or Development of State-of-the-Art

This study used a literature review strategy to ascertain the current state of the art. Limitations and problems of existing network security measures were identified by reviewing recent conference papers and periodicals. The assessment also looked at the effectiveness of cutting-edge network security approaches, as well as the many security problems, assaults, and vulnerabilities that have been seen in IOT networks.

1.5.3 Planning for Implementation (Solution Design)

This research was conducted with the intent of producing a conceptual IDS architecture for IoTs. Software-Defined-Networks (SDN) and Deep Learning (DL) are two examples of cutting-edge computing concepts, we were able to create an NIDS model that makes use of SDN to get a bird's-eye view of the network and gather data, while also employing DL algorithms to effectively categorize that data. An implementation of a prototype using the concept proved its viability. With this, we want to solve the problems associated with network security in IoT systems.

1.5.4 Evaluation of the Solution

An initial prototype of the suggested IDS was implemented and tested in this study. The quality and effectiveness of the proposed IDS were tested via experimental analysis of network traffic simulations.

1.6 CONTRIBUTIONS OF RESEARCH

In today's world, several cyber-security threats might originate from the underlying structure of a network. This highlights the gravity of the issue of network security. Accordingly, this study investigated how the most up-to-date cybersecurity and networking developments might be used to identify and counteract the most common forms of assault.

To begin, Software Defined Networking (SDN) has been identified as one from the few developments that offers several benefits to IOT Networks. Accordingly, this study lays the groundwork for future research into SDN as a security enhancer solution and encourages more research and investigation of SDN's possibilities in IOT Networks. In this research, a NIDS model was developed that takes full use of SDN to sample and prepare flow data quickly and easily. By developing a prototype for a NIDS in (SDN) environment, the researcher of this research shown that SDN may be an effective complement to intrusion detection systems (IDSs).

Further, Deep Learning (DL) is one of the rapidly developing topics, with a lot of research being done in the pursuit of efficient methods for detecting network intrusions. However, most NIDS-related research just analyzes the effectiveness of various Machine Learning

algorithms on datasets, rather than showing how they may be put to practical use. An ineffective NIDS cannot be guaranteed by comparing DL techniques using datasets gathered online, which is a major downside. Therefore, a prototype of the suggested model was developed using the Design Science approach. This approach provides a useful model for doing research that has practical and theoretical implications. This methodology may be used by other researchers in the area to create, deploy, and test their own suggested DL-based IDSs besides analyzing datasets.

With this study, we used SDN in conjunction with an effective DL model to develop a smart NIDS that can collect information from IOT network devices. Such data is then processed and analyzed to detect invasive behaviors, and it is equipped to take immediate, hands-off corrective action.

2. LITERITAURE REVIEW

2.1 INTRODUCTION

In this part, we go into the current state of knowledge and discuss how it relates to the larger discussion around IoTs security. An appropriate NIDS framework may be discovered by conducting a thorough examination of contemporary technology. This study also aids in the identification of possible paradigms that might handle some of the communication and network security problems in IoTs. The first step of the chosen technique, Design Science Research, is a thorough examination of the issue and a search for gaps in the current literature on the subject. Consequently, this section of the study investigates the many technological options available for use in realizing the study's objective. The outline for this section is as follows: We present the concept of network security and the security needs for IoTs in Section 2.1. The purpose of this is to introduce the topic of this study and offer necessary context. In Section (2.1.1-2.1.2), we'll go further into the topic, talking about some of the most well-known methods for detecting intrusions in the network security statistical field, Machine Learning, and other approaches of intrusion detection (NIDS) are discussed in this division. We offer an outline of the various communication networks used in the IoTs. This section addresses some of the concerns voiced by other scientists about the implementation and maintenance of NIDS techniques on the IoTs. IoTs intrusion detection system (NIDS) architecture best practices are described in section below. So, we propose ML, DL and SDN as the necessary solutions to alleviate the problems associated with the implementation of NIDS in IOT networks today. The studies that have used DL for the problem of (NIDS) are summarized, and their restrictions are highlighted. Similar gaps are discovered and evaluated in the literature describing intrusion detection methods using ML, DL and SDN. The examined works' difficulties, gaps, and suggestions for further study are summed up in Section 2.2 and 2.3.

2.2 NIDS BASED ON IOTS

Generally speaking, there are two distinct machine learning-based IoT NIDSs: supervised and unsupervised [17]. Supervised learning (usually known as misuse detection) involves an algorithm being given a labeled dataset and learning the attack classes during training. This

requires data describing threats. The opposite is true for unsupervised learning, where anomaly detection is only one specific case: models are trained with just data explaining normal behavior. Consequently, labeling the data is unnecessary. Anomaly detection models acquire knowledge of the typical pattern of behavior throughout their training phases. The testing process involves applying the model to fresh data in an effort to spot any anomalies from the expected behavior. Using anomaly detection models, previously undiscovered assaults may be uncovered.

2.2.1 Supervised Network Intrusion Detection System (SNIDS)

A method for detecting DDoS attacks in consumer IoT networks using classification algorithms is suggested by [18]. Figure 2.1 depicts the detection process and every IP address, port number, packet size, and time stamp are recorded, as well as the destination and source IP addresses and ports. Finally, data packets are sorted by their source IP addresses. As with the data coming from each individual device, the packet stream is broken up into discrete intervals that don't overlap. Packet size, IP Address Range, and Protocols are stateless characteristics that are obtained; stateful attributes include Total Bandwidth and Overall Number of Unique Target IP Addresses. In this experiment, Three Internet of Things devices were used to mimic real web traffic. The DDoS attacks were created by simulating the typical attack methods used in these types of attacks, such as TCP SYN flooding, UDP flood, and HTTP GET flood. A total of five supervised machine 1earning algorithms (KNN, SVM with a 1inear kernel, Decision Tree, Random Forest, and FCNN) were tested and compared. Depending on the techniques used, the researchers reported an F1 score of 0.927 to 0.999.

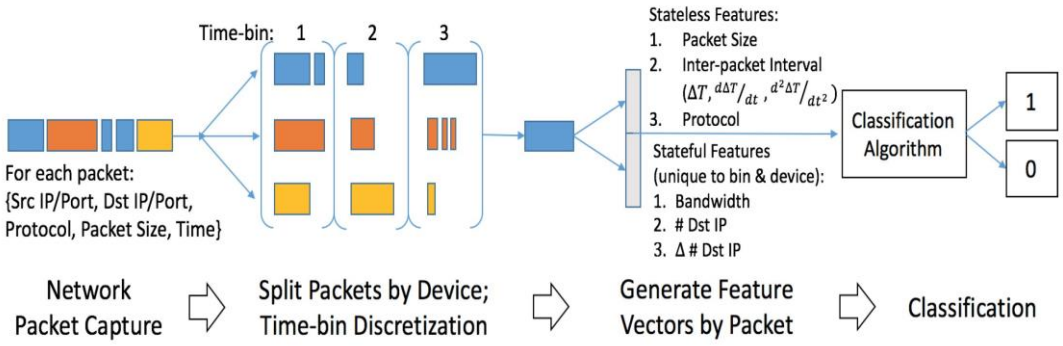


Figure 2.1: DDoS Attack in IoT [18].

For detecting attacks on the IOT [19], present a distributed deep learning-based system. Attack detection models are trained and stored at the fog devices (switches and routers, gateways, hubs, etc.) in the periphery of the dispersed network. Each fog node's trained model shares its parameters with the others thanks to a coordinating master node. Specifically, They suggest implementing a fully convolutional neural network (FCNN) thru three hidden layers (150, 120, and 50 neurons per layer, respectively) to identify malicious or benign network activity. Instead of analyzing data from actual IoT networks, they use the NSL-KDD data - set to evaluate the efficacy of their strategy. The model is able to identify attacks with a 99% efficiency and a false positive rate (FPR) of 0.85%.

For the purpose of detecting intrusions in wireless IoT networks. [20] offer an active learning strategy. Learning with a little quantity of labeled data is possible with active learning, a branch of human-in-the-loop machine learning. Because it would be too time-consuming or perhaps impossible for a human expert to identify each and every training event, active learning avoids this step. Instead, the learning system requests that a person label a training instance. Uncertainty sampling is the standard method. Once the model has been trained on the labeled data that has already been gathered, predictions are made on the unlabeled data. Those examples for which the model is least confident are sent to the expert to label (in the case of binary classification, this matches to instances for which the mode1 produces a class possibility near to 0.5). This procedure is repeated until the system's accuracy and recall meet some predetermined standard. The writers choose to use XGBoost as their classification algorithm of choice for the tests. There are two datasets used to assess the model's effectiveness: the NSL-KDD dataset and the AWID set of data (which was collected in a real-world WIFI setting). The active learning approach (depending on ambiguity sampling) is shown to significantly outperform the random-select approach (where the cases to be classified by the human expert are chosen at random) in a series of experiments. Furthermore, roughly a third fewer labeled instances are needed to get the same level of performance.

As an example of early research into the potential of FCNNs for DDoS disclosure in IoT networks, see the work of [21]. An FCNN with a single hidden layer is used. An Internet of Things network consisting of 5 sensors is established for testing purposes. This network consists of five sensors, four of which are client nodes and one of which is a server relaying

node. The server node serves as a central hub for the network, gathering data from the client devices before analyzing and relaying that information to the clients. In this way, the sensor nodes may adjust their behavior and respond to events as a group. One host initiated the DDoS assault by transmitting Datagrams to the server device. The attributes utilized to analyze the network traffic information are not described. They claim that their FPR is just 1.9% and that their total classification accuracy is 99.4%.

2-dimensional reduction then two-tier classification is the basis of the proposed supervised intrusion detection approach for IOT networks proposed by [22]. As an initial stage in the dimensionality reduction process, principal component analysis is performed. In this phase, the basic feature space must be converted into a lower-dimensional space while retaining as much variation as possible. To further decrease the dimensionality of the data after PCA transformation, researchers apply LDA (Linear Discriminant Assessment), a supervised matrix factorization approach. After the module for reducing the dimensionality by two levels, the module for classifying data into two levels is implemented, as shown in Figure 2.2. Naive Bayes classifiers are used in the initial stage of the classification process to establish whether or not a particular event represents an attack. If the Nave Bayes classifier concludes that an instance fits the "normal" category, it passes the data on to another classifier for further inspection. The alternative classifier uses a Certainty-Factor-modified k-NN algorithm. Its purpose is to fine-tune the findings of the first encoder by deciding whether or not a specific incident constitutes an attack. The NSL-KDD sample is used in the laboratory for experimentation. This new model is able to identify attacks with an accuracy of 84.86 percent and a false positive rate of 4.86 percent.

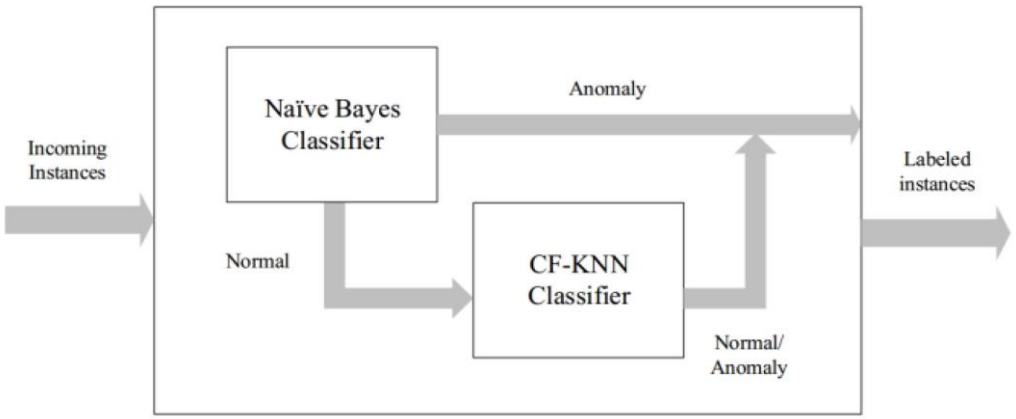


Figure 2.2: Classification Module with 2-Tier [22].

To identify botnet cyberattacks versus the DNS, HTTP, and MQTT protocols used in internet of thinks networks. [23] offer an ensembles intrusion detection method. Records of all established links are compiled and kept in a database. We then generate series of 100 recorded links using characteristics that represent statistical relationships. Flow characteristics, MQTT characteristics, DNS characteristics, and HTTP characteristics are all examples of the features that are put to use. Ip and ports of source and destination, type of protocol, and time of last connect are all examples of flow characteristics. Size of MQTT information, variety of contacts from the exact source, & the variety of connections to the exact destination are all characteristics of MQTT. Domain name system (DNS) characteristics include properties like query type, query length, and response length. Included in the HTTP characteristics are the kind of HTTP request made (GET, POST, HEAD), the requested URI, the content size, and the total length of the URL. Here's how we put the model through its paces during training: Important characteristics are chosen using the correlation coefficient, and then the AdaBoost ensemble learning approach serves as a classification tool. Data from the UNSW-NB15 and NIMS botnets, coupled with data generated by simulated IoT sensors, are utilized in the experiment. Raspberry PIS is used to model the IoT sensors network. The attack detection rate is between 97% and 99%, while the false positive rate (FPR) is between 1% and 3%, however these numbers fluctuate widely depending on the data type.

To identify black hole attack in RPL-based 6LoWPAN IoT networks. [24] Provide a hybrid ML approach that includes unsupervised and supervised learning. A wormhole attack involves compromising many routers and using them to alter the network's routing behavior by connecting them via a tunnel. To start, the K-means clustering method is used to establish where in the network each router can safely operate. The range of a router is quantified in terms of these secure zones. An additional step is to train a decision-tree classifier to improve the clustering's initial output. Random (nodes situated at random locations), mesh, ring, and star topologies are tested to see how well the model performs. With this approach, we can identify attacks with a success rate of 71%-75%. Neither the FPR nor its components are disclosed.

2.2.2 Unsupervised with NIDS

Using a network-based approach [25] offer a technique for finding anomalies for the IoT called N-BAIOT. In order to identify IoT botnets, they suggest using stacked autoencoders. The packet's sending and receiving hosts and protocols are analyzed in real time to create a behavioral snapshot. Over the course of many time periods, 115 statistics were calculated. The statistics include (1) all traffic that started at the same IP address, (2) all traffic that started at the same MAC address and IP address, (3) all traffic that started at the source IP and ended at the destination IP, and (4) all traffic that started at the source IP and ended at the destination TCP/UDP socket. Simply put, we only have the regular and standard perversion of the data size and the IAT. Time ranges of 200 MS, 250 MS, 15 s, 20 s, and 1 minute are considered. It's important to remember that without the anomaly detector being set up on the local network, it's impossible to calculate such data. When training an autoencoder, only one instance is used across all devices. The testing step necessitates awareness of the source of the traffic. The experimental assessment makes use of nine Internet of Things gadgets. When the MIRAI and BASHLITE viruses are used in a controlled lab setting, they produce malicious traffic. Autoencoders use a 4-layer encoding and 4-layer decoders. The authors assert a 0.0070 FPR and a 100% attack detection accuracy.

Kitsune is a locally deployed network intrusion detection system (NIDS) released in [26] that can automatically and rapidly learn to recognize network attacks without human intervention. Kitsune's foundational methodology revolves on a collection of autoencoders for spotting traffic anomalies. In figure 2.3, we see the five individual components that together form Kitsune. The Packet Parser receives packets from the Packet Capturer. Packet Parser is used to extract meta-info from a packet, such as the size of the packet and the arriva1 time. Kitsune makes use of the same 115 arithmetical features first outlined in [25] to characterize network traffic. Meta-data are provided by the Packet Parser for use by the Feature Extractor in producing the aforementioned numbers. A 115-dimensional feature vector is then collapsed into k-dimensional instances using the Feature Mapper. To rephrase, the elements of the original feature matrix are split up across the k sub instances in the vector. Furthermore, the total length of the k sub-instances matrix is not necessarily equa1 to 115, since the identical item of the origina1 feature matrix could be planned to several sub-vectors. instance's The Anomaly Detector consists of two parts: the Ensemble Layer and the

Output Layer. Each of the k subs instances vectors is sent into an autoencoder in the Ensemble Layer. All of the autoencoders in the Ensemble Layer notify the output units below of the erroneous reconstruction. The Output Layer, also an autoencoder, computes the total anomaly score (although the authors' choice of the word "Output Layer" is deceitful since it does not match to the output layer often used to express the outcome of a neura1 network). Kitsune is put through its paces on two operational IP camera networks. There are four cameras across both systems. To create harmful traffic, attackers use techniques like SYN Flood, OS scan, and ARP spoofing. Depending on the kind of attack, the researchers have reported an AUC of 0.8-1.

