

Addis Ababa University

College of Natural Sciences

Department of Computer Science

A Hybrid Deep Learning-based ARP Attack Detection and Classification Method

Yeshareg Muluneh

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia
December 2023

Addis Ababa University

College of Natural Sciences

Department of Computer Science

Yeshareg Muluneh

Advisor: Dr. Solomon Gizaw

This is to certify that the thesis prepared by Yeshareg Muluneh, titled: *A Hybrid Deep Learning-based ARP Attack Detection and Classification Method* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science compiles with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Name	**Signature**
1. Advisor, Solomon Gizaw (PhD)	_____
2. Examiner, _____	_____
3. Examiner, _____	_____

Abstract

To map the Internet Protocol (IP) addresses to the Media Access Control (MAC) addresses and vice versa in local area network communication, the Address Resolution Protocol (ARP) is the most crucial protocol. ARP, however, is an unauthenticated protocol that lacks security features and is stateless in nature. Therefore, ARP is vulnerable to many attacks, and it can be easily exploited to gain unauthorized access to one's sensitive data and transmit bogus ARP messages to poison the ARP caches of the hosts within the local area network. These attacks may result in a loss of data integrity, confidentiality, and the availability of an organization's information.

Many researchers have struggled to detect ARP attacks using different methods. However, some of these papers are not time-effective, require more human effort and involvement, and have high communication overhead. The other works use machine learning and deep learning methods, which have better solutions for detecting ARP attacks. However, those approaches have a significant false alarm rate of 13%, a low attack detection rate, and a classification accuracy of 87%.

This thesis work aims to solve those problems using a hybrid deep learning-based ARP attack detection and classification method. In this work, we used a Sparse Autoencoder for important feature extraction and dimensionality reduction for input data and a Convolutional Neural Network for attack detection and classification to achieve the highest attack detection rate and classification accuracy with a minimized false alarm rate. To evaluate the performance of the proposed model, we used an open-source benchmark NSL-KDD dataset for training and testing. The results obtained by the implementation and evaluation are measured in comparison with a single Convolutional Neural Network model with different evaluation metrics. Hence, the proposed approach scores the highest results for attack detection rate of 98.97%, classification accuracy of 99.26%, and minimum false alarm rate of 0.74%.

Keywords: ARP Attack, Sparse Autoencoder, Convolutional Neural Network, NSL-KDD, hybrid, Machine learning and deep learning

Dedication

I would like to dedicate my thesis to my Mother Aselefech Nigusse, you mean a lot to me, without you nothing will be completed successfully in my life. You are always with me and keep in my heart. I wish and pray, God, keep your soul in haven!

Acknowledgements

The first and the most I would like to Thank Almighty God and His mother St. Marry.

Second, I would like to thank my advisor Dr. Solomon Gizaw for his initial guidance and continuous support, helpful comments, Encouragement, suggestions, patience and full devotions from beginning to end of my thesis work.

Third, I would like to thank my husband Engr. Genene Atlaw and my little son Beamlak Genene for they have showed me their love and support despite the fact that I abandoned them focusing in my work, and I steal my son Mindful Time.

Fourth, I want to express my gratitude for My Mother Aselefech Nigusse. You made me next to God, thus even if I haven't seen you since this MSc class's last exam, you are always with me.

Fifth, I would like to thank to all my family, friends, and colleagues for your continuous support and encouragement, especially my Father Muluneh Dagne. I thank you for you having been encouraging and for your prayer.

Table of Contents

List of Tables iv
List of Figures v
List of Algorithms vi
List of Acronyms and Abbreviations vii
Chapter 1: Introduction 1
1.1 Background 1
1.2 Motivation 2
1.3 Statement of the Problem 4
1.4 Objectives 5
1.5 Methods 5
1.6 Scope and Limitations 7
1.7 Application of Results 7
1.8 Thesis Organization 8
Chapter 2: Literature Review 9
2.1 Address Resolution Protocol (ARP) 9
2.1.1 How ARP Works 9
2.1.2 ARP Cache Table 10
2.1.3 Types of ARP Messages 12
2.2 ARP Attacks 13
2.2.1 ARP Spoofing-based Attacks 14
2.2.2 Tools Used for ARP Spoofing Attacks 16
2.3 ARP Attack Detection and Prevention Methods 17
2.3.1 Static ARP Entire 17
2.3.2 ARP Spoofing Detection Tools 19
2.3.3 Other ARP Attack Detection and Prevention Methods 20
2.4 Machine Learning and Deep Learning Algorithms 22
2.4.1 Machine Learning Algorithms 22
2.4.2 Deep Learning Algorithms 28
2.5 Dataset 36
2.6 Evaluation Metrics 37

2.7 Summary 39
Chapter 3: Related Work 41
3.1 Centralized-based Techniques 41
3.2 Cryptographic-based Techniques 44
3.3 Modified Protocol-based Techniques (State full ARP) 45
3.4 Packet Header Analyzing and Routing Trace-based Techniques 47
3.5 ML and DP Learning-based Techniques 48
3.6 Summary 51
Chapter 4: The Proposed Model Architecture 53
4.1 Overview 53
4.2 Major Components of the Proposed Model 54
4.2.1 Data Preprocessing 54
4.2.2 CNN- Classification 59
4.2.3 Model Testing 60
4.2.4 Classification Result 61
4.3 Summary 61
Chapter 5: Implementation and Evaluation 63
5.1 Overview 63
5.2 Development Tools 63
5.3 Experimental Setting 64
5.4 Data Collection and Preprocessing 65
5.4.1 NSL-KDD Dataset 65
5.4.2 Preprocessing 67
5.5 Implementation of Attack Detection and Classification Model 71
5.5.1 Feature Extraction 71
5.5.2 CNN Classification 73
5.6 Evaluation based on Test Data 74
5.7 Summary 79
Chapter 6: Conclusion and Future Work 81
6.1 Conclusion 81
6.2 Contribution 81
6.3 Future Works 82

References 83
Appendix A 92
Appendix B 94
Appendix C 96

List of Tables

Table 2.1 ARP message format.....10
Table 2.2 Each entry of hosts A and B's ARP cache table under normal conditions 12
Table 2.3 Confusion matrix for five classes' classification evaluation……..37
Table 5.1 NSL-KDD trains and test set attack types with their categories ..... ……66
Table 5.2 Attack class and number of patterns under each class .....…..66
Table 5.3 Hyper parameters used for model building.....72
Table 5.4 Performance evaluation results for proposed model AE-CNN and CNN……..76

List of Figures

Figure 2.1 Sender A broadcasts ARP Request for Receiver B 11
Figure 2.2 Receiver B sends a unicast ARP Reply to Sender A 11
Figure 2.3 ARP Poisoning on A and B by Host C 14
Figure 2.4 Different ML techniques and their required data 22
Figure 2.5 ANN architecture 24
Figure 2.6 Activation function per node or neuron 25
Figure 2.7 Architecture of single layer Autoencoder 29
Figure 2.8 The Sparse Autoencoder architecture 31
Figure 2.9 Architecture of CNN to identify the input images 33
Figure 2.10 ReLU function in CNN 34
Figure 2.11 Process of Max pooling 35
Figure 2.12 Flattening process in CNN 35
Figure 4.1 The proposed ARP attack detection and classification model architecture 54
Figure 5.1 Loading Train set into Panda data frame 68
Figure 5.2 Labeling encoding for each feature naming 68
Figure 5.3 checking any null value for train and test set 69
Figure 5.4 Removing difficulty- level feature from both train and test set 69
Figure 5.5 One hot encoding for training set categorical features 70
Figure 5.6 Attack labeling for one of four categories for training set 70
Figure 5.7 Train data normalization using Min max scalar 71
Figure 5.8 Latent features representation after encoder using t-SNE Visualization 73
Figure 5.9 Test dataset distribution value across each class. 74
Figure 5.10 The 5x5 confusion matrix for AE-CNN classification results 75
Figure 5.11 AE-CNN Classification results in confusion matrix with each class 75
Figure 5.12 AE-CNN classification results evaluation for each attack class in bar graph 76
Figure 5.13 CNN Classification results in a confusion matrix with each class 77
Figure 5.14 CNN classification results evaluation for each attack class in bar graph 77
Figure 5.15 Evaluation results comparison AE-CNN with CNN in bar graph 78

List of Algorithms

Algorithm 4.1 AE feature extraction and reduction algorithm 58
Algorithm 4.2 CNN classification and training algorithm.....58
Algorithm 4.3 CNN model testing and classification algorithm.....59

List of Acronyms and Abbreviations

ACS	ARP Centralized Server
AE	Autoencoder
AKD	Authoritative Key Distributor
ANN	Artificial Neural Network
ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency Network
BSVR	Bayesian Support Vector Regression
CCE	Mean Absolute Error Loss
CIC-IDS2017	Canadian Institute for Cybersecurity, 2017
CNN	Convolutional Neural Network
Colab	Google Collaboratory
DAI	Dynamic ARP Inspection
DHCP	Dynamic Host Configuration Protocol
DL	Deep Learning
DNS	Domain Name system
DOS	Denial of Service
FAR	False Alarm Rate
FPR	False Positive Rate
FTP	File Transfer Protocol
GDPS	Gratuitous Decision Packet System
GPU	Graphical Processing Unit
HTTP	hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection systems
IP	Internet Protocol
IPS	Intrusion Prevention System

KDD'99	Knowledge Discovery in Databases cup'99
KL	Kullback-Leibler Divergence Loss
LAN	Local Area Network
MAC	Medium Access Control
MAE	Categorical Cross Entropy
MD5	Message Digest Method 5
MITM	Man-In-The-Middle
ML	Machine Learning
MSE	Mean Squared Error
NLP	Natural Language Processing
NSL-KDD	Network Security Laboratory-Knowledge Discovery in Databases
OSI	Open System Interconnection
POP	Post Office Protocol
R2L	Remote to Local
ReLU	Rectified Linear Unit
RTNSS	Routing Trace-based Network Security Solution
S-ARP	Secure- Address Resolution protocol
SDN	Software Defined Networking
SSH	Secure Socket Shell
SVM	Support Vector Machine
T-ARP	Ticket-based Address Resolution protocol
TCP	Transport Control Protocol
TSCBA	Time Stamp and Counter Based Approach
U2R	User to Root
UDP	User Datagram Protocol
VM	Virtual Machine

Chapter 1: Introduction

1.1 Background

Networking plays a significant role in current business and communication channel management. The evolution of network architecture can be traced from the ARPANET (Advanced Research Projects Agency Network) of the past to the current Internet, illustrating how theoretical network architecture gave way to practical implementations [1]. Parallel to this, network threats are expanding just as fast and using more sophisticated techniques to affect network communication.

