DILLA UNIVERSITY

COLLAGE OF ENGINEERING & TECHNOLOGY

SCHOOL OF COMPUTING AND INFORMATICS

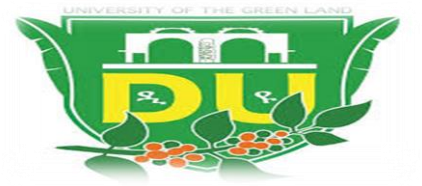
Deep Learning-Based Detection for Black Hole and Gray Hole Attacks in AODV routing protocol MANETs

Mesfin Kiflu

Thesis submitted to the department of Computer science in partial fulfillment for the degree of masters of science in computer science and networking

Dilla, Ethiopia

May,2025



Dilla University

College of Engineering & Technology

School of Computing and Informatics

Mesfin Kiflu

Advisor: Alembante Mulu (Ph.D.)

This is to certify that the thesis prepared by Mesfin kiflu, *Deep Learning-Based Detection for Black Hole and Gray Hole Attacks in AODV routing protocol MANETs* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	**Signature**	**Date**
Advisor:			
Examiner:			
Examiner**:**			

Declaration Sheet

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: _____Signature: _____ Date: _____

Confirmed by advisor:

Name: _____ Signature: _____ Date: _____

Abstract

Mobile Ad Hoc Networks (MANETs) are a prominent category of wireless networks that operate without fixed infrastructure, offering great flexibility in dynamic environments such as battlefields, remote areas, and smart cities. However, due to their decentralized structure and limited security in routing protocols, MANETs are more vulnerable to attacks compared to traditional wired networks. Specifically, black and gray hole attacks represent a serious threat, as malicious nodes exploit network resources, significantly degrading overall network performance. This thesis presents an efficient attack detection system aimed at enhancing the security of the Ad-hoc On-Demand Distance Vector (AODV) routing protocol to detect both black hole and gray hole attacks in MANETs different nodes using deep learning techniques. The research is conducted in two key phases. The first phase involves dataset preparation, where network traffic data is generated through simulations in NS-2 (Network Simulator version 2). These simulations incorporate both normal and malicious behaviors, representing black and gray hole attacks within the context of AODV. After pre-processing and analysis, a refined dataset consisting of 28,691 records with 21 features is extracted from the trace files. In the second phase, the proposed detection system is developed and evaluated. The dataset is divided into training (80%) and testing (20%) subsets. Feature selection is carried out using the Categorical Boosting (CatBoost) importance score, with a threshold applied to select the most relevant features. Additionally, the Synthetic Minority Over-sampling Technique (SMOTE) is utilized to ensure balanced data for the model. The system employs advanced deep learning architectures, including Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and a hybrid CNN-LSTM model. The experimental results show that the proposed system achieves high detection accuracy across various models: LSTM (98.57%), GRU (98.46%), CNN_LSTM (98.60%), and CNN (98.48%). These results demonstrate hybrid CNN_LSTM model show high accuracy and how well the suggested method works to improve MANET security by precisely identifying black hole and gray hole attacks.

Keywords: *MANET, Deep Learning, XGBoost, CatBoost, Black Hole, Gray Hole.*

Acknowledgments

I want to start by expressing my gratitude to the Lord, who is my rescuer and has supported me through all of the highs and lows of my studies. I sincerely thank Dr. Alembant Mulu, my advisor, for his insightful remarks, unwavering support, and direction. Finding the right words to describe how much I appreciate everything you have done for me, family, is nearly impossible. Additionally, I am grateful to Google Collab for offering a robust cloud-based infrastructure that allowed me to train deep learning models without the need for costly local hardware resources. GPUs are available for free use in this research.

Table of Contents

List of tables

List of Figures

List of Abbreviations

ACK:	Acknowledgment
CBR:	Constant Bit Rate
AGT:	Agent
AODV:	Ad hoc On-Demand Distance Vector
AWK:	Abstract Windows ToolKit
CBR:	Constant Bit Rate
CSV:	Comma Separated Values
CNN:	Convolutional Neural Network
GUI:	Graphical User Interface
GRU:	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MANET:	Mobile Ad hoc Network
NAM:	Network Animator
PDR:	Packet Delivery Ratio
RERR:	Route Error
RREP:	Route Reply
RREQ:	Route Request

CHAPTER ONE

1. Introduction

1.1 Background of the study

Mobile ad-hoc networks (MANETs) are dynamic, self-organizing wireless networks that have gained significant attention due to their versatile applications in various areas, such as military operations, emergency response and smart city environments. However, the open and distributed Because of their nature, MANETs are vulnerable to a number of security risks, such as gray and black hole attacks. Attacks can significantly degrade the network's performance, affects to reduced packet delivery and also end-to-end delay becomes increased, and more energy consumption[1].

In recent years, mobile ad-hoc networks, have drawn a lot of attention because of their dynamic and infrastructure-less nature, which allows for flexible and efficient communication in various scenarios [2]. However, the lack of a centralized authority and the reliance on cooperative node behavior make these networks vulnerable to various security threats [3]. now days there has been a lot of interest in mobile ad-hoc networks (MANETs), which are infrastructure-less, self-configuring wireless networks, because of the nodes' flexibility to join or exit the network at any time. These networks are susceptible to a number of attacks, such as the Gray Hole and Black Hole attacks, in which malevolent nodes can seriously harm the network by interfering with routing [4]. A deep learning-based detection Ad-hoc On-Demand Distance Vector (AODV) routing algorithm is suggested as a solution to these problems in order to improve security and identify and lessen these assaults in mobile ad hoc networks.

Due to their independence and self-configuring nature from a permanent infrastructure, mobile adhoc networks (MANETs) are susceptible to a number of security risks, such as gray and black hole attacks. These attacks can severely disrupt the network's functionality by selectively dropping or forwarding packets, leading to reduced throughput and reliability[5].The proposed enhance security algorithm builds upon the Ad-hoc On-Demand Distance Vector (AODV) routing protocol, which is a commonly used reactive routing protocol in mobile networks.

In this research, a MANET (Mobile Ad Hoc Network) dataset was constructed to characterize two types of packet-dropping attacks, in addition to the normal scenario, using the NS-2 (Network

Simulator-2). MANETs are self-configuring, decentralized networks featuring wireless communication between nodes that don't require a fixed infrastructure. Due to their dynamic and open nature, MANETs are highly in secured to various security threats, particularly packet-dropping attacks, which have the potential to significantly impair network performance. The dataset was designed to simulate and analyze these attacks, providing a foundation for developing robust intrusion detection systems (IDS) and mitigation strategies.

Ad-hoc-on-Demand Distance Vector (AODV) security is improved in this study by using the on-demand routing technique to build trust between mobile nodes. Deep learning is used in the detect attacks method to identify network black hole attacks. The Secure AODV (SAODV) security mechanism and the gray hole and black hole attacks used in AODV are used to compare the results.

1.2 Motivation

The primary concern for the fundamental operation of a Mobile Ad-Hoc Network (MANET) is security. By ensuring that security concerns have been addressed, network services, confidentiality, and data integrity can be attained. Because of its open medium, constant topology changes, lack of central monitoring and management, cooperative algorithms, and unclear protection mechanism, MANET frequently experiences security threats. These elements have altered the MANET's battlefield in the fight against security threats.

Ad hoc Mobile Networks (MANETs) are susceptible to security risks including black hole and gray hole attacks, in which malicious nodes drop or selectively forward packets, because they depend on dynamic routing protocols like AODV. security mechanisms often fail to adapt to evolving attack patterns, necessitating more intelligent solutions. Deep learning offers promising capabilities in detecting complex attack behaviors by analyzing network traffic patterns. By review different literature identifying the gap on (MANETs)dataset for malicious attacks, so this research aims to enhance AODV’s security by integrating deep learning models to proactively identify and mitigate black hole and gray hole attacks, ensuring reliable data transmission in MANETs. The motivation of the study is to close the gap between traditional security methods and learning-based, adaptive intrusion detection for reliable ad hoc networking.

1.3 Statement of the problem

The statement of the problem in the thesis was addresses the security vulnerabilities present in Mobile Ad-hoc Networks (MANETs), particularly concerning Black Hole and Gray Hole attacks. MANETs are characterized by their dynamic and decentralized nature, where nodes operate without a fixed infrastructure, leading to challenges in ensuring secure and reliable communication. The fundamental issue highlighted is the susceptibility of MANET routing protocols, such as DSR and AODV, to various kinds of attacks due to their inherent trust assumptions. These protocols rely on the cooperation of every node in the network, making them vulnerable to malicious nodes that may interrupt communication by dropping or altering data packets.

Self-configuring networks of mobile devices connected without the need for a fixed infrastructure are known as mobile ad-hoc networks, or MANETs. Because nodes in these networks serve as both traffic sources and relays without a central authority, they pose special routing and security difficulties. One of the primary security threats in MANETs is the black hole attack, when a malicious node attracts traffic by seeming to have the fastest path to the target, only to drop or alter the packets. Another attack, known as the gray hole attack, is more subtle, where the malicious node selectively forwards packets, making it difficult to detect.

The network layer is more susceptible towards many types of attacks. The most advanced attacks that disrupt a network's regular operation are network layer attacks. By using up the resources of authorized nodes, they will disrupt regular network operations. Its effects include increased latency, increased routing load, decreased throughput, packet loss, and excessive power consumption.

The lack of strong security features in current MANET protocols to efficiently identify and mitigate gray and black Hole attacks has been identified as the research challenge. Gray Hole attacks deliberately discard packets to interfere with communication, while Black Hole attacks cause data loss by a hostile node pretending to have the quickest path to a target. The integrity and dependability of data transmission in MANETs are seriously threatened by these attacks. As a result, IDS has been used by numerous researchers to address these issues and enhance MANET security. However, the current IDS are ineffective in detecting malicious activity that aims to compromise network security objectives, have a high false alarm rate, and have poor accuracy and

detection rates. Additionally, the problem statement highlights the necessity of improving MANET security through the development of deep learning and an algorithm based on trust security that can identify Black and Gray Hole attacks. By attacking these security issues, the study seeks to strengthen MANETs' resistance to malicious activity and guarantee their safe and effective functioning in dynamic and uncertain situations.

1.4 Research Questions

In this study, the following research questions are addressed.

1. Which attributes are relevant to improve attack detection model.
2. How simulated dataset are prepared for attack detection.
3. How to get optimal simulation parameter values for data generated from the simulated network?
4. How to designing deep learning model and techniques for most effective intrusion detecting gray and black hole attacks?
5. What are the performance metrics to evaluate the proposed models?

1.5 Objectives of the Study

1.5.1 General Objectives

The objective of this study is to utilize deep learning techniques to enhance the security of the Ad-hoc On-Demand Distance Vector (AODV) routing protocol by effectively detecting and mitigating black hole and gray hole attacks in Mobile Ad Hoc Networks (MANETs).

1.5.2 Specific Objectives

- To conduct a literature review on conventional and security-based routing protocols of MANETs.
- To make dataset from simulations and identify suitable features presence of attacks and without attacks.
- To determine the optimal features of the dataset using appropriate feature selection technique
- To assess the different algorithms and techniques for reducing the false alarm rate
- To develop a deep learning model for improve detecting gray Hole and black Hole attacks in MANETs.
- To evaluate the efficiency of deep learning algorithms for our dataset.

1.6 Significance of the Study

Research on deep learning with secure-based Ad-hoc On-Demand Distance Vector Algorithm for detecting and mitigating Black Hole and Gray Hole Attacks in Mobile Ad-hoc Networks is crucial to enhancing the security and reliability of MANETs. The study offers a novel algorithmic technique for identifying and removing malicious nodes, safeguarding network integrity and data transport by overcoming the significant shortcomings in Black Hole and Gray Hole attacks.

The final dataset from simulation serves as a critical resource for researchers and practitioners working on MANET security. It not only facilitates the development of advanced intrusion detection systems but also provides a benchmark for comparing the performance of different detection algorithms. Researchers can investigate several machine learning methods, like classification and anomaly detection, to detect and lessen packet-dropping assaults in MANETs by utilizing this dataset. This research contributes to the broader goal of enhancing the security and reliability of wireless ad hoc networks, which are increasingly being used in applications such as military communications, disaster recovery, and IoT (Internet of Things) deployments.

By enhancing the security landscape of MANETs, the research was to contributes to the overall stability and efficiency of wireless networks, benefiting applications across various sectors such as mobile communications, emergency response, and military operations.

1.7 Scope of the Study

MANET is subject to numerous attacks, this thesis was only able to identify wormhole, flooding, and blackhole attacks. AODV is selected from among the several MANET routing protocols. To construct the dataset, ns-2.35 has been used to simulate the routing protocol both with and without assaults. The network was experience congestion as the number of nodes increases during the simulation. However, network congestion was not considered when analyzing the data. The success of the detection system depends on its accuracy, so selecting the appropriate datasets is essential. Due to the lack of publicly available statistics on MANET packet dropping attacks, data was generated using an open-source event-driven simulation.

The algorithms that have been examined for the detection method of Ad hoc Networks are Convolutional Neural Networks (CNN), LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit). The limitations of existing techniques for identifying network layer assaults

within the AODV (Ad hoc On-Demand Distance Vector) routing protocol are also examined due to its extensive use.

1.8 Contributions of the study

- ✓ Improved security and reliability of MANET routing by effectively detecting and mitigating both gray and black hole attacks.
- ✓ creation of a comprehensive and adaptable NS2 simulation environment that produces realistic network traffic data under a range of black hole and gray hole attack scenarios and faithfully simulates MANET behavior.
- ✓ Against malicious node by enhanced routing performance in terms of end-to-end delay, energy efficiency and packet delivery ratio compared to existing secure MANET routing protocols.
- ✓ An Innovative Intrusion Detection System (IDS) for MANETs Based on Deep Learning One particular contribution is the creation of a new deep learning model architecture designed especially for identifying black hole and gray hole assaults in MANETs.
- ✓ A generalizable feature engineering methodology for MANET security is one of the research's main contributions. For better intrusion detection, this methodology offers a framework for extracting useful features from network traffic data that can be modified and used with different deep learning models and MANET setups.

1.9 Thesis Organization

This thesis is organized as following chapters. In chapter 2, was addressed different background information and distinct methods to detect MANET attacks under the network layer. In chapter 3 was are discussed Detailed dataset preparation techniques processes. Chapter 4 shows model development and implementation steps. Experiment results and discussions were described in Chapter 5. Finally, illustrates the conclusion and future work.

CHAPTER TWO

2. Literature Review

2.1 Introduction

This chapter covers the fundamentals of wireless networking, ad hoc networks, MANET vulnerabilities and problems, MANET routing protocols, security attacks, and security attack defence mechanisms. Additionally, relevant work for gray hole and black hole attack detection algorithms in real data packet transmission, in route finding operation, and both are discussed.

A computer network is a collection of two or more machines connected by a wired or wireless connection to share resources. Based on their dependence on the framework, the networks can be divided into wired and wireless networks. Nodes connected by a wireless connection can be connected in a flexible way with wireless networks. Utilizing wireless innovation in the networking domain has led to an expansion in its appeal and the emergence of a new paradigm[6].

2.2 Ad-Hoc Networks

The Mobile Ad Hoc network (MANET) is a vast community of sensor nodes with self-directed commands. Within the network, individual nodes may leave without authorization or notification. As a result, the MANET's dynamic, independent features and simplicity of installation lead to its widespread application in crucial sectors[7] .

In contrast to conventional wireless networks, a wireless connection connects one node to the other. A node may function as a router to send the data to the neighboring nodes. As a result, networks of this type as infrastructure-less networks. There is no central organization in these networks. Ad-hoc networks are capable of handling node malfunctions and progressions brought on by changes in topology. A mobile node in the network may disconnect at any time, breaking the link between the other nodes. As a MANET, new connections are established by the nodes in the network simply requesting new routes[1].

2.3 Mobile Ad-hoc Networks (MANETs) Overviews

The Internet Engineering Task Force (IETF) established the MANETS working group in 1996 to standardize IP routing protocol features appropriate for wireless routing applications. When

MANET was first introduced by the Defense Advanced Research Project Agency (DARPA) in 1970 [24], it was still known as a packet radio network.

The topology of the network is determined by the communication capabilities of its nodes. Because of self-configuration, the nodes are able to set themselves up. Routing protocols are among the most fascinating and difficult academic topics. due to the fact that numerous routing protocols, such as AODV, OLSR, DSR, and others, have been suggested for MANETs.

In Figure 2.1, a MANET example is displayed. It shows a mobile area network (MANET) in which each mobile node is free to go in any direction, resulting in incremental fluctuations in a joint. It consists of a dispersed, autonomous group of mobile nodes that have the ability to occasionally join and leave the network. Each mobile node acts as a router to transport the information, and they all dynamically shift and perform assemblies. The MANET devices are forced to transmit data, memory, energy, and batteries.

In contrast to traditional wired telecommunication networks, which heavily rely on towers and sophisticated base stations, mobile ad-hoc networks, are wireless networks that have gained popularity because of their self-organized nature. They are therefore used various applications [8].

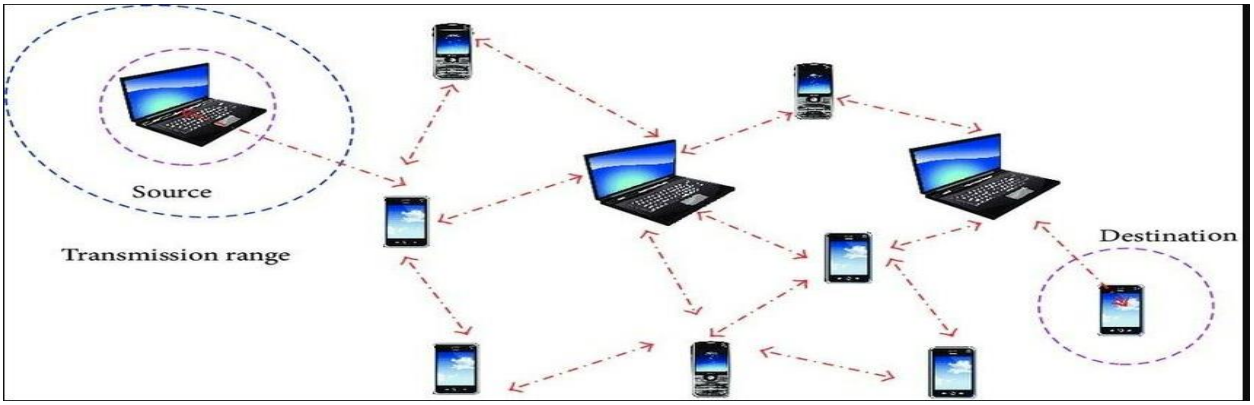


Figure 2.1. Mobile Ad-hoc Network

2.3.1 Vulnerabilities of MANET

The wireless devices are perceived as being more vulnerable to security attacks than the wired medium. These additional wireless connection weaknesses render MANETs essentially helpless.

Even with the accommodation, the characteristics of MANETs are such that security will always be a problem. These vulnerabilities can be enumerated as follows.[9], [10], [11].

- **Wireless Links:** Wireless networks are susceptible to security risks because, in contrast to wired networks, they are accessible to other networked devices. Because there is no distinct secure boundary between MANET and ad hoc networks, nodes using wireless transmission joins leave the latter open to hostile nodes and a variety type of attacks [7]. In a MANET, nodes are free to join and leave the network whenever they want. Any node that is within range of a network can join it right away and begin interacting with other network nodes. MANET's porous borders make it more susceptible to attacks.
- **Dynamic Topology**: Because MANET nodes are allowed to roam about and join or exit the network without restriction, the topology of the network is constantly changing. As a result, neither packet delivery nor the existence of significant pathways leading to the destination is guaranteed. MANET nodes have unrestricted mobility throughout the network. Consequently, the network topology is subject to sudden and arbitrary changes at any time. In this potent network state, it is difficult to distinguish between malicious node behavior and regular variations in the network structure.
- **Absence of Central Governing Body**: there is no central body in charge of overseeing the network's overall operation or keeping an eye out for malicious nodes. It lacks a focused management tool, such as a server or an interruption identification framework, which creates a number of security and vulnerability problems.

2.4 Challenges of MANET

Mobile Ad Hoc Networks (MANETs) present unique challenges due to their decentralized nature and dynamic topology. Here are some of the primary challenges associated with MANETs. The shortcoming of MANET is vulnerable to different challenges. Among those challenges, the following are listed [[12], [13]].

Dynamic Topology: The network topology changes regularly as nodes move. The impact this can lead to frequent route breaks and the need for constant route discovery.

Limited Bandwidth: wireless channels, which have limited bandwidth compared to wired networks. This limitation can lead to congestion and increased latency.

Security Vulnerabilities: Nodes in a MANET are connected to a wireless network. Security is the main priority for any network, but it's particularly difficult for MANETs. MANET allows devices to roam freely within and between one other in an open area, allowing attackers to get access to the network exercises.

Quality of Service (QoS): The complexity of QoS adds another layer of difficulty to the MANET. Since MANETs are wireless networks, it can be challenging to guarantee the administration's nature because wireless connections suffer from greater data loss, distortion, delays, and variations in speed and capacity when compared to conventional connections. Given that MANETs have dynamic topologies, it is difficult to obtain precise information about the state of devices and networks, and providing QoS is even more difficult.

Multi-Hop Routing: In the multi-hop routing mechanism, the destination node can communicate with more than one node to perform routing. There isn't a fixed router in MANETs since the routing algorithm operates under the assumption that any node that is prepared to join the network is trustworthy and capable of assisting with packet routing throughout the networks.

Scalability of Network: Nodes may move out of range, causing temporary disconnections. This can lead to network fragmentation, complicating communication. Both the network's architecture and its scalability were determined at the initial stages of network design. In MANETs, however, the situation is completely different because of the nodes' mobility, which causes the MANETs' scale to fluctuate. It is too challenging to predict the future number of nodes in the MANET.

Maintaining security in a MANET was a challenging because of the potential for unknown issues, such exposed and concealed node concerns, to arise at every level of the communication stack. Links between the nodes can be established and terminated at any time, opening the door to unanticipated threats and attacks under certain situations[14].

Wireless ad hoc networks face several security challenges due to their open and decentralized nature. Since the wireless medium is physically accessible to all, it is vulnerable to interference and bit errors. The absence of secure boundaries allows adversaries to easily join or leave the network, making trust management difficult. Additionally, the lack of infrastructure complicates key distribution, addressing, and certification.

Node limitations further exacerbate security risks, as their constrained resources make them prone to availability attacks. Bandwidth constraints also hinder cooperation-based security mechanisms. Moreover, multi-hop routing introduces vulnerabilities, as malicious nodes can manipulate routes by creating fabricated paths, routing loops, or false routes, disrupting network operations.

2.5 MANET Routing Protocols

Without the requirement for centralized administration nodes, a Mobile Ad-hoc Network, or MANET, is composed of several autonomous mobile nodes without any infrastructure that are connected to one another via a shared wireless channel.[13].

Here, each node serves as both a host and a router at the same time. As a result, every node has a wireless transmitter and receiver. The MANET's nodes are in charge of configuring, running, and maintaining the network. Because distant nodes can join with one another hop by hop by adhering to a set of protocols that define the hopping sequence to be followed, every node in the network has a dynamic topology. It is a process wherein data is exchanged between nodes, one at a time, moving on to the next, or it is a process where packets or messages are transmitted from the source this guiding process in the mobile ad-hoc network is carried out via a routing protocol to the destination node[12]. One area of research in mobile ad hoc networks is secure routing. Ad hoc routing protocols can be divided into three main groups: reactive, proactive, and hybrid. The hierarchy of these protocols shown in Figure 2.2.

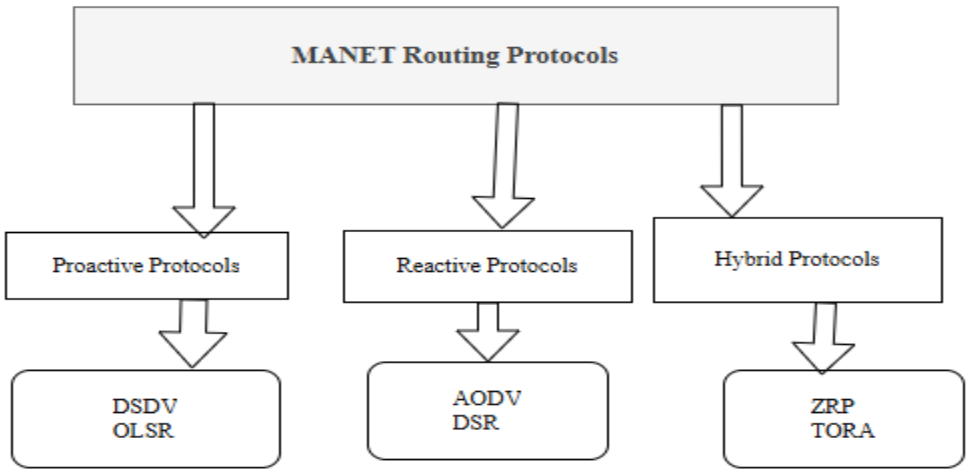


Figure 2.2. MANET Routing Protocols Classification

2.5.1 Proactive Routing Protocols

As the network topology changes, a proactive routing protocol, also called a table-driven routing protocol, updates its routing tables to reflect the most recent routing information from each node to all alternative nodes. It also regularly exchanges routing data from mobile nodes. Compared to other routing methods, this operation causes a significant network overhead [15].

Every node in the network has a table listing all potential routes between source and destination nodes thanks to proactive routing protocols. Since nodes in MANETs move around a lot, there is a good probability that the node table will become invalid. As a result, the tables are constantly updated with the most recent network information.[12]. The proactive routing protocol is not appropriate for networks of greater size because of its substantial control cost. The operation of this protocol is based on link state and distance vector algorithms. DSDV is one of the most popular proactive protocols in MANETs.

2.5.2 Reactive Routing Protocols

Finding the target node's route and maintaining it are the two key responsibilities of on-demand routing protocols [12]. By sending a route request, or RREQ, over the network and waiting for a response from the receiving node, the route discovery method finds the routes between the source and destination nodes. After a route has been found, the act of responding to topological changes that occur is known as route maintenance. Several on-demand protocols include: The most well-known examples of the reactive routing protocol are AODV, DSR[13]

The reactive routing protocol often possesses the following features:

- Find a route only upon request
- Use a flooding mechanism to generate an inquiry when attempting to find the objective "on request".

To find a route, reactive protocols broadcast a large number of route request packets to the network simultaneously. These procedures provide the following benefits.

- Proactive protocols have a significant overhead for maintaining global routing tables
- Reactive systems provide a quicker response to node failure and network reorganization.

This scheme's disadvantage is its large message size, which contains all of the path information. As a result, there may be a lengthy delay before data is transferred between two nodes once a path to the destination has been found.

2.5.3. Adhoc On-Demand Distance Vector Routing Protocol (AODV)

The cutting edge MANET routing protocol Ad hoc On-Demand Distance Vector was developed as an improvement to the DSR and DSDV routing protocols [16]. It is a routing protocol that is put up only when needed. The node has no knowledge of other nodes unless contact with them is established. Each node broadcasts HELLO packets to preserve local connectivity information based on a regular time interval. The node's local connectivity keeps track of all of its neighbors' details. Route Maintenance and Route Discovery are the two stages of the AODV protocol's operation [15].

The AODV routing protocol performs routing operations using three control messages. Route Request (RREQ), Route Reply (RREP), and Route Error (RERR). It makes use of the RERR during the route maintenance phase and RREQ and RREP during the route discovery phase.

Table 2. 1 AODV Packet Formats

Packet Type	Reserved	Hop Count
RREQID		
Destination IP Address		
Destination Sequence Number		
Source IP Address		
Source Sequence Number		

(a) RREQ

Packet Type	Reserved	Hop Count
Destination IP Address		
Destination Sequence Number		
Source IP Address		
Lifetime		

(b) RREP

Packet Type	Reserved	DestCount
Unreachable Destination IP Address (1)		
Unreachable Destination Sequence Number (1)		
Additional Unreachable Destination IP Addresses (if needed)		
Additional Unreachable Destination Sequence Numbers (if needed)		

(c) RERR

I. Route discovery

When using the route discovery method, the route request (RREQ) is broadcast from the source node (S), received by the neighbor node, and then rebroadcasts the message from the neighbor node (with its address). In this manner, the RREQ is sent to the destination node (D) via multi hop communication. Destination node D acknowledges the RREP messages sent to source node S in response to the RREQ. In this manner, out of all the possible paths, the shortest loop-free path is determined between S and D. Here, each node updates its routing table upon successfully receiving and sending a message[12].

II. Route Maintenance

After a route has been built, route maintenance is the process of keeping the route between the source and destination nodes intact. Every node keeps an eye on the active links all the time, updating the routing table's Route Timeout column as they receive and transmit data packets. An intermediate node broadcasts a Route Error (RERR) message with the information about the inaccessible destination if it discovers a broken link or has no valid route to forward the data packet for a destination. The intermediate nodes broadcast the RERR message until it reaches the broken route's originating node.[12]

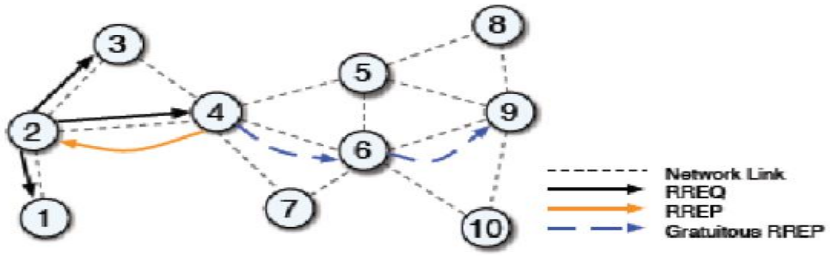


Figure 2.3. An intermediate node generates an RREP.