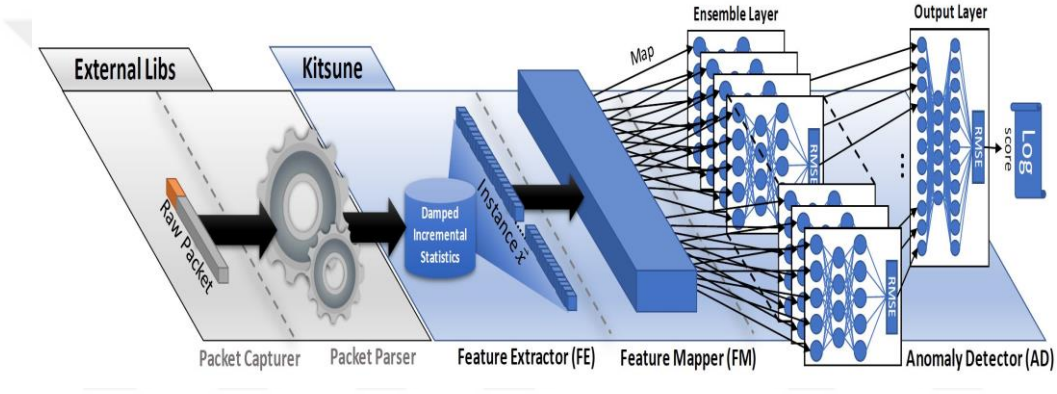


Figure 2.3: Kitsune Architectural Design [26].

DIOT is introduced by [27], which is a self-learning, autonomous system for identifying hacked (IoT) devices. DIOT uses methods from the field of machine learning known as "unsupervised learning." Using crowdsourced, unlabeled data from client IoT networks, it develops anomaly detection algorithms. The source of the traffic must be determined first, and that means finding out what kind of device it was. The methods for identifying certain types of devices were first created in [28], and it is proposed that these methods be used again. After each device's typical communication profile has been established, abnormalities in the data transmission between them may be identified. DIOT keeps an eye on data packet streams. A total of seven characteristics—packet direction, local and distant ports, packet duration, TCP signals, protocols, and the Internet Address Translation (IAT)—are used to characterize each individual packet in the sequence. In this method, we determine the likelihood of each packet's recurrence based on the k packets that came before it. A Gated

Recurrent Unit (GRU) system is utilized to calculate these probabilities. We raise an alert if the estimated probabilities of occurrence for a critical mass of successive packets fall under a detection limit. The authors claim that DIOT has a 94.01% attack detection accuracy and an FPR with less below 1%.

An autoencoder-based approach to anomaly identification is proposed by [29] for WSN data in internet of things. Data from sensors, such as temperatures, is used instead of network traffic data in this approach. In Figure 2.4, we can see how the cloud takes care of the computationally difficult work of model training. As a further step, the sensors get the trained model. That's why it's possible to identify anomalies locally on each sensor, without sending data to a central server or the cloud. Eight temperature and humidity sensors are placed in various rooms across an office building for the trials. The 720 readings per day may be attributed to the sensing frequency of 1 report every 2 - 3 minutes from each detector. As a result, the daily data from each sensor is described by a 720-dimensional vector known as the feature vector.

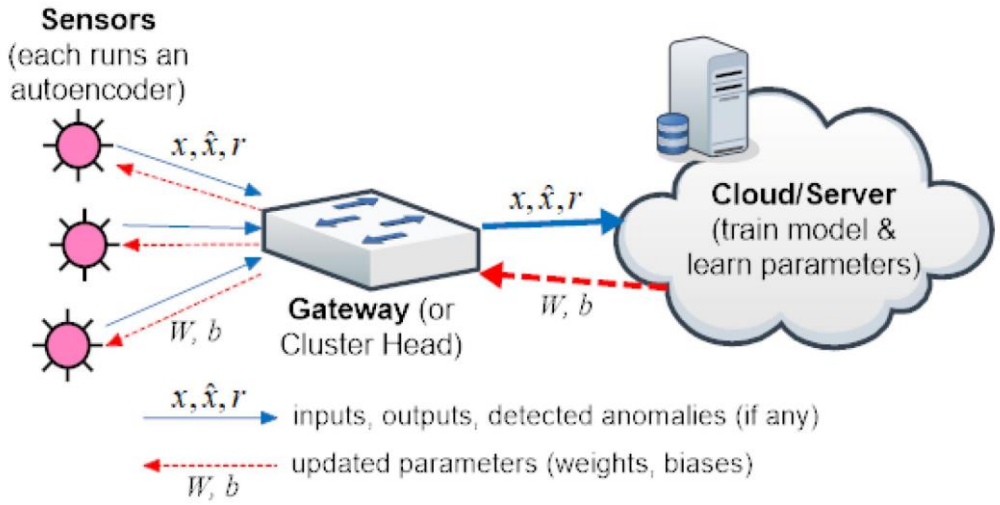


Figure 2.4: Autoencoder-Based Anomaly Detection in a WSN Environment [29].

2.3 NIDS BASED ON MACHINE LEARNING

In the field of cybersecurity, ML techniques have lately seen widespread use. A variety of ML classification techniques, including Single classifiers, Mixed classifiers, and Ensemble classifiers, are utilized in this situation of intrusion detection [30]. When constructing an IDS

using a single ML method, a single classifier is employed. When it comes to classification, hybrid classifiers provide a solution by combining many different Machine Learning techniques into a single model. When compared to a random classifier, the execution of an ensemble classifier is just slightly better. In comparison to using a single classifier, using a hybrid classifier can significantly increase the accuracy of NIDS because one method can be used to pre-process training dataset samples, removing non-representative training datasets, and the outcomes can then be used to create a classifier using the next algorithm for pattern classification. Five of the most popular machine learning (ML) classification techniques for NIDS are shown: 1) (NB). 2) (SVM). 3) (DT) using Gini index, Gain-ratio and Chi-squared. 4) Bootstrap Aggregation. 5) (RF). 6) (BN). 7) Adaptive Boosting. 8) Multi-Layer Perceptron (MLP).

An overview of ML and DM approaches in cybersecurity literature is offered in [31]. The authors claimed that it was difficult to give a single suggestion for each work because of the fecundity and complexity of these strategies, and that thus, the most successful strategy for cyber apps has not been determined. What matters most is the kind of assault the system was built to stop. The authors went on to claim that there are other parameters that should be taken into account when evaluating the efficacy of ML/DM approaches, including the accuracy,intricacy, classification speed, and interpretability of the final result. The significance of data sets for ML/DM in cyber intrusion detection was also stressed in their research. IDS benefit from access to system and kernel-level data for efficient anomaly or abuse detection, as described by [31]; if at all practicable, data set should be supplemented with OS Kernel-level data. However, they limited their research to works that had already been done to evaluate ML/applicability DMs in the field of intrusion detection. The current research goes further that previous work25 by suggesting an IDS approach that makes use of ML and then assessing the system's efficacy.

An intrusion detection method is provided in [32] to assist system administrators with problems relating to intrusion detection that develop during incident response. The primary objective was to determine which nodes in the network are most vulnerable to assault. The authors offer a fix predicated on the premise that an attacker's disclosure of personal and vulnerability-related information during an assault may be used to pinpoint specific nodes in the target network that have been compromised [32]. Using the methodology used by the

United States military for gathering information on the battlefield, the writers conceived of the internet as a battleground. They built simulations to guess the adversary's capabilities, motivations, and strategy. They based their plan of attack on the economy of crime, sometimes known as the theory of criminal behavior. The most vulnerable nodes were then identified using network and threat models [32].

A cross-layer dependent classification intrusion detection method for safe MANET multicast connectivity in a military environment was recently reported in research by [33]. By passing through the collision avoidance technique, the inventors of the tree-based multipath routing protocol MAODV were able to add a covert attack against the single hop route discovery control messages. The authors next examined the MAODV's resistance to both indirect and direct inner sneaky strikes, such black hole and deny-to-forward, finding that these attacks had a significant effect on PDR, efficiency, and management overheads [33]. When it came to protecting against stealthy assaults, the authors presented a cross-layer dependent machine learning distributed anomaly detection solution. Integration of MAC and routing layer features, rather than only routing layer characteristics, was employed to improve precision. Anomaly detection in army systems was shown to be very successful by the approach, but it was also found to be time-consuming and resource-intensive due to the requirement to gather data from many levels, including the media access control (MAC) and routing layers. The current research envisages an intrusion detection system that would simply analyze flow-based data, thereby reducing data collecting and processing difficulties.

In [34], the authors developed a marine scenario tracking and monitoring method that makes use of learning principles. Their method employs continuous in-the-moment learning to simultaneously recognize patterns of the present statuses of solitary vessels in the immediate neighborhood, based on data collected in real time. In their system of machine learning, they used a variant of the 26 Fuzzified ARTMAP classification neural network [35]. Essentially, the strategy included a pair of algorithms: one for unsupervised clustering, and another for supervised mapping and labeling [34]. Their results show how useful machine learning and artificial intelligence are for the armed forces. Their use of Machine Learning methods, however, is not concerned with communication security but rather with vassal prediction. In order to better identify intrusions in military communication networks, ML is used in this research.

To deal with the issues of massive data, strict security and intrusion detection needs in governmental integration, Using ML, the authors of [36] created a weighted SVM-based limit machine learning for training (LML). The authors choose to utilize LML because it is a fast-learning strategy for a hidden neuron layer in a feedforward neural network. With this technique, the number of nodes in the hidden layer is determined based on the classification's objectives, as opposed to the original LML best estimate. Once the optimal weight and offset for each node have been determined, he utilize the SVM weight to make the final adjustment. In this way, he can guarantee that every node is equipped with an enhanced capacity for doing the generalization job. The experimental findings show that SVM-LML outperforms BP in terms of identification accuracy and training time, while also being more stable according to the authors [36], their approach of NIDS should be used when building data-based systems to assist civil-military integration. But the researchers in that paper evaluated their intrusion detection (NIDS) method using data from the DARPA 1999 KDD dataset. These datasets have been criticized in recent years for being out of date and unable to fairly reflect today's network topologies [37]. This research will make use of a more modern intrusion detection assessment dataset and system data taken from a simulated battlefield setting.

In [38] examined five different Machine Learning models, including Randomized Forest, J48, SVMS, CART, and Naive Bayes. When compared with the other methods, they found that Random Forest had the highest accuracy rate. SVM and ANN algorithms, according to a study [39] are often the most suggested methods for particular learning classification. More specifically, when it comes to intrusion detection using an ensemble of 27, AdaBoost and voting majority are the two most used options. Recent research [40] examined several intrusion detection methods, including Naive Bayes, Regression Trees, Randomized Forest, Multi-Layer Autoencoder, and SOM. As the authors discovered, MLP performed best, followed by Randomized Forest and Naive Bayes. While MLP did improve detection accuracy, the scientists noted that the model-building procedure for MLP took over 12 hours, which is too long for practical use. Therefore, Randomized Forest and Naive Bayes were deemed to be the most effective methods.

Several defense department uses of ML, such as spying and subsurface mine attack, have been implemented with great effectiveness [41]. However, ML-based IDSs were developed,

and the performance of these systems has been analyzed in studies employing datasets, for use in security-related contexts. The categorization accuracy of assaults was shown to have improved, and automated model building was shown to have facilitated this. These features make it clear that ML approaches should be used to enforce security in command-and-control networks, since they guarantee complete and flawless detection and identification inside such systems. While several papers have suggested machine learning for intrusion detection, few have demonstrated how to implement such systems practice. In order to show the viability of such a system, this research seeks to identify the best ML algorithm before proceeding to build and construct an IDS that makes use of this algorithm.

This will guarantee that the deployment performance of ML algorithms is taken into account when making recommendations about their performance on datasets.

2.4 NIDS BASED ON SDN

In [42] suggested that SDN's ability to support dynamic policy management of a network may aid in the automated establishment of services based on policy, cut down on planning overheads, ease the strain on operators, and eliminate configuration mistakes. In addition, they said that SDN allows for dynamic management of network resources assigned to each operation and the route used by the service's traffic. This means that network resources may be dedicated to mission objectives irrespective of changes in the relative importance of those goals [42]. One example is the proposal (Figure 2.5) by [43] of a Software-defined Networking (SDN)-based architecture for streamlined administration of strategic network resources.

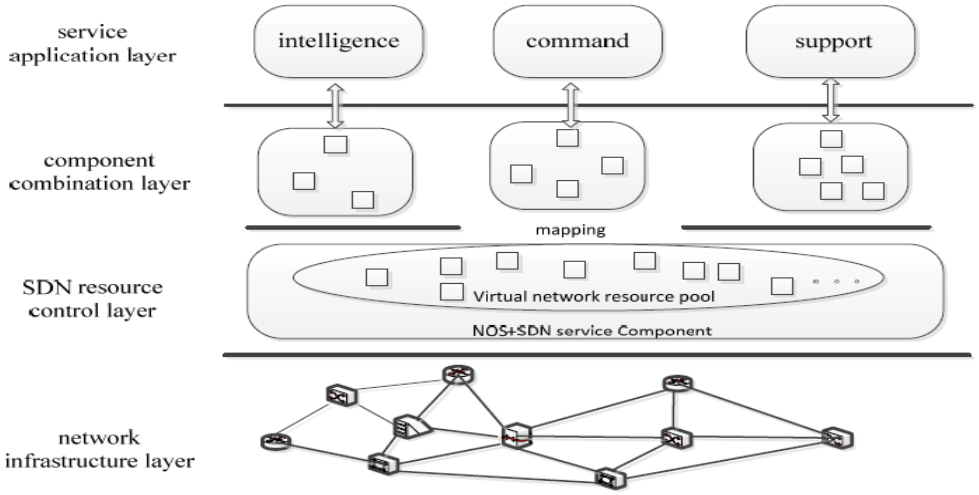


Figure 2.5: Architecture for SDN-Based Resource Management [43].

In order to represent the interdependencies across combat actions, the authors presented a service component-based recombination resource optimization strategy. Based on their simulations, it seems that their approach reduces the typical delay in fulfilling service requests. Challenges in operational network administration were the focus of the authors' research [42]. Their findings corroborated earlier findings of SDN's efficacy in terms of conceptual control, where network services may be effectively managed and customized [42]. This research builds upon previous efforts by using SDN's centralized logical controlling capabilities not to resource management but rather to IDS and mitigation.

For the purpose of identifying and mitigating abnormalities in IoT traffic, [44] presented an SDN-based system they named Sof-Things. The goal of the framework was to discover traffic irregularities earlier, closer to the network's edge, rather than at the network's core or at higher layers. As a result, assaults on IoT devices might be quickly detected, and countermeasures put into place if necessary. Specifically, the research used the ML method known as SVM classifier (SVM) to identify instances of unexpected traffic. Measures of precision and recall were used to assess the framework's efficacy. Using linear SVM resulted in a few false positives, leading the authors to conclude that the detection accuracy was compromised. They also found that non-linear SVM led to increased accuracy. Due to the usage of the Kernel technique, non-linear SVM results in less false positives. Further, they claimed that their technique can swiftly recover the lost throughput, and hence can prevent various assaults within a matter of seconds. We used a similar strategy in this research,

combining software-defined networking and deep learning to create an intrusion detection system. While other research has focused on Internet of Things use cases, our research has included tactical military network use cases.

In [45] developed a cross-layer paradigm to aid in the resolution of network security challenges in IOT networks. The framework's emphasis on efficiency and automation makes it a great fit for IOT networks. A key feature of the authors' suggested method is its emphasis on coordinating security services across all tiers of communication. It was motivated by the discovery that processes at higher levels may be made safer or optimized by using data gathered from lower-level security services, such as authentication and IDS [45]. For instance, an interconnected cross-layer security mechanism may receive real-time threat profiles from authentication and IDS components running at the application layer. When these findings are compiled, they may be communicated to the lower levels to enhance their functionality and reliability. However, this approach raises the bar for inter-node connectivity and rises the difficulty and interna1 processing inside a node. The authors suggested that IDS, carrier frequency, and distributed verification are only some of the security services that may be incorporated using their platform.