Cyber resilience is being advanced by the increasing pace of digitalization driven by the COVID-19 pandemic and the change in our working habits [2]. Cyber security risks are also increasing quickly in both quantity and variety. According to the [2] study, three cyber threats are of particular concern to 80% of cyber leaders. Attacks using ransomware are the first, becoming more frequent and sophisticated. Social engineering attacks and malicious insider activity within the organization are the second and third issues, respectively. According to Positive Technologies' research in [3], frightening cyber figures for the middle of 2022 show that 93 percent of corporate networks are vulnerable to hacking. These findings show that a third party can get past a network's security measures and gain access to internal network resources inside a company.

One of the main causes of Local Area Networks (LAN) being vulnerable to attacks is hardening the Data Link Layer (layer 2 of the OSI Model), which is one area that is typically mishandled, and this leaves the network vulnerable to several assaults and intrusions [4]. Since network administrators believe it is hard for external malicious parties to access a company's internal network, they usually do not concentrate on the data link layer security measures [5]. On the other hand, to secure Layer 3 and upper layers, there are different techniques like packet filtering, access control, Intrusion Detection Systems (IDS), Cryptography methods, and firewalls. However, the data link layer lacks the granular restrictions necessary to avoid attacks against it. This makes it simple to launch attacks on the network, which may result in a loss of data integrity, confidentiality, and availability of the organization's information.

Address Resolution Protocol (ARP) is one of the most vulnerable standards in an Ethernet communication. It is used to map the Media Access Control (MAC) address of the data link layer to the Internet Protocol (IP) address of the network layer, which is kept in the ARP cache of each client machine [6]. To find the MAC address for the machine's IP address, ARP is performed by sending a message request called a broadcast message [7]. Every time a machine wants to communicate with another machine on an identical LAN subnet, it sends an ARP request message along with the recipient machine's IP number. Any machine on the same network segment can view the message, but only the one with that IP number can respond with a reply message that contains its MAC addresses, known as an ARP reply.

The fact that ARP is an authentication-free protocol that responds to all ARP requests without checking the validity of the packets received is a characteristic of this protocol. ARP is the main driver behind the majority of local area network attacks [8]. ARP attacks come in a wide variety of methods, including Man-In-the-Middle (MITM), Denial of Services (DoS), host impersonation, etc. [9]. Attackers utilize a technique called ARP spoofing or ARP cache poisoning, to spoof ARP packets on a LAN [10]. ARP spoofing is a suspicious assault in which bogus ARP packets are sent among employees of the firm [11]. This links the IP address of a trustworthy computer or service on the network to the attacker's MAC address, and then it might be able to interrupt all traffic, modify the traffic, or intercept data packets on a network.

Many studies on detecting and defending against ARP spoofing attacks using various techniques were presented. However, those research studies have less attack detection rate [7, 12], classification accuracy [13, 14], and high false alarm rate [15]. To overcome these problems, we need to implement an intelligent and effective method. In this thesis, we propose a hybrid (Autoencoder and Convolutional Neural Network) deep learning-based ARP attack detection and classification technique to achieve high threat detection rate and classification accuracy.

1.2 Motivation

Most organizations store their private and sensitive information, like bank transactions, business activities, financial data, customer and employee information, inside their local

area network to keep it secure. However, due to the data link layer's insecure nature [4, 5], private data can be accessed by an unauthorized person, stolen, intercepted, modified, or denied by any threats. For those LAN security issues, ARP is one of the data link layer protocols, which is more vulnerable to attacks [8].

It is easy to launch assaults on the network because the data connection layer misses the granular restriction needed to fight off attacks against it. These attacks may result in a loss of data integrity, confidentiality, and availability of an organization's information. On the other hand, insider threats provide substantial difficulties for local area network security since they are difficult to identify and prevent because they already have access to networks and systems. As a result, any adversary can access, steal, intercept, alter, or deny access to the sensitive data that most enterprises keep in their LANs. Staff members of the same organization carry out most attacks inside organizations. Attacks often target layer 2 rather than layer 3 or IP. These assaults make use of the layer 2 switch activity. ARP spoofing is one of the attacks of this type [16].

On the other hand, cyberattack incidents increase from time to time and cause financial costs, infrastructure damage, data loss, denial of service, and reputational damage in many financial and non-financial sectors. According to a checkpoint study report [17], cyberattacks increased by 38% in 2022 over 2021. In this report, education and healthcare sectors were taken the highest in cyberattacks in 2022 relatively to all other sectors. These figures are primarily caused by groups that distribute ransomware and fewer, flexible cybercriminals that focus on taking advantage of collaborative platforms used in remote work settings. Likewise, a LAN attack can also have major financial and reputational impacts on a company. The average cost of a data breach is $3.86 million, according to IBM's 2022 Cost of Data Breach Study [18]. Because LAN attacks frequently result in exposing sensitive data, such as customer profiles or financial information, the cost may even be higher for firms that experience LAN attacks. Additionally, LAN assaults can harm a company's reputation, which implies that if a LAN attack exposes client data, the customers may lose faith in the company and do business elsewhere. A decrease in market share and a loss of revenue may result from this. As a response, addressing the ARP

protocol's weakness by offering a reliable solution reduces the issues with data link layer security.

Generally, the motivation to do this thesis relies on the recent growth of cyberattacks on the company's local area networks and the ARP attack detection and mitigation solution gap in the literature review. So, we need to design an effective ARP attack detection and classification method by using recent and powerful deep learning algorithms.

1.3 Statement of the Problem

ARP is the most crucial protocol in local area network communication for mapping IP addresses to MAC addresses. It has the ability to translate a 48-bit MAC address from a 32-bit IP address [19]. Nevertheless, because ARP lacks security protections, it is open to several attacks. Furthermore, ARP is stateless in nature and an unauthenticated protocol [20]. It replies to any ARP requests without validating the packets received. As a result, attackers can quickly take advantage of ARP to access private information without authorization and send false ARP messages to contaminate the ARP caches of LAN hosts.

A variety of researchers have struggled to prevent and detect ARP attacks using different methods, algorithms, and approaches. All of them, however, have their own drawbacks. For example, if we take [7] and [12], they do not detect insider threats and host impersonation attacks, respectively. Further, those are not time-effective systems, which mean it takes much time to detect and respond to ARP attacks and require more work effort for the network administrator to register all users in the system, making them unsuitable for large-scale local area networks. Additionally, in [15], the researchers used the Autoencoder algorithm, which is unsupervised DL (deep learning) and an effective method for detecting previously unknown attacks. The researchers were able to cluster suspicious ARP activity, but there was a high false-positive detection rate. Whereas in [19], BSVR-ARP is used, this is a supervised ML strategy that works well for previously known attack detection. However, the classification accuracy decreases as the input data size increases. Furthermore, the other recent work in [14] used a convolutional neural network to detect ARP attacks, and it performed around 87% of the classification accuracy. This is because, the Authors used a powerful algorithm but there are weaknesses in their hyperparameter selection and dataset size for training and testing their proposed method. In general, those research studies and

others on ARP attack detection systems have less detection rate, classification accuracy, and high false alarm rate. This research work is aimed to fill the gap of the problems mentioned above.

In our research paper, we suggest a hybrid deep learning algorithms strategy that combines unsupervised Autoencoder for feature extraction and dimensionality reduction with supervised Convolutional Neural Networks for classification technique to achieve the highest threat detection rate and enhance classification performance.

1.4 Objectives

General Objective

The general objectives of this thesis is to design a detection and classification model for ARP attacks using a hybrid method by combining Convolutional Neural Network and Autoencoder deep learning algorithms.

Specific Objectives

To achieve the general objectives of this study, the specific objectives are:

- To carry out a thorough analysis of the literature in order to comprehend the issue area and suggest a suitable deep learning algorithm for effective ARP attack detection and classification methods
- To design a model by using a hybrid (AE with CNN) deep learning algorithm
- To implement the proposed model for testing purpose
- To test and evaluate the proposed system

1.5 Methods

The overall methodology of this research is based on design science methodology principles, which produce an artifact including architecture, algorithm, testing, and evaluation of the result. To achieve the objectives of this study, different methods were used, including:

- **Literature Review**

We perform a comprehensive literature review to gain a clearer understanding of the research field and its problem domains. Through this literature, we identify the strengths and

weaknesses of many existing solutions for ARP attack detection and classification techniques. For this related literature, we used different sources, like journals, conference papers, books, and some other online sources.

- **Experimental Setup and Software Tools to be used**

To experiment with the proposed ARP attacks detection and classification method, we used the following different tools.

- ✓ **TensorFlow:** this is an open-source framework and contains different libraries for machine learning, deep learning, and other statistical and predictive analytics tasks. We're going to employ to put the suggested system into practice.
- ✓ **Keras:** It's an open-source software package that offers a neural network-based Python front end, and it serves as the TensorFlow library's front end.
- ✓ **Python:** Is popular Programming language for machine learning.
- ✓ **Google Drive**: For datasets and model implementation source codes storage.
- ✓ **Google Collaboratory (Colab):** - With its notebook interface, we used for write and run code, view output, and produce visualizations for the proposed model.
- ✓ **Windows 10 Operating system:** For the entire experiment would be carried out.

- **Dataset Collection**

To train and test the proposed system, we select and used an open-source benchmark dataset, NSL-KDD**,** which is heavily used for intrusion detection systems for many scholarly research papers. The University of New Brunswick's KDD'99 has been updated and refined into the NSL-KDD [21]. This dataset, which includes training and testing dataset, has forty-one characteristics and one class feature. There are assaults and regular information in the NSL-KDD dataset. Denial of Service (DOS), Probe, Remote to Local (R2L), and User to Root (U2R) are the four types of assaults.

- **Train the model**

With the help of the preprocessed dataset, the designed model would be trained to understand the traffic data and categorize the behaviors of the collected traffic (whether it is normal or attacks with four categories).

- **Testing and Evaluation**

To test and evaluate the proposed model, we set an experimental or simulation environment by using the TensorFlow framework and Python programming to build the hybrid model. To train and test the proposed model we used the training and testing set respectively. Finally, the model evaluated by using performance measurement metrics like attack detection rate, classification accuracy and false positive rate etc.

1.6 Scope and Limitations

In this research study, we suggest a hybrid deep learning algorithms strategy that combines unsupervised Autoencoder for feature extraction and dimensionality reduction with supervised CNN for classification techniques to achieve the highest threat detection rate and enhance classification accuracy.

The proposed model would not be implemented in a real environment and would not collect real-time network traffic data due to a shortage of time and an organized laboratory constraint. Despite this, we used the NSL-KDD dataset and simulation environment by using the TensorFlow framework and Python programming to examine the designed model. Also, this study does not include attack prevention method.