Figure 2.3 above shows how an intermediate node generates an RREP. Node 4 sends a gratuitous RREP to node 9 and an RREP to node 2, and it has a route to node 9.

2.5.4 Sequence Number in AODV Routing Protocol

The RERR notification indicates an erroneous path from any intermediate nodes to the destination node. Here, a key component of each node's routing table is the Sequence Number, an unsigned, unique number that ensures packets are delivered to their destination in an orderly fashion.

The AODV protocol uses a sequence number, which sets it apart from other on-demand routing technologies. The highest destination in AODV sequence number indicates how recent the path is. An intermediate node may only transmit the RREP if and only if its destination sequence number is higher than the RREQ's, unless it is unable to provide the route reply. If the destination delivers the RRIP packet, the node's sequence number is increased by one more and unicasted back to the originator. It will choose the RREP with the fewest hop counts if the destination sequence numbers are equal [12].

Assume that source node S in Figure 2. 4, broadcasts RREQ to a nearby node with destination node D's sequence number. If the intermediate node has a new path to the target node, it sends back RREP; if not, it rebroadcasts RREQ to nearby nodes. The intermediate node E in this image shows that it has a new route to destination node D by sending back RREP. RREP is returned by the target destination node D.

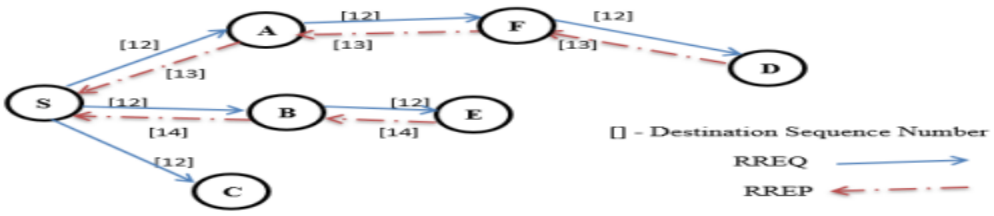


Figure 2.4 .AODV Working Principle

The source node S then uses the destination sequence number to choose one RREP from the two. The path S B E D is chosen in accordance with the AODV working principle since it has a higher destination sequence number, or 14. Because the black hole and gray hole attack features are based on sequence number, the AODV routing protocol is chosen as the base protocol in this research. It uses a destination sequence number to determine the optimal route [19]. When a destination sequence number is higher 5, it determines the data forwarding path. The destination sequence number of RREPs is suggested as a method for detecting and thwarting black hole and gray hole assaults.

2.5.5 Hybrid Routing Protocols

Reactive and Proactive routing systems can be combined to create hybrid routing strategies. Throughout the network, hybrid routing methods are utilized to strike a balance between proactive and reactive routing techniques. Reactive routing protocols are used to locate nodes outside of the small domain that are not located by means of proactive routing protocols, which limits the use of these protocols. This allows for multi-hop communications with multiple topologies across the network.

2.6 Classification of Security Attacks in Manets

The status of the attacker (internal or external), behaviour of the attack (active and passive attack), and purpose of the attack (eavesdropping, denial of service, black hole, and etc.) are additional factors that can be used to further categorize attacks in MANETs. Therefore, all levels of protection must be provided for the networks and the data's availability, integrity, and secrecy. In order to resolve this problem, new routing techniques must be applied at the network layer in order to prevent false routes and routing loops by efficiently detecting the network's frequent topological changes[14] .

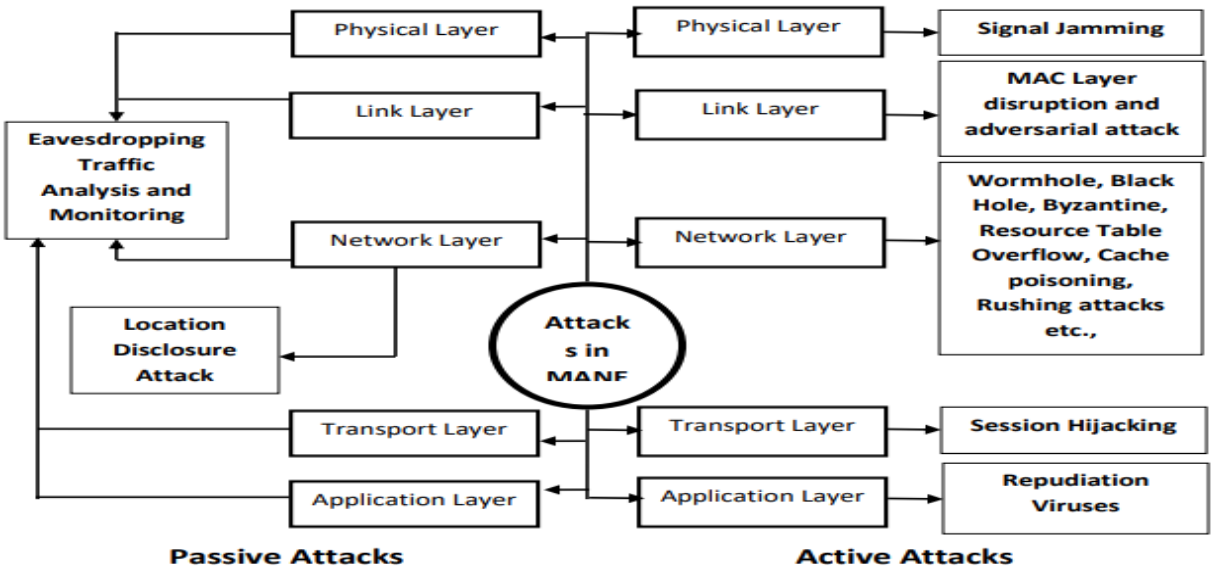


Figure 2.5 . Taxonomy of attacks in MANET

I. Passive Attacks

Passive attacks involve monitoring and intercepting data without altering the data in transit. They are often stealthy and can be challenging to detect.

- Eavesdropping: Unauthorized nodes capture data packets being transmitted across the network. This can lead to sensitive information being compromised.
- Traffic Analysis: Attackers analyse the patterns of data transmission to infer information about the network, such as the location of nodes and the types of data being sent.

II. Active Attack

- **Black Hole Attacks:** The attacker advertises that it has the quickest path and higher destination sequence number to the node whose packets it needs to block in the network, misleading other nodes. An intermediary node in a route to specific destinations is impersonated by a malicious node. An attacker then reroutes every packet that is meant for it.
- **Gray Hole Attacks:** During the route discovery phase, the attacker engages in legitimate activities, but during the actual data transmission, it adopts malevolent behaviours. This allows you to forward the remaining packets from the different nodes to their neighbours after selectively dropping a portion of the incoming packets.
- **Jellyfish Attacks:** During packet transmission, the attacker node exhibits a delay. Packets are arbitrarily delayed in a delay variance attack. In a periodic dropping attack, on the other hand, the attacker regularly drops the packet. The Jellyfish reorder attack rearranges the communication between a source node and a destination node.
- **Wormhole Attacks:** The attacker does harmful exercises by working with two malicious nodes to establish a tunnel between them in order to distort the widely used hop-count statistic. Tunneling routing control messages can interfere with routing, and wormhole assaults can make it impossible to find any other routes
- **Sybil Attack:** Multiple fake identities are presented to the network by a single node, which could give it undue control over communication and routing.
- **Rushing Attack:** An attacker quickly forwards routing requests to gain a competitive advantage and disrupt the normal routing process.

- **Replay Attack**: An attacker captures valid data packets and retransmits them to mislead the network, potentially causing confusion or triggering unwanted actions.

The Gray hole and black hole attacks are the main emphasis of this study's methodology. An active attacker can take advantage of a vulnerability in the AODV routing protocol in a few different ways. For example, the attacker can alter the message before forwarding it, drop the data packets, and then send a forged message without any help, as shown in Figure 6. Because of this, the fake RREP in the route discovery process and the data packet drop in the data transmission are the main topics of this article[17].

All packets that are not meant for compromised or self-serving nodes can be dropped. Dropping attacks have the potential to terminate end-to-end connections between nodes if the dropping node reaches a critical point because the majority of routing protocols lack a way to detect whether data packets have been delivered or not[18].

The attacker node drops the data packets in the data packet dropping assault, but the intermediary node loses the data packet that is seen in Figure 2.6, before it reaches the target node. Consequently, identifying the data packet-dropping adversary and attempting to lower the packet drop ratio become our problems.

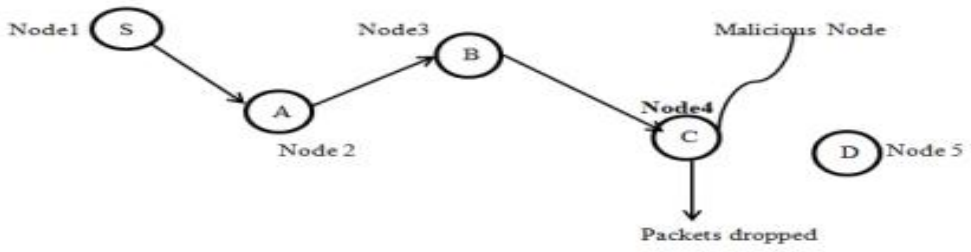


Figure 2.6. Dropping Data Packets by Malicious Node

Two common attack types that perform destructive actions in data packets that are challenging to detect in a MANET are black hole and gray hole attackers. DoS assaults come in two varieties, as seen in Figure 2.7, full packet drops and partial packet drops.

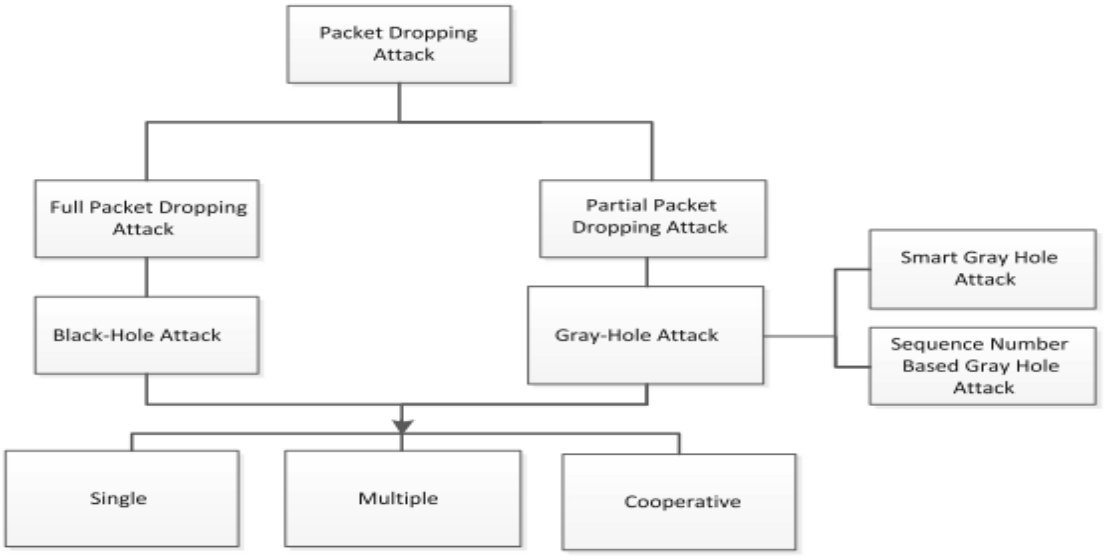


Figure 2.7. Attacker Action to Disrupt Routing in the AODV Protocol

2.6.1 Black Hole Attack

The dispersed routing techniques used in mobile ad hoc networks depend on the genuine collaboration and participation of every network node. However, because of the case of malicious nodes, this assumption is incorrect. These nodes may interfere with routing maintenance and discovery processes, or they may drop packets that are not meant for their destination without forwarding them. These actions could result in abnormal network functioning, which could cause denial of service attacks and hinder network performance[18].

The legitimate target node never receives any data packets since the malicious node never forwards any of them to it. Consequently, the source and destination nodes were unable to establish communication. Consequently, every packet received from the source node will be discarded by the black hole node. As a result, a black hole attack, also known as a packet dropping attack, severely reduces network performance. The objective of the black hole attacker is to attract traffic to it, then block it by dropping data packets. As a result, communication between the source and destination nodes was no longer possible [19].

As shown in Figure 2.8, the source node broadcasts a route request packet over the network to communicate with the destination node.

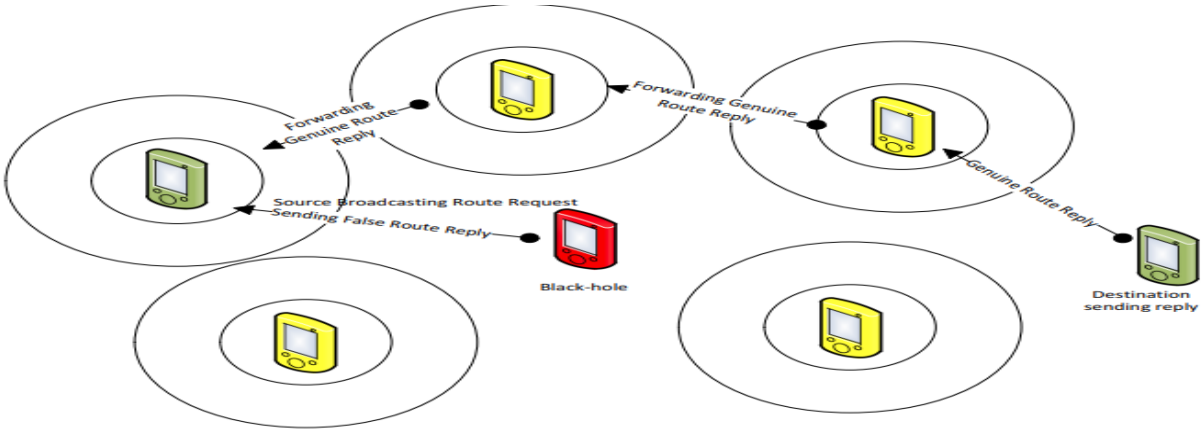


Figure 2.8 . False route reply black-hole node

After creating the path, the source node sends the data packet via the path that contains the malicious node, as shown in Figure 2 9. The network subsequently experiences a denial-of-service attack when the malicious node starts dropping the data packet.

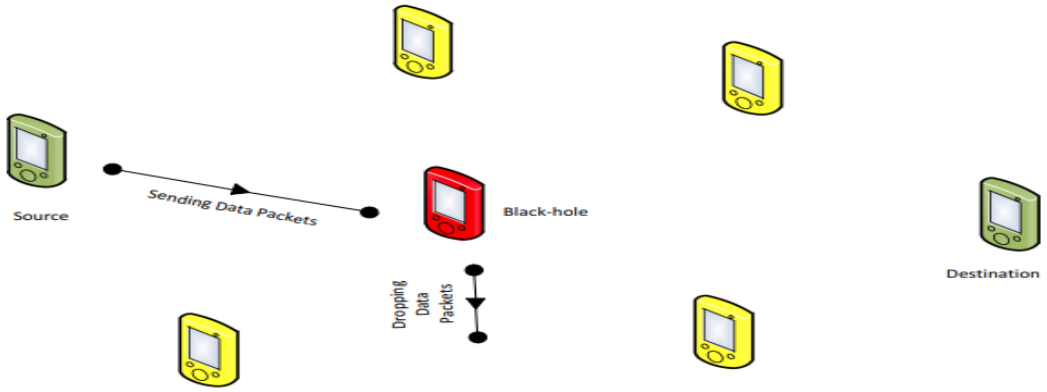


Figure 2.9: Packet dropping by black-hole node

2.6.2 Gray Hole Attack

Black hole attack variants are similar to the gray hole assault [4]. A gray hole attack is a type of denial-of-service (DoS) attack that targets the network layer, much like a black hole assault. Partial packet drop attackers are those who receive data packets, drop parts of them, and then send the remaining packets to their neighbors. An extension of a black hole attack, a gray hole attack is a kind of denial-of-service attack. It is called a partial packet drop attack because it merely omits certain data packets during connection. Since gray hole nodes act properly throughout the path finding phase, they initially seem to be innocuous. Black hole attack variants are similar to the gray hole assault [4]. A gray hole attack is a type of denial-of-service (DoS) attack that targets the

network layer, much like a black hole assault. Partial packet drop attackers are those who receive data packets, drop parts of them, and then send the remaining packets to their neighbors [20].

A. Smart Gray Hole Attack

Before eliminating any of the incoming data packets, the node can actively participate in the route-finding process in the first kind of gray hole attack, called a smart gray hole attack [9]. During the route discovery process, intelligent gray hole nodes do not send false routing information, in contrast to malicious nodes. Because they supplied the correct destination sequence numbers, the attackers are participating in the path discovery stage as a regular node in order to establish a path to the target node. Nevertheless, it changes into a malicious node when it receives the data packet from the source node. Although it has a legitimate path to the destination node, certain data packets are dropped.

In the Figure 2.10, the route discovery operation is similar to a black hole as discussed before in Section 2.5.1. Node G (smart gray hole) are a attacks node that sends true RREP, but changes behaviour in the data broadcast phase. Thus, the reply packet from node G reaches node S ahead of reply packets from other neighbours of node S. Therefore, node S sending packets to node D via node G is considering that node G has a fresher route to node D. Partial packets transmitted from S to D are now absorbed by node G. This is the setup for a smart gray hole attack.

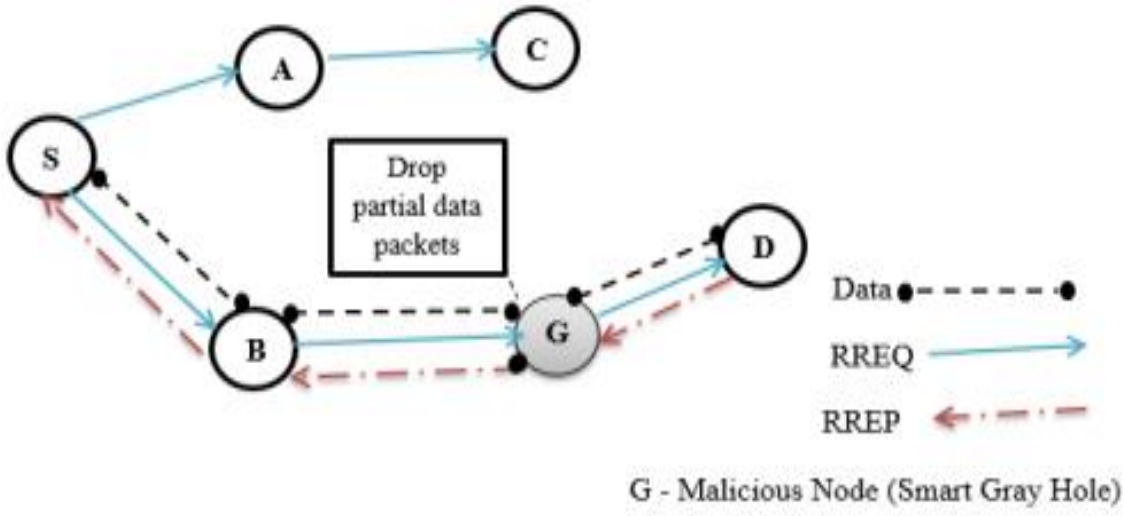


Figure 2.10 Smart Gray Hole Node for Partial Packet Drop

As seen in Figure 11 above, the malicious node G receives the data packet and then begins dropping incomplete data packets from the incoming data.

B. Sequence Number Based Gray Hole Attack

Sequence numbers are the basis of the second kind of gray hole attack, in which the malicious node reroutes data to the source node and provides false route information by sending it a fake destination sequence number [17]. From the attacker node to the destination node, there might or might not be a valid route. The rogue node could launch a black hole attack on the route-finding process.

2.7 Performance Metrics MANET Performance Metrics for AODV in NS-2

Performance is a critical factor in the development of communication protocols for MANETs. The effectiveness of a routing protocol is assessed through various metrics [21]. The following performance metrics are commonly evaluated in NS-2 simulations for AODV in MANETs.

2.7.1 Packet Delivery Ratio (PDR)

The packet delivery ratio is defined as the proportion of packets sent by the source that are successfully delivered to the destination.

$$'PDR = \frac{\text{Received Packets}}{\text{Sent Packets}} \times 100 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{Eq (2.1)}$$

2.7.2 End-to-End Delay

The average time it takes for a packet to travel from its source to its destination is used to calculate the network's latency. This is the total time required for a packet to travel from its starting point to its destination. It is computed as the difference between the packet's transmission and reception times. A lower value indicates better performance of the routing protocol [22].

2.7.3 Throughput

How much data was successfully sent over the network in a specific amount of time. shows how effectively the network handles data. Throughput, which is often expressed in bits per second (bps), measures how quickly information is successfully sent from the source to the destination. The calculation is dividing the entire amount of data received by the recipient's wait time for the final packet. The sum of the sizes of each packet indicates the overall amount of data transferred, whilst the delay between the first packet's sending and the last packet's arrival at its destination represents the total transmission time. Therefore, throughput can be defined as the ratio of the total amount of data sent to the total transmission time.

$$\text{Throughput} = \frac{\text{Total Data Receive}}{\text{Simulation time}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ eq (2.2)}$$

2.7.4 Routing Overhead

Routing overhead, which is generally measured in packets per second, is the total number of routing control packets sent over the network. The normalized routing load is defined as the ratio of all delivered routing packets to all successfully received data packets, including forwarded routing packets. Total data packets received divided by total routing packets sent is how it is calculated. High routing overhead can negatively impact network performance by lowering the overall through-put and data packet delivery ratio. The total number of control packets produced by the AODV protocol, such as RREQ, RREP, and RERR, shows how effectively the routing protocol produces control packets [23].

2.7.5 Energy Consumption

How much energy each node used overall throughout the simulation. assesses the protocol's energy efficiency, which is important for gadgets that run on batteries. Jitter is the variation in end-to-end delay for packets. Indicates the consistency of packet delivery.

2.7.6 Normalized Routing Load (NRL)

In terms of control packet use, the normalized routing load efficiency of the routing protocol is determined by the ratio of the number of data packets delivered to the number of routing control packets.

$$NRL = \frac{\text{Control Packets}}{\text{Data Packets}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{eq (2.3)}$$

2.7.7 Route Discovery Time

The time taken to discover a route from the sender to the receiver. Indicates the responsiveness of the routing protocol.

2.7.8 Packet Loss

The ratio of packets was lost compared to how many were sent. represents the network's dependability. Packet Loss Packets are units of information that are sent and received while connecting to any network. Packet loss is the term used when one or more of these packets fail to arrive at their destination. Any implementation can be affected by packet loss, but real-time packet

processing applications like audio, video, and gaming applications are especially vulnerable. Packet loss affects the program's perceived quality. Two examples of causes of packet loss or corruption include bit errors in a broken wireless network or inadequate buffers [21].

$$\text{Packet Loss Rate} = \frac{\text{Lost Packets}}{\text{Sent Packets}}\ x100 \quad \text{.....eq (2.4)}$$

2.8 MANETs Security Mechanism

The security vulnerabilities of Mobile Ad Hoc Networks (MANETs), highlighting their increased susceptibility to attacks compared to wired networks due to their lack of centralized management and dynamic topology [24]. Typically, there are three levels of security mechanisms for Mobile Ad Hoc Networks (MANETs): mitigation mechanisms, which react to security incidents and lessen their impact; detection mechanisms, which identify and warn against possible security threats; and prevention mechanisms, which try to stop security breaches from happening [25].

2.8.1 MANET Intrusion Detection System

Intrusion detection has become increasingly crucial in the field of network security, particularly for wireless MANET. These networks lack a fixed infrastructure and have constantly changing topologies, which makes them inherently vulnerable to attacks. Often, by the time a countermeasure can be implemented, it may be too late. Additionally, as hacking techniques continue to advance, determined attackers can eventually breach the system. Therefore, in order to identify and resolve suspicious behavior, it is imperative that system operations be continuously monitored, at least periodically [26]. Intrusion detection systems (IDSs) perform the essential function of monitoring audit data to identify intrusions and initiate appropriate responses, such as notifying the systems administrator or launching an automatic countermeasure. Consequently, it is vital to enhance traditional security mechanisms with effective intrusion detection and response strategies[25].

2.8.2 Machine Learning and Deep Learning

Machine learning attack detection with ever growing technology and its increased usage among the users across the world in every imaginable field, privacy and security concerns are exponentially growing with the new range of threats and attacks. Six out of seven billion people use their mobile gadgets for their everyday activities ranging from shopping, banking, social media activities,

healthcare and professional activities. This has increased the chances for information leakage and theft.

Security problems are at large in MANET applications and its operations even with the availability of various dedicated approaches. In the recent years, Machine Learning Algorithms (MLA) has played a vital role in numerous study fields such as wireless networks, wireless sensor networks, data science and computer vision etc. With real-time decision making, vast data processing, shorter learning cycle times, and error-free processing, MLA has largely proven successful in resolving contemporary security concerns and meeting practical demands [1]. To provide security across MANETs, a variety of machine learning algorithms have been used, such as random forest, naïve bayes, k-NN clustering, decision trees, deep learning, support vector machines, and random forest. When compared to alternative approaches, these algorithms have yielded noteworthy results in terms of accuracy, complexity, and memory utilization [14].

2.8.3 Deep Learning

Numerous researchers have applied DL in a so many of fields, such as robotics, autonomous systems, audio processing, image and video recognition, , natural language processing and more [27]. They've achieved the best performance outcome. Additionally, the learning process is more accurate because DL learns the characteristics and representation of the data on its own. Compared to superficial machine learning techniques, it has also proven to be more accurate and efficient. Deep learning techniques can be made to carry out feature extraction and classification tasks simultaneously, in contrast to machine learning algorithms. Furthermore, without the need for human domain knowledge or hand-coded rules, DL automatically learns representations from data like text, photos, and videos. Additionally, it gets around a few of the drawbacks of shallow learning techniques.

Faster learning and more thorough data analysis are the main goals of deep learning. When compared to the effectiveness learning techniques, it employs layer-wise characteristics, has a larger learning capacity, and can produce superior outcomes.

Neurons are at the heart of the learning process in DL. They start with one or more inputs, which are then multiplied by connection weights and added together, either from the raw data set or from neurons positioned in a previous layer.[27], [28] The activation function of the neuron receives

this result, and using equation (2.2), the generated neuron's output is then sent to the following network layer. Equation 2.1 displays the neuron's mathematical form.

The Z is then added to the process by the activation function.

Z=input * weight + bias .....(2.5)

The output is equal to f. Z .....(2.6)

Five primary categories of DL activation functions, including threshold, sigmoid, rectifier, softmax, and hyperbolic tangent functions, were identified by Ochin and Sharma [29]. An activation function that accepts Y is called a threshold function; if Y is a specific value (or threshold), the function's output is activated; if not, it is not. Figure 2.11 displays the graph of the threshold function. The threshold in this picture is denoted by x.

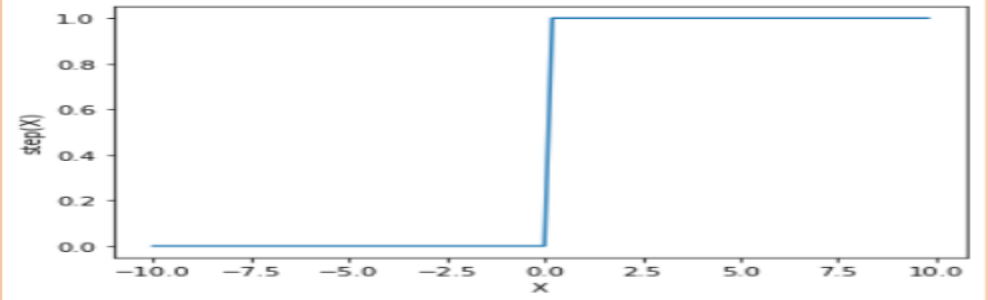


Figure 2.11 Threshold function

The sigmoid function, which is employed in logistic regression, resembles a smoother form of the threshold function. Non-linearity and categorization outcomes are where they diverge. It may have vanishing gradient issues. Figure 2.12. displays the graph of the sigmoid function.