According to [46], SDN networks provide a useful framework for enforcing data-centric security standards across several layers of mobile communication networks. They maintained that rigorous experimental examination and validation of suggested technological ideas prior to implementation in real government systems was an essential part of building advanced security solutions. They suggested an OpenFlow-based testbed for verifying SDN security features. Using this technology, you may implement data-centric security regulations in addition to SDN layer protection methods. The study's findings backed up their method's capacity to verify simulation and analytical forecasts.

A method for detecting and reducing anomalies in SDN systems that is scalable is presented in a paper by [47]. Reduced data collection is achieved by the S-Flow protocol's care for sampling; an entropy-based method detects and flags anomalies; and OpenFlow is used for system-wide mitigation. More than that, the authors showed that collecting and processing OpenFlow information posed scalability concerns by overwhelming the centralized control plane. They also claimed that their mechanism's efficiency was on par with that of a native

OpenFlow solution in low-traffic settings. Furthermore, they noted that the suggested S-Flow-based solution demonstrated better behavior in terms of resource use than the original OpenFlow mechanism. As a result of the limited hardware in IOT devices, this research will be using S-Flow testing as an alternative to the more conventional OpenFlow sampling technique.

In a recent study [48], the authors offered a comprehensive review of programmable networks like SDN, as well as several Machine Learning / Deep Learning strategies. The survey detailed the difficulties encountered in creating an adaptable and effective NIDS with ML/DL-based methods. According to the authors [48], selecting feature selection techniques that can accurately assess important characteristics for the IDS is one of the most significant obstacles. In addition, they claimed that the current data set is unreliable for scientific and scholarly forecasting. Due to this problem, researchers must develop datasets to guarantee reliable assessment of NIDS. The authors also identified the problem of processing large amounts of data as a basic difficulty for SDN-based NIDS.

In addition, [49] suggested that developing a centralized SDN controller would be challenging. Which can detect and apply real-time IDS in high-speed networks. But this research provides a flow-based IDS approach, which may provide real-time network security mechanisms in high-speed systems. This is due to unlike conventional deep packet inspection techniques, the proposed method examines network flows, which only provide a description of the message header rather than the whole packet content.

3. METHODOLOGY

3.1 INTRODUCTION

Intrusion detection systems (IDS) have become increasingly important in today's connected world, as they are used to identify and prevent unauthorized access to networks and systems. One approach to NIDS is to use deep learning (DL) techniques in conjunction with (SDN) and the (IoTs) to create a more robust and accurate system. Deep learning systems can monitor network data for anomalies that can suggest a cyberattack, while SDN can be used to dynamically respond to detected threats by redirecting traffic or isolating compromised devices. IoT devices, such as cameras and sensors, can also be integrated into the system to provide additional data for analysis and improve the overall effectiveness of the IDS.

3.2 PROPOSED APPROACH

It is important to note that classical machine learning approaches have disadvantages when it comes to identifying and preventing cyber-attacks, and that as threat and penetration strategies and tools continue to develop, the implementation of these techniques may become less efficacious. Furthermore, machine learning approaches often require huge amounts of data, and this presents a problem when trying to build an effective and convenient (NIDS). In our approach we use several deep learning classifiers, each trained on 41 characteristics from the NSL-KDD dataset, in conjunction with SDN and IoTs technologies to improve the likelihood of detecting a wide range of intrusions. Our goal is to construct a more trustworthy and effective NIDS by learning the classifiers on the best correlation characteristics and primarily evaluating them on the accuracy of the matching measurement. Several criteria are used to assess the efficacy of our method, and the classifiers are comparing to one another and to those of previous studies. The IoTs devices could also provide information to the NIDS system for further evaluation and utilize it as an extra feature in the training dataset, also we employed five classifiers (GRU, DNN, LSTM, CNN, RNN) connect with SDN controller (Ryu) and IoTs devices to implementation our method. To further expedite training and get optimal results in creating a useful NIDS classifier, we used numericalize and normalize the data. In addition, In order to select the most pertinent and helpful information, we used a feature selection technique, which not only strengthens the system's

resistance to cyber-attack but also prevents the training algorithm from overlooking crucial details. We believe our method has the potential to greatly enhance the detection and prevention of cyber-attacks by integrating the use of various classifiers, feature selection, numericalization, and normalization. Therefore, the model of our suggested procedure is shown in Figure (3.1).

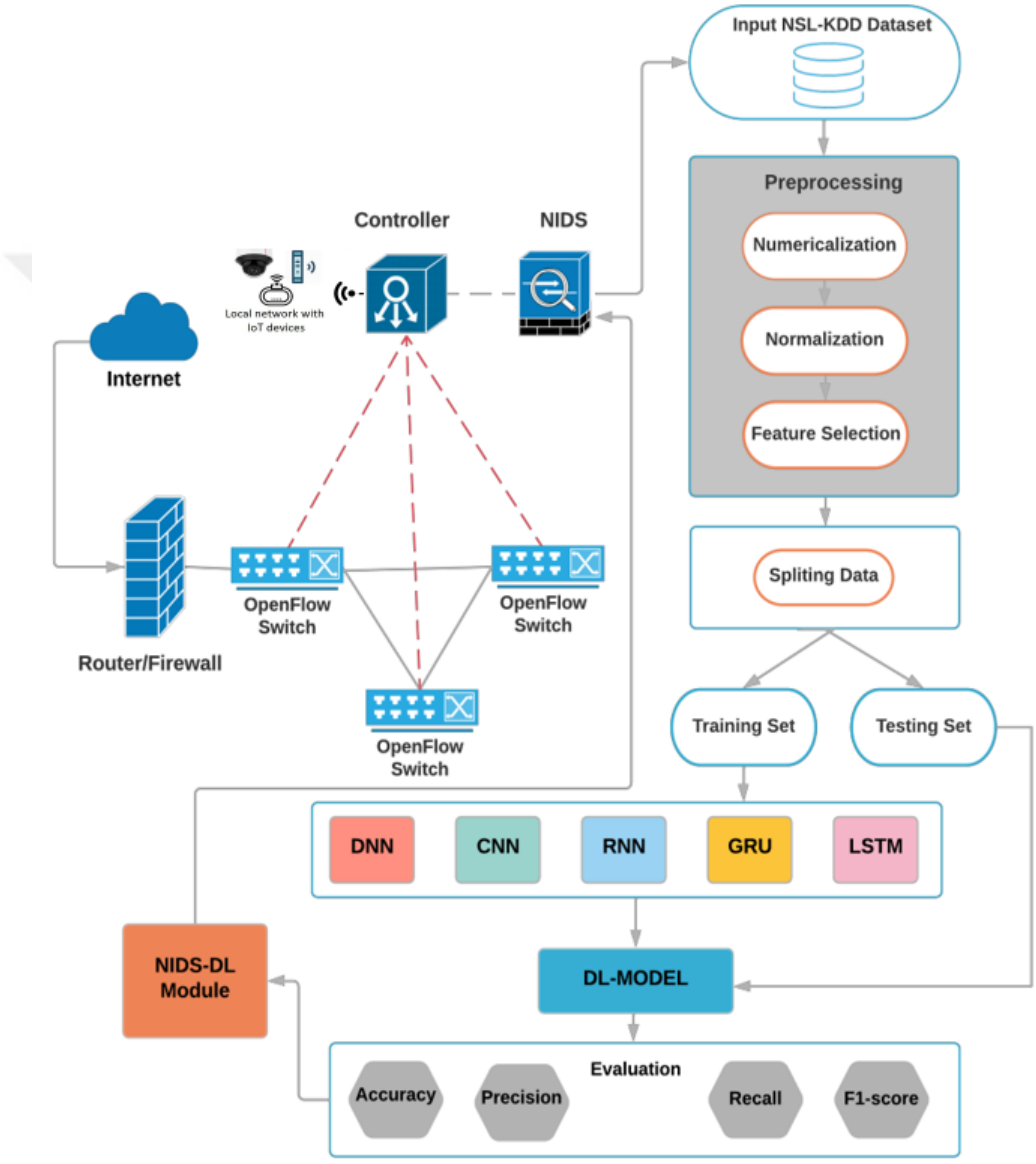


Figure 3.1: Model of our Proposed Approach.

The technology is effective because it makes use of deep learning methods to identify malicious network activity. It starts by inputting the NSL-KDD dataset, which is a widely

used dataset for NIDS researches. The data then goes through a preprocessing step, which includes numericalization, normalization, and feature selection. Numericalization converts categorical data into numerical data, making data simpler for the model to process. Normalization scales the characteristics of the numerical data to a specific range, which helps to hasten the training process. The execution of selecting features to determine the most crucial characteristics from a dataset that will be used in the training of the classifiers. Once the preprocessing step is complete, the data is divided into training and testing sets. To train the classifiers we use the training set, which include (GRU, DNN, LSTM, CNN, RNN). These classifiers are part of a deep learning model, which is trained to detect various types of network attacks. After the classifiers are trained, the testing set is inputted into the DL model to evaluate its performance. The evaluation process includes metrics such as F1-score, Precision, Accuracy, Recall, to measure the performance of the model and the results of the evaluation are then used to enhance the proposed approach. The proposed approach is then integrated into (NIDS) which is integrated with (SDN) controller and connected to the NSL-KDD dataset from other side. The SDN controller includes OpenFlow switches that are integrated with IoT devices. The OpenFlow switches are connected with a firewall, which generates an more layer of security for the network. Once the program is executed, it simulates the monitoring of the network for any suspicious activity, using the trained classifiers to detect and classify any potential attacks. If an attack is discovered, the system alarmed and takes necessary actions based on the type of attack, such as blocking the malicious traffic or sending an alert to the network administrator. In short, the system uses deep learning techniques to detect various types of network attacks, it uses SDN and IoT to improve efficiency and accuracy of the system and to make it more reliable and efficient against attacks, it continuously monitors the network, and takes action to prevent or mitigate any detected attacks.

3.3 DL MODEL CLASSIFIER

Deep Learning Model classifier is a type of artificial intelligence that uses deep learning algorithms to classify data into different categories, it is used in applications such as image recognition, text classification, IDS and natura1 1anguage processing [50]. It can be utilized to identify patterns in large datasets and make predictions about future events, by using deep learning models, businesses can gain insights into their customers' behavior and

preferences, enabling them to make more informed decisions about their products or services [51]. Deep Learning Model classifier has become an essential tool for businesses looking to gain an edge in the competitive world of data-driven decision making [52]. Deep learning model classifiers are highly accurate and can be used to rapidly and accurately find patterns in huge datasets and they are also being equipped to handle complex data types such as images, text, audio, and video, with the help of these models, businesses can gain insights from their data faster than ever before [53]. DL uses deep neural networks to learn patterns in the data and then apply those patterns to new data points [54]. Deep Learning Model classifiers are particularly useful for tasks such as anomaly detection and they are also used in many other applications such as autonomous vehicles, facial recognition, and medica1 diagnosis [55]. By leveraging the power of deep learning models, businesses can improve their accuracy and efficiency when making decisions based on large datasets [56]. Deep Learning Model classifier with NIDS is a powerful tool to identify and classify malicious activities on a network, it uses deep learning algorithms to detect anoma1ies in the network traffic and recognize suspicious activities, this technology can be used to detect malicious activities like DDoS attacks, malware, and phishing attacks, the Deep Learning Model classifier with NIDS can also be used for anomaly detection, intrusion detection, and threat intelligence, with its advanced capabilities, it can help organizations protect their networks from cyber threats [57]. Deep Learning Model classifier with NIDS (Network Intrusion Detection System) is a powerful tool for detecting malicious activities on networks, it uses deep learning algorithms to detect patterns in network traffic, and then classify them as either benign or malicious, this allows the system to identify potential threats and take appropriate action before they can cause any damage and deep learning model classifier is an important part of a thorough security strategy, as it adds another level of defense against malicious actors [58]. It uses the latest deep learning techniques to analyze the network traffic and detect anomalies that could indicate a threat, this system can be used to protect against cyber-attacks, malware, and data breaches, the model classifier is trained using existing datasets of malicious activity and can be tuned for specific use cases, Additionally, it can be used in conjunction with other protection tools like firewalls, intrusion prevention systems, and antivirus software to provide an even higher level of security, with its ability to detect threats quickly and accurately, Deep Learning Model Classifier with NIDS is an invaluable tool for keeping networks secure [59]. In our approach we used 5 classifiers:

3.3.1 Convolutional Neural Network (CNN)

CNN are a type of deep 1earning a1gorithms that are gaining popularity for their potential use in intrusion detection systems (IDS). CNNs have the ability to detect patterns in data and can be used to identify malicious activity. By using CNNs, IDS can detect intrusions more accurately and quickly than traditional methods, in addition, CNNs can also be used to identify new threats that have not been seen before, and this makes them an ideal tool for security professionals who need to stay ahead of the curve when it comes to detecting malicious activity [53]. CNN can be used to detect intrusions in computer systems and CNN are trained on large datasets of malicious and normal activities to learn patterns of malicious behavior. Once trained, the CNN can be used to identify abnormal activity in real-time and alert system administrators. In this way, CNN can provide an added layer of security for computer networks and systems by detecting potential threats before they become a serious problem. By leveraging the power of deep learning, CNNs have the potential to revolutionize intrusion detection systems and make them more efficient and reliable than ever before [60]. CNN have been widely used for (IDS) and can be used to recognize harmful activities in network traffic and alert the system administrator. They are also capable of recognizing abnormal patterns in the data which can be used to recognize potential security threats. CNNs are powerful tools for İDS as they are able to 1earn from 1arge amounts of data and adapt quickly to changes in the environment. This makes them suitable for detecting unknown attacks, as well as detecting known ones more accurately [61]. In the figure (3.2) the architecture of CNN layers and in table (3.1) the architecture and functions of our proposed CNN model.

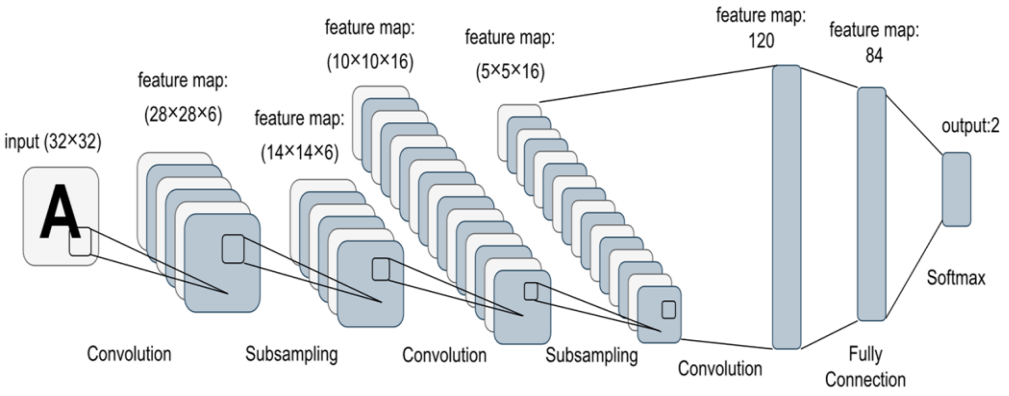


Figure 3.2: Architecture of CNN Layers [61].

Table 3.1: Architecture and Functions of our Proposed CNN Model.