1.7 Application of Results

By suggesting appropriate feature extraction and classifier techniques, this research would improve the effectiveness of ARP attack detection and classification methods.

This thesis study has three primary achievements. It does this by offering network managers an automatic and sophisticated threat detection mechanism, which reduces their workload. The availability, confidentiality, and data integrity of local area networks within enterprises are the second one. In conclusion, it will function as a point of reference for forthcoming

investigations and furnish insights for specialists seeking to enhance their areas of expertise in the security of networks.

1.8 Thesis Organization

This thesis document's subsequent sections are arranged as follows: A review of the ARP protocol, ARP assaults, techniques for detecting and preventing attacks, machine learning, and deep learning algorithms are presented in Chapter 2. Reviews of related studies on the identification and defense against ARP attacks using different strategies are presented in Chapter 3. The overall architecture and design for identifying and classifying an ARP assault are explained in Chapter 4. An Autoencoder and Convolutional Neural Network algorithms are used to identify and categorize attacks. Chapter 5 presents the experimental results and their evaluations, and examines the specifics of the implementation. The thesis's conclusions, contributions, and plans will cover in Chapter 6.

Chapter 2: Literature Review

2.1 Address Resolution Protocol (ARP)

ARP is a mechanism that allows network traffic to reach a specific LAN host. It switches IP addresses into physical addresses and the other way around. ARP is usually used by devices to establish a connection with the network router or entrance that provides them with Internet access [22]. According to researchers, MAC, a distinct hardware-level address issued through the manufacturer, is the fundamental component of ARP [23].

The Internet Protocol (IP) uses the Address Resolution Protocol, as determined in [24], to tie the logical addresses to the physical addresses used by the link layer in LAN environments. At the network layer, this protocol functions as a bridge between the data connection and the network layer. Address resolution is the process of discovering the address of a network node. A network node delivers a piece of information to the destination node as part of a protocol used for address discovering. The destination node's physical address and logical address are contained in the ARP packet's data that it receives [25]. The IP packets are not the ones that ARP uses. ARP frames never leave the local network and are only useful in a single LAN segment. ARP messages are enclosed in Ethernet frames or any other LAN frame, just like IP packets [26]. The ARP frame contains the IP and the MAC address of the sender and receiver; it also has an operation code to clarify the ARP type message (1 for the ARP request, 2 for the replay). The structure of an ARP message is shown in Table 2.1 [12, 26].

2.1.1 How ARP Works

ARP is carried out by generating an information request known as a broadcast message [7]. Every time a host seeks to exchange information with any other on the same subnet, it sends an ARP request message along with the receiving host's IP address. Any host on a local network segment has access to the message, but only the host with that IP address can react with an answer message that contains its physical address, known as an ARP response. The requester host can now send the packet to the destination host since it has all the needed information. The following is a scenario of an ARP request and an ARP reply [27].

Table 2.1 ARP message format

16 bits data		16 bits data
Hardware Type		Protocol Type
Mac Address Length	Protocol Address Length	OP Code Number
Sender MAC Address		
Sender IP Address		
Receiver MAC Address		
Receiver IP Address		

If host A (the sender) wants to communicate with host B (the receiver) inside the same network segment, the sender needs to be aware of the receiver's MAC address, which is equivalent to the logical address (IP). Then, the sender looks up the receiver's MAC address in the ARP cache table. If the receiver's MAC address is not listed in the ARP cache table, it will broadcast an ARP request containing the source IP and MAC addresses, as well as the receiver's IP address, to all other hosts in the same network segment. Figure 2.1 [20] shows this request for the receiver's MAC address. As shown in Figure 2.2 [20], only the receivers will respond to the broadcast request by sending a unicast ARP reply containing their own MAC address. Here, the receiver will send back an ARP reply to the sender with its IP and MAC addresses. The other hosts will ignore the request. Therefore, the sender will store the MAC address of the receiver in its ARP cache table, and it will start communication with the receiver.

2.1.2 ARP Cache Table

The ARP cache table, which is present on each host in a network segment, links IP addresses to the appropriate MAC addresses. ARP request or reply messages have the ability to add new entries to an ARP cache or update ones that are already there. Table 2.2 [28] shows the entries in the ARP cache of hosts A and B for the above example shown in Figure 2.1 and Figure 2.2. An ARP cache contains two different types of entries: static entries, which stay there until the system restarts, and dynamic entries, which stay there for a short

period of time (depending on the OS) and are subsequently deleted if they are not referenced. Small LAN networks frequently employ static entry techniques. The deployment and updating of static ARP cache entries, however, are unusual activities in large networks [29].

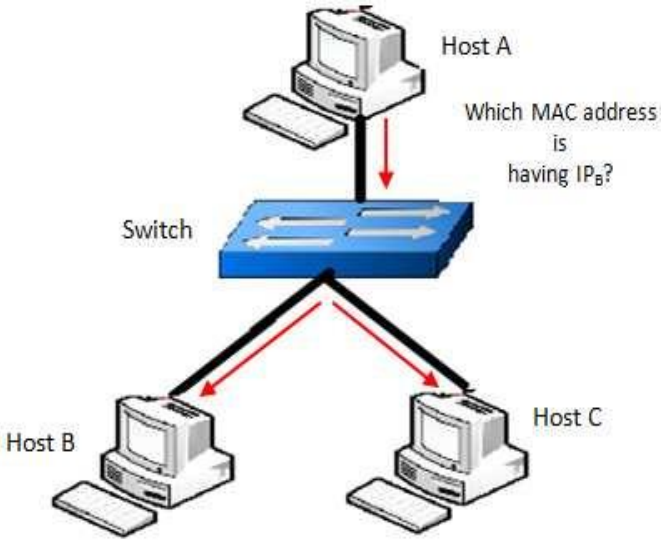


Figure 2.1 Sender A broadcasts ARP Request for Receiver B

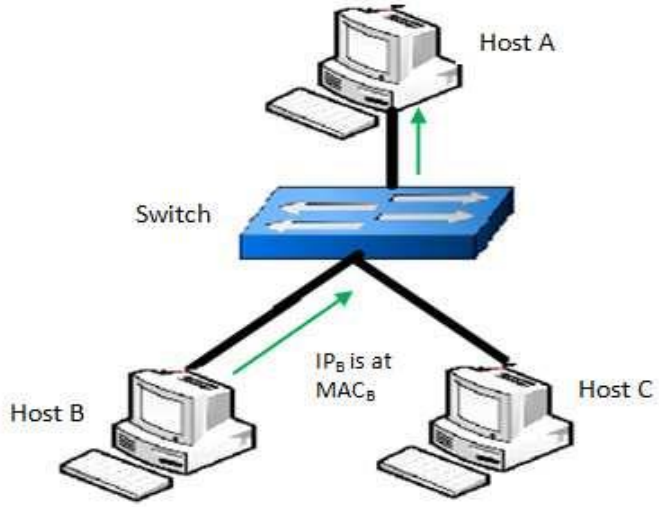


Figure 2.2 Receiver B sends a unicast ARP Reply to Sender A

Table 2.2 Each entry of hosts A and B's ARP cache table under normal conditions

Host A's ARP Cache Table	
IP_of_Host_B	MAC_Address_of_B
IP_of_Host_C	MAC_Address_of_C
Host B's ARP Cache Table	
IP_of_Host_A	MAC_Address_of_A
IP_of_Host_C	MAC_Address_of_C

An ARP table contains either of the two categories of ARP entries: dynamic ARP entries or static ARP entries [30]. By default, ARP creates and updates a dynamic entry. It may become outdated, receive an update from a fresh ARP packet, or be replaced by a static ARP entry. When the connection is offline or the aging timer for a dynamic ARP entry expires, the entry is deleted [31]. A dynamic ARP cache item is one that the protocol learns and has a time-out value attached to it so entries are removed from the cache after a predetermined time [32]. In Static ARP entry, the configuration and maintenance of a static ARP entry are done manually. It cannot be replaced by a dynamic ARP entry or age out. It is not suitable for large networks [30] [31].

2.1.3 Types of ARP Messages

ARP request and ARP replay messages are the two distinct kinds of ARP messages that are used in the LAN segment to carry out the address resolution operation [33]. When a host wants to find out the MAC address of a receiver, it will send out an ARP request. The network node takes this action upon determining from its ARP database that the address is, in reality, unknown. In essence, the ARP request queries, "May I have your MAC address?" After receiving the ARP request message, the receiver will respond with an ARP reply if the requested MAC address matches its own. In response, it said, "Yeah, here it is." If at all possible, hosts never state no.

ARP standard is used to identify the duplicate IP issue in the network. Here, the two ARP packet types that can be used for this are ARP probe and ARP announcement [34]. An ARP request packet is transmitted as a broadcast frame in an ARP probe to see if a network

address is accessible for usage. Therefore, if a host wishes to give itself a new fixed network address, it will first send an ARP probe packet to the network, polling it to see if the requested address is already in use. A host will react with an ARP reply if it already knows this address. If the network address is free and the ARP probe returns no results, the host will begin to send an ARP request as an ARP announcement to let clients know that it has that network addresses [34].

2.2 ARP Attacks

A potential flaw appears as a result of the MAC addresses being kept in the ARP table. A remote hacker has the ability to alter the MAC to IP address entries, which could lead to traffic being diverted away from the intended target and toward a different one. ARP spoofing or ARP cache poisoning are two names for this technique. It is the malicious act of adding a fake IP address to the receiver's ARP cache's MAC address mapping. This can be accomplished by directly changing a receiver's ARP cache, regardless of the ARP messages the receiver sends. The rogue host can either amend a current device with bogus IP and MAC addresses or build a completely new false record in order to achieve this [35]. The terms "ARP Spoofing" and "ARP Poisoning" are frequently used interchangeably in different researches. However, some authors describe them slightly differently, as follows [11] [22].

- ✓ **ARP spoofing:** By sending phony ARP packets, a hacker connects their MAC address to the IP address of a LAN machine that is already in operation. Such an attack poses a serious risk to local area networks since it can be used to intercept data frames on a network, change content, or stop all traffic.

- ✓ **ARP poisoning:** is adding falsified MAC mappings to an organization's ARP table after a successful ARP spoof. As a result, the ARP table that connects the host's IP address to the hacker's MAC address is altered. This gives the hacker access to all LAN information intended for them and the target host, allowing them to read and alter LAN packets during the route.