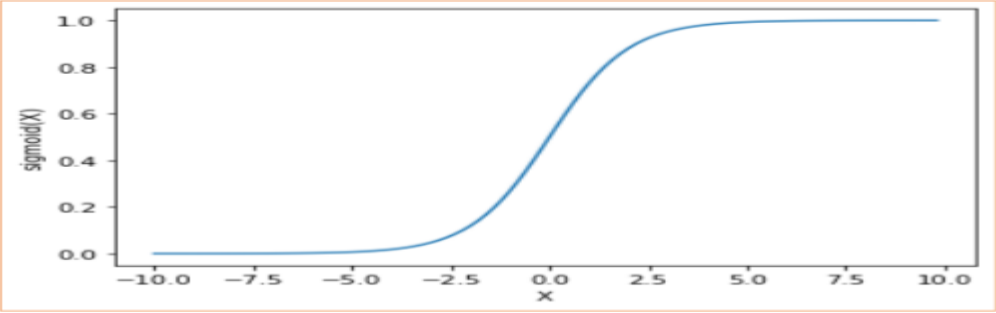


Figure 2.12 .Sigmoid function

A trigonometric identity serves as the foundation for hyperbolic tangent functions (Tanh). Its function is completely similar to that of the sigmoid function. Figure 2.13 shows its bounds (-1, 1). It may also experience vanishing gradient issues, similar to sigmoid activation. For multi-classification situations, the softmax function is employed. Softmax is not typically utilized in the network's hidden layers and is also regarded as an output activation function.

$$F(X_i) = \frac{\mathrm{Exp}(Xi)}{\sum_{j=0}^{k} \mathrm{Exp}(Xi)} \quad i = 0, 1, 2, k \ldots\ldots \qquad \ldots\ldots(2.7)$$

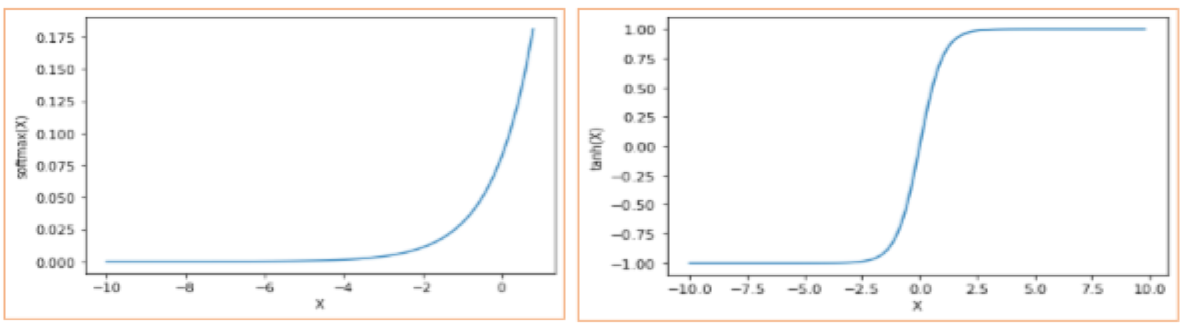


Figure 2.13 softmax graph and Tanh function

The best function for solving the vanishing gradient problem is the Rectified Linear Unit (ReLU). Its smoothness feature differs from that of the sigmoid function. But Relu is still widely used in the deep learning space. At 2.8, the mathematical equation is displayed. Binary classification is a better fit for the ReLU function. Figure 2.14, also displays the ReLU function.

Z = f x = max 0, x ....(2.8)

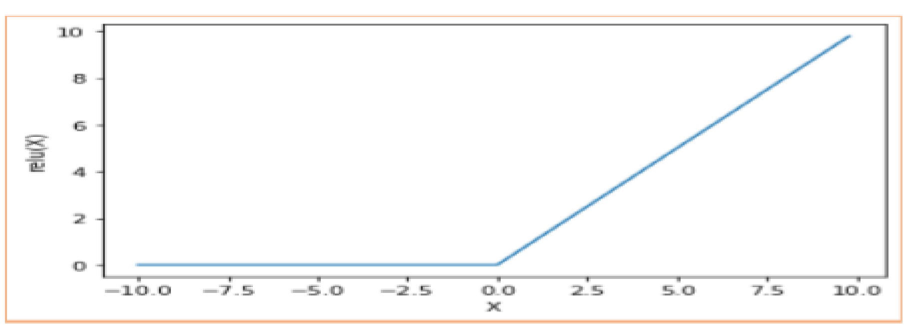


Figure 2.14. ReLU Function

A DL network is composed of neurons, input layers, hidden layers, and output layers. The number of neurons in the input layer, the first or leftmost layer of the network, should match the size or

quantity of features of the dataset. The output layer is the last layer of the deep network. Hidden layers exist between the input and output layers. The two most popular DL methods are recurrent neural networks (RNNs) and convolutional neural networks (CNNs). In computer vision and natural language processing, Deep Belief Networks (DBNs) have shown remarkable performance. particularly in high-level vision applications like understanding and recognition[29] . Nevertheless, it is rarely applied to address security issues, particularly in ad hoc settings.

The vanishing gradient issue in ordinary classical RNN is resolved by LSTM . Because of its robustness while handling nonlinearity and big data, LSTM is suitable for time-series prediction. Additionally, it can employ the gating mechanism to learn long-term dependencies. A memory cell that stores previous states is a feature of every LSTM unit[30] . It is built using a more straightforward architecture that integrates the states and merges the gates [29]. It monitors and controls the information flow between the neural network's cells using gating techniques. Two significant problems that arise during the training phase of DL are under-fitting and over-fitting. To address the problems Many academics proposed many approaches to address the problems of under-fitting and overfitting. For example, by probabilistically eliminating neurons from neural network layers, used dropout to enhance the durability and adaptability of neural networks. Data augmentation, as described by is used to decrease overfitting by employing different transformations to increase the quantity and variety of training samples and enhance accuracy by expanding the size of training data with the information only in the training set[30].

2.9. Related work

Previously research on MANET concentrated on various security threats. Reactive routing protocols such as AODV are used to analyze the black hole and gray hole assaults that are part of MANET. Not all types of packets drop attacks under different networking settings are covered by some of the detection systems that have been created.

The study of [7] have put out a strategy to protect MANET against a black hole node. The creators of this method made the arbitrary assumption that the malicious node will raise the source sequence number by up to 100. If there is a difference of more than 100 between the source and destination sequence numbers, the node is a black hole node. But because the black hole node adds a random number to generate RREP, it is challenging to select an arbitrary value given the behaviors of a black hole in a MANET. As a result, this approach is unable to identify malicious nodes or

malicious nodes that transmit authentic RREP when the difference between the destination and source sequence numbers is smaller than 100.

The study have presented the Mitigating Black Hole impacts through Detection and Prevention (MBDP-AODV) algorithm, a dynamic threshold technique [17]. A malicious (black hole) node that delivers bogus RREP during route discovery is mitigated by the MBDP-AODV algorithm.

The MBDP-AODV technique has the following drawbacks:

- ✓ It is incapable of detecting malicious nodes in the event that the malicious and normal nodes have a little difference in the destination sequence number in the RREP. because the standard deviation is smaller than the mean.
- ✓ This approach is unable to identify the malicious node if it acts normally during route discovery by sending true RREP and maliciously during actual data packet delivery.

To identify black hole attacks in the MANET, the authors of [1] suggested a secure AODV (S-AODV) routing method based on ANN and SVM. According to simulation results, the suggested SAODV outperforms the conventional AODV in a variety of traffic scenarios. It offers defense against attacks like replay, impersonation, and routing information modification, but it doesn't specifically address black hole or gray hole attacks.

In [31] In order to counter black hole attacks in mobile ad hoc networks, the article proposes a trust-based security technique that makes use of the OTB-DSR Protocol in NS-3. The suggested method outperforms benchmark methods with a high packet delivery ratio of over 95% as demonstrated by NS-3 simulations. Reliable data transmission is ensured by using trust between intermediate nodes to identify authentic nodes. The method's ability to minimize data loss even when hostile nodes are present is ascribed to trusted nodes creating source routes. All things considered; the study presents a viable way to improve security in mobile ad hoc networks. The suggested system's need on node trust, which can be difficult to build and preserve in dynamic and sometimes hostile mobile ad hoc network contexts, is its disadvantage. Furthermore, in situations when numerous malicious nodes band together to obstruct communication, the system's efficacy might be restricted.

In mobile ad hoc networks (MANETs), defending the network layer against malicious attacks is a significant and difficult security concern, claim the authors of [32]. In order to protect the widely used AODV routing protocol in MANETs from a coordinated gray hole attack, this study proposes a security measure. In response to a route request message from a source node, a gray hole is a

node that selectively drops and delivers data packets after asserting that it has the quickest path to the destination node.

The study of [33] created the Accurate and Cognitive Intrusion Detection System (ACIDS) as an intrusion detection system to identify the black hole attack. Using the attacking node's destination sequence number and route reply, this method locates the attacker node and removes it from a routing network. A node is eliminated from the routing network and reported to other nodes as an invasive node if it has the highest sequence value in the routing environment. The method's massive amount of control packets causes significant end-to-end latency and routing overhead.

To improve the detection accuracy of IDS[34] ,suggested a unique anomaly detection process utilizing a machine learning technique. Rough set theory (RST) and support vector machines (SVM) are combined in the suggested IDS. From the network layer, they gathered the nodes' actions. To preprocess the data and lower number of features for the training data, the authors employed RST. Furthermore, the SVM model receives the chosen characteristics for classification and training. One drawback is that the researchers did not use a network simulator with different simulation parameters to test the integrated machine learning model, and the data gathered from a single layer might not be enough to identify the suspicious behavior.

The study of Yin et al. [35] use a recurrent neural network (RNN-IDS)-based deep learning technique. They assessed their suggested approach in both binary and multi-class classification using the NSL-KDD dataset, which has a distinct training and testing set. In the task of binary and multi-class classification, they also contrasted deep learning with conventional machine learning methods, finding that the former had achieved a greater accuracy and detection rate with a lower false-positive rate. This study has no bearing on MANET attacks, while being a promising method for RNN-based deep learning IDS.

In author Feng et al. [36] to detect privacy and denial of service (DOS) threats in MANETs. The primary components of this device are the deep learning detection model and the capturing tool. Additionally, it integrates with the functions of data processing, detection, and crawling. The authors' experiment evaluated the deep neural network (DNN) detection model used to detect DOS attacks. According to the results, these deep learning detection models have high accuracy and performance ratings of 98.5%. LSTM offers the highest accuracy among them.

According to study by Nguyen, Thuan MinhVo, Hanh Hong Phuc Yoo, Myungsik,[37] traditional machine learning techniques lose scalability and performance as the amount of created data increases. However, when trained on big produced datasets, the deep learning approach performs better. Recurrent neural networks (RNNs) and their variants are used in our research to develop a deep learning model for a cross-layer based anomalous intrusion detection system.

As a result, in contrast to earlier research, data is gathered from the application, network, MAC, and physical layers. The best features from the cross-layer data are then chosen using Xgboost and catboost, along with feature importance and threshold values. CNN, LSTM, and GRU are then utilized to enhance IDS performance. Furthermore, their results were compared to those of conventional machine learning methods like random forest and SVM.

Communicate forwarders are chosen by the majority of AODV-based intrusion detection systems now in use depending on the number and direction of wireless hops. However, these techniques are unable to adequately resist flooding attacks and are not flexible enough to adjust to a variety of traffic situations. Moreover, current AODV schemes do not simultaneously address both black hole and gray hole attacks. To overcome these limitations, this study proposes a Deep Learning-enhanced AODV Routing Protocol (DL-AODV) for MANETs, designed to detect and mitigate both attacks efficiently.

The deep learning models are trained on network traffic and node behavior data to identify the distinctive patterns of these attacks. This method is effective in detecting the attacks, but does not integrate the detection mechanism into a routing protocol.

Table 2. 2: Summary of Related Work

Authors/year	Article Title	Methodology	Use Features	Findings	Drawbacks
Muhammad .S and **Tao** .F2022	'Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time'	CNN-LSTM	data collected on the website Kaggle	CNN-LSTM method has the highest detection accuracy	Detection only malicious software not include network attacks
Poongothai and Duraiswamy [38] / 2019	'Intrusion Detection in Mobile AdHoc Networks using Machine Learning Approach.	RST and SVM	Routing Control Packets RREQ, RREP, RERR, Hello Data Control Packets Data	A ML technique has been used to offer a novel anomaly detection strategy.	The information gathered from a single layer and small features might not be enough to identify the suspicious activity.
Feng et al. [36]/2019	'Anomaly detection in adhoc networks based on deep learning mode'	LSTM, CNN, and DNN	Kaggle data set and extract and a doubt some features from DOS attacks.	LSTM gets the highest Accuracy, Precision, Recall, and F1 – score.	-They evaluated the suggested detection models using a limited number of malicious nodes.
Yannam .A [39]/2020	'Black hole Attack Detection Using Machine Learning Algorithms in MANET – Performance '	SVM, decision tree classifier, RF classifier, logistic regression algorithm	unknown features were used	Logistic Regression classifier algorithm, an accuracy of 88.23and the SVM and decision tree classifier same accuracy 82.35	The research used machine learning and low detection accuracy
S.Venkatasubramanian, A. Suhasini,S. Hariprasath[40]/20222	'Detection of Black and Grey Hole Attacks Using Hybrid Cat with PSO-Based Deep Learning Algorithm	CNN and LSTM	Packet Delivery Ratio (PDR): Latency: Captures the delay in data packet delivery. Energy Consumption	black & grey hole attacks in MANETs with in accuracy of 80%, utilizing deep learning techniques such as CNN and LSTM	Low accuracy and detection and the primary limitations revolve around moderate performance (80% accuracy), an overly simplistic feature set ignoring critical attack indicators,

CHAPTER THREE

3. Research Methodology

3.1 Introduction

In this chapter, the overall proposed work process and the working mechanism of ns2 simulation for data collection and the proposed network attacks Intrusion Detection System (IDS) of MANET utilizing a deep learning model was presented. In this section, the proposed network attacks Intrusion Detection System (IDS) of MANET utilizing a deep learning model was presented. The model comprises three main components such as data collection, preprocessing, and feature selection and detection. Initially, MANET traffic was generated under both normal and attack conditions, and the process of collecting network-layer data features from the NS-2 network simulation was explained in detail. Following the generation of network traffic network layers under normal and attack scenarios, data preprocessing and feature selection mechanisms were implemented. The preprocessed and extracted data were then used as inputs for the deep learning model to detect attacks. In this study, an IDS for Ad-hoc On-Demand Distance Vector Algorithm for Deep Learning-Based Black Hole and Gray Hole Attack Detection in Mobile Ad-hoc Networks was developed. A well-known and organized methodology called CRISP-DM (Cross-Industry Standard Process for Data Mining) is used to accomplish the goals of the study. CRISP-DM is a robust framework that provides a systematic approach to data mining and analytics projects, ensuring that all critical phases of the research are addressed comprehensively. So, by choosing ns2 simulation and deep learning may have achieved these objectives.

The primary objective of this study was to improve the security of the AODV protocol by putting in place a detection system against black hole and gray hole attacks. The AODV routing protocol in a MANET lacks detection and mitigation techniques to distinguish between malicious and legitimate node information. This research technique improves both the route finding and data transmission phases' ability to detect gray hole and black hole attacker nodes. Therefore, by utilizing a deep learning model, the suggested approach offers a solution that tends to identify the rogue node that is misbehaving during both phases.

The methodology of this research approach experimental used and which involves simulating a MANET environment and generating data for analysis. This approach uses for a comprehensive

and structured process for detecting and classifying attacks in MANETs, leveraging both simulation and deep learning techniques. The data was preprocessed and split into testing and training sets and Various machine learning models (CNN, GUR, LSTM) are optimized, Feature importance is analyzed to improve model performance and evaluated for attack detection and classification.

3.2 Data Set Preparation

Due to the unavailability of public MANET datasets for packet dropping attacks, an open-source event-driven simulation was employed to generate the necessary data. The NS-2 simulation produced raw trace data files in and processed by using AWK and prepared features data in the form of Comma Separated Values (CSV) format, suitable for text-based processing. These files contained network information, including routing updates and attack details, collected from the network, layers. The generated CSV files served as input to the feature preprocessing component of the system.

The preprocessing phase began with the conversion and encoding of certain features in the raw dataset. Subsequently, 21 features were identified as the initial candidate feature set. Feature numericalization was then applied to ensure the input values for the deep learning model were in a numeric matrix format, followed by normalization to mitigate the impact of extreme values. In the feature selection step, the importance of features was evaluated using the extreme gradient boosted tree (Xgboost) and categorical boosted (CatBoost) algorithms. Based on these analyses, some features were eliminated during the selection process. Once preprocessing, including feature selection, was completed, the datasets were fed into the deep learning algorithm. Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) layers were trained with regularization and dropout mechanisms, their weights were adjusted, and the proposed models were developed.

3.3.1 Proposed solution

The proposed model contains three mechanisms, data collection, preprocessing, and feature selection and detection. First, MANET traffic was generated under normal and attack situations, and the way to collect cross-layer data features from the NS-2 network simulation was described in detail. After generating network traffic under normal and attacks scenarios from multiple-layer,

data preprocessing and feature selection mechanisms were applied. Then, the preprocessed and extracted data were used as inputs into the deep learning model for attacks detection.

An open-source event-driven simulation was used to generate data because of the lack of availability of public MANET packet dropping attack datasets. The proposed model includes three feature selection and detection, preprocessing, and data collecting. Initially, MANET traffic was created in both normal and attack scenarios, and a detailed description of how to gather cross-layer data characteristics from the NS-2 network simulation was given. Data preparation and feature selection procedures were used after creating network traffic in both conventional and multiple-layer attack scenarios. The retrieved and preprocessed data were then fed into the deep learning model to detect assaults.

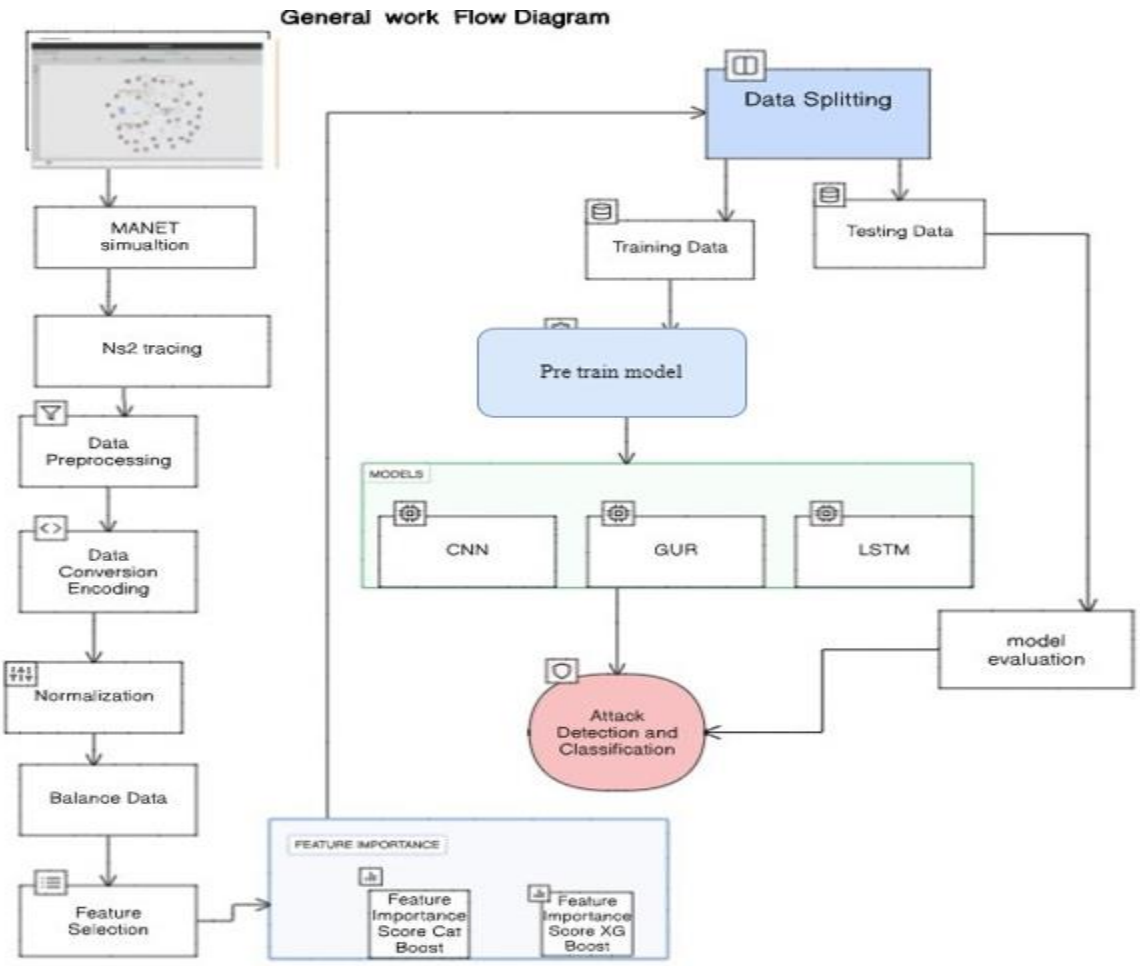


Figure 3. 1 General work Flow Diagram

3.2. Data Collection

The dataset can generally be gathered in a many of ways, including sanitized, simulated, testbed, and standard datasets. While the real traffic method is risky, the area traffic method is costly. The success of the detection system depends on its accuracy, so selecting the appropriate datasets is essential. Due to the absence of publicly available statistics on MANET packet dropping attacks, data was generated using an open-source event-driven simulation.

The simulation environment was set up using NS-2, a widely used network simulator that allows researchers to model and analyze network behavior under various conditions. The tracing system in NS-2 was utilized to capture detailed information about network events, such as packet transmissions, receptions, and drops. These traces were then expanded and processed to create a complete dataset that includes both normal network behavior and malicious activities. This study simulates two different kinds of packet-dropping attacks: gray hole and blackhole assaults. A rogue node loses all packets it receives in a blackhole attack, whereas in a gray hole assault, the node drops only some packets, making identification more difficult.

The dataset records a number of network traffic-related characteristics, including throughput, packet delivery ratio, end-to-end delay, and routing behavior. These characteristics are crucial for differentiating between benign and malevolent actions. An anomalous increase in end-to-end latency or a notable decline in the packet delivery ratio, for example, may be signs of a packet-dropping attack. The dataset is a useful resource for training and assessing machine learning and deep learning models intended to identify and mitigate such threats by recreating these attacks in a controlled setting.

One of the main challenges in constructing this dataset was ensuring its realism and relevance to real-world MANET scenarios. To achieve this, the simulation parameters were carefully chosen to reflect typical MANET conditions, such as node mobility, network density, and traffic patterns. The NS-2 tracing system was configured to record detailed logs of network events, which were then processed and labeled to create a structured dataset. The labels indicate whether a given instance represents normal behavior, a blackhole attack, or a gray hole attack, enabling supervised learning approaches to be applied effectively.

A popular and highly efficient network simulator (NS-2) was utilized to run the simulation. Both normal network activity and network layer attacks (black hole and gray hole) must be replicated. The necessary features were extracted from the trace file. To investigate the effects of MANET assaults, the AODV routing protocol was simulated both with and without an attack. The trace file was examined using the AWK (Abstract Windows ToolKit) tool.

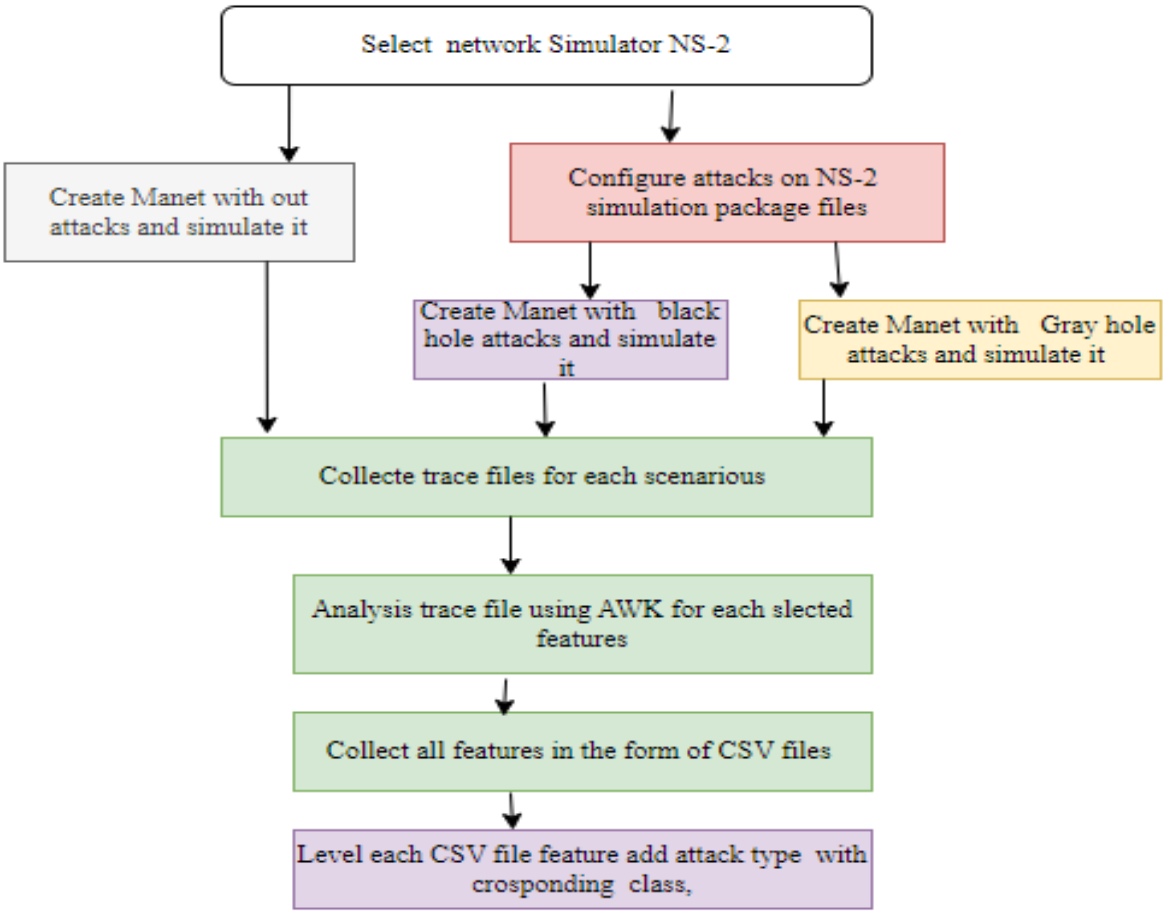


Figure 3. 2 Steps of dataset preparation for proposed work

To prepare the dataset, features are obtained from MANET performance metrics and AODV control messages. To calculate the feature values, the first step is making a simulation using network simulator (NS-2). Once the simulation is done, important network features were collected. Simulations are performed for normal, blackhole and gray hole cases independently. Depending on the number of nodes, we injected different number of attack nodes.

To create a malicious node, the built-in c++ code of AODV routing protocol is modified for both black hole and gray hole attacks. Declaration for an attacker node, conditions for the occurrence of an attacks are main activates of the modifications.

During simulation, number of nodes, pause time, simulation time, number of an attacker nodes, number of connections and other relevant parameters are tested. We have collected 28691 trace files from 28691 simulations. The trace file is analyzed using AWK (Abstract Windows ToolKit) tool. Finally, the number of records from each class is collected in the form of CSV (Comma Separated Version) file

3.2.1 Network Simulator

Network Simulator (NS) is a discrete, object-oriented event simulator used in networking research. As indicated in Table 3.1, the NS-2 provides a variety of features to build the network environment with all network layers. It may be used to imitate both wired and wireless networks [42]. With more than 88% of users, NS2 is among the most widely used network simulators [43].

Table 3. 1: Top five network simulators

Simulator	Popularity	Availability
NS2	88.8%	Open source
GloMoSim	4%	Open source
OPNET	2.61%	Commercial
QualNet	2.49%	Commercial
OMNET++	1.04%	Free for academic use

The simulation environment it offers is versatile and promising. The user's perspective of ns2 is explained in Figure 3.3.

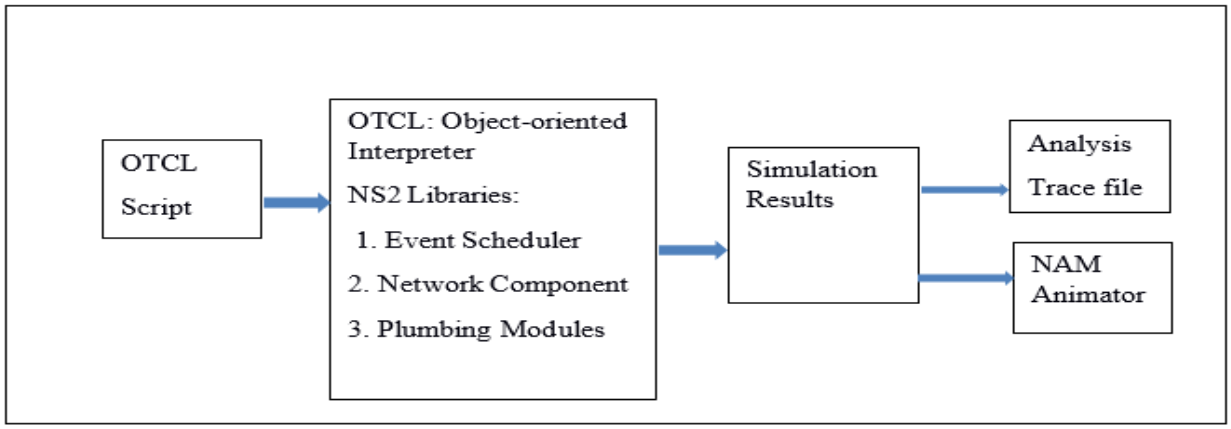


Figure 3. 3 view of ns2.