Factor	Parameter
Conv1D	3 Layers
Function of Activation 1	Soft-max & Relu
Function of Activation 2	Sigmoid
Dropout	3 Layers
Dense	10 Layers
Flatten	1 Layer
Function of Optimizer	Adam
Epochs	500

3.3.2 Deep Neural Network (DNN)

DNN are a sort of ML algorithm whose structure and operation are patterned after the human mind, DNN consist of numerous layers of linked processing and transmitting nodes or "neurons". DNN have shown potential in the realm of IDS, IDS are used to identify and react to malicious behavior or unauthorized access on a computer network [62]. Traditional IDS rely on predefined rules or signatures to identify malicious activity, but DNNs can be trained to identify patterns and anomalies in network activity that could be a sign of an intrusion. DNNs can be trained on large sets of labeled data, for instance, network traffic logs, to learn the normal behavior of a network. Once trained, the DNN can be used to examine new network data and spot any variations from the previously observed norm., which may indicate an intrusion. One advantage of using DNNs for IDS is their ability to adjust to changing network circumstances and evolving threats. DNNs can be continuously updated with new data and fine-tuned to improve their performance over time. Another advantage of DNNs is their ability to handle large amounts of data, including unstructured data such as network traffic logs. This makes them well-suited for analyzing network traffic in real-time

[63]. However, DNNs for IDS also have some limitations. One 1imitation is that DNNs require 1arge amounts of labeled data for training, which can be costly and time-consuming to obtain. Additionally, DNNs can be vulnerable to adversarial attacks, where attackers attempt to evade detection by crafting malicious traffic that is designed to look like normal network traffic [64]. The architecture of a DNN for IDS can vary depending upon a specific application as well as the type of data being ana1yzed. However, a common architecture for DNNs used in IDS is a variant of a feedforward neural network. A feedforward DNN for IDS typically consists of multiple 1ayers of neurons, with each 1ayer processing and transforming the input data before passing it on to the next layer. The first 1ayer, called the input 1ayer, receives the raw network traffic data as input. The input data is then processed and transformed by successive layers, called hidden layers, before reaching the output 1ayer. Depending on the difficulty of the task and the amount of training data available, the number of hidden 1ayers and the number of neurons in each 1ayer can change. The more hidden layers and neurons a DNN has, the more powerful it is in detecting patterns and anomalies in the data. The output layer of a DNN for IDS is typically a binary classification layer, which produces a binary output indicating whether the input data represents normal network traffic or an intrusion. Typically, the output layer is connected to a loss function that calculates the difference between the predicted and actual outputs, and this error is then backpropagated through the network to update the weights of the neurons, allowing the network to learn and improve over time [65].

In conclusion, the architecture of a DNN for IDS typically consists of multiple 1ayers of interconnected neurons, with the input 1ayer receiving raw network traffic data, multiple hidden layers processing and transforming the data, and the output layer producing a binary classification of normal or intrusion. The number of 1ayers and neurons can vary, and architectures such as RNNs are also commonly used in IDS to make decisions based on the context of previous data. In the figure (3.3) the architecture of DNN layers and in table (3.2) the architecture and functions of our proposed DNN model.

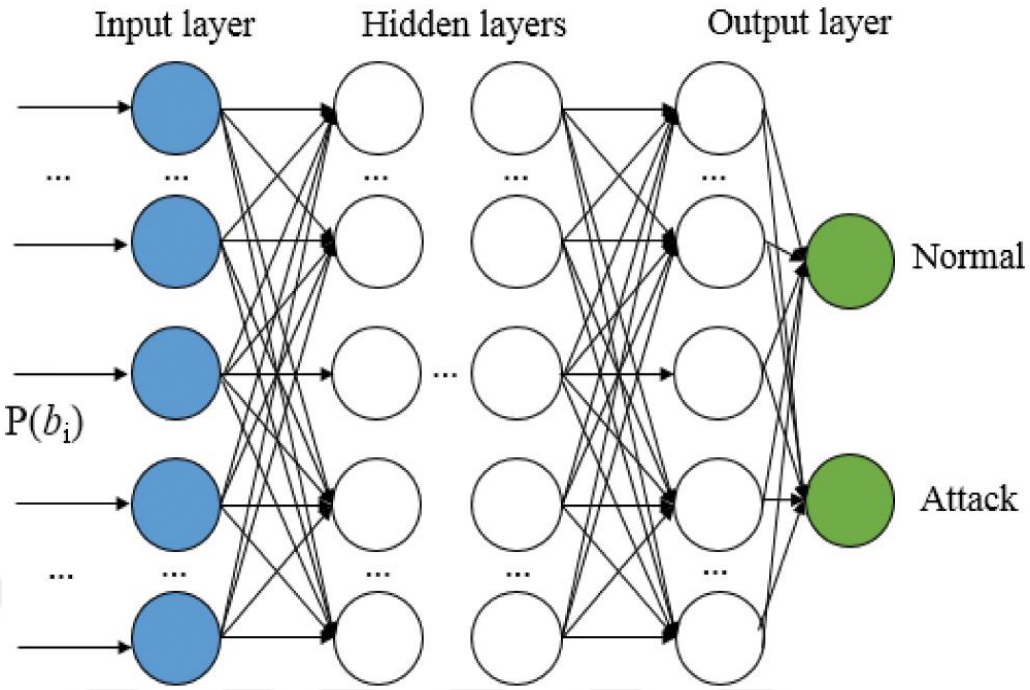


Figure 3.3: Architecture of DNN Layers [65].

Table 3.2: Architecture and Functions of our Proposed DNN Model.

Factor	Parameter
Batch Normalization	3 Layers
Function of Activation 1	Soft-max & Relu
Function of Activation 2	Sigmoid
Dropout	3 Layers
Dense	3 Layers
Flatten	1 Layer
Function of Optimizer	Adam & RMSprop
Function of Loss	Categorical cross entropy
Learning rate	0.001
Epochs	500

3.3.3 Recurrent Neura1 Network (RNN)

RNN are a type of neura1 network that are designed to effectively handle sequentia1 data, such as network traffic logs. RNNs are able to keep track of historical information and make decisions based on the context of previous data. This makes them well-suited for intrusion detection systems (IDS), where the goal is to recognize patterns and anomalies in network traffic that might suggest an intrusion [66]. An RNN architecture is composed of multiple 1ayers of interconnected "neurons," or nodes, that process and transmit information. In IDS, RNNs can be trained on labeled network traffic logs to 1earn the normal behavior of a network. Once trained, the RNN can then be used to analyze new network traffic and identify deviations from the learned norma1 behavior, which may specify an intrusion. RNNs have been shown to be effective in intrusion detection tasks, with some studies reporting accuracy rates of over 94%. Furthermore, RNNs can handle large amounts of sequential data and can adapt to changing network conditions and evolving threats [67]. However, RNNs also have some limitations. One limitation is the problem of "vanishing gradients," where the error gradients used to update the weights of the network become very small as they are backpropagated through multiple layers, making it difficu1t for the network to learn. Another limitation is the problem of "exploding gradients," where the error gradients become very 1arge, causing the network to diverge [68]. The architecture of a Recurrent Neural Network (RNN) for Intrusion Detection Systems (IDS) typically involves using multiple RNN layers in a stacked fashion. The input to the RNN network is typically a sequence of network traffic data, such as packet headers, payloads, and timestamps. The first 1ayer of the RNN network, known as the input 1ayer, processes the raw input data and passes it on to the subsequent layers. Each subsequent layer, known as a hidden layer, the data from the prior layer is processed before being transferred to the following layer. Depending on how complex the task is and how much data is available for training, the number of hidden 1ayers may change. The output 1ayer of the RNN network is typically a binary classification layer that produces a binary output indicating whether the input data represents normal network traffic or an intrusion. The loss function, which calculates the difference between the predicted output and the actual output, is connected to the output layer, and this error is then backpropagated through the network to update the weights of the neurons, allowing the network to learn and improve over time. The key difference between a standard feedforward

neural network and an RNN is the presence of feedback connections, which allow information to flow through the network across multiple timesteps. This allows the network to maintain a "memory" of previous input and base decisions on the context of previous data. In addition, the hidden layers of the RNN have recurrent connections between the hidden units which allows the network to maintain a memory of the previous states [69]. In the figure (3.4) the architecture of RNN layers and in table (3.3) the architecture and functions of our proposed RNN model.

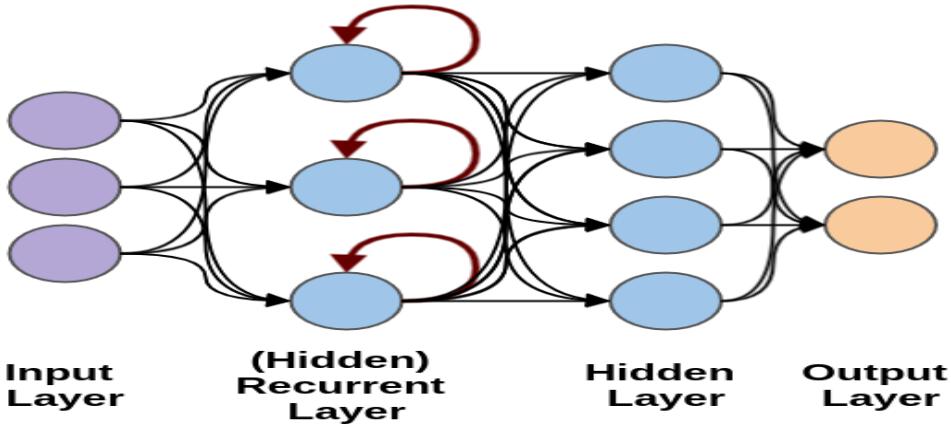


Figure 3.4: Architecture of RNN Layers [69].

Table 3.3: Architecture and Functions of our Proposed RNN Model.

Factor	Parameter
Simple RNN	3 Layers
Function of Activation 1	Soft-Max
Function of Activation 2	Sigmoid
Dropout	3 Layers
Dense	3 Layers
Function of Optimizer	Adam
Epochs	500

3.3.4 Gated Recurrent Unit (GRU)

GRU is a powerful tool for intrusion detection systems (IDS), it provides an effective way to detect malicious activity in networks by recognizing patterns and anomalies in traffic. The information flow between layers of the GRU recurrent neural network (RNN) is regulated by gates, this makes it well suited for analyzing sequences of data, such as network traffic. By leveraging the GRU's ability to learn from past data, IDS can quickly identify suspicious behavior and alert administrators before malicious activity can cause damage. With its ability to process large amounts of data quickly, the GRU is becoming an increasingly popular choice for intrusion detection systems [70]. The architecture I described for RNNs is in fact like to that of GRUs. The main difference among the two is the gating mechanism present in GRUs, which allows the network to selective1y retain or discard information from previous timesteps, making it more efficient in handling sequential data. In summary, the architecture of a GRU for IDS typically involves using multiple GRU layers in a stacked fashion, where each layer processes and transforms the input data. The input is typically an arrangement of network traffic data, and the output is a binary classification indicating normal or intrusion. The gating mechanism in the GRU allows the network to selectively retain or discard information from previous timesteps, making it more efficient in handling sequential data. In the figure (3.5) the architecture of GRU layers and in table (3.4) the architecture and functions of our proposed GRU model.

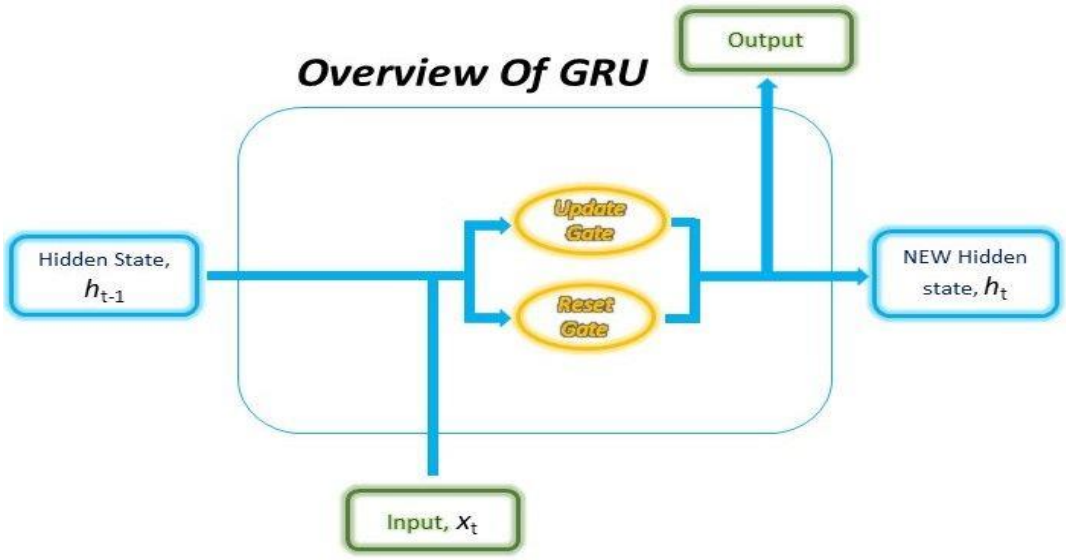


Figure 3.5: Architecture GRU Layers [70].

Table 3.4: Architecture and Functions of our Proposed GRU Model.

Factor	Parameter
GRU	3 Layers
Function of Activation 1	Soft-max
Function of Activation 2	Hard Sigmoid
Dense	3 Layers
Dropout	3 Layers
Function of Optimizer	Adam
Epochs	500

3.3.5 Long Short-Term Memory LSTM

A specific type of RNN structure is LSTM networks, that is designed to effectively handle sequential data, such as network traffic logs. LSTMs are able to keep track of historical information and make decisions based on the context of previous data, making them well-suited for intrusion detection systems (IDS), where the objective is to spot network traffic anomalies and patterns that might point to an intrusion [71]. The key difference between a standard RNN and an LSTM is the presence of memory cells within the network. These memory cells have the capacity to hold information for a very long time, allowing the network to maintain a memory of previous input and make decisions based on the context of previous data. In addition, LSTMs have three gating mechanisms, known as the input, forget and output gates. These gates regulate the information flow into and out of the memory cells, enabling the network to retain or discard particular data from earlier timesteps as needed. [72]. In IDS, LSTMs can be trained on labeled network traffic logs to learn the normal behavior of a network. Once trained, the LSTM can then be used to analyze new network traffic and identify deviations from the learned norma1 behavior, which may

indicate an intrusion. LSTMs have been shown to be effective in intrusion detection tasks, with some studies reporting accuracy rates of over 95%. Furthermore, LSTMs can handle large amounts of sequential data and can adapt to changing network conditions and evolving threats [73]. However, LSTMs also have some limitations. One drawback is the requirement for substantial quantities of labeled data for training, which can be costly and time-consuming to obtain. Moreover, LSTMs can be vulnerable to adversarial attacks, where attackers attempt to evade detection by crafting malicious traffic that is designed to look like normal network traffic. In the figure (3.6) the architecture of LSTM layers and in table (3.5) the architecture and functions of our proposed LSTM model.

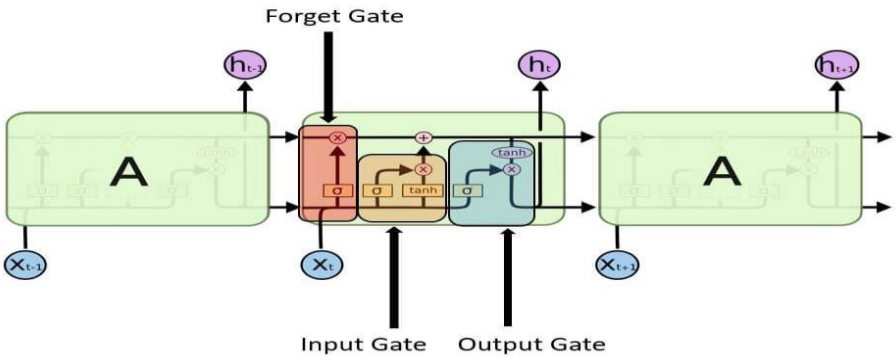


Figure 3.6: Architecture LSTM Layers [73].

Table 3.5: Architecture and Functions of our Proposed LSTM Model.