Knowing that ARP does not offer an honest way of confirming the sender's identity, it is well known to be susceptible to poisoning attacks. ARP frequently triggers more potent

assaults that can seriously harm a company's network, such as DOSs, MITM attacks, or session hijacking [25]. ARP is open to assaults because it lacks security measures in its design. It replies to any ARP requests without validating the packets received. As a result, attackers can quickly take advantage of ARP to access private information without permission and send phony ARP messages to contaminate the ARP caches of hosts. As illustrated in Figure 2.3 [20], C initiated an ARP Poisoning attack. A receives a bogus ARP packet from C claiming that B's IP address is associated with C's MAC address. Similarly, C impersonates B and sends a fake ARP message claiming that C's MAC address is A's IP address. Consequently, C will be positioned as the Man-In-the-Middle (MITM) throughout all of the communication between A and B.

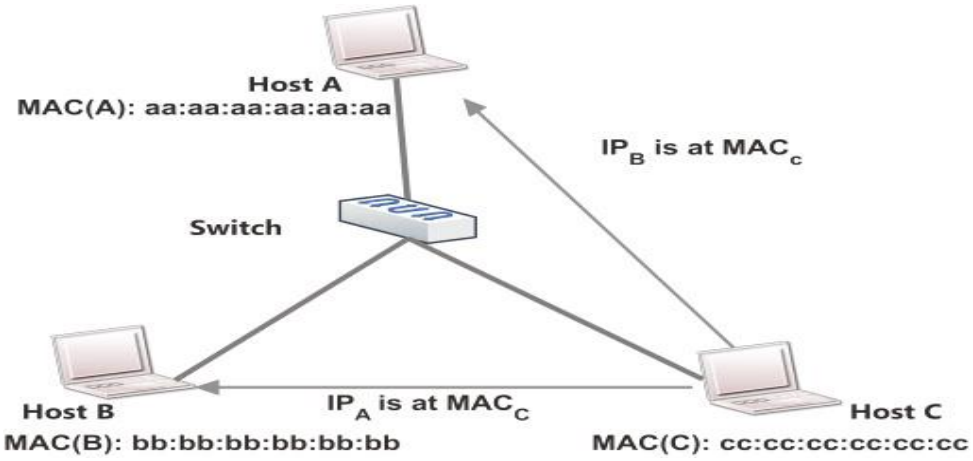


Figure 2.3 ARP Poisoning on A and B by Host C

2.2.1 ARP Spoofing-based Attacks

ARP was not created to deal with threatening hosts because it is a trusting protocol. There are a number of methods a malicious host can force a careless host to update or add an entry with an IP-MAC mapping in its ARP cache so that other more skilled attackers can use it. So, one of the following attack types can be launched using ARP spoofing [4, 36, 37, 38].

I. **Man-in-the-middle (MITM) attacks**

An attacker can secretly occupy the space between the two hosts by simultaneously spoofing two hosts on the network, misleading them into believing they are in communication. Then, the attacker has access to all traffic sent in both directions. An attacker can launch an attack between any host within the LAN and any host outside of it, just as they can launch an

attack between the host and the standard gateway. Through a MITM attack, the attacker can acquire confidential data such as passwords, or even alter the data being delivered, compromising its integrity.

II. **Denial of Service (DoS) attacks**

On a network with a fake MAC address, an attacker can transfer an ARP response with the network address. For instance, a bogus ARP response using the network's router IP and a fake MAC will break network communication throughout the entire system. ARP poisoning is frequently used in DOS attacks to connect many IP addresses to the MAC address of one host. Because of this, the target will be overloaded with traffic that is seen for numerous IP addresses will be transmitted repeatedly to the host's MAC address. A rogue system creates a lot of false identities during a DOS assault. In other words, it prevents the intended users from accessing system resources. Attacks involve flooding the target (victim) machine with requests for external communication [4].

III. **Host Impersonation**

Through the use of ARP spoofing, the ARP cache table of hosts that are interacting with one another is modified such that each packet transmitted by the host is now directed to the attacker. The attacker responds to the packet after receiving it from the host and gives the impression that the host is in communication with the intended destination [37].

IV. **Connection Hijacking**

A connected node can be tricked into having its connection modified and having complete control over it through packet or connection hijacking. Attackers can enter private systems and hijack data connections by using ARP poisoning to steal session IDs during connection hijacking operations. Another term for it is TCP session hijacking, which basically means gaining access to an Internet user session by silently obtaining the session number and assuming the role of the legitimate user. Once the user's session number has been bought, the attacker can take the user's place and carry out any actions that would be permitted [4] [36].

V. **MAC Flooding**

ARP cache poisoning is a technique used at an Ethernet switch called MAC flooding. Switches typically enter hub mode when they are overwhelmed. In hub mode, the switch broadcasts the entire network information to every node in your network while providing

port security measures. It overloads a switch's ARP table by continuously flooding it with faked ARP responses. MAC flooding causes the switch to become overloaded with data packets, disrupting the usual source-to-destination data flow that is common with MAC addresses [4].

2.2.2 Tools Used for ARP Spoofing Attacks

Different kinds of tools are used in ARP spoofing attacks to spoof the network or test network penetration. Some of these tools are listed below [4] [39].

- **Wireshark:** An open-source packet filter tool used to examine network traffic and analyses all of the communication on that interface. It is familiar with the layout of various networking protocols. It records every packet delivered and received via the network. Also, it used as an ARP spoofing detection tool.

- **Ettercap:** It is a helpful instrument for MITM attacks on LANs and in charge of content screening, sniffing live connections, and many more intriguing things. It records passwords for numerous protocols, including FTP, HTTP, POP, SSH1, and others. It serves as a Swiss Army knife for network sniffing and ARP poisoning.

- **ARPwner:** It is a tool for ARP and DNS poisoning attacks. It is completely coded in Python, available on GitHub, and includes an implementation of SSLStrip so that you can adapt it to meet your needs.

- **Window ARP Spoofer:** It is a LAN tool and able to change the ARP table of another system. This tool may effectively gather all packets on the LAN, especially by altering the ARP database of a router.

- **Cain & Abel:** It is a tool for general network tracking and ARP spoofing. It has a wide range of capabilities including, sniffing traffic, recovering passwords, attacking clients and servers, and cracking encryption keys.

- **Arpspoof:** It is a straightforward tool for testing and combining it with other tools for more complex assaults. By faking ARP responses and transmitting them to either a specified target or everybody on the internal network's connections, it has the ability to divert traffic on the local network.

2.3 ARP Attack Detection and Prevention Methods

ARP attacks can be identified and prevented using a variety of methods, techniques, and tools. The following are some methods for detecting and preventing ARP spoofing.

2.3.1 Static ARP Entire

Static ARP entries are permanent and manually added using a TCP/IP configuration. It is possible to statically map every MAC address in a network to its corresponding IP address. These will act as MAC address to IP address mappings that are permanent. ARP requests for regularly used internal network addresses, such as switches, routers and servers are avoided by using static ARP cache entries. Additionally, it safeguards device-to-device communication because attack packets cannot alter a static ARP entry's IP-to-MAC mapping. Static ARP entries have the drawback of requiring manual updating whenever network interface equipment changes [31] [32]. A device does not have any static ARP entries set up by default. There are two types of static ARP entry configuration methods basically, Manual and Automatic configuration of static ARP entries. We discuss each of them in the following manner.

A, **Manual Static ARP Entry Configuration**

The simplest and fastest way to prevent ARP spoofing is to add constant ARP entries to every host. Nevertheless, this approach is not scalable and is not easy to utilize, especially in companies where network administrators are in high demand and user density is high. If there are few entries, it is suggested to manually set up static ARP entries. Manually configure static ARP entries are shown below [31].

1. Execute the following command if the outgoing interface is a Layer 2 Ethernet interface.

   arp static *ip-address mac-address* **vid** *vlan-id* [cevid *cevid*] **interface** *interface-type interface-number* [*subinterface-number*]

2. Use the following command if the outgoing interface is a Layer 3 Ethernet interface.

   arp static *ip-address mac-address* **interface** *interface-type interface-number*

3. Use the following command to specify the VPN connection translating the ARP entries.

   arp static *ip-address mac-address* **vpn-instance** *vpn-instance-name*

4. Perform the following command to configure simple static entries. For instance, if a device is connected to a Network load-balancing cluster and multi-interface ARP has been utilized.

arp static *ip-address mac-address*

B, Automatic Static ARP Entry Configuration

Automatic scanning and permanent ARP can also be used to configure static ARP entries in a repeated manner [31]. We use the following commands to configure this entry automatically.

i. Execute the following command to enter the interface or subinterface view.

Interface *interface-type interface-number* [. *subinterface-number*]

ii. Execute the following command to configure automatic ARP scanning.

arp scan [*start-ip-address* **to** *end-ip-address*]

- After obtaining the adjacent' MAC addresses, the device sends out ARP Request messages to them in order to build dynamic ARP entries.
- You can use the **display arp dynamic** command to see the dynamic ARP entries that the machine has learned, or you can click **Ctrl+C** to halt the scanning process.
- For automatic ARP scanning, the IP address of the interface needs to be in the same network segment as the initial and finish IP addresses. Additionally, the start and end IP addresses must be less than or equal.

iii. Execute the following command to set up fixed ARP and change the dynamic ARP entries that the interface learned into static ARP entries.

arp fixup

The number of static ARP entries that fixed ARP is able to convert depends on the amount of static ARP entries the device supports. If this number exceeds what the device can handle, extra dynamic ARP entries won't be converted; an error notice will be displayed instead. The ARP entries that are explicitly supplied are the same as the static ARP entries that are

translated using fixed ARP. Use the **reset arp static** command or the **undo arp static** command to wipe items one at a time or all at once.

In addition to these, a client-server approach was put out in the work in [6] that stops ARP attack by spontaneously configuring static ARP entries. This method can function in both dynamic and permanent ARP entry in large networks without requiring any administrative work. In this paper, they states 3 different messages including, register message: which includes the client's IP and MAC addresses as well as a hashed authentication key, is sent from the client to the server, modify message: which is a broadcast notice message from the server informing everyone on the network that a fresh host has joined, register response message: an ARP entries for all users who successfully registered at the server are included in this unicast message, which the server sends to the fresh host. This method also specifies two distinct entities, one of which is the ARP Client, a piece of software that runs on users' computers and performs a number of functions and the second entity is ARP Server, a server application that can be set up on any network device as well as on a dedicated server and perform a number of tasks. This method has the disadvantage of having minimal scalability, which makes it ineffective in a dynamic context. When the network interface equipment changes, manual upgrading is required.

2.3.2 ARP Spoofing Detection Tools

A user can use a variety of software or tools to monitor, update, modify, check IP and MAC addresses, and perform other functions related to ARP cache; these tools are discussed in the following sections [36, 40, 41, 42, 43].