3.2.2 Tool Command Language (TCL)

The string-based command language known as Tcl (Tool Command Language) is intended for desktop and web applications. With a limited syntax and few fundamental constructs, Tcl is easy to learn, yet its full power is realized with experience. It is widely used for rapid prototyping, scripted applications, and testing. Tcl allows dynamic overriding and redefining of components, treats all data types as strings, and follows simple syntactic rules. It supports event-driven programming with flexible scope and restricted variable visibility, and it features basic exception handling through return codes from commands. There are at least three considerations to be carried out on simulation:

1.) The network appearance

This indicates the overall topographical arrangements of the network. The network appearance can be set by the following parameters. The position of nodes with coordinates, the starting time, the movement of nodes to which direction, the node speed and pausing time.

2) Internal of the network

In order to establish a connection, source nodes and destination nodes should be identified. The type of traffic, connection types, and maximum connections are the internals of the network.

3) Node Configuration

This involves the detailed setup of network parts on the mobile node. To create MANET simulations, the following steps should be taken place:

Step1: Create an instance of the simulator

Step2: Setup traces support by opening a file and call the procedure trace-all

Step3: Create a topology object that keeps track number of all the nodes within the boundary

Step4: Set up the operations director

Step5: Node configuration

Step6: Create nodes and the random-motion

Step7: Give nodes

Step8: Setup nodes

Step9: Give the traffic flow

Step10: Set halt/stop time

Step11: Define the command to start the simulation

3.2.3 Network Node Movement Scenario and Traffic Generation

In ns2 to create a TCL (Tool Command Language) script to manually set up a mobility file, but it generates it automatically using a specific command. Here's a breakdown of the parameters:

'/setdest [-n <number_of_nodes>] [-p <pause_time>] [-s <speed>] [-t <simulation_time>] [-x <width>] [-y <height>]'

Where n: Number of 'nodes**,** p: Pause time**,** **s**: Speed**,** **t**: Simulation time and x & y: Dimensions of the simulation area.'

Additionally, network traffic can be generated using the cbrgen.tcl script, located in the '.../indep-utils/cmu-scen-gen/cbrgen.tcl' directory of your ns2 installation. This script is easily readable and can be modified to better suit your needs. To automatically create a traffic connection file with 'cbrgen.tcl', you need to define parameters such as the traffic type, number of nodes, maximum connections, random seed, and rate. To use the cbrgen.tcl script in ns2, the command syntax is as follow.

'ns cbrgen.tcl [-type] [-nn] [-seed] [-mc] [-rate]

- **type**: Indicates the kind of traffic, which may be TCP or CBR (Constant Bit Rate).
- **nn**: Shows how many nodes there are in the network. The traffic pattern that will emerge when all other parameters are the same is defined by the **seed option**.
- The maximum number of connections between source and destination nodes is denoted by the **symbol -mc**.
- **rate:** This is used to calculate the interval of time for the packet sending rate.

After generating the mobility and traffic files, you can merge them into the original TCL files. This allows you to run simulations multiple times while varying the simulation parameters

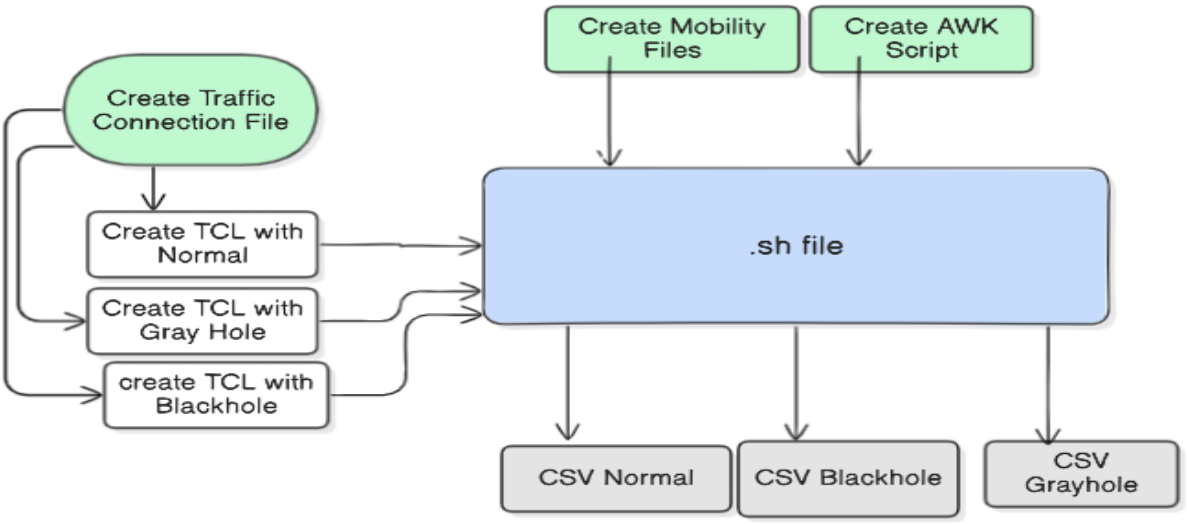


Figure 3. 4 Dataset automation ns2 scenarios for simulations

3.2.4 MANETs Attack Modeling

It is crucial to concentrate attack modeling on a particular layer of the protocol stack since Mobile Ad Hoc Networks (MANETs) are vulnerable to a variety of assaults across multiple layers. A major difficulty is making sure a network layer routing protocol functions properly and effectively when there is a rogue node trying to interfere with routing services.

3.2.5 Blackhole Attack Modeling

To create an attacker node, the following steps are undertaken

Step 1: Modify the aodv.h File: In the aodv.h header file, declare a Boolean variable to represent the attack state.

Step 2: Modify the aodv.cc File: Initialize the attacker variable, assuming that the network is initially free of attacks, set the variable to false. Once the attacker is introduced, packets will be dropped, so update the variable to true and implement the condition for packet dropping.

Step3: Compile the Files: Use the "make" command to compile the changes made in the aodv.cc and aodv.h files. If there are no errors, the compilation should be successful.

Step 4: Modify the Normal Tcl File: In the normal Tcl file, configure the attacker node as follows: <<**$ns at 1.0 "[$node(3) set ragent] blackhole".>>**

Step 5: Run the Tcl Script: Execute the modified Tcl script to initiate the simulation.

Step 6: Analyze the Trace File: Use the AWK tool to analyze the trace file and collect the results in a CSV format.

3.2.6 Gray hole Attack Modeling

To create an attacker node that simulates a gray hole attack, follow these steps

Step1: Modify the aodv.h File: In the aodv.h header file, declare a Boolean variable to represent the gray hole state.

Step 2: Modify the aodv.cc File: Initialize the gray hole variable, assuming the network is initially attack-free, and set the variable to false.

Step 3. Compile the Files: Use the "make" command to compile the changes in the aodv.cc and aodv.h files.

Step 4: Modify the Normal Tcl File: In the normal Tcl file, set up the attacker node by adding a command like; <<**$ns at 1.0 "[$node(3) set ragent] grayhole"**>>

Step 5: Run the Tcl Script: Execute the modified Tcl script to start the simulation with the grayhole attacker.

Step 6: Analyze the Trace File: After the simulation, use the AWK tool to analyze the trace file and collect relevant metrics in a CSV format.

3.2.7 Parameter Selection for Simulation

To create and simulate a Mobile Ad-Hoc Network (MANET), various parameters need to be selected. These include the node number of nodes, pause time, mobility speed, mobility model, transmission range, and others. Appendix A provides some of the parameter settings used in the simulation.

Mobility Model: The mobility is a critical factor in MANETs, as real-world mobility is unpredictable and significantly impacts the protocols designed to support node movement. Mobile nodes move randomly, and the Random Waypoint Model (RWP)[43] , introduced by Johnson and Maltz, is commonly used to simulate this randomness. In our simulations, we employed the RWP model.

Number of Nodes: is a primary parameter in MANET simulations. While any number of nodes can be chosen, a large number can lead to network congestion, degrading overall performance.

Pause Time: the duration a mobile node remains stationary before moving to a new location.

Traffic Type: In ns2, traffic is typically generated using traffic agents like TCP and UDP, which are based on specific statistical distributions. TCP is a reliable, bidirectional, connection-oriented transport layer protocol that uses a handshake mechanism to establish connections. FTP, an application layer protocol, is often used with TCP. UDP, on the other hand, is an unreliable, unidirectional, and connectionless transport layer protocol, often paired with CBR (Constant Bit Rate), an application layer protocol.

Simulation Time: This is the total time required to execute the simulation. **Simulation Area:** The simulation area defines the boundaries within which mobile nodes move, specified by X and Y dimensions. This is a crucial parameter in MANET design.

Transmission Range: is represented as an ideal circle around the source and destination nodes. we used a default transmission range of 250 meters, which is standard for wireless networks.

Routing Protocol: The performance of a MANET is heavily influenced by the routing protocol used. Table 3.2, provides an example of the simulation parameters employed.

Table 3. 2: Simulation Parameters

Parameter	Values	Remark
Simulator	ns2.35	--
Examined Protocol	AODV	--
Traffic Type	CBR and FTP	--
Agent	UDP and TCP	--
Transmission Range	250 m	Variable values used
Packet Size	512 bytes	Variable values used
Pause Time	5, 10, 20, 50 s	Variable values used
Simulation Time	40, 50, 60, 100 s	Variable values used
Number of Nodes	20, 40, 50, 60, 100	Variable values used
Simulation Area	1440 x 1000	Variable values used
MAC Layer	802.11	--
Movement Model	Random Way Point (RWP)	--
Channel Type	Wireless	--
Link-Layer Type	LL	--
Antenna Type	Omni antenna	--
Types of Attack	Blackhole attack, Gray hole attack	--

3.3.NAM (Network Animator)

The Network Animator (often referred to as "nam") is a standalone program distributed alongside the network simulator. It is designed to visually represent the movement of packets throughout the network. The program can read an input file that contains packet transmission events and graphically display these network events. It is one of the most commonly used tools in network simulation. Users can execute it directly from a scripting language, allowing them to control playback, pause, and monitor packets effectively. To visualize a "nam" file, you simply need to

run the command nam followed by the file name with a ".nam" extension. In the Figure 19 bellow shows network animation.

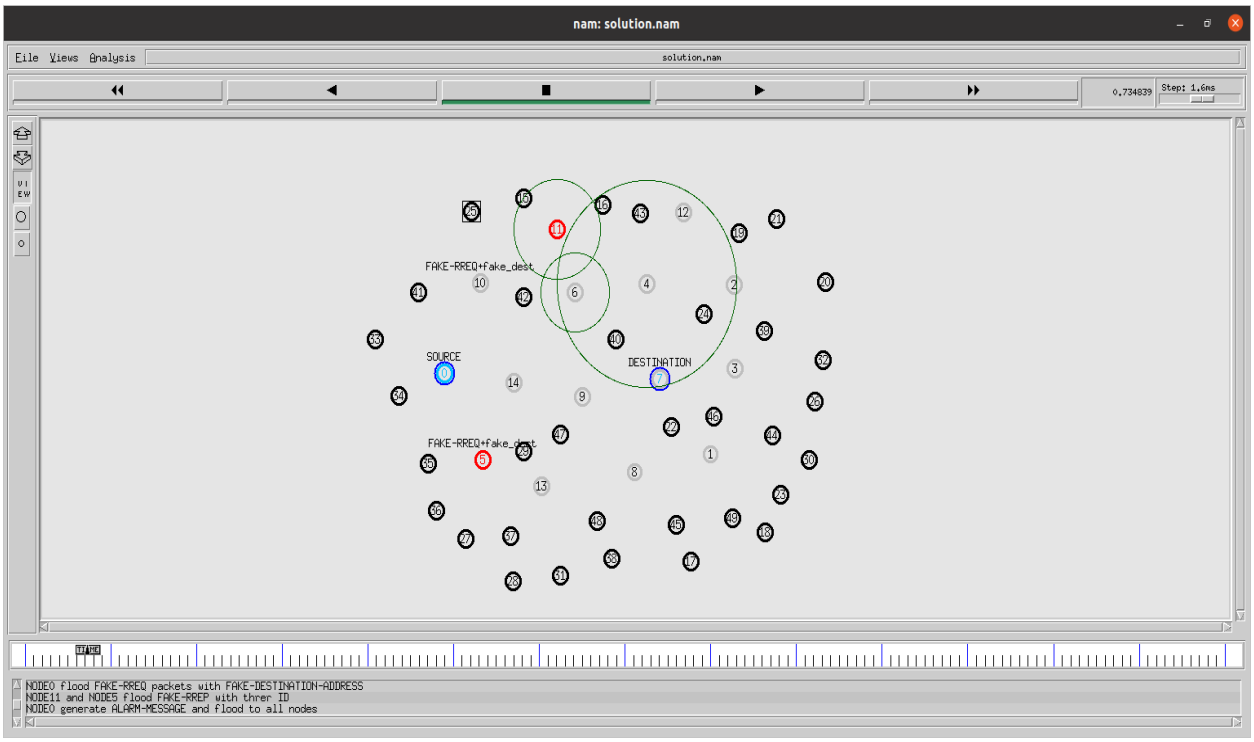


Figure 3. 5 An example of network animation.

A trace file in ns2 is created by an application to store coverage or overall network information. After executing the Tcl file, this trace file is generated and can be analyzed to extract various network features. The following is an example of a row from the trace file for clarification:

s 1.000000000 _0_ AGT --- 0 cbr 1000 [0 0 0 0] ------- [0:0 3:0 32 0] [0] 0 0

r 1.000000000 _0_ RTR --- 0 cbr 1000 [0 0 0 0] ------- [0:0 3:0 32 0] [0] 0 0

s 1.000000000 _0_ RTR --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [3 0] [0 4]] (REQUEST)

s 1.007371858 _5_ RTR --- 0 AODV 44 [0 0 0 0] ------- [5:255 0:255 30 2] [0x4 1 [3 4] 10.000000] (REPLY)

To further clarify, table 3.3, explains the common naming practices for NS2 trace files. The

computation of the required network performance metrics, which will be covered in the ensuing subsections, depends on this naming scheme.

Table 3. 3: Simulation setup and Parameter Selection for Simulation

Column	Description	Example Value
$1	Event Type	s (send), r (received), f (forward), d (drop)
$2	Time at which event happens	1.001408772
$3	Node_ID	10
$4	Layer event occurred	AGT, RTR, LL, IFQ, MAC
$5	Flags	---
$6	Sequence No. of Packets	0
$7	Traffic Type	CBR, AODV, TCP, ARP
$8	Packet Size	44
$9	Additional Information (e.g., duration, MAC addresses, MAC type)	[...]
$10	Additional Flags	---------

3.3.1 Abstract Window ToolKit (AWK)

The programming language AWK was created specifically for text processing. In the context of ns2 trace files, each line is treated as a record in a sequence. The AWK tool can compute all necessary features that serve as inputs for machine learning algorithms. This flowchart captures the essential workflow of an AWK script, highlighting the flow from the BEGIN block through line processing to the END block. It's a useful tool for understanding how AWK processes input data systematically. To conduct the simulation, we implemented various scenarios as outlined in Figure 3.3. The traffic-connection file, along with the standard TCL script file, mobility file, and awk file, were executed using a '.sh file'. The simulation was initiated by running the command "./allinone.sh," and the resulting feature values were captured in a CSV file labeled as CSV-

normal. This process was repeated multiple times by adjusting different simulation parameters to ensure a comprehensive dataset.

The described methodology ensures that the simulation covers a wide range of scenarios, including both normal and attack conditions, by systematically varying parameters such as the number of attack nodes and other control variables. This approach enhances the reliability and generalizability of the dataset, which is critical for developing an effective intrusion detection system. By collecting data under diverse conditions, the model can better learn to distinguish between normal and malicious activities, enhancing its precision and resilience in actual MANET settings. Additionally, the use of automated scripts (‘.sh fil).

Therefore, to analyze and calculate the necessary features from the trace file using AWK tool. Features that we have used for dataset preparation are listed in Table 3.4, below This table summarizes the features along with their short descriptions, making it easy to reference and understand each metric's significance.

Table 3. 4: Feature Name and Short Description

Feature Name	Short Description
Packet Sent	Total data packets sents by the source node
Packet Received	Total packets of data that the target node received
Packet Loss	Total number of network data packets drop
Throughput	The data rate generated by the application
Control Overhead	Total no.of routing packets, including forwarding packets
Normalized Routing Load	Total no.of routing packets transmitted per data packet
Average Delay	Total time required for the packet to arrive at the destination
Packet Delivery_ Ratio	The ratio of transmitted packets to sent packets
Remaining Energy	Amount of energy left after a node has consumed
Number of RREQ	Total route requests generated by the mobile node
Number of RREP	Total route replies generated by the mobile node
Traffic Type	Traffic agent (CBR and FTP)

3.4 Distribution of Dataset

Data collection automates the process of organizing all relevant files in order to finalize the dataset. The dataset was finally assembled into a CSV file. Dataset simulation features were gathered

28,691 occurrences in all, together with the classes that corresponded to them, as shown in Table 3.5, This procedure's code can be found in Appendix A, B and C

Table 3. 5: Distributions of the Dataset

Class name	Number of instances
Blackhole	9748
Gray hole	6890
Normal	12053
Total	28691

3.4.1 Dataset Description

When assessing MANET attacks Intrusion Detection Systems (IDS), the dataset selection is essential for verifying the suggested system's technique. Datasets such as, ADFA13, KDD99, ISC2012, and DARPA98 have been used by a number of researchers to evaluate the effectiveness of security measures. However, because of their shortcomings such as a lack of traffic diversity, limited feature sets, and poor metadata. so, this research not to include these datasets in our analysis. Furthermore, some datasets do not represent current network topologies and attack trends, nor do they capture a wide range of attacks. Additionally, they are not designed with wireless networking, especially (MANETs).

In this study, were built a MANET dataset to describe two kinds of packet-dropping assaults in addition to typical attack-free conditions. Results from the simulation were produced using the NS-2 simulator, and the NS-2 tracing system gathered unprocessed data for both attack and normal conditions. This contained information about packets that were received, sent, and dropped, including RREQ, RREP, RREPACK, and RERR. To collect data about transport connections between nodes, such as the volume of data sent and received, packet sizes, and delays, application layer traffic was created. Our dataset's labeling was determined by the general state of network behavior, considering both typical circumstances and packet-dropping attacks.

Compared to node-level labeling, which is susceptible to modifications in node functioning, this method is more reliable. In our dataset, every data input is classified as either normal or attacks (for black-hole and gray-hole assaults, respectively. Each traffic record in the created dataset has 21 features and one class label, covering all the necessary details.

3.4.2 Preprocessing Phases Data

Following the simulation, as detailed in sections 3.5.1, we were given CSV raw data files, which include lists of integers and numbers as well as text that cannot be directly fed into a deep learning model. Furthermore, because the gathered raw dataset includes features like attack type and protocol type that are discussed in this section, the raw data files are not suitable input for the learning method. The dataset, which is in CSV format, must be loaded as the initial stage in the data pre-processing procedure. This can be accomplished using the 'pandas' library. To do this efficiently, it is advisable to keep the dataset in the same directory as the Python code, allowing for easy access via the read_csv method. The steps of dataset pre-processing are shown in the flow chart 3.6.

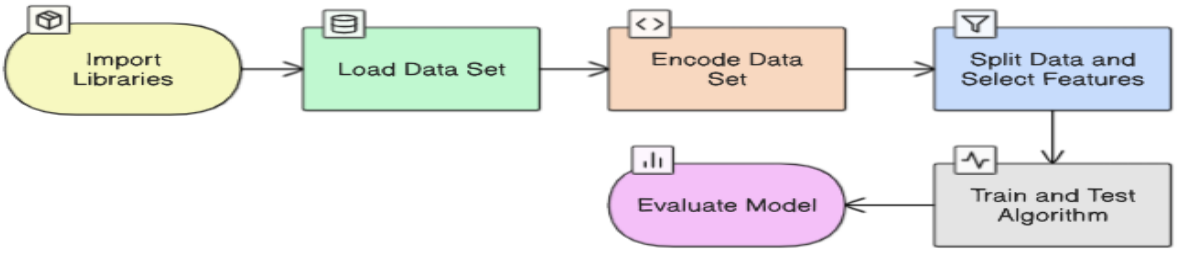


Figure 3. 6 Dataset preprocessing steps

A specific MANET dataset is created for this study in order to describe two different kinds of packet dropping assaults as well as typical behavior. Both number and non-numeric values can be found in these built dataset properties. Therefore, the encoding/numericalization technique was used to convert non-numeric properties to numeric ones. Data standardization was then limited to numerical attributes, and data balancing was employed to address the problem of the unbalanced dataset that leads to biases in deep learning Data normalization, particularly using the standard scaling method, is a crucial preprocessing step in machine learning and deep learning processes[44].

Table 3. 6: Data Set Data Types

Feature Name	Types	Short Description
Protocol type	Object	Protocol types UDP ant TCP
Packet Sent	Int	Total packets of data that the source node sent
Packet Received	Int	Total packets of data that the target node received
Packet Loss	Int	Total number of network data packets dropped
Throughput	Float	The data rate generated by the application
Control Overhead	Float	Total no. of routing packets, including forwarding packets
Normalized Routing Load	Float	Total no. of packets routing transmitted per data packet
Average Delay	Float	Total time required for the packet to arrive at the destination
Packet Delivery Ratio	Float	The ratio of transmitted packets to send packets
Remaining Energy	Float	Amount of energy left after a node has consumed
Number of RREQ	Int	Total route requests generated by the mobile node
Number of RREP	Int	Total route replies generated by the mobile node
Traffic Type	Object	Traffic agent (CBR and FTP)
Label	Object	Class: normal, black-hole and wormhole

3.4.3 Data Normalization and Data Label Encoding

In simulation data set Both numerical and non-numerical properties can be found in the raw datasets that were gathered. But for the learning algorithms to function properly, just numerical values are needed. In order to change the unified format of non-numeric features, we used feature conversion. Our MANET dataset has 2 non-numeric features and 19 numeric features. The datasets contain normal, black, gray classes. Attacks are assigned as labels 1 and 2 and normal is labeled as 0. And protocol type UDP and TCP assigned 3 and 4 respectively.

The collected raw datasets contain both numerical and non-numerical features. But for the learning algorithms to function properly, just numerical values are needed. Therefore, in order to change their unified format, we used feature conversion to non-numeric characteristics. There are 19 numeric features and two non-numeric features in our MANET dataset.

Detect common network layer attacks, including blackhole, and gray hole attacks, which constitutes a multiclass problem with three categories, including normal node behavior. Consequently, the class labels for the dataset were designated as blackhole, gray hole, and normal.

Table 3. 7 Encoded attributes labels of the textual data after encoding.

Textual data	Blackhole	Gray hole	Normal	UDP	TCP
Label	1	2	0	3	4

Transforming category names, such as "normal," "blackhole," and "grayhole," into a numerical representation that machine learning models can comprehend is usually required. One-hot encoding, also referred to as label encoding, is a widely used technique that converts categorical labels into a binary vector representation. Transforming category names, such as "normal," "blackhole," and "grayhole," into a numerical representation that machine learning models can also referred to as label encoding, is a widely used technique that converts categorical labels into a binary vector representation. Another non-numerical value is assigned in the labeled encoder UDP and TCP values encoding 3 and 4 respectively.

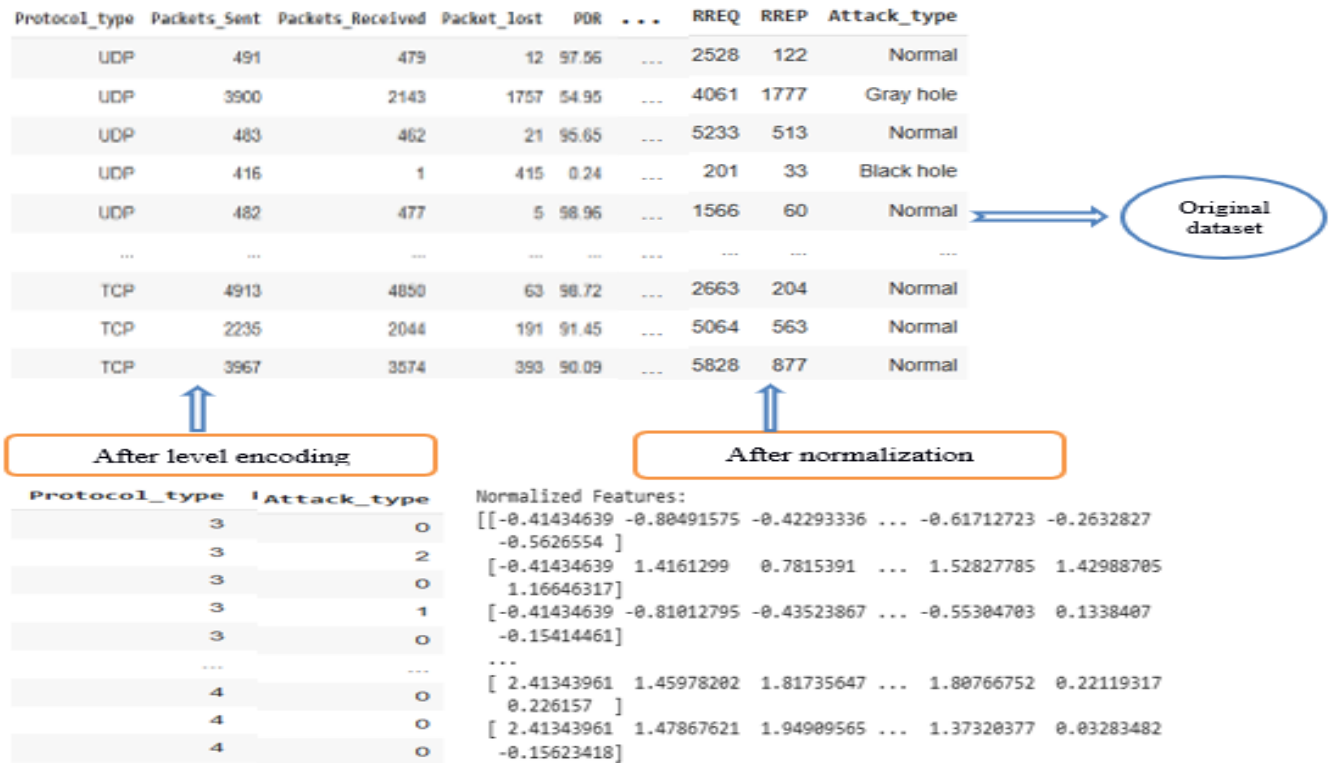


Figure 3. 7 Sample output of dataset before and after preprocessing

3.6.4 Data Balancing

Biases in machine learning are caused by the issue of unbalanced datasets. When one class is far more prevalent than the others, the dataset is said to be unbalanced. Prior to training a classifier, the data must be balanced. The problem of class imbalance, where one class (majority class) has significantly more samples than another (minority class). This imbalance can hinder the performance of classifiers, especially for the minority class, which is often the focus of interest in various applications. challenges associated with imbalanced datasets in machine learning and data mining[45].

When training on an unbalanced dataset, classifiers often predict the majority class with high accuracy and the minority class with relatively low accuracy. The biased result is caused by the majority class's higher error signals to the loss function, which classification algorithms try to reduce during training. Researchers have suggested three methods to deal with this problem: sampling-based, ensemble-based, and cost-sensitive using class weight computing.

The researcher [37], Deep learning encounters obstacles and hurdles in DDoS detection, including high computing needs, lengthy training periods, and intricate model interpretation. overcome these obstacles by putting out a practical method for identifying DDoS attacks in network systems that are out of balance[46]. By enabling models with LSTM to efficiently learn time anomalies during DDoS attacks, SMOTE can broaden the class distribution of the data set. They include accuracy of 99.50 and 97.50 with validation loss results of 0.048 for LSTM SMOTE and 0.1943 for LSTM without SMOTE. The F1 score, however, increased from 93.4% to 98.3%. SMOTE is a useful technique for enhancing models' robustness and dependability as well as their ability to identify DDoS attacks on heterogeneous networks. [47].

The distribution of our dataset samples is not uniform. For instance, almost 40.5% of the data are “normal”, 32.6% “Black hole and 27.0%” Gray hole”. However, training results are biased in favor of larger data categories.