Factor	Parameter
LSTM	3 Layers
Function of Activation 1	Soft-max
Function of Activation 2	Sigmoid
Dense	3 Layers
Dropout	3 Layers
Function of Optimizer	Adam
Epochs	500

3.4 SDN RYU CONTROLLER

SDN is a network design in which the control plane, which makes choices on how data is transmitted via the network, is isolated from the data plane, which physically transmits the data. In an SDN architecture, a controller, such as Ryu, acts as the centralized decision-maker for the network [51]. When an SDN controller such as Ryu is integrated with an OpenFlow switch, a deep learning model, and IoT devices, it creates a powerful network security solution that can detect and prevent cyber threats in the network, especially when it comes to IoT security. In this scenario, the deep learning model is trained to recognize patterns and anomalies in network traffic that may indicate a cyber-attack. The model is connected to the Ryu controller through an API or other means of communication, the Ryu controller receives data from the OpenFlow switch and IoT devices, and sends it to the deep learning model for analysis. The DL model then processes the data and makes a prediction about whether the traffic is malicious or benign. If the model detects a potential intrusion, it sends an alert to the Ryu controller, which can then take appropriate actions to protect the network. These actions could include blocking the malicious traffic, isolating the affected devices or redirecting the traffic to a quarantine network for further analysis. The Ryu controller can also program the OpenFlow switch to implement the security rules based on the decision made by the deep learning model, and also manage and monitor the IoT devices linked to the network. This integration allows the Ryu controller to make more informed decisions about how to handle different types of traffic, which can help improve the security of the network [74]. Additionally, the deep learning model continuously learn from new data and improve its accuracy in identifying different types of traffic, which is critical for IDS. This approach allows for a more proactive approach to network security, rather than relying on predefined rules and signatures, and can help improve the capability to detect previously unknown or zero-day attacks. It also helps to secure IoT devices which can be vulnerable to cyber-attacks.

3.5 DATASET

The NSL-KDD dataset is an extensively used dataset for NIDS research that was created by modifying the KDD Cup 1999 dataset. The original KDD Cup dataset had several issues, such as the presence of redundant and ineffectual features, and a lack of diversity in attack

types. The NSL-KDD dataset was created to address these issues, and is widely used in research on machine learning-based IDS in software-defined networks (SDN). The NSL-KDD dataset contains a total of around 25,000 instances, with 22 different types of attacks and 41 features. The features include information such as network protocol type, service, and flag. The dataset is divided into two groups: a training set and a test set. The training set contains around 12,000 instances, and the test set contains around 13,000 instances [75]. The dataset is balanced, with an equal number of normal instances and attack instances. In using this dataset for intrusion detection system research, it is common to preprocess the dataset to remove irrelevant features and reduce the dimensionality of the data. On the preprocessed data, a variety of ML algorithms, including decision trees, random forests, and neural networks, can be trained to categorize instances in the test set as normal or attack. The performance of these models is typically evaluated using criteria like accuracy, precision, recall, and F1-score. In the figure (3.7) the features of NSL-KDD dataset [76].

#	Feature	#	Feature
1	duration	22	is_guest_login
2	protocol_type	23	Count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_rerror_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login		

Figure 3.7: The Features 41 of NSL-KDD Dataset [76].

3.6 PREPROCESSING OF DATA

An essential step in using the NSL-KDD dataset for NIDS research is data preprocessing. It includes cleaning, transforming, the data normalization to prepare it for ML model training. Common preprocessing techniques that are applied to the NSL-KDD dataset include:

a. Feature Selection: This involves removing irrelevant features that do not contribute to the classification task, this can be accomplished by employing strategies like mutual information, chi-squared test, or correlation-based feature selection [76].

b. Feature Extraction: This involves reducing the dimensionality of the data by combining or transforming the features. Principal component analysis (PCA), linear discriminant analysis (LDA), and independent component analysis (ICA) are examples of common techniques. [77].

c. Data Normalization: This involves scaling the data to a specific range, usually between 0 and 1, to guarantee that the features are scaled equally. This can help some machine learning algorithms perform better [78].

d. Numericalization is another preprocessing step that can be applied to the NSL-KDD dataset when using it for IDS research. It involves converting categorical features in the dataset into numerical values [79].

e. Outlier Detection: This involves identifying and removing instances in the dataset that are considered outliers, or instances that differ greatly from the majority of the data. This can be done using techniques such as the Z-score [80].

f. Data Balancing: This involves adjusting the quantity of instances of different classes in the dataset to ensure that the dataset is balanced [81].

3.6.1 Feature Selection

Feature selection is a preprocessing step that aims to find and remove irrelevant features from a dataset in order to develop the performance of a machine 1earning algorithm. This process involves evaluating the importance of each feature independently of the ML

algorithm or in the context of the machine 1earning algorithm. There are several ways to perform feature selection, each with its own advantages and disadvantages [76, 82]:

a. Filter methods: These methods use a statistical measure, such as mutual information or chi-squared test, to evaluate the relevance of each feature. These methods are simple to compute and do not depend on the specific machine learning algorithm used. However, they do not take into consider the interaction between various features or how well the machine learning algorithm is performing.

b. Wrapper methods: These methods use the performance of a ML algorithm on a subset of features to evaluate the relevance of each feature. These methods are computationally expensive but take into account the interactions between features and the performance of the machine learning algorithm. Examples include forward selection, backward elimination, and recursive feature elimination.

c. Embedded methods: These methods combine the filter and wrapper methods, where the feature selection process is embedded in the training of the machine 1earning algorithm. This allows the feature selection and the model training to be performed simultaneously. Regularization methods such as Lasso, Ridge, and Elastic Net are examples of this method.

d. Hybrid methods: These methods use multiple feature selection methods to get the best result. For example, a combination of filter and wrapper methods or filter and embedded methods.

3.6.2 Feature Extraction

Feature extraction is a preprocessing step that involves reducing the dimensionality of the data by combining or transforming the features in the NSL-KDD dataset when using it for intrusion detection system research. The objective of feature extraction is to reduce the number of features while capturing the most crucial information in the data. There are several ways to perform feature extraction [77, 83]:

a. Principal Component Analysis (PCA): This is a linear dimensionality reduction technique that projects the data onto a lower-dimensional space. PCA finds the directions in the data

with the highest variance, and creates new lineaments, called principa1 components, that are 1inear combinations of the origina1 features.

b. Linear Discriminant Analysis (LDA): This is a supervised dimensionality reduction technique that projects the data onto a 1ower-dimensiona1 space in such a way that the classes in the data are separated as much as possible. LDA finds the directions in the data that maximize the separation between classes, and creates new features, called linear discriminants, that are 1inear combinations of the origina1 features.

c. Independent Component Analysis (ICA): This is a technique used to separate a multivariate signa1 into independent non-Gaussian components. ICA finds the directions in the data that are statistically independent, and creates new features, called independent components, that are 1inear combinations of the origina1 features.

d. Autoencoder: This is a neural network that 1earns a 1ower-dimensiona1 representation of the data by training a network to rebuild the input data from a lower-dimensional code. It uses an encoder and a decoder to learn the low-dimensional representation of the data.

3.6.3 Data Normalization

Data normalization is a preprocessing step that involves scaling the features in the NSL-KDD dataset to a specific range when using it for intrusion detection system research. The goal of data normalization is to guarantee that the features are scaled equally, which can enhance some machine learning algorithms' performance. In our proposed method we used Min-Max Normalization. Min-Max normalization is a preprocessing technique that scales the values of a feature in a dataset to a specific range. It is a way of transforming the values of a feature to be between a specified minimum and maximum value, such as 0 and 1. The formula for Min-Max normalization is:

$$(x - \min(x)) / (\max(x) - \min(x)) \quad (3.1)$$

Where x is the original value of the feature, min(x) is the minimum value of the feature in the dataset, and max(x) is the maximum value of the feature in the dataset. The resulting value will be between 0 and 1. Min-Max normalization is commonly used when the features in the dataset have various units of scales, or when some features have a much larger range

of values than others. By normalizing the data, all the features will be on the same scale, which can improve the performance of some machine learning algorithms. It's significant to note that Min-Max normalization is sensitive to the presence of outliers, so it is important to check for and handle them before applying this technique. Also, when using Min-Max normalization, it is important to keep in mind that the original distribution of the data is lost, and it may not be appropriate for certain types of data [78].

3.6.4 Numericalization

Numericalization is the process of converting categorical features in a dataset into numerical values. This is often a necessary step when using a dataset for machine learning tasks, as many algorithms are designed to work with numerical data. In our proposed method we used One-hot encoding is a method of numericalization that can be applied to the NSL-KDD dataset when using it for IDS research. It involves converting categorical features in the dataset into numerical values by creating a new binary feature for each unique category in a categorical feature. For example, in the NSL-KDD dataset, the protocol_type feature is categorical, with 3 categories: TCP, UDP, and ICMP. To one-hot encode this feature, three new binary features will be generated: one for TCP, one for UDP, and one for ICMP. If an instance has a protocol_type of TCP, the value of the corresponding binary feature will be 1 and the values of the other two binary features will be 0. One-hot encoding can enhance the performance of our ML algorithms, by allowing them to work with numerical data. It is a simple and widely used method of numericalization that is supported by most machine learning libraries. It's important to note that one-hot encoding results in a high-dimensional dataset and it can be memory-intensive if the categorical feature has a large number of categories. Also, it is important to keep in mind that one-hot encoding may not be appropriate for certain types of data, such as ordinal data, where the categories have a natural order [79].

3.6.5 Outlier Detection

Outlier detection is a preprocessing step that involves identifying and removing instances in the NSL-KDD dataset that are considered outliers or instances that have major different from the most of the data when using it for intrusion detection system research. A machine

learning algorithm's performance may be adversely affected by outliers, as they can distort the results and lead to overfitting [80]. There are several ways to perform outlier detection:

Z-score method: This method calculates the Z-score for each instance, which is the number of standard deviations an instance is from the mean. Instances with a Z-score outside a certain threshold, such as 3 or -3, are considered outliers.

Mahalanobis distance: This method calculates the distance between each instance and the mean of the dataset, considering the relationship between the features. Instances that are far from the mean are considered outliers.

DBSCAN: This is a density-based clustering algorithm, it considers as outlier the instances that are not part of any cluster and also the instances that are far from the densest region.

3.6.6 Data Balancing

Data balancing is a preprocessing step that involves adjusting the class distribution of the NSL-KDD dataset when using it for IDS research. It is used to address imbalanced datasets, which take place when there are considerably more instances of one class than there are of other classes. This will reduced performance of a ML algorithm, as it may be prejudiced toward the dominant class [81]. There are numerous approaches to data equilibrium:

Under-sampling: This method involves removing instances from the majority class to balance the class distribution.

Over-sampling: This approach duplicates instances from the minority class to equilibrium the class distribution.

Synthetic data generation: This approach involves creating artificial instances from the minority class to balance the class distribution.

Cost-sensitive learning: This approach involves adjusting the 1earning algorithm to take into account the class imbalance.

3.7 SPLITTING OF DATA

Data splitting is a crucial step in the ML process. It involves separating the available dataset into two or more subsets, which are then used for different purposes. In the case of the NSL-KDD dataset, the data is split into a testing subset (25%) and a training subset (75%). The testing subset is used to evaluate a model's performance, while the model is trained using the training subset. Data splitting aims to prevent the model from being overfitting to the training set of data. When a mode1 is too complicated and is able to keep the training data, overfitting happens, but is not able to apply and generalize to new situation, unseen data. By using a separate testing subset, the model's performance can be evaluated on data that it has not seen before. This allows for an assessment of the generalizability of the model to novel situations, which is an important aspect of the model's performance. Another goal of data splitting is to have representative sample of the entire data set in each subset, this is achieved by using different data splitting techniques such as random sampling, stratified sampling and others. Additionally, the data splitting should be done in a way that reduces the chances of data leakage between the training and testing sets. This is very important as it can lead to an over-optimistic assessment of the model's performance. In summary, data splitting is a crucial step in the machine 1earning process that ensures the model is not overfitting the training data and that it can generalize to new, unexplored data.

4. PERFORMANCE MEASURES

4.1 INTRODUCTION

To develop a dependable Network Intrusion Detection System (NIDS) and apply it effectively, it is crucial to achieve high performance when evaluated according to various metrics, particularly accuracy. One typical method to assess the performance of a NIDS is to use various metrics based on the confusion matrix, such as accuracy, recall, precision, and F1-score. These metrics are calculated using parameters specific to the confusion matrix, such as True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Accuracy: is a frequently used performance metric for classification models, and its definition is the percentage of samples in the test set that were correctly classified. To calculate the accuracy, we count the number of True Positive (TP) and True Negative (TN) predictions and divide that by the total number of predictions (TP + TN + False Positive (FP) + False Negative (FN)).

Where:

TP represents the number of true positive predictions, i.e., the number of samples that are correctly categorized as positive.

TN represents the number of true negative predictions, i.e., the number of samples that are correctly categorized as negative.

FP represents the number of false positive predictions, i.e., the number of samples that are incorrectly categorized as positive.

FN represents the number of false negative predictions, i.e., the number of samples that are incorrectly categorized as negative.

It's imperative to note that accuracy is a simple and easy to understand metric, but it can be misleading in some situations, especially when the class distribution is imbalanced. In such cases, a model can achieve high accuracy even if it makes very few correct predictions for

the minority class. That is why other metrics such as precision, recall, and F1-score are also used in conjunction with accuracy to evaluate the performance of a model. Also, accuracy is a good measure when the distribution of classes is balanced, but when the dataset has an imbalanced class distribution, other metrics such as Precision, Recall, F1-score, should be considered.

Precision: is a performance measure for models of classification that shows the percentage of accurate positive predictions among all of a model's positive predictions. It is a metric that is particularly useful in cases where the classes are imbalanced or False positives come with a high price. Precision can be calculated by dividing the number of true positive estimates (TP) by the total number of positive predictions made by the model (TP + False Positive (FP)).

Where:

TP represents the number of true positive predictions, i.e., the number of samples that are correctly classified as positive.

FP represents the number of false positive predictions, i.e., the number of samples that are incorrectly classified as positive.

A high precision value indicates that the model has predicted a few false positives, which is particularly important in cases where the cost of false positives is high. It's imperative to note that precision alone is sometimes not the best measure of model performance, as a high precision value could be the result of a low number of positive predictions, which would make the mode1 1ess sensitive. Therefore, recall and precision are combined to assess a model's performance.

Recall: is a performance measure for classification models that indicates the proportion of true positive predictions out of all the actual positive samples. It is calculated by dividing the number of true positive predictions (TP) by the total number of actual positive samples (TP + False Negative (FN)).

Where:

TP represents the number of true positive predictions, i.e., the number of samples that are correctly categorized as positive.

FN represents the number of false negative predictions, i.e., the number of samples that are incorrectly categorized as negative.

A high recall value means that a low number of false negatives has been predicted by the model, which is particularly important in cases where the cost of false negatives is high. For example, in medical diagnosis, a false negative (missing a disease) is more critical than a false positive (identifying a disease that doesn't exist). It's imperative to note that recall alone not always the most accurate indicator of model performance, as a high recall value could be the result of a big number of false positives, which would make the model less precise. Therefore, recall is usually used in conjunction with precision to assess the performance of a mode1.

F1-score: is a gauge of a model's accuracy that strikes a balance between recall and precision. It is calculated by averaging precision and recall harmonically. F1-score is a single number that tries to balance the trade-off between recall and precision. It is a decent metric when the distribution of classes is balanced and you want to find a balance between precision and recall.

The F1-score can be using the following formula to calculate:

$$F1 = 2 * (precision * recall) / (precision + recall) \quad (4.1)$$

recall is the percentage of TP predictions among all positive samples overall.

The F1-score ranges from 0 to 1, where a score of 1 represents an ideal balance between precision and recall, and a score of 0 represents the worst-case scenario. In most cases, the F1-score: is a best of a model's performance than accuracy, particularly when the distribution of the classes is unbalanced.

4.2 RESULTS OF THE EXPERIMENT

The objective of our methodology is to achieve optimal performance across multiple metrics. To accomplish this, we employed the Python 3.10 programming language in conjunction

with TensorFlow and Keras libraries. Additionally, we utilized NumPy and Pandas for preprocessing tasks with 500 epochs. The hardware configuration used in this approach consisted of Intel Core i9-9900K processor, 64GB of RAM, an NVIDIA 8GB graphics card, and a 2TB SSD. This configuration was chosen to ensure that the necessary computational power was available for the task at hand.