- **ARPwatch:** It used to maintain track of Ethernet activity on a network and logs every IP/MAC address match that it notices and notify the system administrator of any changes.
- **ARP-Guard:** It acts as an active defense against internal ARP assaults and can identify and localize using a sensor-based architecture.
- **XArp:** It is a security tool used to find ARP related attacks. It uses both active and passive components to locate attackers within the network you are using.

- **ARP AntiSpoofer:** It is a lightweight tool that can identify ARP poisoning. It presents an exhaustive user interface and makes all of its features accessible from the primary window.
- **ArpON:** In static, dynamic, and hybrid networks, it is a portable handler application for protecting ARP against poisoning, cache poisoning, or poison-route attacks.

2.3.3 Other ARP Attack Detection and Prevention Methods

In the literature, researchers proposed different techniques rather than the above discussed methods for identifying and defending against ARP attacks including kernel-based patches, securing switch ports, using the ICMP protocol, and IDS and snort approaches, among others. The Anticap and Antidote [20] strategies are kernel-based patches and the most preferred at the individual host level. These scheme states that a host should attempt to define whether the past learned MAC address is still alive when it receives an ARP reply whose MAC address changes from before cached one. The modified binding is denied and the illegal MAC address is joined to a list of forbidden addresses if the previously learned MAC address is still alive. Its cure, however, is limited to a specific kernel, and this approach has drawbacks because it needs different patches for various operating systems, which issues communication overall. Additionally, there are two workarounds for this, but they both depend on the ARP entry that is cached being the right one. As a result, there is now a computation among the attacker and the target, which is a very risky situation. If an attacker places his fake ARP entry in the host's cache before obtaining it from the real host, MAC addresses may be forbidden. Administrative action is the only way to undo this. The suggestion is that mistaken knowledge may prevent these approaches from identifying ARP spoofing [44].

In other research discussed, a feature called port security [45] can stop MAC cloning by restricting the quantity of MAC addresses that can be serviced on a particular port and restricting changes to a switch's MAC tables unless they are made by a network administrator. This approach needs a lot of manual configurations, such as static ARP binding, and it only detects MAC addresses and cannot stop attackers from using legitimate MAC addresses. Furthermore, most controlled Ethernet switches come with defenses against ARP attacks. These features are called dynamic ARP inspection (DAI). To prevent DoS

attacks effectively, DAI can frequently configure to limit the rate at which ARP messages can pass through the switch. Generally speaking, the best practice is to enable DAI across every port, with the exception of those connected to other switches. The feature doesn't significantly affect achievement; however, it might need to be enabled in addition to other capabilities like DHCP snooping [46].

In addition to these, an ICMP protocol-based detection technique for ARP attacks was proposed [47]. In order to search for malicious hosts based on their response packets, this program first collects and evaluates the ARP messages before injecting ICMP echo request messages. The technique, however, is built on a record that is accessible at the detecting device. It results in the single-point failure issue. The technique also ignores the problem of ARP poisoning by means of fictitious ICMP echo queries. With the source MAC address of itself and the spoof source IP address of an authorized host, an attacker can submit a false ICMP echo request. As a result, the target host will modify its ARP cache with dishonest mapping so that the attacker's MAC address binds to a fake legal IP address assigned to another host in the subnet when it receives this message.

Likewise, Snort is a potent open-source intrusion detection and prevention system (IDS/IPS) that provides real-time packet monitoring and network activity assessment. It uses a rule-driven language that combines abnormality, procedure, and signature examination approaches to detect possibly fraudulent activity [48]. Also, IDSs like Snort can frequently identify ARP assaults and alert the administrator by producing the proper alarm messages. The fundamental issue with IDSs is that they frequently produce false positives, therefore, to be effective; the business should assign someone to handle these situations. Paying someone to undertake this job is typically not an option for businesses. By transmitting TCP reset packets or instructing a switch to cut off contact with the attacker, IPSs can defend the host or network [14]. Furthermore, in [49], a technique for resolving the ARP attacks issue with snort IDS and static ARP entries is proposed. The person in charge must still manually put the static mappings, though. It is also limited to use on static IP network configuration environments.

2.4 Machine Learning and Deep Learning Algorithms

2.4.1 Machine Learning Algorithms

It is a branch of artificial intelligence that aims to enable computers to do jobs with understanding by using intelligent programs [50]. Statistical learning techniques are the basis of intelligent programs, which are used to generate AI for machines. The use of ML allows computer software to gain knowledge from a training dataset. In order to forecast or decide, algorithms utilize training data sets. A training dataset often contains information about prior occurrences. As a result, it is possible to consider the previous training data set's quality as a key determinant of the data's prediction. Classification issues have usually been addressed via ML. However, as ML techniques have advanced, they are currently utilized in the several applications, including recommendation systems, image recognition, data mining, and email spam filtering [51].

As seen in Figure 2.4 [50], ML approaches are divided into four categories based on the data they require as input: supervised, unsupervised, semi-supervised, and reinforcement learning. Each of them is described as below manner [51, 52, 53].

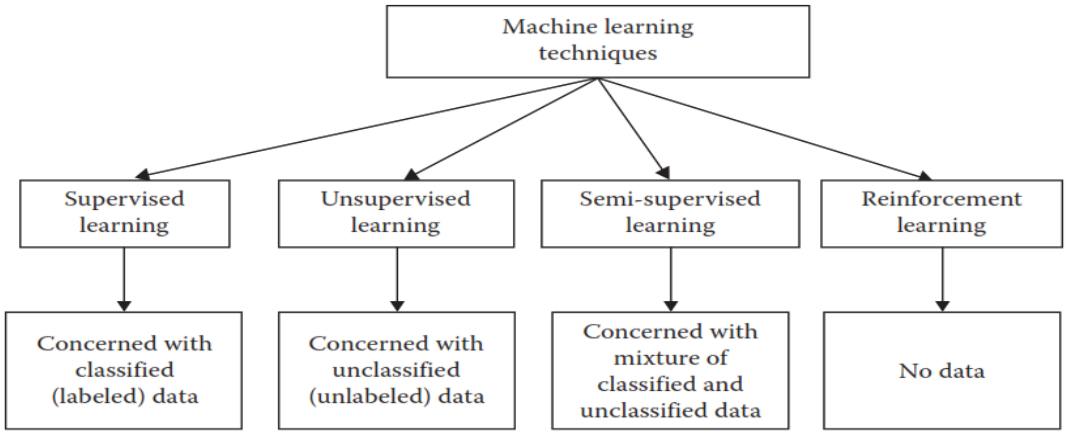


Figure 2.4 Different ML techniques and their required data

- **Supervised learning**: It is the process of deriving a function from labeled training information. A collection of training examples made up the training data. These sets of labeled examples are provided to the learner as training data, and then produce predictions for all invisible points. "**Regression**" and "**Classification**" problems are

typically at the heart of these concerns. Whereas the goal of a regression problem is to predict results within a continuous output, the goal of a classification challenge is to forecast outcomes inside a discrete output.

- **Unsupervised learning:** It is the process of inferring a function to describe a hidden structure from unlabeled data. The learner produces predictions for all unknown points using just unlabeled training data, and it enables us to approach issues without having a clear vision of the final product. "**Clustering"** and "**dimensionality reduction"** are example of this type of learning problems.

- **Semi-supervised Learning:** Unlike other, it makes use of both labeled and unlabeled data for problem solving. ML algorithms can learn to categorize unlabeled data by employing this hybrid technique.

- **Reinforcement learning**: It focuses on structured learning processes in which a set of actions, parameters, and end values are provided to a ML algorithm. Additionally, after developing the rules, it makes an effort to investigate many alternatives and possibilities, evaluating and assessing each result to determine which is optimal.

Numerous significant and commonly used machine learning algorithms are employed to address various issues, including Naive Bayes classifiers, K-means clustering, support vector machines, linear regression, decision trees, random forests, and artificial neural networks [53]. In this section, we select only the Artificial Neural Networks ML algorithm for discussion.

Artificial Neural Networks (ANN)

It is a type of computation that creates a number of interconnected computational units. The network is made up of a certain number of nodes, also known as neurons, which link the input set to the output. It is a component of a computer system that simulates the analysis and data-processing functions of the human brain [54]. As shown in Figure 2.5[55] an artificial neural network can be divided into three layers, named: Input layers, hidden layers and output layers [54].

1. **Input Layer:** This is in charge of accepting any input data or information from the external environment and fed to the algorithm.

2. **Hidden Layer:** Neurons in these layers are in charge of extracting patterns associated with the processor system under consideration. These layers take care of the majority of a network's internal operations.
3. **Output Layer:** The last layer of the neural network that produces the specified output classification result.

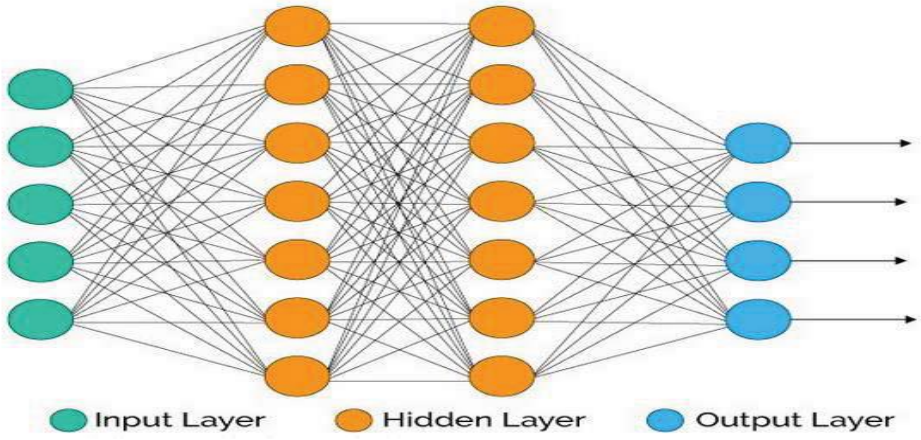


Figure 2.5 ANN architecture

A**, Activation Functions**

The ANN's neurons operate in accordance with their corresponding activation functions, weight, and bias [56]. As seen in Figure 2.6 [57], the activation function computes the weighted sum and then applies bias to it to decide whether or not to activate a neuron. This function's primary goal is to add nonlinearity to a neuron's output. Its primary duty is to transform the weighted input sum of the node or neuron into an output value that can be utilized as an output layer or supplied into the following hidden layer [57]. In a neural network, the weights and biases of the neurons would be adjusted based on the reconstruction error. We call this process backpropagation. Because they provide the gradients and errors needed to update the weights and biases, activation functions in this instance permit it [56]. Activation functions in neural networks come in a variety of forms. We attempt to explain the three that are most commonly used [57, 58].