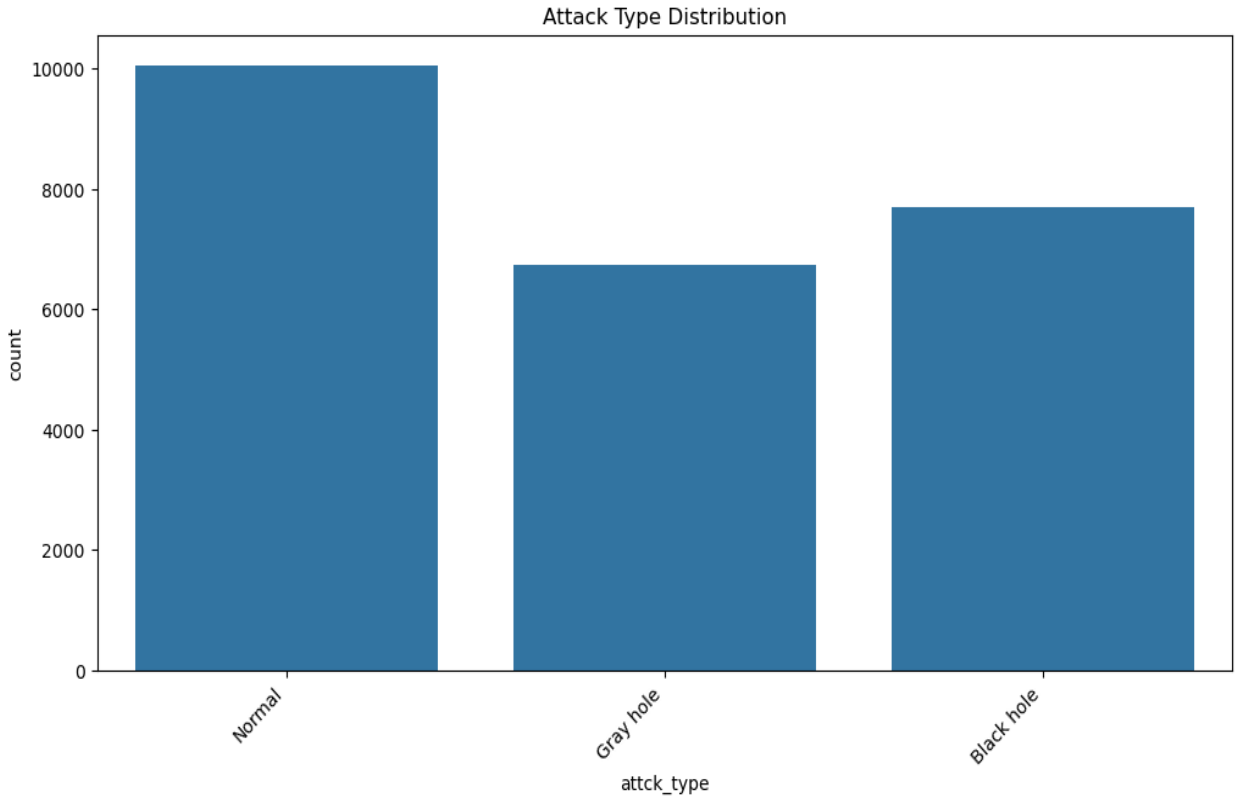


Figure 3. 8 Distribution of dataset

When working with uneven datasets that contain underrepresented classes in one or more categories, data balancing is especially crucial. Biased models that perform badly on minority groups can result from unbalanced data. Inconsistency or misunderstanding in the SMOTE data balancing procedure. To balance the dataset, SMOTE (Synthetic Minority Oversampling Technique) is typically employed to enhance the number of samples in the minority class.

3.4.4 Feature Selection

A crucial stage in machine learning is feature selection, which selects the best subset of features based on factors like correlation and relevance score, among others. It is the process of selecting significant parameters or features from the sizable dataset that has been gathered. Its benefits include improving the generalization and comprehension of the data, decreasing complexity, removing unnecessary information, and boosting the performance of learning algorithms[48].

MANET dataset is built in this study to describe two types of packets dropping attacks and normal behavior without deployed attacks. Building the classification model takes a lengthy time when the datasets contain features that are useless, redundant, or only marginally significant, which lowers the classification accuracy. In order to improve the model's performance and reduce the computational cost of modeling, fewer input variables are required.

In this research were uses in order to improve the classification of attacks and typical network activities, we employed extreme gradient boosting (XGBoost) and categorical boosting techniques (CatBoost) to extract the most significant features from the MANET dataset. To determine the greatest accuracy for identifying the two packet dropping assaults and regular traffic, the impact of these MANET features with varying thresholds was investigated.

The Gradient Boosting Decision Tree (GBDT) technique is effectively implemented by GBoost, also known as XGBoost (Extreme Gradient Boosting). Its design prioritizes portability, flexibility, and excellent performance. By employing strategies like regularization to avoid overfitting and improve model generalization, XGBoost optimizes the gradient boosting framework[49].

Additionally, each feature's value or usefulness in building the model's boosted decision trees is indicated by its score or rank in feature importance. Additionally, other relevance indicators, including gain, frequency and cover, are used to determine the importance score values[50].

The main algorithmic methods underlying CatBoost, a novel gradient boosting toolbox, enable CatBoost to outperform other boosting implementations that are made publicly available in terms of quality across a range of datasets Two important algorithmic innovations introduced by CatBoost are the use of ordered boosting, a permutation-driven alternative to the conventional method, and a novel methodology for handling categorical data [51].

In this research, we used CatBoost to solve this problem suggested algorithms successfully resolve it, producing superior empirical outcomes. With CatBoost, the evaluation index can be used to determine the feature coefficient or feature importance score after the model has been trained using a loss function and prediction values have changed successfully, producing outstanding empirical results.

To find the best subset of features from the dataset, the feature selection procedure was put into place. This was achieved using the XGBoost and CatBoost algorithms, which involved conducting

extensive experiments with various thresholds to evaluate feature importance through gain metrics and changes in prediction values, respectively. The dataset initially contained 21 features, some of which were irrelevant or redundant. These unnecessary features introduced noise, negatively impacting the performance of the proposed models. Additionally, the computational cost associated with training on a dataset with a large number of features is a significant consideration, as it can prolong the training duration. Therefore, the feature importance scores for the 21 features, including those that were selected or discarded, were carefully evaluated.

Based on these results, CatBoost is identified as the better-performing model, and the optimal feature set of **16 features** (out of the original 21) is selected for further analysis

. These selected features, as detailed in Table 4.2, are then used as input for a deep learning models, including:

- Convolutional Neural Networks (CNN) and its variations.
- LSTM (Long Short-Term Memory)
- GRU (Gated Recurrent Unit)

These models are employed for the classification of packet-dropping attacks and normal network traffic. The goal is to leverage the optimal feature set to improve the performance of these classifiers in detecting and distinguishing between malicious and normal network behavior.

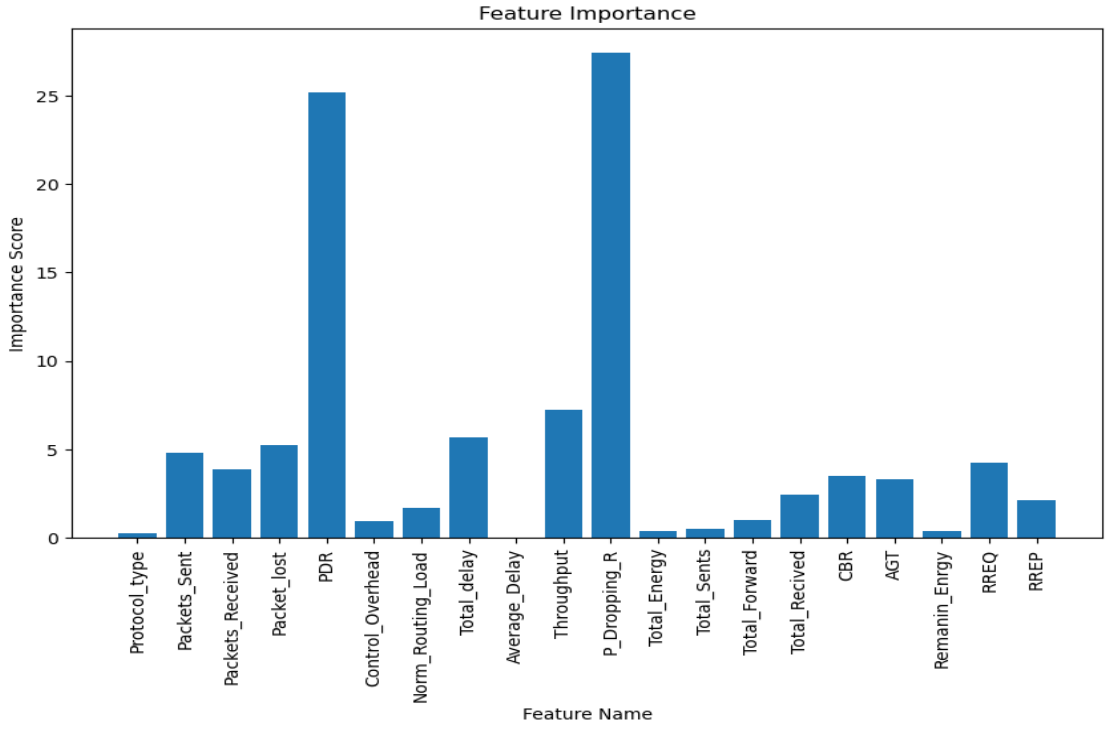


Figure 3. 9 Feature Importance

Table 3. 8: Selected and not selected features by using feature importance

Feature Name	Importance Score	Remark	Feature Name	Importance Score	Remark
PDR	25.413987	Selected	Received	1.999999	Selected
P_Dropping.R	22.737688	Selected	Control Overhead	1.989392	Selected
RREQ	10.090874	Selected	Packets Sent	1.441129	Selected
CBR	8.952772	Selected	AGT	1.415479	Selected
RREP	6.394513	Selected	Norm_Routing_Load	1.386471	Selected
Packet lost	4.165510	Selected	Forward	1.024969	Selected
Throughput	4.102728	Selected	Sents	0.774288	Not selected
Packets Received	3.063815	Selected	Protocol type	0.613783	Not selected
Av_Delay	2.208146	Selected	Total Energy	0.593067	Not selected
Total Delay1	2.014811	Selected	Remain Energy	0.516580	Not selected

Deep learning is fed the chosen features displayed in Table 3.10. For the purpose of classifying packet dropping attacks and typical in machine learning and deep learning, feature scaling is an essential preprocessing step. It entails converting your dataset's features, or input variables, to a comparable scale or range. This is key since a lot of machine learning algorithms are sensitive to the size of the input data, and higher-magnitude features may overpower lower-magnitude ones, producing biased or less-than-ideal outcomes. A method for adjusting feature values to a predetermined range, usually between 0 and 1 (or between -1 and 1 for datasets with negative values), is feature scaling, also known as normalization.

3.5 Classification Stage

The classification procedure determines if a particular record is an attack types or normal. Our goal is to examine how well CNN, GUR and LSTM on recent years, deep learning has been used to detect intrusions on the MANET network. Among the many deep learning techniques available are Convolutional Neural. These selected features, as detailed in Table 4.2, are then used as input for a deep learning models, including:

- Convolutional Neural Networks (CNN) and its variations.

- LSTM (Long Short-Term Memory)
- GRU (Gated Recurrent Unit)

These models are employed for the classification of packet-dropping attacks and normal network traffic. The goal is to leverage the optimal feature set to improve the performance of these classifiers in detecting and distinguishing between malicious and normal network behavior. Networks (CNN), Gated Recurrent Units (GRU), and Long-Short Term Memory (LSTM. how the proposed network-based anomaly detection system works.

Using the multiple threshold values for feature selection to improve the accuracy of the learning process, we assess the performance of the Xgboost and catboost algorithms in Low-importance features are eliminated the best feasible combination of the two algorithms is Xgboost and catboost to prevent bias in the values and to make it appropriate for the models, the preprocessed dataset was scaled using a conventional scaler before being directly fed into the deep learning model.

Therefore, to demonstrate the validity of the approach for model construction, normalized data must be converted into a 3-dimensional data format. The deep learning method then incorporates the 3-dimensional vectorized information. Python libraries are used to implement the learning algorithm. Both LSTM and GRU cells use the gating mechanism for long-term dependence. The experiment was conducted by the authors of Naseer et al. [52] and Kasongo and Sun [53],using the NSL-KDD benchmark dataset and LSTM and GRU, thus both did well and they couldn't decide which was superior. LSTM and GRU are therefore used as multi-classification classifiers in the suggested model.

In our case we experimented to determine hyperparameter settings to get the best IDS performance measures in order to build CNN, LSTM, and GRU classifiers. Accordingly, the following hyper-parameter is used by three of the models. The input layer sends preprocessed input data into the network, but the input data needs to be a 3D array in order to fit the model and produce predictions. In this case, we convert the two-dimensional input features into a three-dimensional array using a 28691 sample, one-time step, and 21 features at each time step.

The input layer of the network feeds preprocessed data into the model, requiring the input to be structured as a 3D array. In this case, we reshape the 2D input features into a three-dimensional format with 28,691 samples, 1-time step, and 21 features at each time step. The input layer is defined using the “input_shape” parameter in the first hidden layer. In the hidden layer, weighted

inputs are randomly initialized and transformed into a format usable by the output layer, employing the ReLU activation function. This layer is fully connected to both the input and subsequent layers, and its output dimensionality is specified accordingly.

The dense layer, which is sometimes thought of as an additional hidden layer, uses the ReLU activation function to process the output from the hidden layer in order to improve learning and generate a more stable output. Lastly, the output layer is made up of three neurons that produce probabilities for the normal, black, and gray categories. The softmax activation function is useful in this layer, allowing it to produce a probabilistic output for each category.

The training process is influenced by the learning rate, a crucial hyperparameter for CNN, LSTM, and GRU models. Selecting an effective neural network architecture and corresponding parameters is essential for minimizing errors, as these factors significantly impact the network's performance. To address this issue, the adaptive moment estimation technique is used in conjunction with the Adam optimizer. It uses the gradients' first (mean) and second (variance) moments to determine the learning rate [54]. Since it measures how well the algorithm performs on the dataset, the loss function is essential to model optimization. Since categorical cross-entropy is specifically made for multiclass classification tasks, we chose it as the loss function. Compared to previous options, this loss function works better for multi-class issues.

3.5.1 Data Separation for Datasets used for Testing and Training

The dataset's 21 features which include two categorical or symbolic variables and 19 continuous or discrete numerical properties were derived from network layers and MANET ns2 simulation datasets, as shown in Table 3.11.

Table 3. 9 Table representing the training and testing dataset samples

Class	No. of Training Samples	No. of Testing Samples	Total Samples
Normal	9664	4030	10694
Black	7805	1951	9751
Gray	5496	1377	6873
Total	**22965**	**5728**	**28691**

3.6 Methods for Experiments

All deep learning There were experiments using Keras, with TensorFlow as the backend, within the Anaconda3 environment running Python 3.9. Keras was chosen for this research due to its capabilities as a deep learning framework for constructing and evaluating complex neural networks. Its advantages include being an open-source library, integration with Python packages, having comprehensive user documentation, and its user-friendly nature. The experiments were carried out on an Intel(R) Core (TM) i3-8100 CPU operating at 3.60 GHz, with 4.0 GB of RAM, on a 64-bit Windows 10 operating system. The NS2 simulator was employed to generate and extract data in CSV format. The implementation of RNN and LSTM its variants was carried out using Keras, while Scikit-learn, XGBoost, CatBoost, and other machine learning libraries were utilized for data preprocessing, feature selection, classification, and data visualization

3.7 Metrics for Evaluation

To assess the performance of the produced model, we employ key assessment parameters such as accuracy (Ac), detection rate (DR) or true positive rate (TPR), false alarm rate (FAR) or false positive rate (FPR), recall, precision, and F1-score

- **True Positive (TP)** represents the number of correctly classified samples for each class. For instance, it includes normal samples classified as normal, black-hole samples classified as black, and worm-hole samples classified as worm.
- **False Positive (FP)** the number of samples incorrectly classified.
- **True Negative (TN)** for a specific class is calculated by excluding the rows and columns corresponding to that class in the confusion matrix and summing the values of the remaining entries, which represent the correct classifications of the other classes.
- **False Negative (FN)** denotes the number of samples incorrectly classified as another class.

The formulas for the performance metrics are as follows:

1. **Accuracy (Ac)**: Measures the proportion of correctly classified records out of the total records.

   Ac=TP+TNTP+TN+FP+FN……..(3.3)
2. **True Positive Rate (TPR)**: Also call as recall or detection rate (DR), it indicates the percentage of correctly identified attack records out of the total attack records.

TPR=TPTP+FN……. (3.4)

3. **False Positive Rate (FPR)**: Equivalent to the false alarm rate (FAR), it represents the percentage of normal records incorrectly classified as attacks.

   FPR=FPFP+TN……. (3.5)

4. **Precision**: Reflects the model's ability to correctly predict positive instances, measuring the accuracy of positive predictions.

   Precision=TPTP+FP……..(3.6)

5. **F1-Score:** A reasonable evaluation of the model's performance based on the precision and recall harmonic means.

   F1-Score=2×TP2×TP+FP+FN……(3.7)

These metrics were chosen to evaluate the effectiveness of our cross-layer anomaly detection models. They help demonstrate the models' performance in distinguishing between normal behavior and various types of attacks.

CHAPTER FOUR

4. Experimental Results and Discussion

4.1 Experimental Results

We conducted two sets of experiments to evaluate the models. In the first experiment, we used all available features from the dataset, which included 21 features. In the second experiment, we applied feature selection to reduce the dimensionality of the dataset, retaining only 16 optimal features. Both experiments were performed using the same model parameters and architecture to ensure a fair comparison.

An input layer with neurons equal to the dataset's feature count (21, depending on the experiment) makes up the model architecture. The two hidden layers that follow have 50 neurons in the second hidden layer and 64 neurons in the first. Three neurons, representing the three types of normal behavior, blackhole attack, and gray hole attack, make up the output layer, the last layer. 20% of the training data was set aside as a validation set prior to training in order to track and adjust the model's performance. Because the softmax activation function works well for multi-class classification issues, it was chosen for the output layer. The softmax function provides a probabilistic output, assigning probabilities to each class such that the sum of all probabilities is 1. This function is a smoothed version of the argmax function, where the neuron with the highest input value outputs a value close to 1, and the others output values close to 0.

The testing accuracies after features selection achieved by the models were 98.57% for LSTM, 98.48% for CNN, 98.60% CNN_LSTM and 98.46%for GRU when using the softmax function in the output layer and with optimal 16 features. During training, we observed a gradual increase in validation accuracy, avoiding sharp jumps that could indicate overfitting. Specifically, the validation accuracy increased from before features selection approximately 98.33% to 98.62% for LSTM, 98.00 % to 98.34% for CNN, 98.14% to 98.60% for CNN_LSTM and 98.36% to 98.59% for GRU.

Table 4. 1: Hyper parameters

Parameters	Value
Activation function	ReLU and softmax
Optimizer	Adam
Rate of learning	0.0001
Dropout	0.2
Loss function	Categorical cross-entropy
Size Batch	32

4.2 Experimental Results and Evaluation

From the trained CNN, LSTM, and GRU networks using a three-dimensional preprocessed dataset consisting of 16 optimal features. As a result, attack detection models for CNN, LSTM, hybrid CNN_LSTM and GRU. The performance metrics for each model are as follows: the CNN achieved a training accuracy of 98.34% with a training loss of 0.0510, while its testing accuracy was 98.48% and testing loss was 0.0499. The LSTM demonstrated a training accuracy of 98.62% and a training loss of 0.0360, with a testing accuracy of 98.57% and testing loss of 0.0374. The GRU recorded a training accuracy of 98.59% and a training loss of 0.0411, while its testing accuracy and loss were 98.46% and 0.0430, the CNN_LSTM hybrid model achieved a training accuracy of 98.66% with a training loss of 0.0382, while its testing accuracy was 98.60% and testing loss was 0.0401 respectively. From the above results CNN_LSTM are the most accurate second LSTM and CNN third and GUR lowest accuracy.

Figure 4.3, shows the CNN training and validation loss learning curves for the same number of epochs. It also shows the learning curve for CNN training and validation loss learning curves over 100 epochs. Effective learning and convergence are indicated by the graph's steady decline in training loss. Additionally, validation loss indicates possible overfitting as it first declines before stabilizing at a higher value than training loss. Both losses converge towards comparable values across the graphs, demonstrating the stability and convergence of the model.

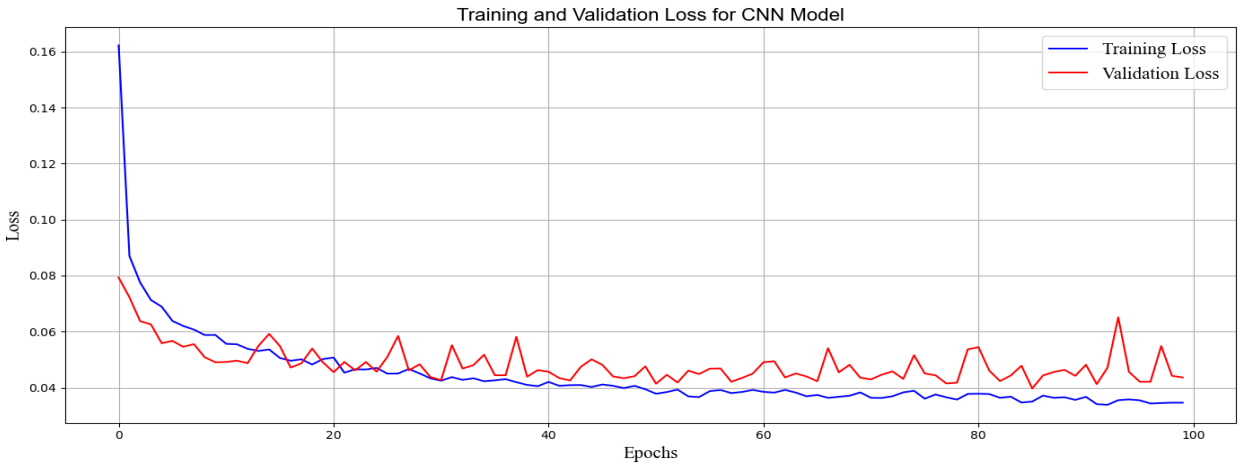


Figure 4. 1. Learning curve for CNN training and validation loss

Figure 4.4, displays the graph that illustrates the CNN training and validation accuracy learning curve over 100 epochs. Training Accuracy shows good learning and a good fit to the training data when it increases gradually and stabilizes at about 98%. Additionally, validation accuracy increases and stabilizes at a comparable level (around 98%), demonstrating how effectively the model generalizes to unknown data. However, there appears to be no discernible overfitting as both training and validation accuracies track closely. For all training and validation sets, the CNN model performs well with high accuracy, suggesting efficient learning and great generalization to new data.

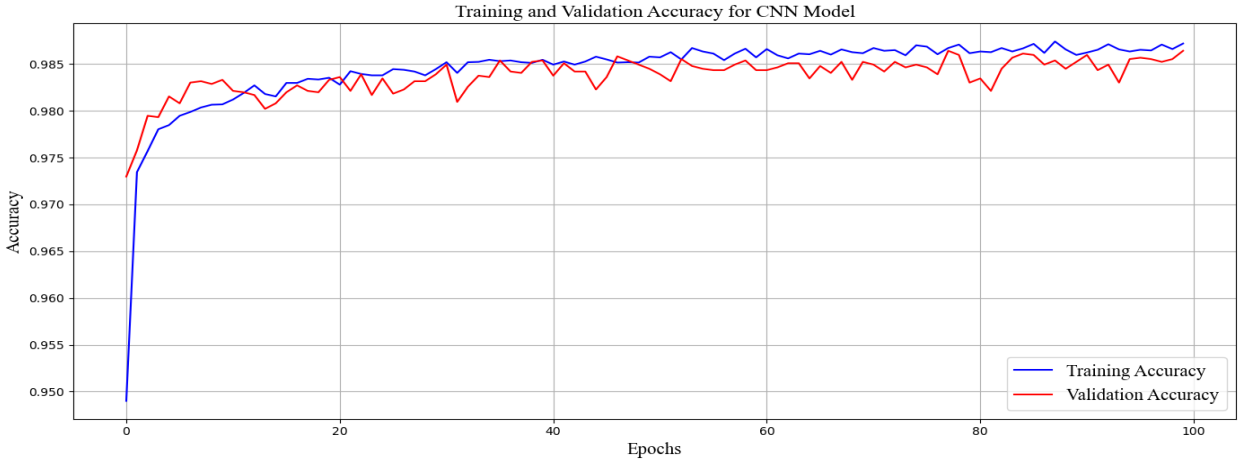


Figure 4. 2. Learning curve for CNN training and validation accuracy

LSTM training and validation accuracy learning and training and validation loss learning curve using 100 epochs are provided in Figure 4.5.

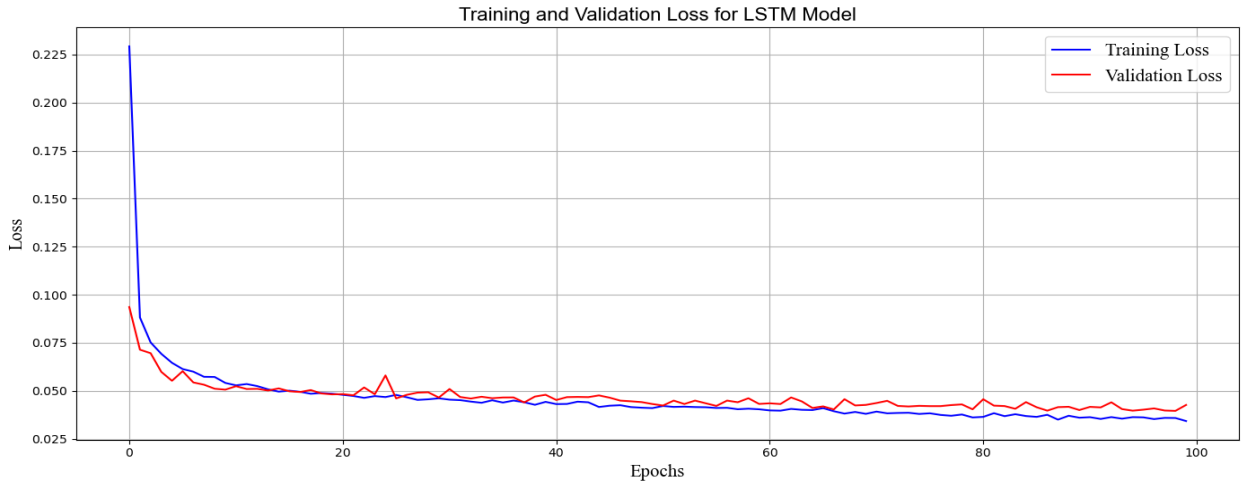


Figure 4. 3. Learning curve LSTM training and validation loss

The plot depicts the training and validation loss of an LSTM (Long Short-Term Memory) model across 100 epochs. The elements the number of training epochs, or how many times the model has run over the whole training dataset, is represented by the X-axis (Epochs). The loss values, which measure the model's performance, are displayed on the Y-axis (Loss). Better performance is indicated by lower values. Loss of Training (Blue Line). This line monitors the training dataset's loss. The model often declines as it learns from the training data. Loss of Red Line Validation. This line helps assess how effectively the model generalizes to new data by displaying the loss on an alternative validation dataset.

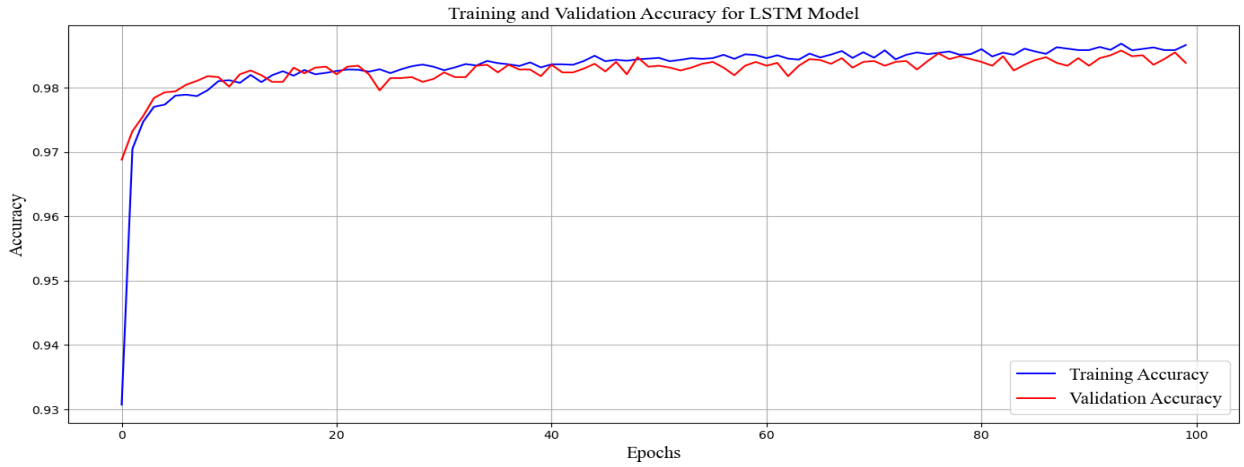


Figure 4. 4. learning curve LSTM training and validation accuracy

The plot shows the training and validation accuracy of an LSTM (Long Short-Term Memory) model over 100 epochs. The X-axis (Epochs) displays the number of training epochs, or the

number of times the model has processed the whole training dataset. The Y-axis (Accuracy) displays the model's accuracy, which measures the proportion of correctly predicted cases. Greater accuracy numbers, which range from 0 to 1, imply better performance. Accuracy of Blue Line Training This line tracks the accuracy on the training dataset. If there is an upward trend, the model is effectively learning from the training data. Accuracy of Red Line Validation By showing the accuracy on an alternative validation dataset, this line assesses the model's performance on unseen data.

GRU training accuracy and training loss learning curve using 100 epochs are presented in Figures 4.7, below. The GUR model shows effective learning with a strong decrease in training loss, but the higher and stabilizing validation loss suggests a need for monitoring to avoid overfitting. Adjustments may be necessary to improve generalization.