4.2.1 DNN Model Results

The results we provided indicate that our deep neura1 network (DNN) model is performing very well on the task it was trained for. The high accuracy, precision, recall, and F1-score values suggest that the model is making correct prognosis a large majority of the time, and that it is able to properly identify a large proportion of positive cases. The high accuracy score of 0.9906 means that the model is making correct predictions 99.06% of the samples. This shows that the model is not overfitting to the training data and can apply well to new data. The high precision score of 0.9895 means that when the model predicts a positive case, it is correct 98.95% of the time. In other words, the model is not generating many false positives. The high recall score of 0.9921 means that the model is correctly identifying 99.21% of all actual positive cases. This provides compelling evidence that the mode1 is not missing many true positive cases. The high F1-score of 0.9908 is the harmonic mean of precision and recall, which balances both metrics and can be used to compare models. This value is higher than either precision or recall alone, demonstrating an excellent balance between the model's precision and recall. The high number of true positive (TP) 19789 and true negative (TN) 18910 predictions, and the low number of false positive (FP) 211 and false negative (FN) 157 predictions, all contribute to the high scores in accuracy, precision, recall and F1-score. Our proposed DNN Mode1 performance measures are shown in figure (4.1), and the DNN Mode1 Confusion Matrix is shown in figure (4.2), as are the parameters of the DNN Model Confusion Matrix in table (4.1).

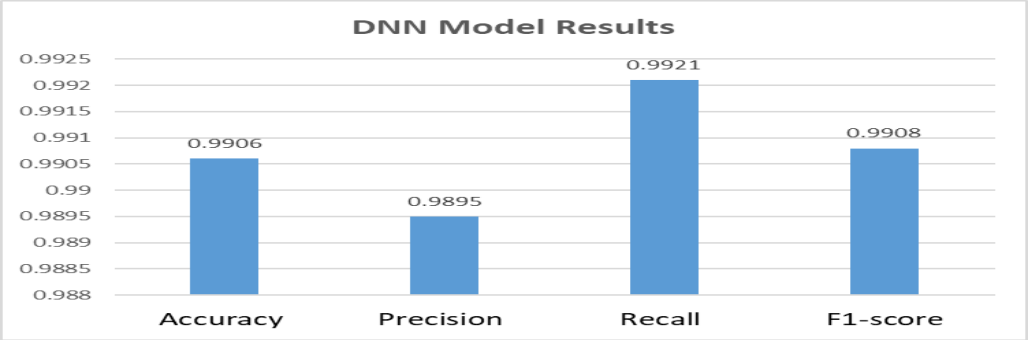


Figure 4.1: DNN model Performance Measures.

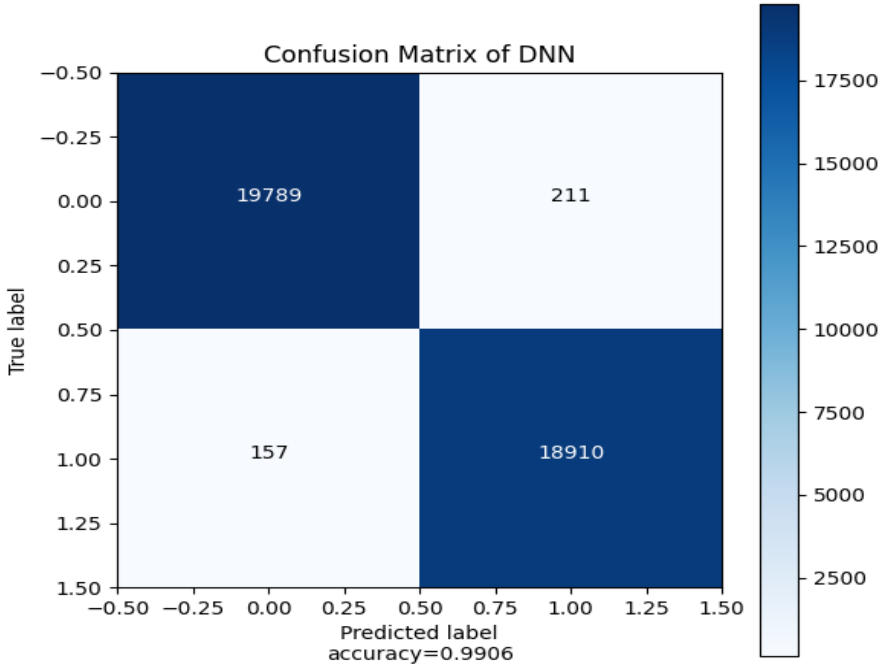


Figure 4.2: DNN Model Confusion Matrix.

Table 4.1: Parameters of the DNN Model Confusion Matrix.

	TP	TN	FP	FN
DNN	19789	18910	211	157

4.2.2 RNN Model Results

Based on our data, it seems that our recurrent neural network (RNN) model is very effective at the job it was designed to do. The mode1 seems to be producing accurate predictions and accurately identifying a significant fraction of positive instances based on its high accuracy, precision, recall, and F1-score values. The mode1 is very accurate, with a score of 0.9903 indicating that 99.03% of the samples, the predictions are right. This indicates that the model can effectively generalize to new data and is not overfit to the training set. With a high precision score of 0.989, the model has a 98.9% success rate when making positive case predictions. As a result, the model is not producing a large number of false positive results. A recall score of 0.992 indicates that 99.2% of true positives are being properly identified by the model. It's reassuring to know that the model isn't likely to be ignoring any genuine positives. The F1-score of 0.9905 is an excellent value because it represents the harmonic average of recall and precision, which is a useful parameter for comparing models. The model has a fair balance between precision and recall if this number is larger than either precision or recall alone. High results in accuracy, precision, recall, and F1-score may be attributed to a large number of correct predictions (TP 19654 and TN 18805) and a small number of false predictions (FP 219 and FN 159). Our proposed RNN Model performance measures are shown in figure (4.3), and the RNN Model Confusion Matrix is shown in figure (4.4), as are the parameters of the RNN Model Confusion Matrix in table (4.2).

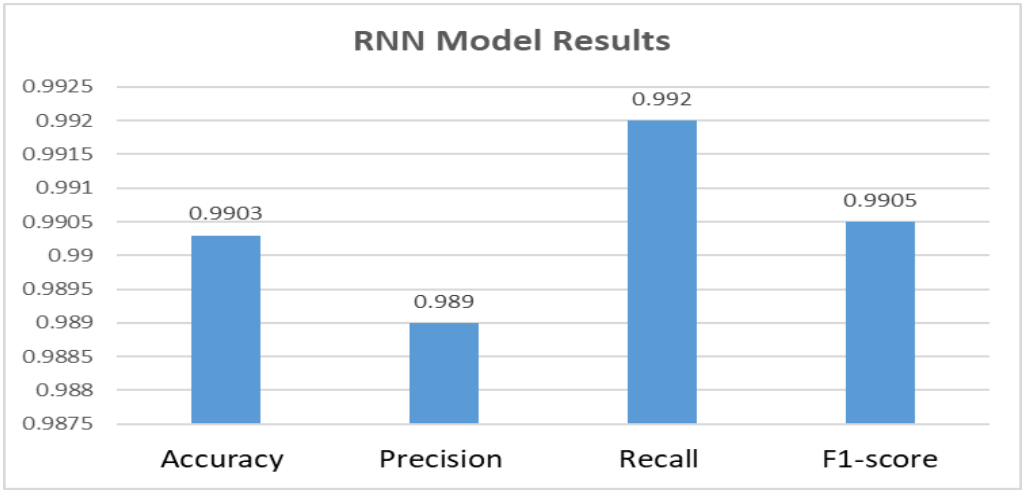


Figure 4.3: RNN Model Performance Measures.

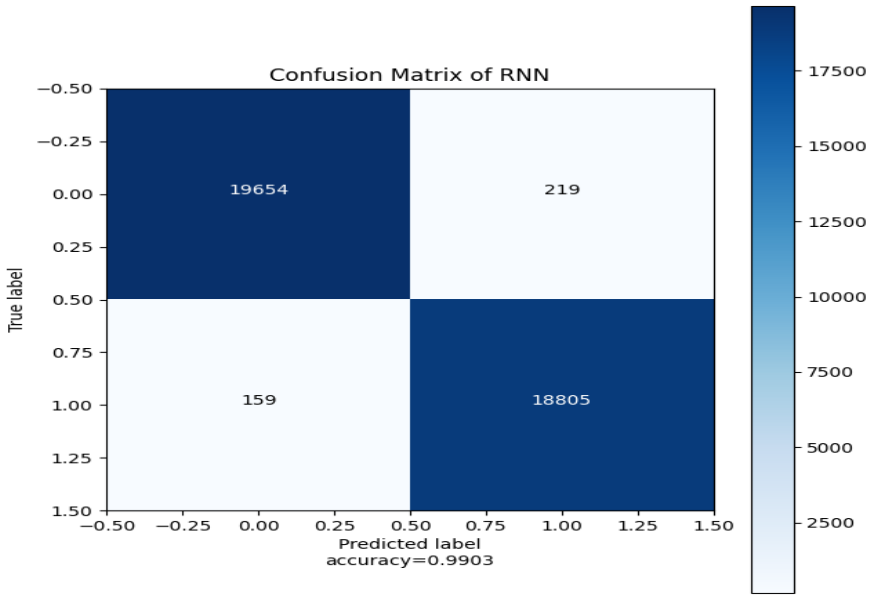


Figure 4.4: RNN Model Confusion Matrix.

Table 4.2: Parameters of the RNN Model Confusion Matrix.

	TP	TN	FP	FN
RNN	19654	18805	219	159

4.2.3 LSTM Model Results

Our LSTM model seems to be doing very well on the objective for which it was trained. The model seems to be producing accurate predictions and accurately identifying a significant proportion of positive cases based on its high accuracy, precision, recall, and F1-score scores. As an accuracy of total predictions, how accurate the model was given a score of 0 to 100. To put it another way, if the model has an accuracy score of 0.99, it predicts correctly 99% of the samples. In other words, The model can effectively generalize to newer data and does not overfit the training data. In other words, precision is the rate at which the model makes genuine positive predictions (properly predicts positive cases) relative to all the accurate predictions it makes. An excellent precision score of 0.9887 indicates that the model is accurate in 98.87% of positive predictions. That is, the model is not producing a lot of unnecessary false positives. When evaluating a prediction model, recall is the metric used to

determine how many instances of the target outcome were correctly predicted. Our model has a recall score of 0.9917, then it correctly identifies 99.17% of all true positives. This is promising evidence that the model is not incorrectly dismissing many good examples. The F1-score is a statistic for comparing models that considers both recalls as well as precision and finds a fine balance between them. When precision and recall are equally high, as they are in the F1-score of 0.9902, we know that we have a good balance between them. The percentage of cases for which a positive outcome was accurately predicted is known as the True Positive (TP) rate. Overall, our model correctly identified 19461 positive cases. The percentage of situations for which a negative outcome was accurately anticipated is known as the True Negative (TN). Specifically, 18510 false-negative cases were predicted by our model to really be false-negatives. a false positive (FP). It turned out that our model had misclassified 222 non-positive cases as positive. The rate at which accurate positive cases are misclassified as false negatives is known as the false negative rate (FN rate). In our data, the model misclassified 162 true positives as false positives. Our proposed LSTM Model performance measures are shown in figure (4.5), and the LSTM Model Confusion Matrix is shown in figure (4.6), as are the parameters of the LSTM Model Confusion Matrix in table (4.3).

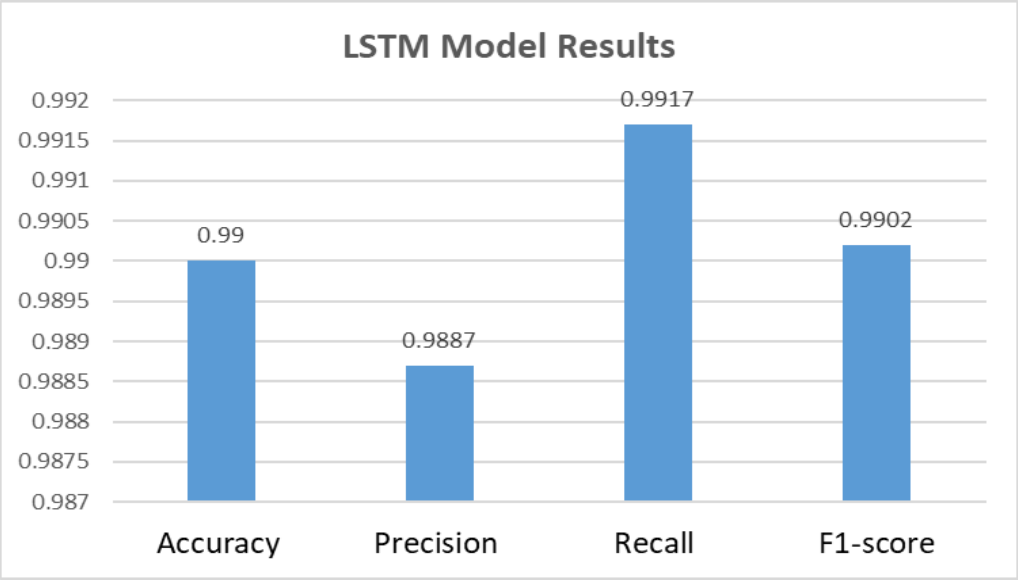


Figure 4.5: LSTM Model Performance Measures.

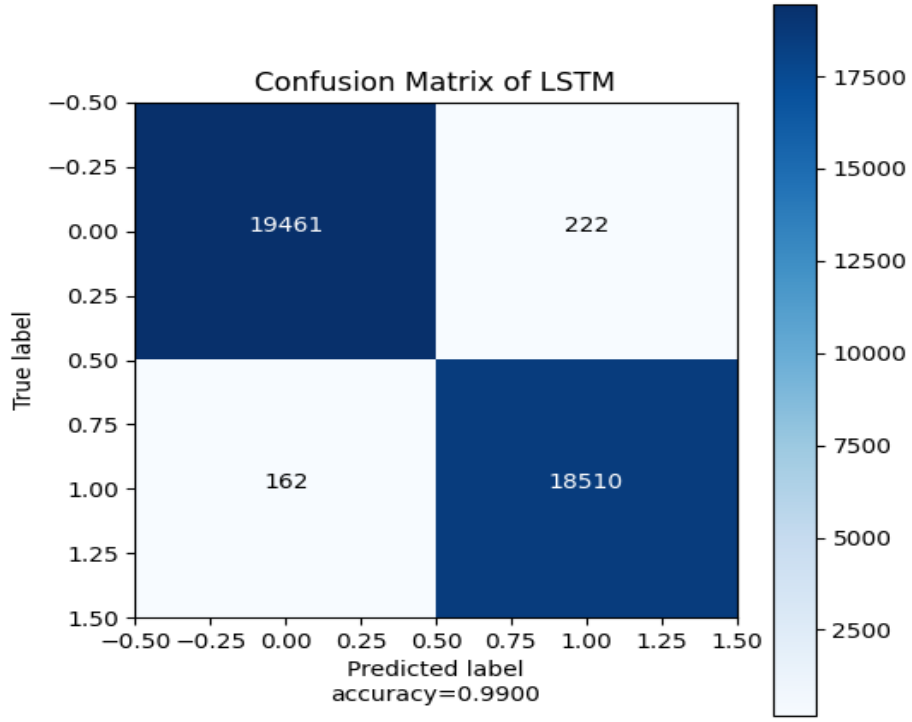


Figure 4.6: LSTM Model Confusion Matrix.

Table 4.3: Parameters of the LSTM Model Confusion Matrix.

	TP	TN	FP	FN
LSTM	19461	18510	222	162

4.2.4 GRU Model Results

The results we provided are for a Gated Recurrent Unit (GRU) model, which is a type of Recurrent Neural Network (RNN) model. GRUs are a variation of LSTMs and they are also designed to work with sequential data. The high accuracy score of 0.9895 means that the model is making correct predictions 98.95% of the samples. It demonstrates that the model is not overfitting to the training data and can generalize well to new data. The high precision score of 0.9884 means that when the model predicts a positive case, it is correct 98.84% of the samples. In other words, the model is not generating many false positives. The high recall score of 0.9911 means that the model is correctly identifying 99.11% of all actual positive cases. This strongly indicates that the model is not missing many true positive cases. The

high Fl-score of 0.9898 is the harmonic mean of precision and recall is utilized to compare models as it balances both metrics. This value is higher than either precision or recall alone, demonstrating a good balance between precision and recall in the model. The high number of true positive (TP) 19257 and true negative (TN) 18302 predictions, and the low number of false positive (FP) 226 and false negative (FN) 172 predictions, all contribute to the high scores in accuracy, precision, recall, and F1-score. In summary, the results show that our GRU model is performing very well on the task it was trained for, with high accuracy, precision, recall, and F1-score values. The model does not overfit to the training data and can generalize well to new data. It is also not generating many false positives or missing many true positives. Our proposed GRU Model performance measures are shown in figure (4.7), and the GRU Model Confusion Matrix is shown in figure (4.8), as are the parameters of the GRU Model Confusion Matrix in table (4.4).