Sigmoid activation function: It is frequently employed in models when the output must be a probability prediction. Sigmoid function is the best choice, because of its range and the probability of anything occurring only between ranges of 0 and 1. As

shown in equation (1) [57] is the sigmoid function and very common choice for feed-forward neural networks.

$$g(z) = \frac{1}{1+e^{-z}} \tag{1}$$

Where **z** is the input to the sigmoid function, **e** is Euler's number, which is a constant number.

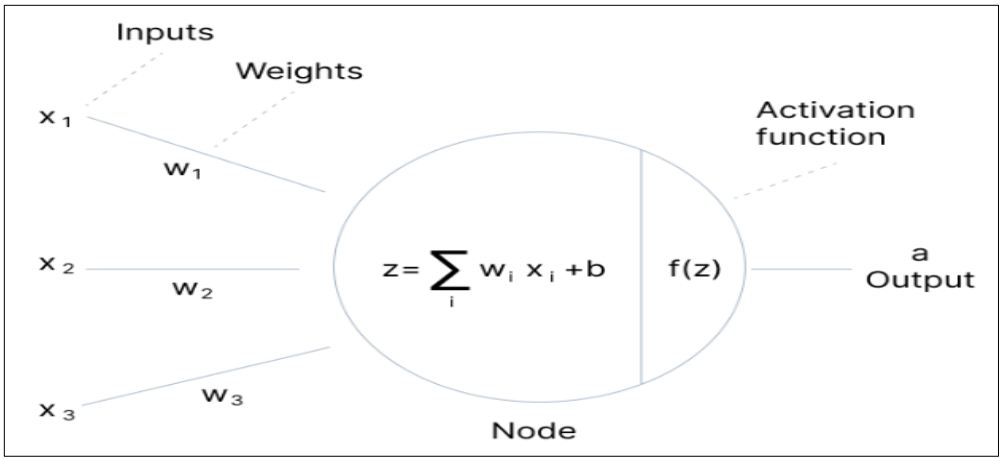


Figure 2.6 Activation function per node or neuron

Where, x_i=x_1, x_2, x_3, etc., is the input vector, w_i=w_1, w_2, w_3, etc. is the connection weight of each neuron, **b** is the bias term, which is an additional parameter that is added to the weighted sum of the inputs of a neuron to shift the activation function's output, enabling the network to learn complex functions and make accurate predictions, and **f (z)** is the activation function.

ReLU (Rectified Linear Unit) activation function: It is a linear function with a piecewise structure that, in the event that the input is positive, will output the input directly; if not, it will output zero. Compared to other non-linear activation functions, the model with many hidden layers can be trained much more quickly and easily, and it also aids in minimizing error gradient and state of vanishing problems [59]. It is mathematically written as follows in equation (2) [59], and it operate based on this formula.

$$f(x) = \max(0, x) \tag{2}$$

Where, x is the input vector, max is the maximum value.

SoftMax activation function: Most often, it is used as an activation function for the neural network's last layer in multi-class classifications. It computes the relative probability and is described as a combination of various sigmoid. For each class, it generates a probability-like prediction as shown in Equation (3)[56].

$$\text{SoftMax}(xi) = \frac{e^{xi}}{\sum_{j=1}^{N} e^{xj}} \quad (3)$$

Where, the variable x_i is an input value, e^{xi} is a function for input value, e^{xj} is a function for output value, and N is the number of classes in the multi-class classifier.

B**, Feedforward and Backpropagation**

When studying neural networks, two terms that describe information transmission are called feedforward and backpropagation [57] [60].

- **Feedforward Propagation**: The data is forwarded, and the input is used to build an intermediary function in the hidden layer that then computes an output. In feedforward propagation, the Activation Function serves as a logical gate between the input that enters the present neuron and the outcome that moves on to the subsequent layer.
- **Backpropagation**: It is a technique for adjusting the network's biases and weights to reduce training errors. The weights of the connections are continually altered in order to reduce the difference between the generated output vector and the actual output vector of the network. It modifies the network's weights and biases in an attempt to reduce the cost function.

C, **Loss Functions**

An expression or mathematical function known as a loss function is used to direct the model training process towards accurate predictions. The training aims to lower the variance by measuring the discrepancy between the models predicted and expected outputs. Researchers built different loss functions for different problems to ensure stable gradient flow during training. We try to describe some of them in the next section [61] [62].

- **L1 loss function (Mean Absolute Error Loss):** To calculate the Mean Absolute Error (MAE), it adds up all the values returned from each absolute difference computation and takes the average of this sum value. The measurement is the mean of the absolute variations between the estimated and real values. As shown in equation (4) [62] MAE loss can define as

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |xi - \hat{x}\, i| \quad (4)$$

Where xi and $\hat{x}i$ are the real and anticipated values of the i^{th} sample respectively, and n is the number of samples.

- **Mean Squared Error (MSE):** The mean of the squared variances among the real and anticipated values is what's measured. The MSE loss function can be defined mathematically in equation (5) [62].

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (xi - \hat{x}\, i)^2 \quad (5)$$

Where xi and $\hat{x}i$ are the real and anticipated values of the i^{th} sample respectively, and n is the number of samples.

- **Categorical Cross Entropy (CCE):** For multi-class classification issues, this function also referred to as multi-class log loss or negative log-likelihood loss is used. It is described as being the average negative log-likelihood of the true class given the expected probability distribution. The formula for categorical cross-entropy loss is expressed in equation (6) [62].

$$CCE = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} yi,j \log(pi,j) \quad (6)$$

Where N is the number of samples, C is the number of categories, yi is the real tag, and p is the anticipated possibility of the actual class.

- **KL divergence loss:** The Kullback-Leibler divergence loss, as it is also referred to, computes the discrepancy among the anticipated possibility distribution and the real value possibility distribution of the classes [63]. It makes sure that the predicted probabilities are very near to the actual probabilities, which can be helpful when the actual probabilities are known or can be approximated. The KL divergence loss function can be defined mathematically in equation (7) [62].

$$KL\ (p\|q) = \sum_{i=1}^{n} p(xi)\ \log(\frac{p(xi)}{q(xi)}) \quad (7)$$

Where p(xi) is the actual probability of class xi and q(xi) is the predicted probability of Class xi.

2.4.2 Deep Learning Algorithms

Machine learning has a subfield known as deep learning (DL) that moves machine learning closer to artificial intelligence. It enables the use of different levels of representation to represent complex relationships and ideas [64]. DL is a collection of supervised or unsupervised nonlinear techniques for multi-layered models. Deep learning is an essential element of the data science field, which also includes mathematics and forecasting. It simplifies and accelerates this process, which is highly beneficial to data scientists who handle the collection, processing, and interpretation of enormous volumes of data. It is currently being used in all kinds of big data analytics programs, including those that focus on Natural Language Processing (NLP), medical diagnosis, stock market trading signals, network security, and picture identification [65]. Deep learning methods require large amounts of data and computing capacity to address complicated issues [65].

To address various issues, many potent deep-learning algorithms are used. However, we restricted the scope of our discussion to the two powerful deep learning algorithms, the Autoencoder and Convolutional Neural Network, which are applied to our suggested methodology.

1, Autoencoder

Unsupervised learning data encoding is accomplished through the use of an artificial neural network known as an Autoencoder. An autoencoder's job is to teach the network to extract the most important components of the input data in order to learn an encoding, or lower-dimensional representation, for higher-dimensional data often for the purpose of decreasing dimensionality [66]. However, to duplicate input data and condense it, self-supervised machine learning models known as autoencoders are used. Since these algorithms are trained as supervised machine learning algorithms that function as unsupervised approaches during decision-making, they are known as self-supervised models [67].

The architecture of a single layer Autoencoder is shown in Figure 2.7 [68] below, and it has three important layers that are discussed in the next section [66, 67, 69].

- **Encoder:** The given input data has been reduced and represented in latent space. It transforms the input data into a compressed form that is significantly smaller than the original data and has a decreased dimension.
- **Bottleneck:** A component that contains the condensed cognitive representations, making it the most important part of the network. Another name for it is latent space representation.
- **Decoder:** This component helps the network reconstruct the data from its condensed form by decompressing learning representations. Next, the output is assessed using the actual data.

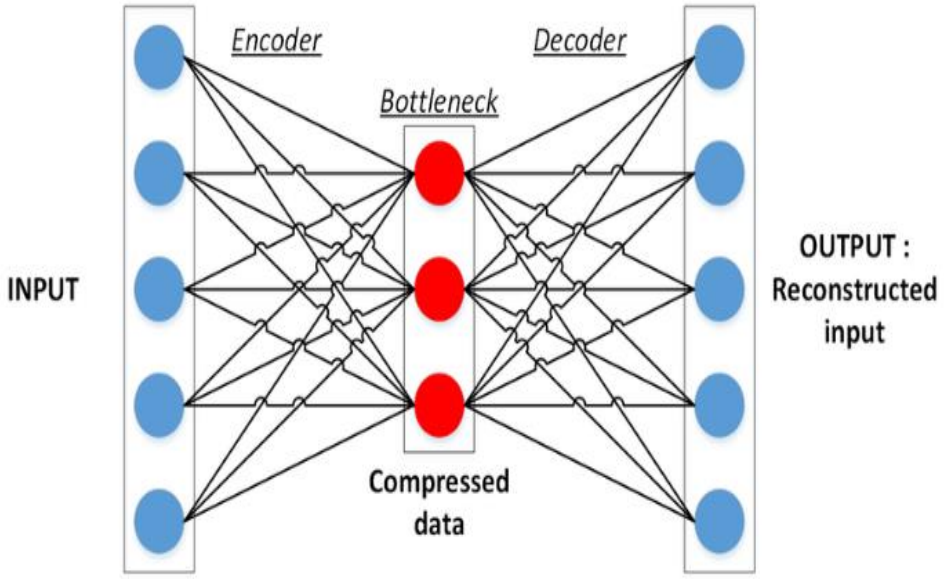


Figure 2.7 Architecture of single layer Autoencoder

Types of Autoencoders

There are five types of popular autoencoders, and each of them is discussed in the following sections [66] [69].

I. **Undercomplete Autoencoder**

An unsupervised neural network called undercomplete autoencoders can be used to create a compressed version of the input data. By using an image as input and trying to forecast the same picture as outcome, the picture is then rebuilt from the condensed latent space region. These types of autoencoders are mostly used to create a latent space representation, which serves as a condensed version of the input data and is rapidly and freely decompressed with the assistance of a network when required. This type of data condensation can be treated as a **dimensionality reduction** technique.