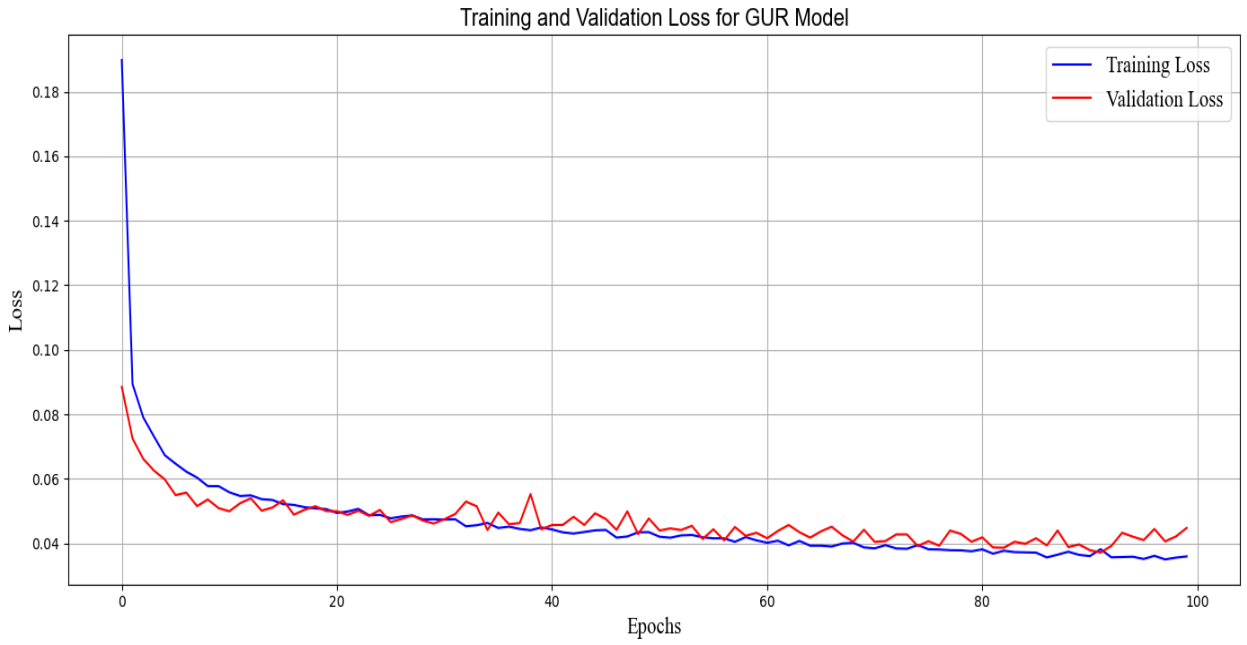


Figure 4. 5. GRU training accuracy and training loss learning curve

The accuracy of a model's training and validation across 100 epochs is depicted in the graph. The graph displays the following important findings. Effective learning of the model is indicated by the convergence of both training and validation accuracy towards a high value.

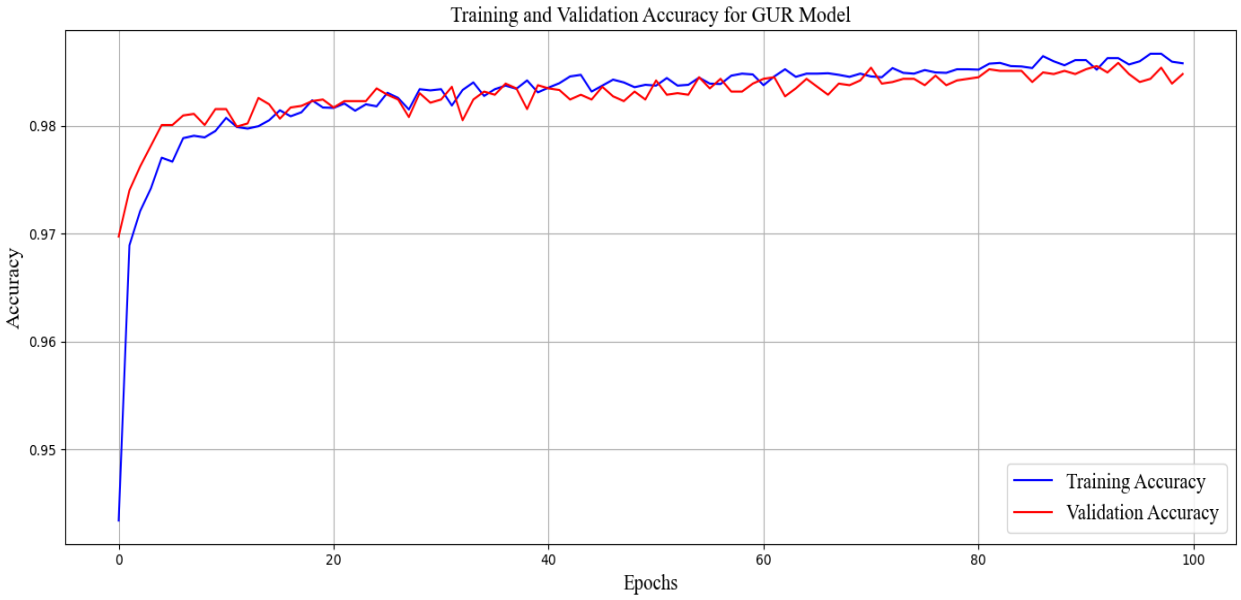


Figure 4. 6. GRU training accuracy and training accuracy learning curve

The graph displays the training and validation losses for the CNN-LSTM model over 100 epochs. A consistent drop in training loss (blue line) indicates effective learning. The validation loss (red line), which initially decreases before beginning to fluctuate, indicates possible overfitting. It is anticipated that both losses will decrease and stabilize with the validation loss soon after the training loss.

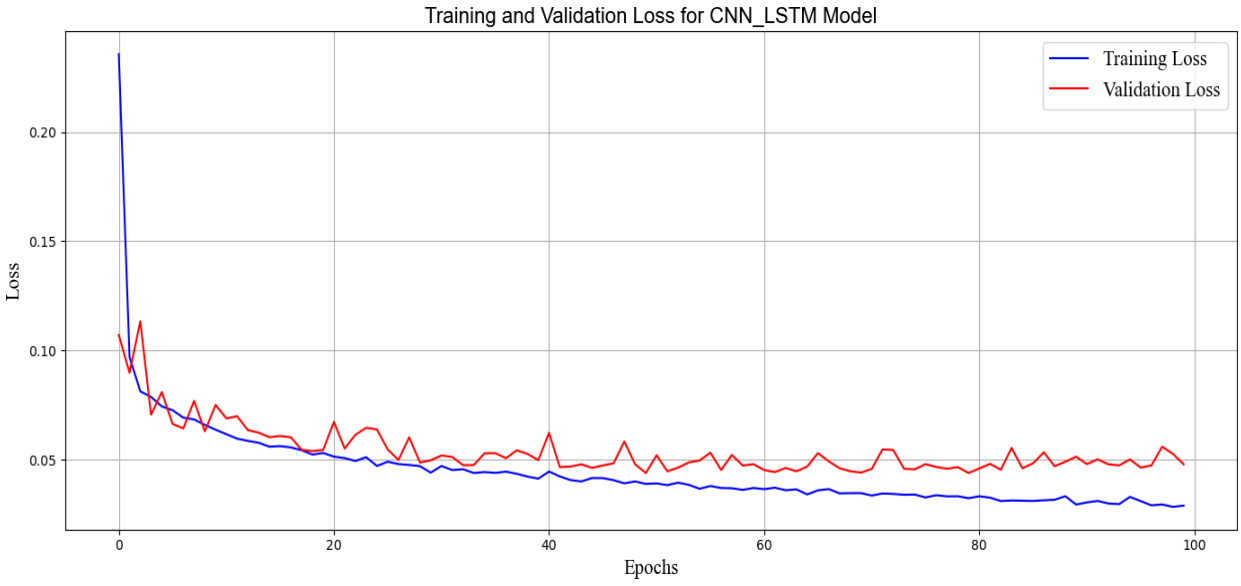


Figure 4. 7 . CNN-LSTM model's training and validation losses

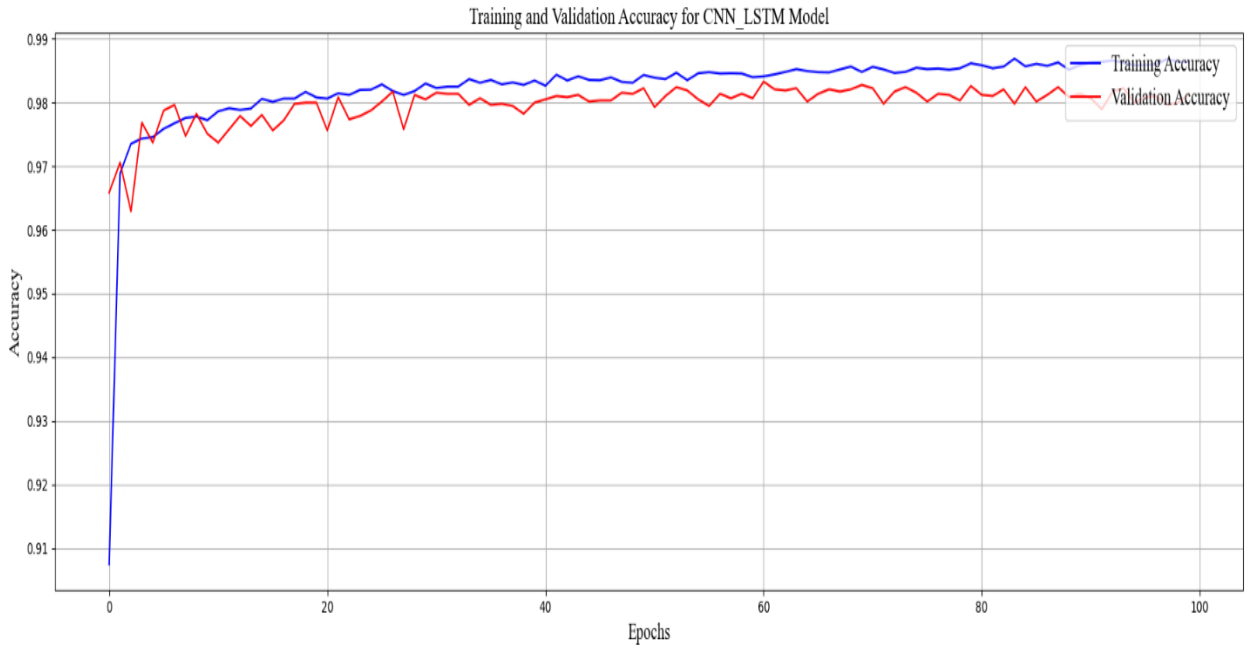


Figure 4. 8 . CNN-LSTM model's training and validation accuracy

The CNN-LSTM model's training and validation losses over epochs are displayed in the graph. The number of epochs, or iterations, in the training dataset is shown on the X-Axis. The model's performance is shown by the loss value, which is displayed on the Y-Axis (the lower the better). The model is learning from the training data, as indicated by the blue line (training loss), which decreases and then stabilizes. Good model generalization is shown when both losses reduce and converge without noticeable gaps, which is the optimum situation overall.

4.3. Model Performance Evaluation Metrics

The analysis and performance assessment of the deep learning-based model for packet-dropping attack detection are presented in this subsection. Accuracy, Classification Report, and Confusion Matrix are used to assess the classification performance of CNN, LSTM, CNN_LSTM, and GRU models. The analysis shows that feature selection greatly enhances CNN, LSTM, and GRU models' ability to identify packet-dropping attacks. LSTM and GRU are expected to outperform CNN among the three models due to their ability to detect long-term dependencies in sequential data. The model that performs the best can be determined by looking at its accuracy and F1-score.

4.3.1 Classification Report

Before implementing the feature selection method, the proposed deep learning algorithms CNN_LSTM, CNN, LSTM, and GRU were evaluated using all 21 preprocessing features available in the dataset. Table 4.5 and 4.6 presents the performance metrics, including Precision, Recall (Detection Rate), F1-score, FRA, accuracy, and error rate, for two types of attacks (blackhole and gray hole) and normal behavior, based on the test data without feature selection. The results indicate that the CNN, LSTM_CNN, LSTM and GRU models achieve a higher detection rate for normal behavior, blackhole attacks, and gray hole attacks. Additionally, the F1-score, Precision, and detection rate for wormhole attacks are consistently higher across all five classifiers compared to blackhole attacks and normal behavior. In Tables 4.5, the labels Normal, Black hole, and gray hole represent normal, blackhole, and gray hole, respectively. The CatBoost feature selection technique was used in the studies, and the most pertinent features were found by utilizing its feature importance scores. To identify the ideal subset of features, several threshold values were applied to the feature importance scores.

Different feature combinations were chosen by adjusting the thresholds, and the effect of these feature combinations on the model's performance was assessed. This method made it possible to systematically examine how feature selection affects the suggested deep learning models' accuracy and effectiveness (CNN, LSTM, hybrid CNN_LSTM, and GRU). The findings showed that choosing features according to their relevance scores greatly enhanced the models' overall performance and detection rate, especially when it came to recognizing normal behavior, blackhole attacks, and gray hole attacks. Additionally, the use of CatBoost for feature selection ensured that the most discriminative features were retained, leading to better generalization and reduced computational complexity. The findings highlight the effectiveness of combining CatBoost-based feature selection with deep learning models for enhanced packet-dropping attack detection.

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2445
1	1.00	0.96	0.98	1921
2	0.97	0.99	0.98	2405
accuracy			0.99	6771
macro avg	0.99	0.98	0.99	6771
weighted avg	0.99	0.99	0.99	6771

Confusion Matrix:
[[2432 0 13]
[17 1853 51]
[8 8 2389]]
Accuracy: 0.9857
Precision: 0.9858
Recall: 0.9857
F1-score: 0.9857
Error Rate: 0.0143

A) LSTM after feature selection

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2457
1	0.97	0.98	0.98	1928
2	0.98	0.97	0.97	1354
accuracy			0.98	5739
macro avg	0.98	0.98	0.98	5739
weighted avg	0.98	0.98	0.98	5739

Confusion Matrix:
[[2431 14 12]
[9 1899 20]
[8 35 1311]]
Accuracy: 0.9829
Precision: 0.9830
Recall: 0.9829
F1-score: 0.9829
Error Rate: 0.0171

B) LSTM before feature selection

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2445
1	0.99	0.96	0.98	1921
2	0.97	0.99	0.98	2405
accuracy			0.98	6771
macro avg	0.99	0.98	0.98	6771
weighted avg	0.98	0.98	0.98	6771

Confusion Matrix:
[[2425 2 18]
[17 1850 54]
[3 10 2392]]
Accuracy: 0.9846
Precision: 0.9848
Recall: 0.9846
F1-score: 0.9846
Error Rate: 0.0154

C)GUR after feature selection

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2457
1	0.98	0.98	0.98	1928
2	0.97	0.98	0.97	1354
accuracy			0.98	5739
macro avg	0.98	0.98	0.98	5739
weighted avg	0.98	0.98	0.98	5739

Confusion Matrix:
[[2436 14 7]
[9 1882 37]
[11 16 1327]]
Accuracy: 0.9836
Precision: 0.9837
Recall: 0.9836
F1-score: 0.9836
Error Rate: 0.0164

D) GUR before feature selection

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2445
1	0.99	0.97	0.98	1921
2	0.98	0.99	0.98	2405
accuracy			0.98	6771
macro avg	0.99	0.98	0.98	6771
weighted avg	0.98	0.98	0.98	6771

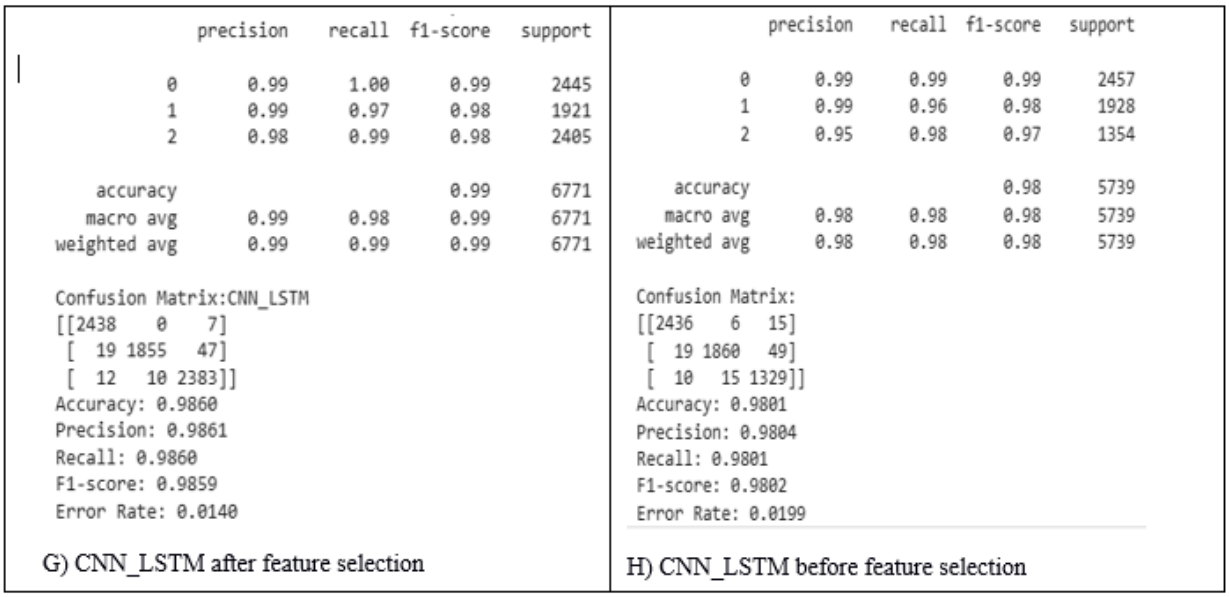
Confusion Matrix CNN:
[[2435 4 6]
[19 1861 41]
[15 18 2372]]
Accuracy: 0.9848
Precision: 0.9848
Recall: 0.9848
F1-score: 0.9848
Error Rate: 0.0152

E) CNN after feature selection

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2457
1	0.98	0.98	0.98	1928
2	0.97	0.97	0.97	1354
accuracy			0.98	5739
macro avg	0.98	0.98	0.98	5739
weighted avg	0.98	0.98	0.98	5739

Confusion Matrix CNN:
[[2439 12 6]
[12 1880 36]
[12 35 1307]]
Accuracy: 0.9803
Precision: 0.9803
Recall: 0.9803
F1-score: 0.9803
Error Rate: 0.0197

F) CNN before feature selection



	precision	recall	f1-score	support
0	0.99	1.00	0.99	2445
1	0.99	0.97	0.98	1921
2	0.98	0.99	0.98	2405
accuracy			0.99	6771
macro avg	0.99	0.98	0.99	6771
weighted avg	0.99	0.99	0.99	6771

Confusion Matrix:CNN_LSTM
[[2438 0 7]
[19 1855 47]
[12 10 2383]]
Accuracy: 0.9860
Precision: 0.9861
Recall: 0.9860
F1-score: 0.9859
Error Rate: 0.0140

G) CNN_LSTM after feature selection

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2457
1	0.99	0.96	0.98	1928
2	0.95	0.98	0.97	1354
accuracy			0.98	5739
macro avg	0.98	0.98	0.98	5739
weighted avg	0.98	0.98	0.98	5739

Confusion Matrix:
[[2436 6 15]
[19 1860 49]
[10 15 1329]]
Accuracy: 0.9801
Precision: 0.9804
Recall: 0.9801
F1-score: 0.9802
Error Rate: 0.0199

H) CNN_LSTM before feature selection

Figure 4. 9 Before and After feature selection results compression

Table 4. 2: Table confusion matrix before feature selection(21 features)

Model	Class	Precision (%)	Recall (%)	F1Score (%)	Accuracy (%)	Error Rate (%)
LSTM	Normal	99.00	99.0	99.00	98.29	1.71
	Black Hole	97.00	98.00	98.00	98.29	1.71
	Gray Hole	98.00	97.00	97.00	98.29	1.71
GUR	Normal	99.00	99.10	99.02	98.36	1.64
	Black Hole	98.00	98.00	98.00	98.36	1.64
	Gray Hole	98.00	97.08	98.00	98.36	1.64
CNN	Normal	99.02	99.00	99.01	98.03	1.97
	Black Hole	98.00	98.00	98.00	98.03	1.97
	Gray Hole	97.00	97.00	97.00	98.03	1.97
LSTM_CNN	Normal	99.00	99.00	99.10	98.01	1.99
	Black Hole	99.00	96.00	98.00	98.01	1.99
	Gray Hole	95.74	98.00	97.00	98.01	1.99

Table 4. 3 Table confusion matrix after feature selection(16 optimal features)

Model	Class	Precision (%)	Recall (%)	F1Score (%)	Accuracy (%)	Error Rate (%)
LSTM	Normal	99.47	99.0	99.23	98.57	1.43
	Black Hole	96.42	99.57	97.96	98.57	1.43
	Gray Hole	99.33	97.4	98.34	98.57	1.43
GUR	Normal	98.86	99.18	99.02	98.46	1.54
	Black Hole	96.30	99.36	97.80	98.46	1.54
	Gray Hole	99.79	97.08	98.41	98.46	1.54
CNN	Normal	99.02	98.62	98.82	98.48	1.52
	Black Hole	96.88	98.83	97.84	98.48	1.52
	Gray Hole	99.21	98.06	98.63	98.48	1.52
LSTM_CNN	Normal	99.00	100	99.10	98.60	1.40
	Black Hole	99.00	98.01	99.00	98.60	1.40
	Gray Hole	99.00	99.00	99.00	98.60	1.40

Overall, all models show high-performance metrics after feature selection, especially in accuracy and precision, suggesting that some valuable characteristics may have been left out during the selection process.

4.3.2 Comparative Model Analysis

The suggested deep learning models' performance was examined, with particular attention paid to important evaluation measures like accuracy, precision, recall, F1-score, and FAR (False Acceptance Rate). When compared to conventional machine learning techniques, the results, which are based on the average values of these parameters, unequivocally show that the deep learning models performed better. Specifically, the analysis revealed that deep learning models achieved higher accuracy, greater precision, improved recall, better F1-scores, and lower FAR, demonstrating their enhanced capability in handling complex data patterns and delivering more reliable outcomes. These findings are visually represented and further elaborated in Figures bellows which provide a detailed comparison of the performance metrics between deep learning

models. This underscores the effectiveness of deep learning techniques in achieving more robust and accurate results across various evaluation criteria.

The precision of the four deep learning models (LSTM, GRU, CNN and CNN_LSTM) on the testing dataset before and after feature selection is shown in the figure below. When using optimal features, we can see that the precision of CNN_LSTM is the highest while that of LSTM is the lowest. When using all available features, GUR gives better precision as compared to the other models. However, the precision of CNN and GUR are lower when using all features.

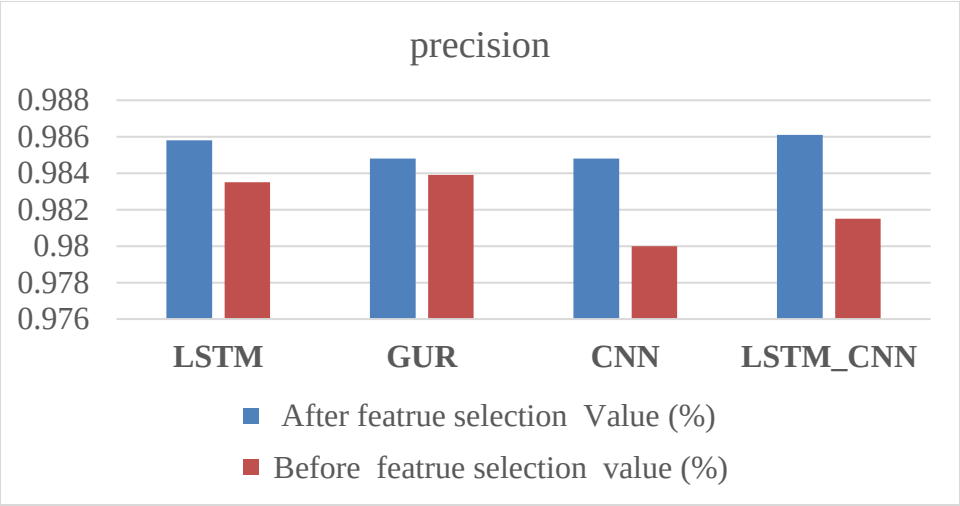


Figure 4. 10. Precision before and after feature selection

The accuracy of the four deep learning models (LSTM, GRU, CNN) on the testing dataset before and after feature selection is shown in the figure below. When using optimal features, we can see that the accuracy of LSTM is the highest, while that of CNN_LSTM is the lowest. When using all available features, GUR gives better accuracy compared to the other models. However, the accuracy of CNN is lower when using all features.

The accuracy of the GUR before feature selection was marginally higher than that of the LSTM and CNN_LSTM models. However, GRU achieves relatively better accuracy compared to the other two classifiers when utilizing all 16 features after the feature selection process. Extensive experimental results involving deep learning algorithms such as CNN_ LSTM, LSTM, GRU, and CNN demonstrate that employing CatBoost for feature selection based on importance scores yields improved outcomes. Among these models, GUR proves to be the most accurate, highlighting the effectiveness of using CatBoost for feature selection in enhancing model performance.

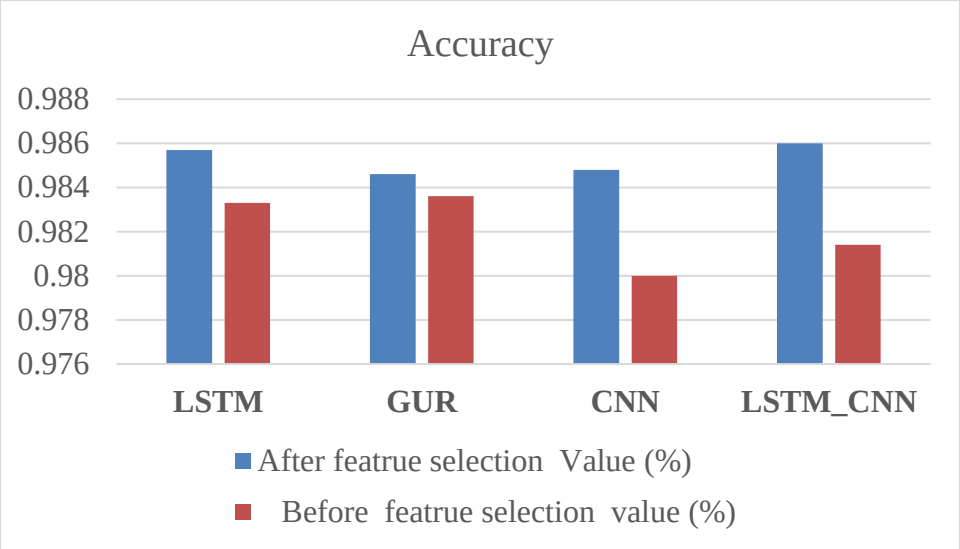


Figure 4. 11.Accuracy before and after feature selection

The recall of the four deep learning models (LSTM, GRU, CNN_LSTM and CNN) on the testing dataset before and after feature selection in the figure below. When using optimal features, we can see that the recall of CNN_LSTM is the highest, while that of CNN is the lowest. When using all available features, GRU gives better recall compared to the other models. However, the recall of CNN is lower when using all features.

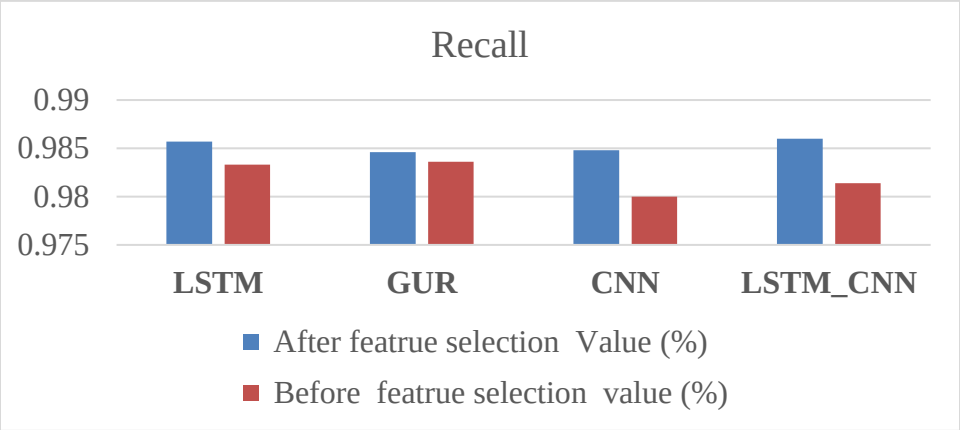


Figure 4. 12. Recall before and after feature selection

The F1-score of the four deep learning models (LSTM, GRU, LSTM_CNN and CNN) in the graph below. After feature selection, all models achieve high F1-scores, with CNN_ LSTM and LSTM

showing similar performance. However, when considering the values before feature selection, the F1-score of CNN is noticeably lower compared to the others. Overall, after optimal feature selection LSTM_CNN, CNN, GUR and LSTM approximately similar results the highest F1-score, indicating its robustness in the given dataset.