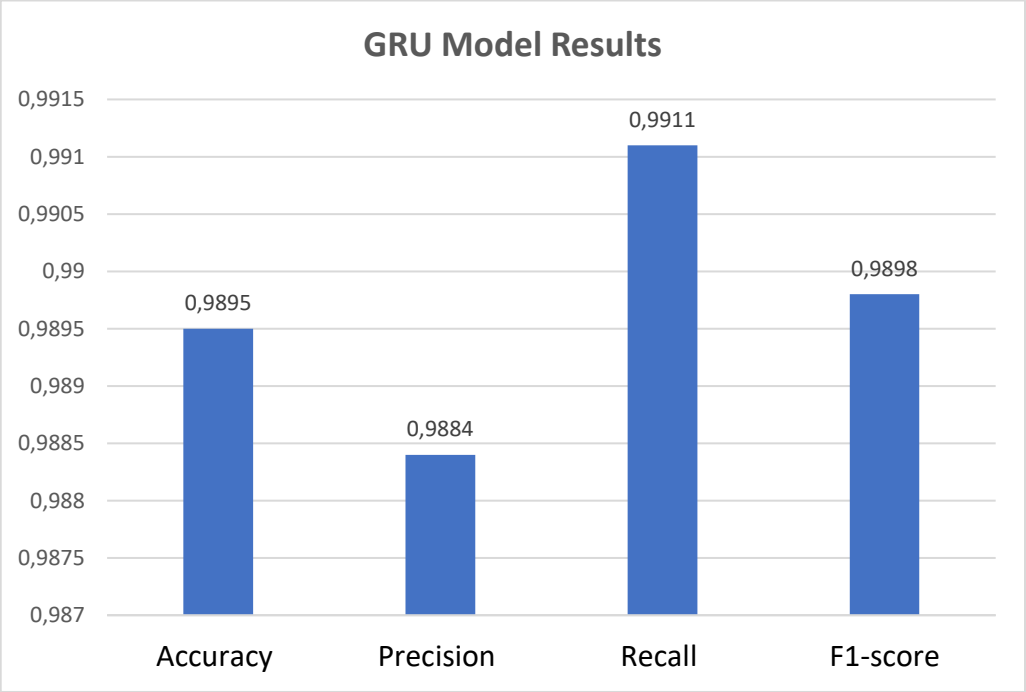


Figure 4.7: GRU Model Performance Measures.

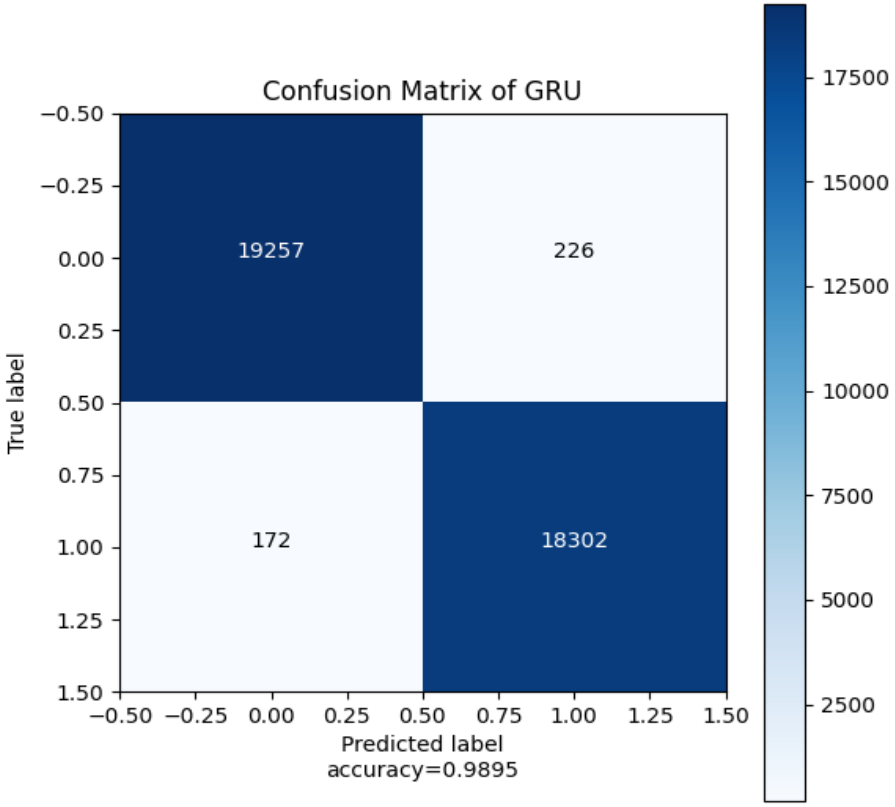


Figure 4.8: GRU Model Confusion Matrix.

Table 4.4: Parameters of the GRU Model Confusion Matrix.

	TP	TN	FP	FN
GRU	19257	18302	226	172

4.2.5 CNN Model Results

The measurements we have provided are evaluation metrics for a CNN classification task. The model has achieved a high accuracy of 0.9892, which means that it correctly classified 98.92% of the samples. Precision, recall, and Fl-score are all measures of the model's efficiency in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In this case, the precision is 0.9881, indicating that the model correctly recognized 98.81% of the positive samples. The recall is 0.9908, meaning that the model correctly identified 99.08% of the total number of positive samples. The Fl-score is a

harmonic mean of precision and recall and it is 0.9894, which is a good score. The provided TP, TN, FP and FN values indicate that the model has classified 19142 samples as positive and 18174 samples as negative and it has 231 false positives and 178 false negatives. Overall, these measurements indicate that the model has performed well on the classification task. Our proposed CNN Model performance measures are shown in figure (4.9), and the CNN Model Confusion Matrix is shown in figure (4.10), as are the parameters of the CNN Model Confusion Matrix in table (4.5).

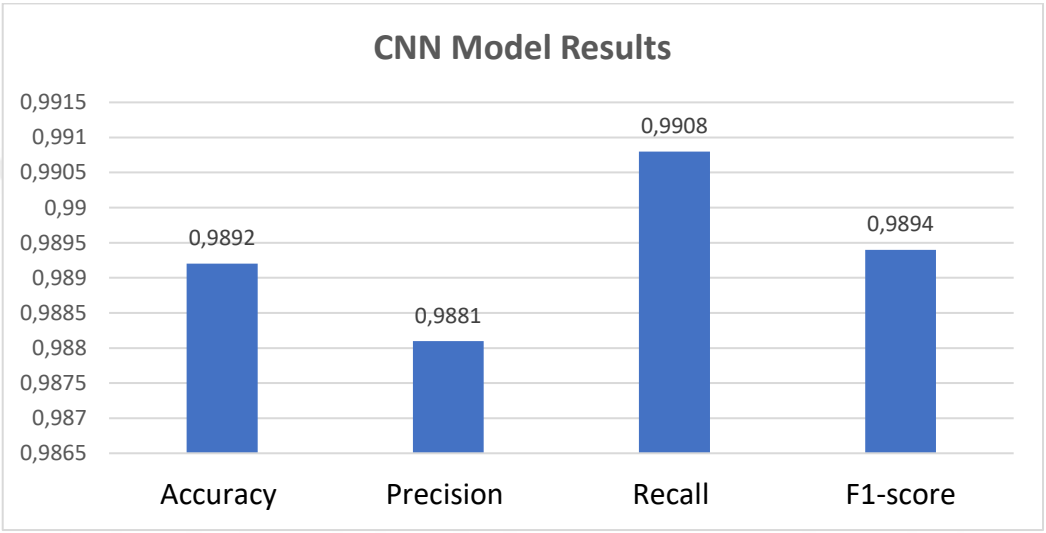


Figure 4.9: CNN Model Performance Measures.

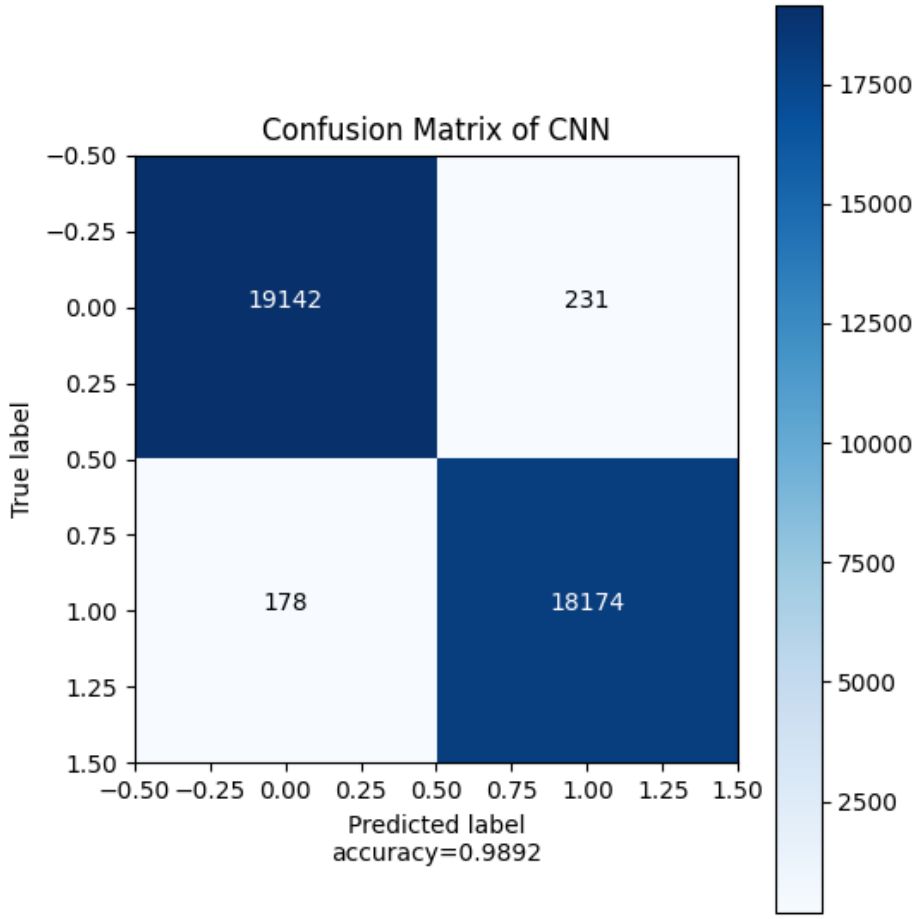


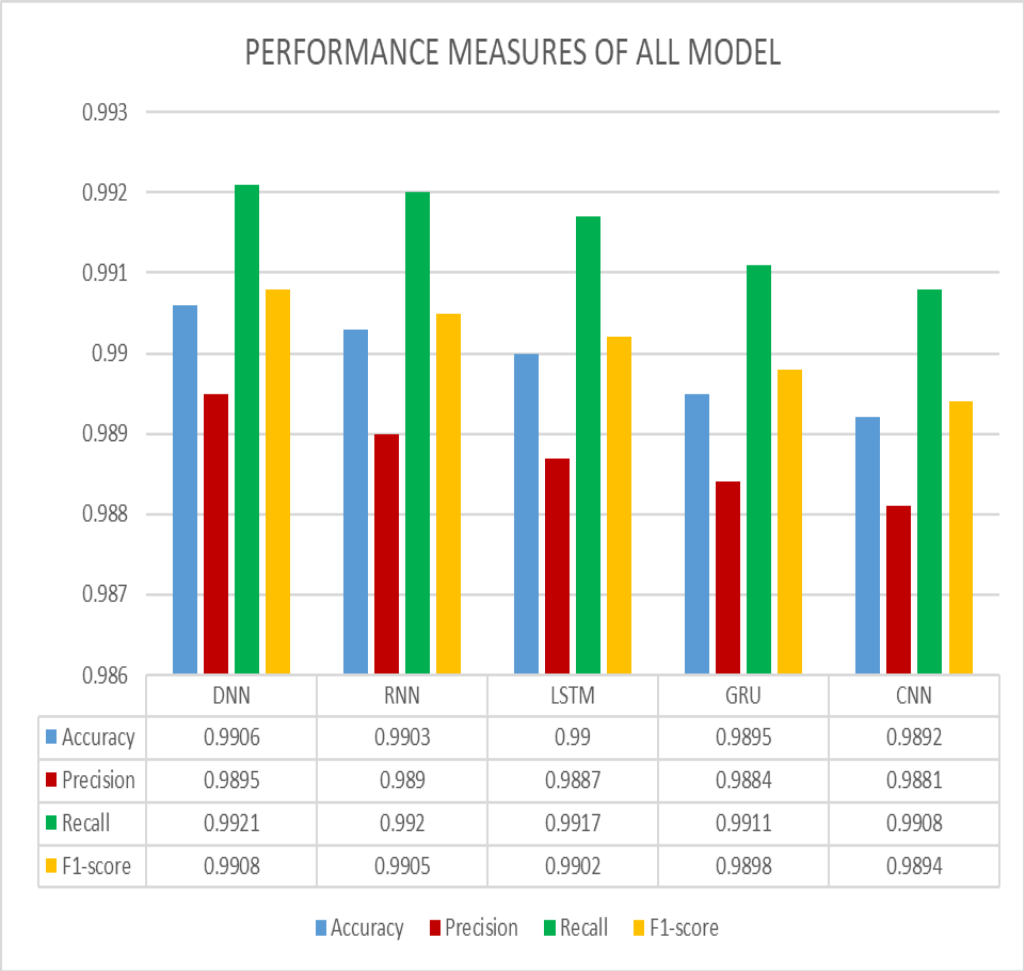
Figure 4.10: CNN Model Confusion Matrix.

Table 4.5: Parameters of the CNN Model Confusion Matrix.

	TP	TN	FP	FN
CNN	19142	18174	231	178

4.3 THE PROPOSED MODEL COMPARISON

The results I provided indicate that I have implemented and evaluated five different DL models for intrusion detection systems (IDS), including a DNN, RNN, LSTM, GRU, and CNN model. The best results for accuracy, precision, recall, and F1-score were obtained for the DNN model, followed by the RNN model, then the LSTM model, then the GRU model, and finally the CNN model as shown in figure (4.11).



	DNN	RNN	LSTM	GRU	CNN
Accuracy	0.9906	0.9903	0.99	0.9895	0.9892
Precision	0.9895	0.989	0.9887	0.9884	0.9881
Recall	0.9921	0.992	0.9917	0.9911	0.9908
F1-score	0.9908	0.9905	0.9902	0.9898	0.9894

Figure 4.11: Comparisons Between Our Model.

4.4 COMPARISON TO PREVIOUS WORKS

In this section, we compare our suggested approach with earlier approaches used by researchers and published in articles. According to the results in table (4.6), it turns out that our method has proven high efficiency in training, testing and accuracy in prediction compared to other methods by researchers.

Table 4.6: Comparison to Previous Works.

Reference	Approach	Accuracy
Our Approach	DNN, RNN, LSTM, GRU, CNN	99.6% 99.3% 99% 98.95% 98.92%
[21]	FCNN	99.4%
[22]	Naive Bayes and CF-KNN	84.86
[59]	RNN-GRU	83.5%

4.5 CONCLUSION

In conclusion, we have implemented a NIDS For SDN with IoTs based on DL by using various types of models including DNN, CNN, GRU, LSTM, and RNN. The system takes in input from an NSL-KDD dataset and enables the user to select the type of run, dataset, and model to use. The proposed system uses KERAS, TensorFlow and other library to implement the models and Sklearn library to examine the performance of the models. The performance of the models is measured by recall, precision, accuracy, F1-score, and confusion matrix. The best model is the one that has the highest recall, precision, accuracy and F1-score and it was DNN model. However, the model's performance will depend on the data's quality and quantity, as well as the specific problem we are trying to solve. Overall, we have implemented a DL-based NIDS and SDN with IoTs that can be used as a tool for detecting anomalies and intrusions in the network. In addition to what I previously mentioned, it's important to note that deep 1earning-based NIDS have many advantages over traditional methods such as rule-based systems. They can handle high-dimensional and complex data, and can detect unknown intrusions that traditional methods may miss.