II. **Sparse Autoencoders**

Sparse Autoencoders operate by regularizing and penalizing the activation of a certain number of neurons in hidden layers. The sparsity function in this situation prevents extra neurons from activating, regularizing the neural network. The Autoencoder will still look for interesting structures in the data if we place a sparsity constraint on the hidden units, even if there are many hidden units [66]. Due to this capability, sparse Autoencoder is mainly used for dimensionality reduction and feature extraction purpose. Thus, the sparse Autoencoder can extract meaningful and efficient latent space representation features for the classification process by learning valuable reduced-size characteristics from the provided data. It involves two steps, which is encoder and decoder for feature extraction and dimensionality reduction process as shown in Figure 2.8 [66] the architecture of sparse Autoencoder. In the encoding stage, the input data xi is translated into the representations of the hidden units, which are then subjected to the nonlinear activation function as displayed in equation (8).

$$H = f\,(WX + b1) \tag{8}$$

Then, the decoding sage translates the hidden units' representations into the recreated data, which is output data that processed by output vector activation function as shown in equation (9).

$$Z = g\,(VH + b2) \tag{9}$$

In the above two equations, where $X = (x_1; x_2; x_3,....; x_i)$ is the large size input data vector, $H = (h_1, h_2, h_3,....,h_k)$ is the reduced small size vector, which is the latent space representation or bottleneck layer, $Z = (\hat{x}_1, \hat{x}_2, \hat{x}_3 \ldots \hat{x}_i)$ is the reconstruction vector of the input data, which is the output layer.

Where, $W= (w_1, w_2, w_3, \ldots, w_n)$ and $V= (v_1, v_2, v_3,...,v_n)$ are the weight matrix of the hidden layers and output layer, respectively. And also, **b1** and **b2** are bias vector of the hidden layer and output

layer, respectively, and **n** is the summed of weights or network connections for all hidden neurons from all input neurons and the summed of weights or network connections for all output neurons from all hidden neurons respectively. Additionally, ***f*** and ***g*** denote the nonlinear activation function of the encoder and decoder respectively, which is used to convert the input data into the output data through nonlinear transformation.

Sparse autoencoder performs backpropagation techniques to achieve the greatest values for its weight matrices W and V, and bias vectors b1 and b2, which try to acquire and recreate its output values $\hat{x}_i$ to be equal to its inputs x_i, and also to minimize the loss function or reconstruction loss function [70].

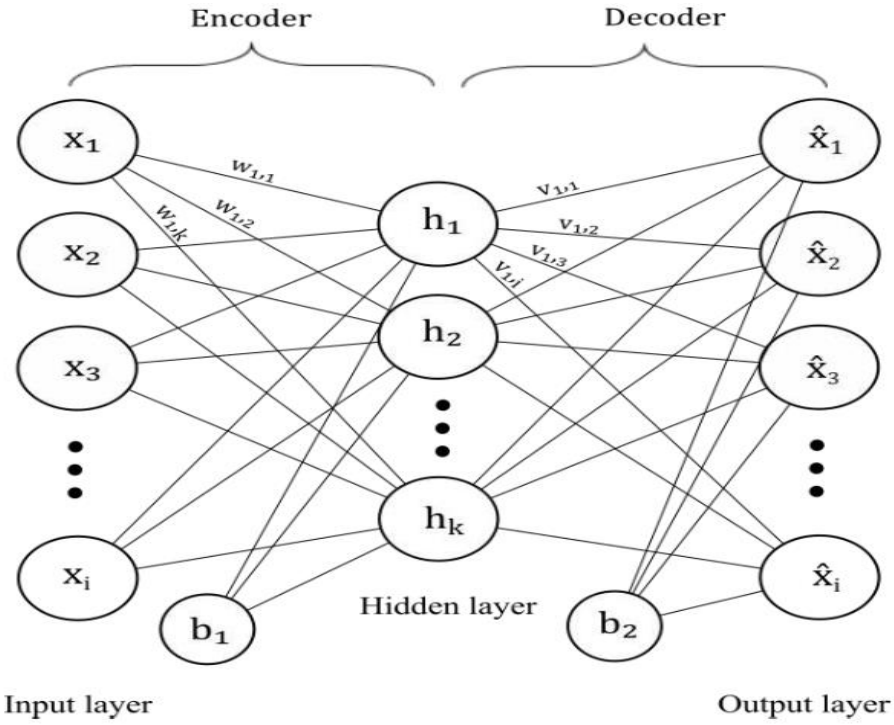


Figure 2.8 The Sparse Autoencoder architecture

To regularizing the neural network, sparse Autoencoder used sparsity function to stops or penalize additional neurons from activation by applying one of the two regularize methods in to loss function.

- **L1 Loss**: Method as a broad regularization, we can apply to increase the model's performance by adding l1 loss to the MSE loss function, which is used as a normal

reconstruction error function. Mathematically this function is defined in equation (10).

$$L=MSE+L1=\frac{1}{n}\sum_{i=1}^{n}(xi-\hat{x}i)^2+\frac{1}{n}\sum_{i=1}^{n}|xi-\hat{x}i| \quad (10)$$

Where L is the total loss of reconstruction error of output from its input value.

- **KL-divergence:** Unlike the L1 loss method, which simply adds up the activations across all samples, this approach takes all of them into account simultaneously. We set limits on the average activity of each neuron throughout this group. It expresses as in equation (11).

$$L=MSE+KL(p\|q)=\frac{1}{n}\sum_{i=1}^{n}(xi-\hat{x}i)^2+\sum_{i=1}^{n}p(xi)\log(\frac{p(xi)}{q(xi)} \quad (11)$$

III. Contractive Autoencoders

A contractive autoencoder passes the input through a bottleneck before reconstructing it in the decoder. While the data is being processed, the bottleneck function is being utilized to learn a representation of the input data. In order to stop the network from learning the identity function and turning input into output, it additionally incorporates a regularization term. We must make sure that the derivatives of the hidden layer activations are minimal in comparison to the input in order to train a model that conforms to this threshold.

IV. Denoising Autoencoders

Similar to standard autoencoders, denoising autoencoders accept input and produce output it. In fact, that denoising autoencoders substitute a noisy image as input and don't take it as actual value. Therefore, denoising autoencoders are able to reproduce the clear image output from the noisy image input through the encoder and decoder data processing step.

V. Variational Autoencoders

Models known as variational autoencoders (VAE) are designed to solve a particular issue with traditional autoencoders. During training, an autoencoder discovers how to express the input in a condensed format called the latent space. However, this post-training latent space is not necessarily continuous, making it difficult to interpolate. The latent properties of variational autoencoders, which focus on this particular subject, are expressed as probability distributions, creating a continuous latent space that is easy to sample and estimate.

2. Convolutional Neural Network

CNN is one of the most popular supervised DL algorithms that used to solve classification problems. It is mostly used for object detection and image processing, and it has several layers [71]. This kind of feed-forward ANN can recognize complex objects and patterns since it has millions of parameters, multiple hidden layers, an input layer, an output layer, and other features. The CNN's detection of image architecture is shown in Figure 2.9 [72] below. Prior to employing an activation function, the input size is reduced using the pooling and convolution processes. The output layer is the outcome of the last fully linked layer that results from these hidden layers, which are only slightly connected. The original image's size and its final shape are similar.

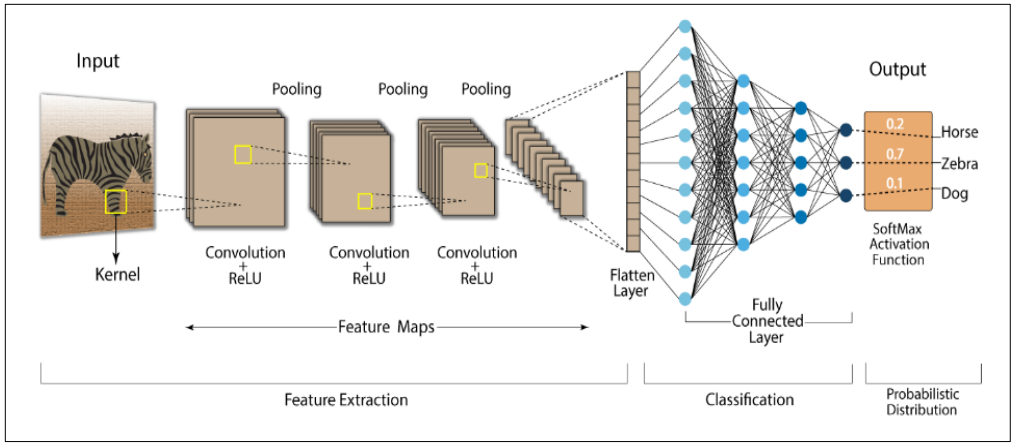


Figure 2.9 Architecture of CNN to identify the input images

CNN contains a number of hidden layers that aid in information extraction from the input data. The following are the five crucial CNN layers [71, 72]

A, **Convolution layer**

CNN is composed of convolutional layers. Input variables like images, filters like a feature detector, and result variables like a feature map are typically included in this layer. The input data is abstracted to a feature map also known as an activation map -after passing through a convolutional layer, as indicated by the mathematical explanation below.

Feature Map = Input data x Feature Detector

The input is condensed by this layer before the result is sent to the next layer. Each multilayer neuron is assigned a certain responsive field, and only that data is processed. A convolution is a grouping function in mathematics. It occurs in CNNs when two matrices are rectangular arrays of numbers arranged in columns and rows are combined to generate a third matrix.

B, **Rectified Linear Unit (ReLU) layer**

In CNN, the ReLU function has become the standard activation function for many different kinds of neural networks since a model that uses it to trains more quickly and performs better overall. Figure 2.10 [72], shows ReLU function in CNN.

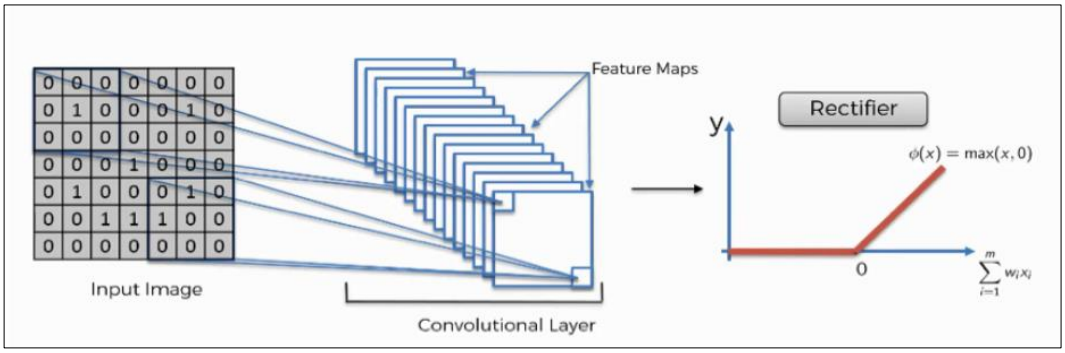


Figure 2.10 ReLU function in CNN

C, **Pooling layer**

The down sampling process of pooling lowers the feature map's dimensionality. The corrected feature map is processed through a pooling layer to create a pooled feature map, as shown in Figure 2.11 [71] below. In order to reduce the number of variables and computations in the network, its objective is to progressively reduce the spatial aspect of the representation. The pooling layer treats each feature map independently.