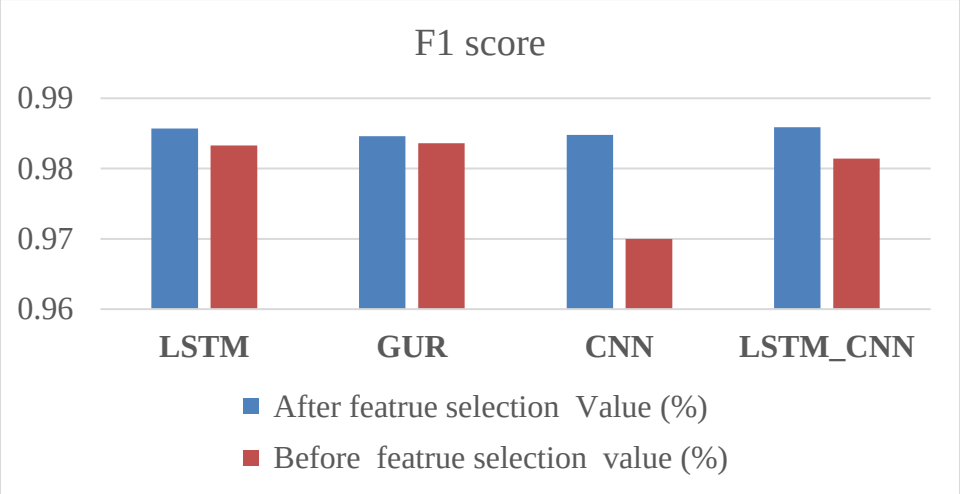


Figure 4. 13. F1 -score before and after feature selection

The graph displays the Error Rate of three models: LSTM, GRU, LSTM_CNN and CNN, before and after feature selection. After feature selection, the CNN_ LSTM model shows the lowest error rate, indicating improved performance and after next LSTM. CNN_LSTM less error rate. In contrast, the CNN and GUR model has the highest error rate after feature selection, suggesting a decrease in accuracy compared to the other models. Overall, using optimal features generally results in lower error rates across all models compared to using all available features.

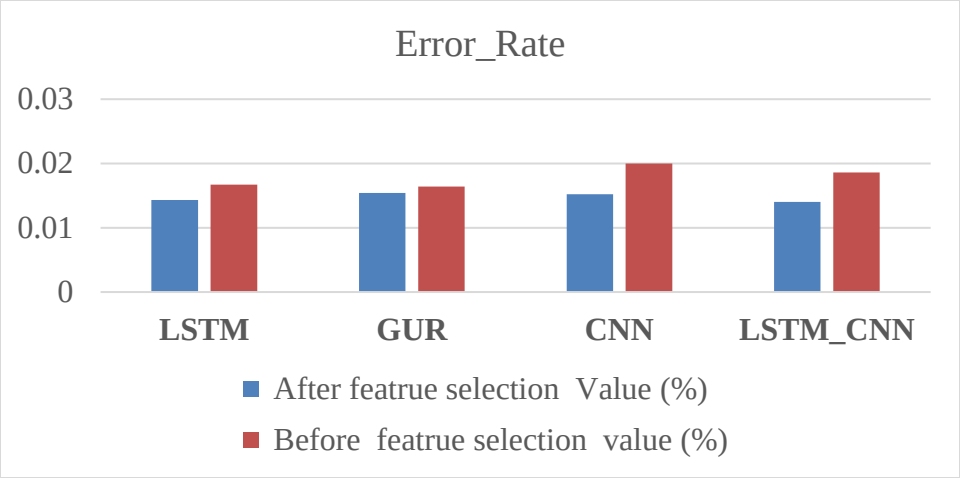


Figure 4. 14. Error rate before and after feature selection

In summary, the comparative analysis from the result where gate After Feature Selection GUR is the best model across all metrics. Before Feature Selection: GUR is the best model for precision, recall, F1 score, and error rate, while after feature selection optimization process get highest result CNN_LSTM has the best FAR accuracy while as CNN highest error rate compare to all models. In finally, CNN_LSTM shows superior performance after feature selection, while GUR was best before feature selection in most metrics.

Table 4. 4 Summary of Performance of Metrics

Metrics	LSTM	GUR	CNN	LSTM_CNN
Accuracy	0.9857	0.9846	0.9848	0.9860
Precision	0.9858	0.9848	0.9848	0.9861
Recall	0.9857	0.9846	0.9848	0.9860
F1-score	0.9857	0.9846	0.9848	0.9859
Error Rate	0.0143	0.0154	0.0152	0.0140

4.4 Result and Discussion

In the previous section chapter 3, a dataset containing information on attacks and normal network behavior, collected from multiple layers using the NS-2 simulator, was analyzed using deep learning algorithms such as CNN, LSTM, and GRU. The dataset was divided into 28,691 samples (80%) for training and 28,691 samples (20%) for testing. Initially, the experiments were conducted using all 21 available features. Categorical features, including attack type and protocol type, were

converted into numerical form using the label encoder method. The data was then reshaped from 2D to 3D to suit the deep learning models. During training, the data was divided into batches of 64 samples each, with 100 iteration steps. The models were structured with 16 optimal units in the input layer activation function use Relu, three nodes in the output layer, and two hidden layers containing 64 and 50 neurons, respectively. A learning rate of 0.001 was used, and kernel dropout regularization (0.2) was applied to the hidden layers to reduce overfitting and enhance generalization. The models were compiled using categorical cross-entropy as the loss function and the Adam optimizer for optimization. The experiments aimed to compare the performance of CNN, LSTM, CNN_LSTM and GRU in detecting attacks and normal network behavior.

The training results showed that the CNN_ LSTM model achieved the highest accuracy of 98.66%, followed by LSTM (98.62%), GUR (98.59%) and CNN (98.34%). In terms of recall, GRU (98.52%), CNN (98.51%), LSTM_CNN (98.60%) and LSTM (98.48%) performed well in detecting normal behavior, blackhole attacks, and gray hole attacks. From the Table 4.1 and 4.2, result shows provide detailed performance metrics, including precision, recall, F1-score, false alarm rate (FAR), accuracy, and error rate. The CNN_LSTM model demonstrated superior performance in learning complex features and preserving long-term historical behaviors, while the LSTM model, with its shallow structure, struggled to capture complex patterns effectively.

Subsequently, feature selection was performed using CatBoost to identify the 16 most important features out of the original 21. The models were retrained using these optimal features, maintaining the same architecture, learning rate, and optimization settings.

The testing accuracies achieved by the models in Table 4.7, were shows 98.57% for LSTM, 98.48% for CNN, 98.60% CNN_LSTM and 98.46%for GRU when using the SoftMax function in the output layer. During training, we observed a gradual increase in validation accuracy, avoiding sharp jumps that could indicate overfitting. Specifically, the validation accuracy increased from approximately 98.33% to 98.62% for LSTM, 98.00 % to 98.34% for CNN, 98.14% to 98.60% for CNN_LSTM and 98.36% to 98.59% for GRU. The CNN_LSTM model also achieved the lowest error rate (1.4%), while CNN highest error rate and CNN had the lowest detection rate (98.34%). Overall CNN_LSTM and LSTM performed best in terms of accuracy, precision, and F1-score, whereas CNN_LSTM and LSTM good in recall.

According to the result shown in Table 4.3, not only CNN_LSTM but also LSTM classifier was achieved a better detection rate of black hole class than the other classes. The detection rate of the classifiers for blackhole was better than gray class. Both CNN_LSTM and LSTM classifier detects gray hole attack with the low detection rate compared to blackhole attack because gray hole attack is harder when it is launched by collaborative attackers. For those model's CNN_LSTM becomes lowest alarm rate of 1.4%, and LSTM the second low false alarm rate 1.43% and in comparison, GUR and CNN high false alarm rate 1.54% and 1.52% respectively.

From the result, we found that the CNN_ LSTM and LSTM is the most efficient algorithm for Manet attack detection. The performance of LSTM and other deep learning model's classifier was improved compared to the related work in [36,39,40.41]. LSTM are better in detection so, similar to this research works but the accuracy of this research is high when we compare to other related works.

Finally, using the AODV routing protocol, this thesis proposes a deep learning-based intrusion detection system for detecting packet dropping attacks in MANETs, particularly black-hole and gray-hole attacks. Given the scarcity of suitable public datasets (such as KDD99 and DARPA98, which lack relevant features), a custom dataset was generated by simulating attack and normal traffic scenarios in NS2 (Network Simulator 2). Network traces were extracted, preprocessed, and used to train and evaluate four deep learning models: CNN, LSTM, a hybrid CNN-LSTM, and GRU. Key Findings

- LSTM achieved the highest performance with a training accuracy of 98.62% (loss: 0.360) and a testing accuracy of 98.57% (loss: 0.0374).
- GRU closely followed, with a training accuracy of 98.59% (loss: 0.0411) and a testing accuracy of 98.46% (loss: 0.0430).
- CNN showed slightly lower performance, attaining a training accuracy of 98.34% (loss: 0.0510) and a testing accuracy of 98.48% (loss: 0.0499).

The hybrid CNN-LSTM model recorded a training accuracy of 98.66% (loss: 0.0382) and a testing accuracy of 98.60% (loss: 0.0401).

The results confirm by using different preprocessing and features selection and extraction techniques get high accuracy that hybrid CNN-LSTM outperformed other models, while LSTM

demonstrated comparable efficiency. The use of NS2-simulated data significantly enhanced detection accuracy, validating the proposed approach as an effective solution for AODV-based attack detection in MANETs.

4.4.1 Research question and answers

Q1. Which attributes are relevant to improve attack detection model?

Based on the research question Q1: Which attributes are relevant to improve attack detection model? the following attributes, categorized by their nature, are identified as relevant for enhancing models designed to detect network intrusions, particularly gray and black hole attacks:

1. **Network Traffic Features:** These attributes capture fundamental communication characteristics within the network. Key relevant features include counts of specific packet types (such as Route Requests - RREQ, Route Replies - RREP, DATA packets, and ACK packets), overall throughput, communication delay, bandwidth consumption, and the total number of Packets Received. These metrics provide a baseline understanding of normal and abnormal traffic flow.
2. **Routing Behavior:** Attributes in this category focus on the operation of the routing protocol itself, crucial for detecting attacks that manipulate routing. Relevant features encompass the frequency of route discovery processes, the volume of Route Requests (RREQ) generated, the volume of Route Replies (RREP) generated, the Normalized Routing Overhead (indicating the efficiency of routing control traffic relative to data traffic), and the Total Routing Control Traffic (representing the absolute volume of control messages like RREQ and RREP). Anomalies in these are strong indicators of malicious routing activity.
3. **Node Behavior:** This category contains attributes directly reflecting the actions and performance of individual nodes, which is paramount for identifying malicious nodes involved in gray or black hole attacks. Critically relevant features include the **Packet Drop Ratio (crucial for detecting gray/black holes)**, the Packet Forwarding Ratio, metrics indicating sent/received packet imbalance, the node participation rate in RREQ/RREP processes, the Packet Dropping Rate (a direct measure of malicious behavior), the count of Packet lost, the Packet Forwarding Rate, and the count of Packets Sent by the node. These metrics directly expose nodes failing to forward data as expected.

In summary**,** the most relevant attributes for improving an attack detection model targeting gray and black hole attacks encompass detailed network traffic statistics (packet types, throughput, delay, received packets), routing protocol dynamics (RREQ, RREP, routing overhead), and, most critically, node-level behavioral metrics, especially the Packet Drop Ratio, Packet Dropping Rate, and Packet Forwarding Rate/ratio**,** which directly quantify the malicious actions characteristic of these attacks.

Q2: How are simulated datasets prepared for attack detection?

Based on research question **Q2: How are simulated datasets prepared for attack detection?** the dataset preparation process involves the following key steps:

1. A **network simulator** (specifically NS-2) is selected to generate the data.
2. **Scenario parameters are defined**, including:
    - **Topology**: Configuring the number of nodes, their placement (using random, grid, or real map models), and mobility patterns (e.g., Random Waypoint, Gauss-Markov).
    - **Traffic**: Setting application types (CBR, TCP, UDP), traffic patterns, packet rates, and source-destination pairs.
    - **Protocol**: Implementing the AODV routing protocol, which is commonly targeted for gray/black hole attacks.

3. **Attack simulation** is performed by designating specific nodes as malicious:
    - **Gray Hole** attackers selectively drop DATA packets
    - **Black Hole** attackers drop all DATA packets and may actively attract traffic.

4. **Feature extraction** is conducted using AWK scripts to process raw trace files and logs, extracting the attributes identified in Q1. Each data point is explicitly labeled as **Normal**, **GrayHole**, or **BlackHole** for supervised learning.

Q3. How to get optimal simulation parameter values for data generated from the simulated network?

For Maximize detection accuracy and Minimize false positives and Identify Key Parameters such as Number of nodes, % malicious nodes, mobility speed, traffic load, attack intensity (drop rate), attack duration/frequency 21 features are collected for our datasets. Ensure optimal parameters yield realistic network behavior and generalize to slightly different scenarios.

Q4. How to designing deep learning model and techniques for most effective intrusion detecting gray and black hole attacks?

Data Representation- Sequence data (time-series of features per node/flow) or tabular data (feature vectors per sample).

Final after selected optima 16 features Based on the comparative performance results observed in the study, the effectiveness of the deep learning model architectures for detecting gray and black hole attacks can be described as follows, presented in order of descending accuracy.

- **CNN-LSTM Hybrid (Highest Accuracy)** Combines CNN's spatial/temporal feature extraction with LSTM's sequential dependency modeling, yielding superior accuracy through synergistic analysis of dynamic attack patterns.
- **LSTM (Second Highest)** Effectively captures long-term temporal dependencies in network traffic but lacks CNN's localized feature extraction, limiting its performance versus the hybrid.
- **CNN (Third)** Excels at identifying local patterns within fixed data windows but underperforms on time-evolving attacks due to weaker sequential modeling.
- **GRU (Lowest)** Though efficient for sequence data, its simplified gating mechanism struggles to model complex temporal attack signatures compared to LSTM-based approaches.

By using the following techniques to achieve high detection accuracy for models

- **Feature Engineering/Selection** using (Cat Boost) features selection get optimal 16 features from 21.
- **Class Imbalance Handling** Oversampling (SMOTE), under sampling, cost-sensitive learning, focal loss.
- **Regularization:** Dropout, regularization to prevent overfitting.

Q5. What are the performance metrics to evaluate the proposed models?

Based on research question **Q5: What are the performance metrics to evaluate the proposed models?** the following metrics are essential for assessing the effectiveness of intrusion detection models targeting gray and black hole attacks.

1. **Standard Classification Metrics:** Accuracy, Precision, Recall and F1-Score
2. **Confusion Matrix:** This visualization tool is critical for detailed per-class performance analysis, explicitly showing how predictions for each category—**Normal**, **Gray Hole**, and **Black Hole**—compare against the true labels. It reveals specific mis patterns (e.g., false positives/negatives) and helps diagnose model strengths/weaknesses across distinct attack types.

These metrics collectively validate model reliability, efficiency, and practical utility in real-world attack detection scenario.

CHAPTER FIVE

5. Conclusion and Recommendation

5.1. Conclusions

In this study, the MANET attack detection model using a deep learning algorithm in MANET under AODV routing has been developed. In this research, anomaly detection system using deep learning algorithms in MANET has been proposed. The proposed system is used to detect the two packet-dropping attacks (black-hole and gray-hole attacks) as well as normal traffic. In the MANET security area, the lack of an appropriate public dataset is one of the biggest barriers, and also the public dataset like KDD99 and DARPA98 is not preferable due to unsuitable features. Blackhole attack and gray hole attack along with the normal traffic were identified as classes of a dataset. AODV routing protocol has been implemented during simulation using network simulator (NS2). Simulations with the presence of attack and without attack under different scenarios were conducted.

MANET attacks features were extracted from trace file and the dataset was prepared. The experiments aimed to compare the performance of CNN, LSTM, CNN_LSTM and GRU were trained and tested. As it is described in Table 4.1 and 4.2, after and before important n features, used for training and testing, were successfully identified. As a result, attack detection models for

CNN, LSTM, hybrid CNN_LSTM and GRU. The performance metrics for each model are as follows: the LSTM achieved a training accuracy of 98.62% with a training loss of 0.360, while its testing accuracy was 98.57% and testing loss was 0.0374 The GUR demonstrated a training accuracy of 98.59% and a training loss of 0.0411, with a testing accuracy of 98.46% and testing loss of 0.0430. The CNN recorded a training accuracy of 98.34% and a training loss of 0.0510, while its testing accuracy and loss were 98.48% and 0.0499, and LSTM_CNN were training accuracy 98.60% and training loss 0.0382 while tasting accuracy 98.60%. and tasting loss 0.0401 respectively. From the result showed in Table 4.7, CNN_LSTM classifier was achieved the best result while GUR was less performed. However, the performance metrics of CNN_LSTM and LSTM were approximately similar the former performs better than the later. For those model's CNN_ LSTM becomes lowest alarm rate of 1.4%, and LSTM the second low false alarm rate 1.43% and in comparison, CNN and GUR high false alarm rate 1.52% and 1.54% respectively. It was found that the data collection from simulation using NS2 has played a great role to improve the detection model. Therefore, the proposed study was an efficient method to detect the aodv routing attack in MANET.

5.2. Recommendation and Future works

This study demonstrates that the proposed approach effectively detects a specific set of attacks; however, its performance could be further enhanced by incorporating additional threats, whether executed collaboratively or sequentially. To improve robustness, network-layer attacks such as Sybil and rushing attacks should be integrated into the dataset. Moreover, exploring alternative deep learning algorithms could optimize detection accuracy. Future research should extend this framework beyond MANET to other wireless networks, including WSNs and VANETs, while also examining attacks across different protocol layers.

Currently, the work focuses on identifying black-hole, gray-hole, and normal behaviors, but it could be expanded to investigate impersonation, denial-of-service attacks, and unintentional packet drops caused by factors like low battery, channel interference, mobility-induced link errors, or buffer congestion. Additionally, evaluating the approach with alternative routing protocols such as DSR, DSDV, and OLSR would provide a more comprehensive assessment of its applicability.

6. Reference

[1] S. Shafi, S. Mounika, and S. Velliangiri, “Machine Learning and Trust Based AODV Routing Protocol to Mitigate Flooding and Blackhole Attacks in MANET,” *Procedia Comput. Sci.*, vol. 218, no. 2022, pp. 2309–2318, 2022, doi: 10.1016/j.procs.2023.01.206.

[2] J. Jayashri M. Boopathi, “Energy efficiency for mobile using cloud and stochastic wireless channel with deadline," *Global Journal of Engineering Science and Research Management*, vol. 1, no. 7, pp. 9–13, 2019.

[3] S. Younas, F. Rehman, T. Maqsood, S. Mustafa, A. Akhunzada, and A. Gani, “Collaborative Detection of Black Hole and Gray Hole Attacks for Secure Data Communication in VANETs,” *Appl. Sci.*, vol. 12, no. 23, 2022, doi:10.3390/app122312448.

[4] T. Harshini and S. Sinha, “"Recognition and Avoidance of Blackhole and Grayhole Attacks in MANET",” *International Journal of Management (IJM*), vol. 11, no. 8, pp. 2204–2215, Aug,2020., doi: 10.34218/IJM.11.8.2020.191.

[5] A. Abdelhamid, M. S. Elsayed, H. K. Aslan, and M. A. Azer, “Anomaly-Based Intrusion Detection for Blackhole Attack Mitigation," in Proc. 19th Int. Conf. Avail., Rel., Secur. (ARES), New York, NY, USA, 2024, Art. no. 124, pp. 1–9.doi: 10.1145/3664476.3670941.

[6] F. C. Korir and W. Cheruiyot, "A survey on security challenges in the current MANET routing protocols," *Global Journal of Engineering and Technology Advances*, vol. 12, no. 1, pp. 78–91, Jul. 2022, doi: 10.30574/gjeta.2022.12.1.0114.

[7] S. Shrestha, R. Baidya, B. Giri, and A. Thapa, “Securing Blackhole Attacks in MANETs using Modified Sequence Number in AODV Routing Protocol,” *2020 8th Int. Electr. Eng. Congr. iEECON 2020*, pp. 0–3, 2020, doi: 10.1109/iEECON48109.2020.229555.

[8] U. D. Kolekar, V. S. Jadhav, and P. V Sapkale, “a novel nature inspired secure and trustworthy routing protocol for,” *Indian J. STEAM*, vol. 1, no. 2, Aug. 2021.

[9] M. S. Abood, H. F. Mahdi, M. M. Hamdi, O. J. Ibrahim, R. Q. Mohammed, and S. F. Ahmed, “Black/Gray Holes Detection Tools in MANET: Comparison and analysis,” *7th IEEE Int. Conf. Eng. Technol. Appl. Sci. ICETAS 2020*, no. July, 2020, doi:

10.1109/ICETAS51660.2020.9484203.

[10] Y. H. Robinson and E. G. Julie, “MTPKM : Multipart Trust Based Public Key Management Technique to Reduce Security Vulnerability in Mobile Ad - Hoc Networks,” *Wirel. Pers. Commun.*, no. 0123456789, 2019, doi: 10.1007/s11277-019-06588-4.

[11] R. K. Sidhu and R. Krishan, “Security Threat of MANET : A Comprehensive Architecture Security Threat of MANET : A Comprehensive Architecture,” vol. 3075, no. 5, pp. 14–21, 2020, doi: 10.35940/ijitee.E1961.039520.

[12] S. Lanka and I. Ikechukwu, “An Effective Performance of Dynamic and Ad-Hoc On-Demand Routing Protocols for Mobile Ad-Hoc Networks Under NS Environment,” *nt. J. Netw. Virtual Organ.*, vol. 22, no. 4, pp. 325–333, Apr. 2020: https://doi.org/10.1504/IJNVO.2020.107558

[13] S. E. Bouzid, Y. Serrestou, K. Raoof, and M. N. Omri, “Efficient Routing Protocol for Wireless Sensor Networks Based on Reinforcement Learning,” *in Proc. 5th Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, Sousse, Tunisia, Sep. 2020, pp. 1–5, doi: 10.1109/ATSIP49331.2020.9231883.

[14] V. Subbiah, “A novel approach for detecting flooding attacks in MANET using machine learning methods,” *Linguistica Antverpiensia*, vol. 2021, no. 3, pp. 620–633, 2021..

[15] A. Alshammari et al, “ Real-Time Simulation of Black Hole Attacks on AODV in Urban VANETs” Proc. IEEE Int. Conf. Inf. Secur., 2020, pp. 71–80. DOI: 10.4236/jis.2020.111004.

[16] M. Mishra, P. K. Dash, L. Hota, and M. Panda, “Analyze the network layer protocols on the basis of mobility, pause time and simulation time in MANET,” *IEEE Int. Conf. Power, Control. Signals Instrum. Eng. ICPCSI 2017*, no. September 2022, pp. 530–538, 2022, doi: 10.1109/ICPCSI.2017.8392350.

[17] S. Gurung and S. Chauhan, “A dynamic threshold based approach for mitigating black-hole attack in MANET,” *Wirel. Networks*, vol. 24, no. 8, pp. 2957–2971, 2018, doi: 10.1007/s11276-017-1514-1.

[18] Moumen et al., “AODV-based Defense Mechanism for Mitigating Blackhole Attacks in

MANET," in E3S Web of Conferences, vol. 412, p. 01094, 2023, doi: 10.1051/e3sconf/202341201094.

[19] S. Gurung and S. Chauhan, "A dynamic threshold based algorithm for improving security and performance of AODV under black-hole attack in MANET," *Wirel. Networks*, vol. 25, no. 4, pp. 1685–1695, 2019, doi: 10.1007/s11276-017-1622-y.

[20] Z. A. Zardari *et al.*, "A dual attack detection technique to identify black and gray hole attacks using an intrusion detection system and a connected dominating set in MANETs," *Futur. Internet*, vol. 11, no. 3, 2019, doi: 10.3390/fi11030061.

[21] S. Mohapatra and P. Kanungo, "Comparative performance analysis of MANET routing protocols using NS2 simulator," *Commun. Comput. Inf. Sci.*, vol. 250 CCIS, pp. 731–736, 2011, doi: 10.1007/978-3-642-25734-6_127.

[22] M. Sharma, M. Singh, K. Walia, and K. Kaur, "A Comprehensive Study of Performance Parameters for MANET, VANET and FANET," *2019 IEEE 10th Annu. Inf. Technol. Electron. Mob. Commun. Conf. IEMCON 2019*, pp. 643–646, 2019, doi: 10.1109/IEMCON.2019.8936159.

[23] N. Kanimozhi, S. Hari, and B. Karthikeyan, "Performance Analysis of MANET Routing Protocols," *Int. J. Comput. Appl.*, vol. 185, no. 50, pp. 44–50, 2023, doi: 10.5120/ijca2023923329.

[24] M. Ichaba, "Security Threats and Solutions in Mobile Ad Hoc Networks; A Review," *Univers. J. Commun. Netw.*, vol. 6, no. 2, pp. 7–17, 2018, doi: 10.13189/ujcn.2018.060201.

[25] S. Mamatha and A. Damodaram, "Behavioral intrusion detection in mobile ad hoc networks," *International Journal of Mobile Network Communications & Telematics*, vol. 3, pp. 31–40, 2013, doi: 10.5121/ijmnct.2013.3103.

[26] Ü. Çavuşoğlu, "A new hybrid approach for intrusion detection using machine learning methods," *Appl. Intell.*, vol. 49, no. 7, pp. 2735–2761, 2019, doi: 10.1007/s10489-018-01408-x.

[27] S. Naseer *et al.*, "Deep Neural Networks," *IEEE Access*, vol. PP, no. 8, p. 1, 2018, doi: 10.1109/ACCESS.2018.2863036.

[28] O. Sharma, “A New Activation Function for Deep Neural Network,” Conference: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). July, 2023, doi: 10.1109/COMITCon.2019.8862253.

[29] A. Aldweesh, A. Derhab, and A. Z. Emam, “Intrusion detection system for detecting distributed denial of service attacks using machine learning algorithms,” *Knowledge-Based Syst.*, p. 105124, 2019, doi: 10.1016/j.knosys.2019.105124.

[30] K. A. Alafandy, H. Omara, and M. Al Achhab," *Deep Learning*,"in Approaches and Applications of Deep Learning in Virtual Medical Care, S. Dash, B. R. Acharya, and J. J. P. C. Rodrigues, Eds. Hershey, PA, USA: IGI Global, 2022, ch. 6, pp. 127–167. doi: 10.4018/978-1-7998-8929-8.ch006. .

[31] E. G. Mwangi, G. M. Muketha, and G. Ndung, “Trust-Based Security Technique to Curb Cooperative Blackhole Attacks in Mobile Ad Hoc Networks using OTB-DSR Protocol in NS-3 ,” no. April, 2021.

[32] M. S. Shaik and F. Mira, “A Comprehensive Mechanism of MANET Network Layer Based Security Attack Prevention ,” *Int. J. Wirel. Microw. Technol., vol. 1, pp. 38–47, Feb. 2020. doi:* 10.5815/ijwmt.2020.01.04.

[33] S. Sivanesh and V. R. S. Dhulipala, “Accurate and Cognitive Intrusion Detection System (ACIDS): a Novel Black Hole Detection Mechanism in Mobile Ad Hoc Networks,” *Mob. Networks Appl.*, vol. 26, no. 4, pp. 1696–1704, 2021, doi: 10.1007/s11036-019-01505-2.

[34] Y. Zhang, Q. Yang, S. Lambotharan, K. Kyriakopoulos, I. Ghafir, and B. AsSadhan, “Anomaly-Based Network Intrusion Detection Using SVM," in Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP), Xi’an, China, Oct. 2019, pp. 1–6. doi: 10.1109/WCSP.2019.8927907.

[35] S. M. Kasongo, “A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks,” *A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework," Comput. Commun., vol. 199, pp. 113–125, 2023. doi: 10.1016/j.comcom.2022.12.010.*

[36] F. Feng, X. Liu, B. Yong, R. Zhou, and Q. Zhou, “Anomaly detection in ad-hoc networks

based on deep learning model: A plug and play device," *Ad Hoc Networks*, vol. 84, no. August 2022, pp. 82–89, 2019, doi: 10.1016/j.adhoc.2018.09.014.

[37] T. M. Nguyen, H. H. P. Vo, and M. Yoo, "Enhancing Intrusion Detection in Wireless Sensor Networks Using a GSWO-CatBoost Approach," *Sensors*, vol. 24, no. 11, pp. 1–26, 2024, doi: 10.3390/s24113339.

[38] N. Zagorodna, M. Stadnyk, B. Lypa, M. Gavrylov, and R. Kozak, "Network Attack Detection Using Machine Learning Methods," in *Challenges to National Defence in Contemporary Geopolitical Situation*, 2022, pp. 55–61. doi: 10.47459/cndcgs.2022.7.

[39] M. K. Tejaswini and Y. Adilakshmi, "Black hole Attack Detection Using Machine Learning Algorithms in MANET – Performance Comparision," *Int. Res. J. Eng. Technol.*, no. June, pp. 6047–6051, 2020.

[40] S. Venkatasubramanian, A. Suhasini, and S. Hariprasath, "Detection of Black and Grey Hole Attacks Using Hybrid Cat with PSO-Based Deep Learning Algorithm in MANET," *Int. J. Comput. Networks Appl.*, vol. 9, no. 6, pp. 724–735, 2022, doi: 10.22247/ijcna/2022/217705.

[41] P. Nigam, "Utilizing Machine Learning to Mitigate Attack Risks in MANET," *African J. Biol. Sci.*, vol. 6, no. 14, pp. 1116–1122, 2024, doi: 10.48047/afjbs.6.14.2024.1116-1122.