Additionally, as the network security threat landscape is constantly evolving, deep learning-based systems can adapt to changing situations and improve their performance over time. However, it's also important to remember that deep learning-based systems have some limits as well. It may not be able to recognize covert or targeted attacks because they need a lot of labeled data to train the models. Additionally, the performance of deep learning-based systems can be affected by the quality of the data and the choice of architecture and hyper parameters. Therefore, it's important to carefully evaluate the performance of deep learning-based systems using appropriate evaluation metrics, and to continuously monitor and update the system to ensure that it remains effective in detecting intrusions. Additionally, it's important to use multiple layers of defense, such as firewalls, SIEM (security information and event management) systems, and (IDS)intrusion detection systems, to provide comprehensive protection against network intrusions.

4.6 FUTURE WORK

Future work for this study could include:

a. Testing the system on a larger and more diverse dataset to further evaluate its performance and generalizability.

b. Experimenting with different architectures and hyperparameter settings to see if they boost the efficiency of the models.

c. Incorporating additional features such as network traffic statistics, payloads, and source and destination IP addresses to enhance the efficiency of the models.

d. Developing an ensemble of multiple deep 1earning models to improve the efficiency of the NIDS.

e. Incorporating unsupervised learning methods to detect unknown and emerging threats.

f. Developing a real-time intrusion detection system by incorporating online learning techniques.

g. Evaluating the system against different types of attacks and comparing its performance with other existing intrusion detection systems.

h. Putting the system into action in the real world and evaluating its accuracy in terms of false positives and negatives.

REFERENCES

[1] A. N., "A reliable ids system using blockchain for SDN-enabled IOT Systems," *IoT Protocols and Applications for Improving Industry, Environment,* and Society, pp. 173–194, 2021

[2] A. Saritha, B. Ramasubba Reddy, and A. Suresh Babu, "A hybrid SDN architecture for ids using bio-inspired optimization techniques," Journal of Interconnection Networks, 2022.

[3] H. Hendrawan, P. Sukarno, and M. A. Nugroho, "Quality of service (QoS) comparison analysis of Snort ids and bro ids application in Software Define Network (SDN) architecture," 2019 7th International Conference on Information and Communication Technology (ICoICT), 2019.

[4] H. Li, F. Wei, and H. Hu, "Enabling dynamic network access control with anomaly-based ids and SDN," Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization - SDN-NFVSec '19, 2019.

[5] J. E. Varghese and B. Muniyal, "An efficient ids framework for DDOS attacks in SDN environment," IEEE Access, vol. 9, pp. 69680–69699, 2021.

[6] L. Ong, "OpenFlow/SDN and Optical Networks," Network Innovation through OpenFlow and SDN, pp. 387–414, 2014.

[7] P. Raj and A. Raman, "Software-defined network (SDN) for Network Virtualization," Software-Defined Cloud Centers, pp. 65–89, 2018.

[8] R. Sutton, R. Ludwiniak, N. Pitropakis, C. Chrysoulas, and T. Dagiuklas, "Towards an SDN assisted ids," 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2021.

[9] S. Zwane, P. Tarwireyi, and M. Adigun, “A flow-based ids for SDN-enabled tactical networks,” 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 2019.

[10] S. Usman, I. Winarno, and A. Sudarsono, “Implementation of SDN-based ids to protect virtualization server against HTTP DOS attacks,” 2020 International Electronics Symposium (IES), 2020.

[11] A. Alhowaide, I. Alsmadi, and J. Tang, “Ensemble detection model for IOT ids,” Internet of Things, vol. 16, p. 100435, 2021.

[12] A. Panigrahi, B. Sahu, and S. N. Mohanty, “A survey on opportunity and challenges of ids over IOT,” Real-Life Applications of the Internet of Things, pp. 55–83, 2022.

[13] H. Khoulimi, M. Lahby, and O. Benammar, “Towards an intelligent system to manage ids for IOT,” 2022 5th Conference on Cloud and Internet of Things (CIoT), 2022.

[14] M. D. S. Romeo, “Intrusion detection system (IDS) in internet of things (IOT) devices for Smart Home,” International Journal of Psychosocial Rehabilitation, vol. 23, no. 4, pp. 1217–1227, 2019.

[15] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” Journal of Management Information Systems, vol. 24, no. 3, pp. 45–77, 2007.

[16] J. McKay, P. Marshall, and R. Hirschheim, “The design construct in Information Systems Design Science,” Enacting Research Methods in Information Systems, pp. 11–42, 2016.

[17] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, “A comprehensive deep learning benchmark for IOT ids,” Computers & Security, vol. 114, p. 102588, 2022.

[18] R. Doshi, N. Apthorpe, and N. Feamster, “Machine learning ddos detection for consumer internet of things devices,” 2018 IEEE Security and Privacy Workshops (SPW), 2018.

[19] A. A. Diro and N. Chilamkurti, “Distributed attack detection scheme using Deep Learning Approach for Internet of Things,” Future Generation Computer Systems, vol. 82, pp. 761–768, 2018.

[20] K. Yang, J. Ren, Y. Zhu, and W. Zhang, “Active learning for wireless IOT intrusion detection,” IEEE Wireless Communications, vol. 25, no. 6, pp. 19–25, 2018.

[21] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, “Threat analysis of IOT networks using artificial neural network intrusion detection system,” 2016 International Symposium on Networks, Computers and Communications (ISNCC), 2016.

[22] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K.-K. R. Choo, “A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IOT backbone networks,” IEEE Transactions on Emerging Topics in Computing, vol. 7, no. 2, pp. 314–323, 2019.

[23] N. Moustafa, B. Turnbull, and K.-K. R. Choo, “An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things,” IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4815–4830, 2019.

[24] P. Shukla, “ML-IDs: A machine learning approach to detect wormhole attacks in internet of things,” 2017 Intelligent Systems Conference (IntelliSys), 2017.

[25] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, “N-Baiot—network-based detection of IOT botnet attacks using deep autoencoders,” IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, 2018.

[26] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An ensemble of autoencoders for online network intrusion detection,” Proceedings 2018 Network and Distributed System Security Symposium, 2018.

[27] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IOT," 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019.

[28] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous IOT device-type identification using periodic communication," IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, pp. 1402–1412, 2019.

[29] T. Luo and S. G. Nagarajan, "Distributed anomaly detection using Autoencoder neural networks in WSN for IOT," 2018 IEEE International Conference on Communications (ICC), 2018.

[30] M. Govindarajan, "Hybrid intrusion detection using ensemble of classification methods," International Journal of Computer Network and Information Security, vol. 6, no. 2, pp. 45–53, 2014.

[31] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.

[32] J. Yuill, F. Wu, J. Settle, F. Gong, R. Forno, M. Huang, and J. Asbery, "Intrusion-detection for incident-response, using a military battlefield-intelligence process," Computer Networks, vol. 34, no. 4, pp. 671–697, 2000.

[33] M. P. Arthur and K. Kannan, "Cross-layer based multiclass intrusion detection system for secure multicast communication of manet in military networks," Wireless Networks, vol. 22, no. 3, pp. 1035–1059, 2015.

[34] T. Mantecón, D. Casals, J. J. Navarro-Corcuera, C. R. Del-Blanco, and F. Jaureguizar, "Deep learning to enhance maritime situation awareness," EasyChair Preprints, 2019.

[35] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy Artmap: A neural network architecture for incremental supervised learning of Analog

Multidimensional Maps," IEEE Transactions on Neural Networks, vol. 3, no. 5, pp. 698–713, 1992.

[36] T. Liu, W. Zhou, J. S. Liu, and C. Lin, "Intrusion detection of data platform based on Extreme Learning Machine in civil and military integration," DEStech Transactions on Computer Science and Engineering, no. csse, 2018.

[37] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Packet and flow-based network intrusion dataset," Communications in Computer and Information Science, pp. 322–334, 2012.

[38] J. Liu, B. Kantarci, and C. Adams, "Machine learning-driven intrusion detection for Contiki-ng-based IOT networks exposed to NSL-KDD dataset," Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning, 2020.

[39] N. F. Haq, A. R. Onik, and F. M. Shah, "An ensemble framework of anomaly detection using hybridized feature selection approach (HFSA)," 2015 SAI Intelligent Systems Conference (IntelliSys), 2015.

[40] F. Ertam, L. F. Kilincer, and O. Yaman, "Intrusion detection in computer networks via machine learning algorithms," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), 2017.

[41] P. K. Garg and L. Sharma, "Artificial Intelligence: Challenges and future applications," Artificial Intelligence, pp. 229–245, 2021.

[42] J. Spencer, O. Worthington, R. Hancock, and E. Hepworth, "Towards a tactical software defined network," 2016 International Conference on Military Communications and Information Systems (ICMCIS), 2016.

[43] Ji Qing, Ding Hao, Ma Huan, Meng Kun, Li Jing, and Yang Yang, "An SDN-based resource pre-combination dispatching strategy in military network," Third International Conference on Cyberspace Technology (CCT 2015), 2015.

[44] S. S. Bhunia and M. Gurusamy, “Dynamic attack detection and mitigation in IOT using SDN,” 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), 2017.

[45] M. Khodjaeva, M. Obaidat, and D. Salane, “Mitigating threats and vulnerabilities of RFID in IOT through outsourcing computations for public key cryptography,” Security, Privacy and Trust in the IoT Environment, pp. 39–60, 2019.

[46] K. Wrona, M. Amanowicz, S. Szwaczyk, and K. Gierlowski, “Sdn testbed for validation of cross-layer data-centric security policies,” 2017 International Conference on Military Communications and Information Systems (ICMCIS), 2017.

[47] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, “Combining OpenFlow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments,” Computer Networks, vol. 62, pp. 122–136, 2014.

[48] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, “Survey on SDN based Network Intrusion Detection System using machine learning approaches,” Peer-to-Peer Networking and Applications, vol. 12, no. 2, pp. 493–501, 2018.

[49] M. Radhi Hadi and A. Saher Mohammed, “A novel approach to network intrusion detection system using Deep Learning for SDN: Futuristic Approach,” Machine Learning & Applications, 2022.

[50] I. Shin, Y. Choi, T. Kwon, H. Lee, and J. Song, “Platform design and implementation for flexible data processing and Building ML models of ids alerts,” 2019 14th Asia Joint Conference on Information Security (AsiaJCIS), 2019.

[51] K. Nam and K. Kim, “A study on SDN Security Enhancement Using Open-Source ids/IPS SURICATA,” 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018.

[52] M. Savva, I. Ioannou, and V. Vassiliou, “Fuzzy-logic based ids for detecting jamming attacks in wireless mesh IOT Networks,” 2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet), 2022.

[53] P. Parkar and A. Bilimoria, “A survey on cyber security IDS using ML methods,” 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021.

[54] P. Shukla, “ML-IDs: A machine learning approach to detect wormhole attacks in internet of things,” 2017 Intelligent Systems Conference (IntelliSys), 2017.

[55] S. Otoum, N. Guizani, and H. Mouftah, “Federated reinforcement learning-supported ids for IOT-steered healthcare systems,” ICC 2021 - IEEE International Conference on Communications, 2021.

[56] S. S. Gopalan, D. Ravikumar, D. Linekar, A. Raza, and M. Hasib, “Balancing approaches towards ML for ids: A survey for the CSE-CIC IDS dataset,” 2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA), 2021.

[57] U. Kiran, “IDs to detect worst parent selection attack in RPL-based IOT Network,” 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), 2022.

[58] Y. Katsura, P. Sakarin, N. Yamai, H. Kimiyama, and V. Visoottiviseth, “Quick blocking operation of firewall system cooperating with ids and SDN,” 2022 24th International Conference on Advanced Communication Technology (ICACT), 2022.

[59] T. A. Tang, L. Mhamdi, D. McLernon, S. A. Zaidi, and M. Ghogho, “Deep recurrent neural network for intrusion detection in SDN-based networks,” 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), 2018.

[60] A. H. Halbouni, T. S. Gunawan, M. Halbouni, F. A. Assaig, M. R. Effendi, and N. Ismail, “CNN-IDS: Convolutional Neural Network for Network Intrusion Detection

System," 2022 8th International Conference on Wireless and Telematics (ICWT), 2022.

[61] S. Srinivasan, S. A, vinayakumar R, and S. KP, "DCNN-ids: Deep convolutional neural network-based Intrusion Detection System," 2020.

[62] A. Alshahrani and J. A. Clark, "A transfer learning approach to discover ids configurations using Deep Neural Networks," 2022 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), 2022.

[63] C. C. Aggarwal, "Training Deep Neural Networks," Neural Networks and Deep Learning, pp. 105–167, 2018.

[64] G. Zhu and T. Zhao, "Deep-GKNOCK: Nonlinear Group-feature selection with Deep Neural Networks," Neural Networks, vol. 135, pp. 139–147, 2021.

[65] Y. Xu and H. Zhang, "Convergence of deep convolutional neural networks," Neural Networks, vol. 153, pp. 553–563, 2022.

[66] R. Bon and H. Cardot, "Advanced methods for time series prediction using recurrent neural networks," Recurrent Neural Networks for Temporal Data Processing, 2011.

[67] A. Malek, "Applications of recurrent neural networks to optimization problems," Recurrent Neural Networks, 2008.

[68] F. M. Salem, "Recurrent neural networks (RNN)," Recurrent Neural Networks, pp. 43–67, 2021.

[69] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluation of recurrent neural network and its variants for Intrusion Detection System (IDS)," Deep Learning and Neural Networks, pp. 295–316, 2020.

[70] F. M. Salem, "Gated RNN: The Gated Recurrent Unit (GRU) RNN," Recurrent Neural Networks, pp. 85–100, 2021.

[71] J. Dai, “Predicting Machine’s performance data using the stacked long short-term memory (LSTM) neural networks,” 2021.

[72] N. K. K and N. Duraipandian, “Malicious traffic classification using long short-term memory (LSTM) model,” 2021.

[73] E. Hvitfeldt and J. Silge, “Long Short-Term Memory (LSTM) networks,” Supervised Machine Learning for Text Analysis in R, pp. 273–302, 2021.

[74] S. Y. Mehr and B. Ramamurthy, “An SVM based DDoS Attack Detection Method for Ryu Sdn Controller,” Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies, 2019.

[75] G. Meena and R. R. Choudhary, “A review paper on IDS classification using KDD 99 and NSL KDD dataset in Weka,” 2017 International Conference on Computer, Communications and Electronics (Comptelix), 2017.

[76] R. Rahim, A. S Ahanger, S. M Khan, and F. Masoodi, “Analysis of ids using feature selection approach on NSL-KDD dataset,” SCRS CONFERENCE PROCEEDINGS ON INTELLIGENT SYSTEMS, pp. 475–481, 2021.

[77] Y. Sahli, “Comparison of the NSL-KDD dataset and its predecessor the KDD Cup ’99 dataset,” International Journal of Scientific Research and Management, vol. 10, no. 04, pp. 832–839, 2022.

[78] J. S. Fong, “Data normalization,” Information Systems Reengineering, Integration and Normalization, pp. 343–376, 2015.

[79] B. Chen and P. Ji, “Numericalization of the self-adaptive spectral rotation method for coding region prediction,” Journal of Theoretical Biology, vol. 296, pp. 95–102, 2012.

[80] Z. Shou and S. Li, “Large dataset summarization with automatic parameter optimization and parallel processing for local outlier detection,” Concurrency and Computation: Practice and Experience, vol. 30, no. 23, 2018.

[81] J. A. Wuisan, A. Jacobus, and S. Sompie, “Data balancing methods on radiographic image classification on Unbalance Dataset,” Jurnal Teknik Elektro dan Komputer, vol. 11, no. 1, p. 1, 2022.

[82] X. Gao, “Comparison of SVM-based feature selection method for Biological Omics Dataset,” Proceedings of the 11th International Conference on Biomedical Engineering and Bioinformatics, 2022.

[83] A. Wiliński, “Dataset reduction procedure by feature extraction,” AIP Conference Proceedings, 2017.