[42] S. Boschi, M. Di Ianni, P. Crescenzi, G. Rossi, and P. Vocca, "MOMOSE a mobility model simulation environment for mobile wireless ad-hoc networks," *SIMUTools 2008 - 1st Int. ICST Conf. Simul. Tools Tech. Commun. Networks Syst.*, 2008, doi: 10.4108/ICST.SIMUTOOLS2008.3036.

[43] C. Science and M. Studies, "Performance Analysis and Simulation of Reactive Routing Protocols (AODV , DSR and TORA) in MANET using NS-2 ," I*nternational Journal of Advance Research* in pp. 256–264, 2014.

[44] S. Kaur and M. Singh, "Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 7859–7877, 2020, doi: 10.1007/s00521-019-04187-9.

[45] S. S. Yadav* and G. P. Bhole, "Learning from Imbalanced Data in Classification," *Int. J.*

Recent Technol. Eng., vol. 8, no. 5, pp. 1907–1016, 2020, doi: 10.35940/ijrte.e6286.018520.

[46] R. Alshamy, M. Ghurab, S. Othman, and F. Alshami, "Intrusion Detection Model for Imbalanced Dataset Using SMOTE and Random Forest Algorithm," *Commun. Comput. Inf. Sci.*, vol. 1487 CCIS, no. January 2022, pp. 361–378, 2021, doi: 10.1007/978-981-16-8059-5_22.

[47] R. Efendi, T. Wahyono, and I. Ratna Widiasari, "LSTM SMOTE: An Effective Strategies for DDoS Detection in Imbalanced Network Environments," 2024, doi: 10.20944/preprints202407.1825.v1.

[48] R. Makani and B. V. R. Reddy, "Taxonomy of Machine Leaning Based Anomaly Detection and its suitability," *Procedia Comput. Sci.*, vol. 132, pp. 1842–1849, 2018, doi: 10.1016/j.procs.2018.05.133.

[49] K. A. Binsaeed and A. M. Hafez, "Enhancing Intrusion Detection Systems with XGBoost Feature Selection and Deep Learning Approaches," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 5, pp. 1084–1098, 2023, doi: 10.14569/IJACSA.2023.01405112.

[50] P. Devan and N. Khare, "An efficient XGBoost–DNN-based classification model for network intrusion detection system," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12499–12514, 2020, doi: 10.1007/s00521-020-04708-x.

[51] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. Section 4, pp. 6638–6648, 2018.

[52] Y. Chen, Y. Zhang, and F. Yang, "Different Approaches Towards Vertical Track Irregularity Prediction A Comparative Study," 2020, [Online]. Available: http://arxiv.org/abs/2012.03062

[53] S. M. Kasongo and Y. Sun, "A Deep Gated Recurrent Unit based model for wireless intrusion detection system," *ICT Express*, vol. 7, no. 1, pp. 81–87, 2021, doi: 10.1016/j.icte.2020.03.002.

[54] C. Yu, X. Qi, H. Ma, X. He, C. Wang, and Y. Zhao, "LLR: Learning learning rates by LSTM for training neural networks," *Neurocomputing*, vol. 394, 2020, doi: 10.1016/j.neucom.2020.01.106.

APPENDIX A TCL script

#===

```
#Set all the options

set val(chan)           Channel/WirelessChannel  ;# channel type
set val(prop)           Propagation/TwoRayGround ;# radio-propagation model
set val(ant)            Antenna/OmniAntenna      ;# Antenna type
set val(ll)             LL                       ;# Link layer type
set val(ifq)            Queue/DropTail/PriQueue  ;# Interface queue type
set val(ifqlen)         150                      ;# max packet in ifq
set val(netif)          Phy/WirelessPhy          ;# network interface type
set val(mac)            Mac/802_11               ;# MAC type
set val(rp)             AODV                     ;# routing protocol
set val(x)              1440                     ;# X length
set val(y)              1000                     ;# Y length
set val(finish)         50        ;# Finish time
set val(nn)             40                       ;# number of mobilenodes
set val(t1)     0.0                          ;
set val(t2)     0.0                           ;

set mobility [lindex $argv 0]                   ;# Dynamic mobility file
set ns_ [new Simulator]

set f [open out_[regexp -all -inline -- {[0-9]+} $mobility].tr w]
$ns_ trace-all $f
set namtrace [open out_[regexp -all -inline -- {[0-9]+} $mobility].nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
set chan_1 [new $val(chan)]

# CONFIGURE AND CREATE NODES

$ns_ node-config  -adhocRouting $val(rp) \
           -llType $val(ll) \
                   -macType $val(mac) \
                   -ifqType $val(ifq) \
                -energyModel "EnergyModel" \
                -initialEnergy 100.0 \
                -txPower 0.9 \
                -rxPower 0.5 \
                -idlePower 0.45 \
                -sleepPower 0.05 \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
                 -channel $chan_1

for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns_ node]
 $ns_ initial_node_pos $node_($i) 40
    }


proc finish {} {
    global ns_ namtrace filename
    $ns_ flush-trace
    close $namtrace
   # exec nam out.nam &
    exit 0
}

source $mobility
source cb.tcl
#attack balck and gray hole
 #$ns_ at 1.0 "[$node_(2) set ragent_] black hole"
 #$ns_ at 30.0 "[$node_(3) set ragent_] black hole"
 #$ns_ at 25.0 "[$node_(5) set ragent_] balck hole "
 #$ns_ at 1.0 "[$node_(35) set ragent_] balck hole"

  #$ns at 1 "$ns trace-annotate \"$n21 transfers data to attacker $n1 which does not have shorest route to $n17\""
#$ns at 1 "$ns trace-annotate \"Attacker $n1 selectively forwards data to Destination $n17 and drops the remaining data creating gray hole attack\""
#$ns at 1 "$ns trace-annotate \"$n21 transfers data to attacker $n3 which does not have shorest route to $n17\""
#$ns at 1 "$ns trace-annotate \"Attacker $n3 selectively forwards data to Destination $n17 and drops the remaining data creating gray hole attack\""
#=====================================================================

$ns_ at $val(finish) "finish"
puts "Start of simulation..."
$ns_ run
```

#==========

APPENDIX B Analysis script
AWK

#==========

Open

mesfin.awk
~/mk/Mesfin simulation

```
1 ######################################
2
3 ##### Created by mesfin du  #####
4
5 ######################################
6
7
8
9 BEGIN {
10
11        send = 0;              #variable for storing number of packets sent
12
13        recv = 0;              #variable for storing number of packets received
14
15        bytes = 0;             #variable for storing number of bytes transmitted
16
17        st = 0;                #variable for start time
18
19        ft = 0;                #variable for end time
20
21        rtr = 0;               #variable for number of routing packets
22
23        delay = 0;
24
25 #========== for new=========================
26
27          #drop=0;
28
29        sents=0;
30
31        recived=0;
32
33        forward=0;
34
35        cbr=0;
36
37        agt=0;
38
39        Enrgy=0;
40
41         estimationE=0;
42      eEco=0;
43    remaining_energy=0;
44    intiail_energy=0;
```

```
58
59 #=========================================
60
61   #### new #####
62
63
64
65    $0~/^s.*(REQUEST)/{
66
67       rreq++;
68
69     }
70
71    $0~/^s.*(REPLY)/{
72
73     rrep++;
74
75  }
76
77 {#------>
78
79 #============================new=======================
80
81 if($1=="s")
82
83         {
84
85                 sents=sents+1;
86
87         }
88
89
90
91         if($1=="f")
92
93         {
94
95                 forward=forward+1;
96
97         }
98
99         if($1=="r")
100
101         {
```

```
91         if($1=="f")
92
93         {
94
95                 forward=forward+1;
96
97         }
98
99         if($1=="r")
100
101        {
102
103                recived=recived+1;
104
105        }
106
107
108
109        if($7=="cbr")
110
111        {
112
113                cbr=cbr+1;
114
115        }
116
117        if($4=="AGT")
118
119        {
120
121                agt=agt+1;
122
123        }
124
125       if($14>0)
126
127        {
128
129                Enrgy =+$14;
130             eEco =+$20;
131           estimationE =+$18;
132       remaining_energy =+$22;
133      intiail_energy =+$16;
134
```

```
148
149         #Final value of rtr will be the total number of routing packets sent.
150
151
152
153         if (( $1 == "s" || $1 == "f" )  && $4 == "RTR" && $7 == "AODV")
154
155         {
156
157                 rtr++;
158
159         }
160
161
162
163         #If event is "sent" and trace level is AGT(transport layer packet) then increment send by 1
164
165         #Final value of send will be the total number of data packets sent.
166
167
168
169         if ( $1 == "s" && $4 == "AGT" &&( $7 == "cbr"|| $7 =="tcp"))
170
171         {
172
173                 if(send == 0)
174
175                 {
176
177                         st = $2;        #Starting time of packet transmission will be assigned to st
178
179                 }
180
181
182
183                 ft = $2;                #End time of packet transmission will be assigned to ft
184
185                 st_time[$6] = $2;       #This array holds sending time for each packet and $6 is unique ID such as 0,1,and so on.
186
187                 send++;
188
189         }
190
```

```
196
197
198
199         if ( $1 == "r" && $4 == "AGT" &&( $7 == "cbr"|| $7 =="tcp"))
200
201         {
202
203                 recv++;
204
205                 bytes+=$8;      #$8 is packet size and final bytes value is the total number of bytes tranmitted from source to destination.
206
207                 ft_time[$6] = $2;  #This array holds receiving time for each packet and $6 is unique ID such as 0,1,2,3 and so on.
208
209                 delay += ft_time[$6]-st_time[$6]        #Final value of delay is the average delay.
210
211         }
212
213 }
214
215
216
217 #printing results
218
219 #Packet delivery ratio is the ratio of total number of data packets received at destination to the total number of data packets sent from source
220
221 #Control overhead means the number of routing packets which are required in the network for data transmission.
222
223 #Normalized routing overhead is the ratio of number of routing packets to the total number of data packets received.
224
225 #Throughput means the average number of bits transmitted per unit time.
226
227 #Delay is the time taken by packet to reach to destination from source
228
229 #Packet dropping ratio is the ratio of total number of data packets dropped to the total number of data packets sent from source
230
231
232
233 END {
234
235 if(recv == 0)
236
237         recv=1; #Handled issue when recv value is 0, which causes other values to raise a divide by zero error.
238
239 printf("Packets_Sent: \t%.2f\n",send); #correct
```

```
234
235 if(recv == 0)
236
237         recv=1; #Handled issue when recv value is 0, which causes other values to raise a divide by zero error.
238
239 printf("Packets_Sent: \t%.2f\n",send); #correct
240
241 printf("Packets_Received: \t%.2f\n",recv); #correct
242
243 printf("Packet_lost: \t%.2f %%\n",(send-recv));# recently added
244
245 printf("PDR: \t%.2f %%\n",recv/send*100);#correct
246
247 printf("Control_Overhead: \t%.2f\n",rtr);
248
249 printf("Norm_Routing_Load: \t%.2f %%\n",rtr/recv*100);
250
251 printf("Delay1:\t%.2f Seconds\n",delay);# recently added
252
253 printf("Delay: \t%.2f Seconds\n",delay/recv);
254
255 printf("Throughput: \t%.2f Kbps\n",bytes*8/(ft-st)/1000);
256
257 printf("P_Dropping_R: \t%.2f %%\n",(send-recv)/send*100);
258
259 printf("Enrgy: \t%.2f \n",(100 - remaining_energy));
260
261 #printf("energy_con: \t%.2f \n",( Enrgy + intiail_energy + estimationE + eEco)(-remaining_energy));
262
263 #new==============================
264
265 printf("Sents:\t%.2f\n", sents);
266
267 printf("Forward:\t%.2f\n", forward);
268
269 printf("Recived:\t%.2f\n", recived);
270
271 printf("CBR:\t%.2f\n", cbr);
272
273 printf(" AGT:\t%.2f\n", agt);
274
275 printf("Remaining_Energy:\t%.2f\n", remaining_energy);
276
277 printf("RREQ:\t%.2f\n",rreq);
```

#==========

APPENDIX C shell scrip

#==========

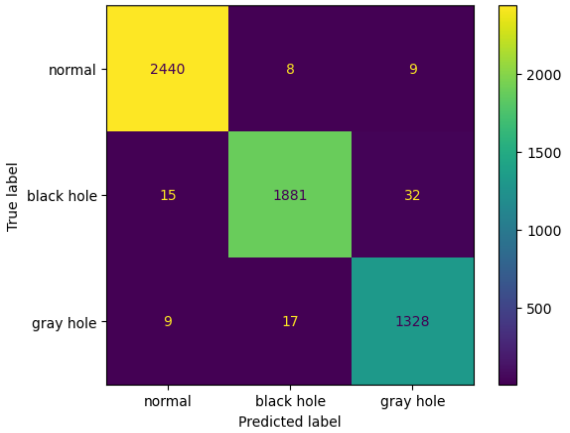
```
 1 #!/bin/bash
 2 #Create required directories and set permissions
 3 mkdir mobility_files
 4 mkdir -p Output/nam
 5 mkdir -p Output/trace
 6 mkdir -p Output/Results
 7 chmod 777 -R mobility_files
 8
 9 #Create required variables to store the parameter values.
10 send=0
11 recv=0
12 pdr=0
13 co=0
14 nro=0
15 dl=0
16 tp=0
17 pdrop=0
18 #Recently added
19 energy=0
20 delay_1=0
21 pkt_lost=0
22 jt=0
23 avg_gt=0
24 #==========================
25          #drop=0
26        sents=0
27        recived=0
28        forward=0
29        cbr=0
30        agt=0
31        enrgy=0
32  energy_con=0
33 remaining_energy=0
34  estimationE=0
35      eEco=0
36    intiail_energy=0
37     energy_con=0
38 rreq=0
39 rrep=0
40 #==================================
41
42 #Create header of CSV file.
43 echo ""| awk 'BEGIN{printf
   "SN,Packets_Sent,Packets_Received,Packet_lost,PDR,Control_Overhead,Norm_Routing_Load,Delay1,Delay,Throughput,P_Dropping_R,Enrgy,Sents,Forward,Recived,CBR,AGT,
```

```
#Create header of CSV file.
echo ""| awk 'BEGIN{printf
  "SN,Packets_Sent,Packets_Received,Packet_lost,PDR,Control_Overhead,Norm_Routing_Load,Delay1,Delay,Throughput,P_Dropping_R,Enrgy,Sents,Forward,Recived,CBR,AGT,
  Remaining_Energy,RREQ,RREP"}'>Output/Results/Final_Result.csv


#Main loop which runs 20 times.
 for (( i=1; i<=20; i++))
       do
               #Creates mobility files. Each of which is named as mob1, mob2, ... ,
               ./setdest -v 2 -n 40 -m 1 -M 18 -t 50 -p 10 -x 1440 -y 1000 > mobility_files/mob$i

               #Call TCL script with mobility file as a parameter.
               ns mkbh.tcl mobility_files/mob$i

               #Call analysis file and redirect the parameter values to file Result.txt
               printf "########## Simulation number $i ##########\n\n" >> Result.txt
               awk -f mesfin.awk out_$i.tr >> Result.txt
               printf "\n\n" >> Result.txt

               #Call analysis file and redirect the parameter values to file tmp_result.
               awk -f mesfin.awk out_$i.tr > tmp_result

               #Extract the parameter value from the file tmp_result and redirect it to the temporary file 1.txt.
               grep "Packets_Sent:" tmp_result |  awk 'BEGIN{}{print $2;}' > 1.txt
               send=$(<1.txt)

               #Add all the values to average variable.
               #avg_send=$( bc <<< "$send + $avg_send ")

               grep "Packets_Received:" tmp_result |  awk 'BEGIN{}{print $2;}' > 1.txt
               recv=$(<1.txt)

                #recently added
                grep "Packet_lost:" tmp_result |  awk 'BEGIN{}{print $2;}' > 1.txt
                pkt_lost=$(<1.txt)
               #====================================

               grep "PDR:" tmp_result |  awk 'BEGIN{}{print $2;}' > 1.txt
               pdr=$(<1.txt)

               grep "Control_Overhead:" tmp_result |  awk 'BEGIN{}{print $2;}' > 1.txt
               co=$(<1.txt)
```

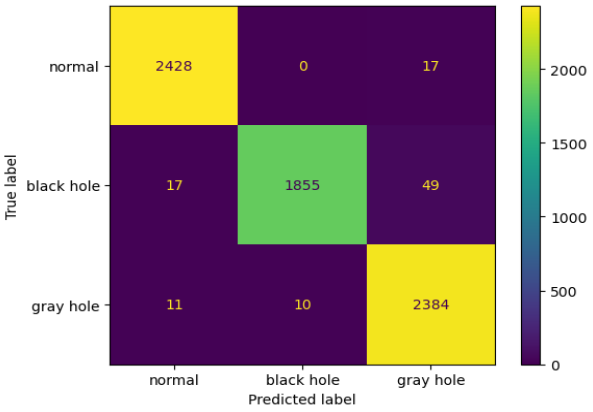
```
               #=======================================================


               #Print all the values to CSV file.
               echo  "$i" | awk '{printf "\n%.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$send" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$recv" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
                   echo  "$pkt_lost" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$pdr" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$co" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$nro" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
                   echo  "$delay_1" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$dl" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$tp" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$pdrop" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
                   echo  "$enrgy" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
   #=============================================================================================
                   echo  "$sents" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$forward" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
                   echo  "$recived" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$cbr" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
               echo  "$agt" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
                  echo  "$remaining_energy" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv

                   echo  "$rreq" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv
                   echo  "$rrep" | awk '{printf ", %.5f", $1}' >> Output/Results/Final_Result.csv


       done
#Remove all temporary files.
rm 1.txt
rm tmp_result

#Move all output files to their locations.
mv *.nam Output/nam
mv *.tr Output/trace
mv *.txt Output/Results

#Generate and print graphs to their own .png file.
gnuplot gnuplot_script

echo "Completed successfully!"
```

#==========

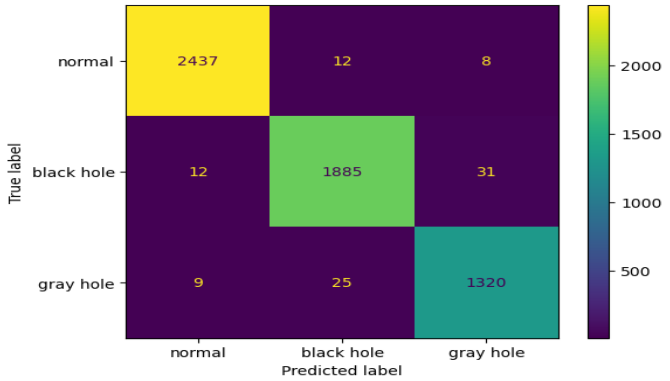
APPENDIX C: CONFUSION MATRIX FOR LSTM, GUR ,CNN and CNN_LSTM



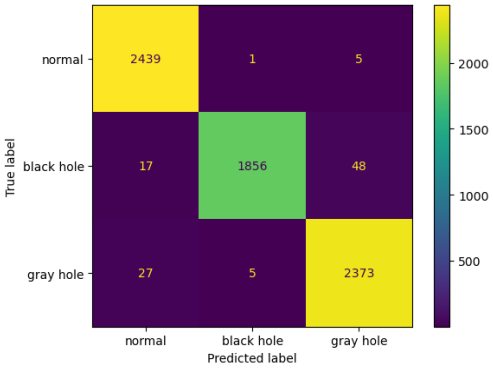
A). Before feature selection LSTM



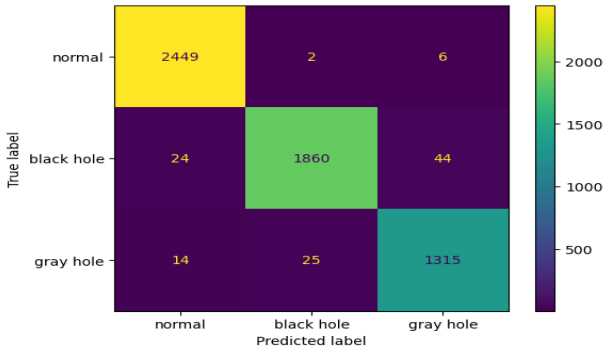
B). After feature selection LSTM



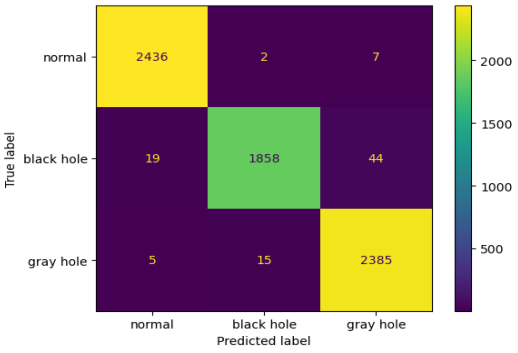
C) Before feature selection GUR



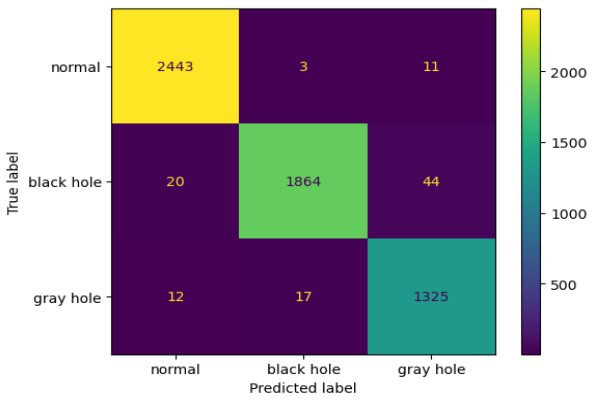
D). After feature selection GUR

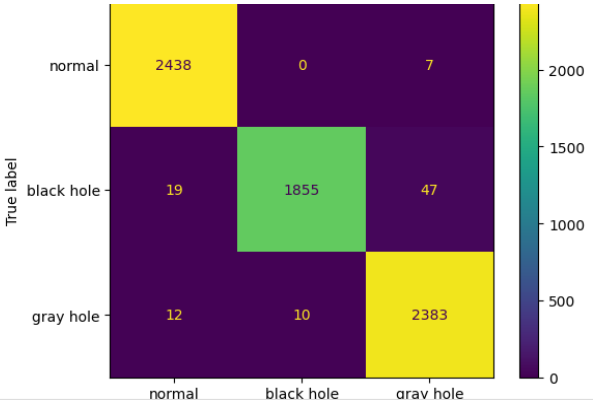


E) Before feature selection CNN



F). After feature selection CNN





CNN_LSTM before feature selection CNN_LSTM after feature selection

#==========

Importing Libraries

[2]:
```
import glob
import pandas as pd
import numpy as np
import dask.dataframe as dd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
```

```
# Loading data set
```

[5]:
```
dataset = pd.read_csv('Dataset_final12.csv', low_memory=False)
df=dataset
```

Attack type Distribution

[10]:
```
df['Attack_type'].value_counts()
```

[10]:
```
Attack_type
Normal        12053
Black hole     9748
Gray hole      6890
Name: count, dtype: int64
```

Preprocessing lavel encoding

[26]:
```
import glob
import pandas as pd
import numpy as np
import dask.dataframe as dd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
dataset = pd.read_csv('Dataset_final12.csv', low_memory=False)
df = dataset
df['Attack_type'] = df['Attack_type'].replace({'Normal': 0, 'Black hole': 1, 'Gray hole': 2}).astype(int)
df['Protocol_type'] = df['Protocol_type'].replace({'UDP': 1, 'TCP': 2}).astype(int)
# Save the preprocessed dataset to a new CSV file
df.to_csv('preprocessed_dataset_MANET1.csv', index=False)
```

Data normalization

[66]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import joblib
data = pd.read_csv('preprocessed_dataset_MANET1.csv')
df=data
X = data.iloc[:, 0:20]
y = data['Attack_type']
scaler = StandardScaler()
X = scaler.fit_transform(X)
# Display normalized features
print("Normalized Features:")
print(X)
joblib.dump(scaler, 'standard_scaler.save')
```

Blancing the data set by using smote

[38]:
```
import numpy as np
from collections import Counter
from imblearn.over_sampling import SMOTE
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical

smote = SMOTE(sampling_strategy='minority',random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
y_resampled = to_categorical(y_resampled)

# class distribution and shape of y_resampled
print("Original dataset shape:", Counter(y))
print("Resampled dataset shape:", Counter(y_resampled.argmax(axis=1)))
print("Shape of y_resampled:", y_resampled.shape)
```

Feature selecting by using catBoost

[74]:
```
from catboost import CatBoostClassifier
import matplotlib.pyplot as plt
import joblib
catboost_model = CatBoostClassifier(iterations=300, learning_rate=0.1, depth=6, verbose=0)
catboost_model.fit(X_resampled, y_resampled.argmax(axis=1))

# Extract Feature Importance
importance = catboost_model.get_feature_importance()

feature_names = df.columns[0:20]
plt.figure(figsize=(10, 6))
plt.bar(feature_names, importance)
plt.xticks(rotation=90)
plt.title('Feature Importance')
plt.xlabel('Feature Name')
plt.ylabel('Importance Score')
plt.show()
feature_names = data.columns[:-1]
feature_importance_df = pd.DataFrame({
    'Feature Name': feature_names,
    'Importance Score': importance
})
```

```
feature_importance_df = feature_importance_df.sort_values(by='Importance Score', ascending=False)
print(feature_importance_df)

# Select Important Features
threshold = 1
selected_features = np.where(importance > threshold)[0]

# Create a subset of the dataset with selected features
X_selected = X_resampled[:, selected_features]
feature_importance_df.to_csv('feature_importance.csv', index=False)
joblib.dump(selected_features, 'catboost_selector.save')
joblib.dump(catboost_model, 'feature_selector.save')
```

Split preprocessed data set train and test ¶

[42]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_selected, y_resampled, test_size=0.2, random_state=42)
```

Bulid LSTM model

[49]:
```
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
# define and bulid the LSTM Model
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))

lstm_model = Sequential()
lstm_model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=True ,activation='relu'))
lstm_model.add(Dropout(0.2))
lstm_model.add(LSTM(50, activation='relu'))
lstm_model.add(Dropout(0.2))
lstm_model.add(Dense(y_train.shape[1], activation='softmax'))
lstm_model.summary()
```

Compile and Train the LSTM model ¶

[52]:
```

# Compile and train the model
lstm_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
lstm_history = lstm_model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))
lstm_model.save('lstm_model.h5')
print("LSTM model saved successfully.")
```

Bulid model ¶

[66]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense, Dropout
from tensorflow.keras.utils import to_categorical
gur_model = Sequential()
gur_model.add(GRU(64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=True,activation='relu'))
gur_model.add(Dropout(0.2))
gur_model.add(GRU(50,activation='relu'))
gur_model.add(Dropout(0.2))
gur_model.add(Dense(3, activation='softmax'))
gur_model.summary()
```

Compile and Train the GUR model

[69]:
```
#  Compile the Model
gur_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#  Train the Model
gur_history = gur_model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))
gur_model.save('gur_model.h5')
print("GUR model saved successfully.")
```

CNN

[92]:
```
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout

cnn_model = Sequential()

#  1D CNN layers
cnn_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Flatten())

#  Dense layers
cnn_model.add(Dense(128, activation='relu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(y_train.shape[1], activation='softmax'))
```

```
# Compile the model
cnn_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
cnn_history = cnn_model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))

# Save the model
cnn_model.save('cnn_model.h5')
print("CNN model saved successfully.")
```

Evaluate the CNN model

[95]:
```
cnn_loss, cnn_accuracy = cnn_model.evaluate(X_test, y_test)
print(f'Test Loss: {cnn_loss:.4f}, Test Accuracy: {cnn_accuracy:.4f}')
#  predictions
y_preds = cnn_model.predict(X_test)
y_pred_classes = np.argmax(y_preds, axis=-1)
```

CNN_LSTM

```
[ ]  from keras.models import Sequential
     from keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Dropout, Flatten, Reshape

     # Initialize the model
     cnn_lstm_model = Sequential()

     # Feature Selection: CNN Layers
     cnn_lstm_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
     cnn_lstm_model.add(MaxPooling1D(pool_size=2))
     cnn_lstm_model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
     cnn_lstm_model.add(MaxPooling1D(pool_size=2))
     cnn_lstm_model.add(Flatten())  # Flatten the output

     # Reshape to 3D for LSTM input
     cnn_lstm_model.add(Reshape((-1, 128)))  # Adjust to match the desired time steps and features

     # Detection: LSTM Layers
     cnn_lstm_model.add(LSTM(64, return_sequences=True))
     cnn_lstm_model.add(Dropout(0.2))
     cnn_lstm_model.add(LSTM(50))  # LSTM for sequence detection
     cnn_lstm_model.add(Dropout(0.2))

     # Dense Output Layer for Classification
     cnn_lstm_model.add(Dense(y_train.shape[1], activation='softmax'))
```

```
# Train the model
cnn_lstm_history = cnn_lstm_model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))

# Save the model
cnn_lstm_model.save('cnn_lstm_model.h5')
print("Hybrid CNN-LSTM model saved successfully.")
# Evaluate the Hybrid CNN-LSTM Model
cnn_lstm_loss, cnn_lstm_accuracy = cnn_lstm_model.evaluate(X_test, y_test)
print(f'Test Loss: {cnn_lstm_loss:.4f}, Test Accuracy: {cnn_lstm_accuracy:.4f}')
#  predictions
y_preds = cnn_lstm_model.predict(X_test)
y_pred_classes = np.argmax(y_preds, axis=-1)
```