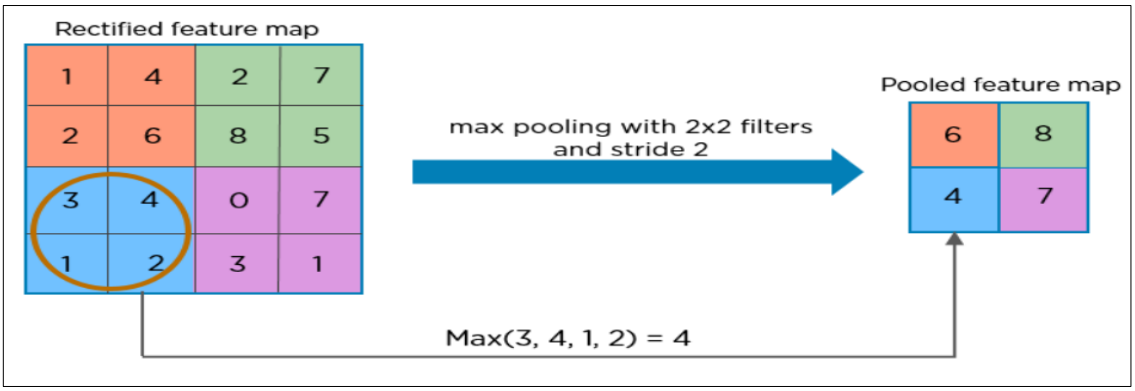


Figure 2.11 Process of Max pooling

Flattening is employed to create a single, long, continuous linear vector from all of the 2-dimensional arrays, from pooled feature maps, as shown in Figure 2.12 [71]. The fully connected layer receives the flattened matrix as input in order to classify the data.

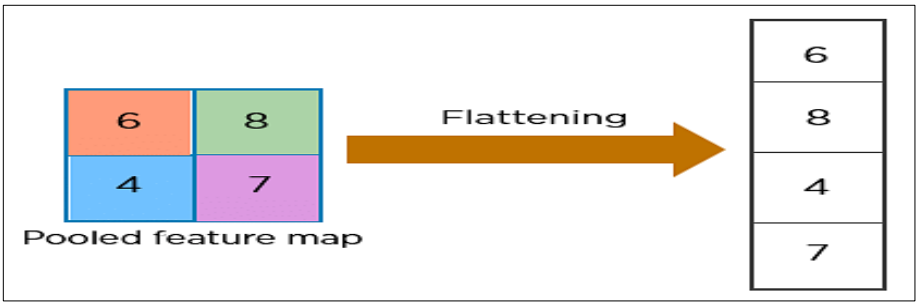


Figure 2.12 Flattening process in CNN

D, **Fully Connected Layer**

Neurons in a completely linked layer function similarly to conventional neural networks in that they have complete connections to all activations in the layer before. This layer, which groups and tags the output, develops when the flattened matrix from the pooling layer is provided as an input [70, 71].

E, **Output Layer**

Last but not least, soft-max activation function serves as a classification algorithm and is frequently used in conjunction with an activation function layer. The task of categorizing the input into multiple types falls to this layer [71].

2.5 Dataset

There are a number of datasets used for intrusion detection systems; some of them are self-created and used for private purposes, while others are created in different laboratories and universities and are publicly available. Some publicly available datasets are [21]: KDD 99, NSL-KDD, CIC-IDS2017, and like. These datasets are used to train and evaluate machine-learning IDS models. From these datasets, we select and discuss only NSL-KDD, which is an open-source benchmark dataset that is heavily used for intrusion detection systems in many scholarly research papers. The University of New Brunswick's KDD'99 has been updated and refined into the NSL-KDD [21]. This dataset includes training and testing sets and has forty-one features and one class attribute.

Compared to the original KDD data collection, the NSL-KDD data set offers the following benefits [73].

- The train set does not contain repeated entries, preventing the classifiers from preferring more frequent records.
- The suggested test sets contain no repeated information; therefore, the achievement of the learner's is unaffected and come up with higher detection rates for common records.
- Because there are a sufficient number of records in both the train and test sets, it is feasible to conduct the tests on the entire set without limiting the sample size. As a result, evaluation findings from various research projects will be consistent and equivalent.

There are four attack classes (categories) and one normal class in the NSL-KDD dataset. These four types of attacks classes are defined in the following manner:

A. **Denial-of-service (DoS) attacks**: It occurs when a malevolent actor attempts to stop the intended users of a device, such as a computer, from utilizing it by interfering with its normal operation. In order to deny service to more users, this attack often involves overloading a targeted computer with requests until regular traffic is unable to be handled.

B. **Probe:** In order to launch assaults on the intended system later on, this attack gathers information about possible weaknesses of the system that is targeted.

C. **Remote to Local (R2L):** the capacity to transmit data packets over a network to a remote system, whereupon one can obtain illegal access as an end-user or a root account and carry out criminal activity.

D. **User to Root (U2R):** By using a regular user account to access the system, these attackers exploit security holes to obtain administrative powers. These attacks first use a dictionary attack or credential sniffing to get access to a regular user. Once it has managed to obtain the credentials of a valid user, it tries to take advantage of a system vulnerability to obtain super user access.

2.6 Evaluation Metrics

The effectiveness of a classifier can be evaluated using a variety of metrics. The following evaluation metrics are often used in binary and multiclass classification techniques like confusion matrix, accuracy, error rate, precision, recall, F-1 score, and false positive rate or false alarm rate [74].

❖ **Confusion matrix**

A confusion matrix is a square matrix where the rows represent the actual classes of the examples and the columns represent the expected classes of the instances. It is a standard metrics to evaluate the intrusion detection model. The confusion matrix for multiclass classification consists of a number of rows and columns equal to the total number of distinct classes involved in the classification task. In our study, the confusion matrix is a 5x5 matrix, or a classification with five classes as shown in Table 2.3.

Where the following definitions apply to the denotations True Negative (TN), True Positive (TP), False Positive (FP), and False Negative (FN):

- True Positive (TP): The quantity of malicious records accurately detected.
- True Negative (TN): The quantity of normal records that were appropriately categorized as such.
- False Positive (FP): The number of records that are mistakenly classified as attacks when, in fact, they are just routine activity.
- False Negative (FN): The numbers of records that are malevolent yet are mistakenly categorized as regular actions.

Table 2.3 Confusion matrix for five classes' classification evaluation

	Predicted Class					
Actual Class		Normal	Dos	Probe	R2L	U2R
	Normal	TN	FP	FP	FP	FP
	Dos	FN	TP	FP	FP	FP
	Probe	FN	FP	TP	FP	FP
	R2L	FN	FP	FP	TP	FP
	U2R	FN	FP	FP	FP	TP

The TP, TN, FP, and FN of five classes are represented by the confusion matrix in Table 5.4. Except for the TN, the total of the values of the associated rows will determine the total number of FN values for a class. The total of the numbers in the corresponding column represents the false-positive value for a class, with the exception of the TP value for each assault class. The diagonal value represents the total number of TPs for each class.

- **Accuracy**

How many data points are correctly forecasted is what is meant by accuracy. It is one of the most popular categories of evaluation metric. As stated in the equation below, it is equal to the number of accurate forecasts divided by the total number of predictions, multiplied by 100. In our instance, it illustrates how the classifier labels "attack" for abnormal network traffic data and "normal" for regular network traffic data.

$$\text{Accuracy} = \frac{TP+TN}{\text{TP+FP+TN+FN}} \text{ x } 100 \quad (12)$$

- **Error Rate**

The following formula allows us to calculate the Misclassification rate in this metric based on the accuracy of the classification:

$$\text{Error Rate} = \frac{FP+FN}{\text{TP+FP+TN+FN}} \text{ x } 100 \quad (13)$$

Or Error Rate = 1- Accuracy

❖ **Precision**

The ratio of the number of correctly identified positive classes to the total number of positive classes is used to compute precision. The precision measures the model accuracy in classifying the model as positive. Mathematically precision is represented as follows.

$$\text{Precision} = \frac{TP}{TP+FP} \text{x}100 \quad (14)$$

❖ **Recall**

This metric measure how many accurate positive predictions were made out of all possible positive forecasts. How accurate is the classifier in our context in labeling all instances of attacks as "attack"? Recall can also be referred to as sensitivity, detection rate, or true positive rate (TPR). The mathematical representation of recall is as follows.

$$\text{Recall} = \frac{TP}{TP+FN} \text{x}100 \quad (15)$$

❖ **F1-score**

It can be described as the harmonic mean of recall and precision. It rates the performance of the model statistically. This is how the F1-score was computed.

$$\text{F1- score} = \frac{2\text{XPrecisionXRecall}}{\text{Precision+Recall}} \quad (16)$$

❖ **False Alarm Rate** (FAR)

False alarm rate, or false positive rate (FPR), is a metric that quantifies the fraction of normal data that is mistakenly recognized as an attack relative to the total number of samples. The following is the mathematical representation of FPR.

$$\text{FPR or FAR} = \frac{FP}{FP+TN} \text{x}100 \quad (17)$$

2.7 Summary

Generally speaking, we tried to discuss and review different literatures on ARP attacks, and their detection and defense methods in detail, in the Literature Review Chapter. Starting from introducing ARP protocols and their role in layer two networks, how to be poisoned by attackers and types of attacks that occur in this protocol are discussed. We also reviewed about different types of tools that are used to perform ARP spoofing, and other tools that are

used to detect and defend ARP attacks. Additionally, different techniques and ways of ARP attack detection and prevention are discussed. Furthermore, Machine Learning and Deep Learning algorithms are explained in detail, especially in our proposed method area.

The literature that is revised in the initial sections of this research work happens to have a greater impact on the effort to know about ARP attacks' natures and previously done solutions. Similarly, literature written about machine learning in general and Autoencoder and convolutional neural network algorithms improved our level of understanding on domain specific topics.

Chapter 3: Related Work

There are various researchers' works that are more related to our research field, based on different detection techniques, discussed in this related work. The techniques for identifying and preventing ARP attacks are categorized into five main categories including: centralized-based techniques, cryptographic-based techniques, modified protocol-based techniques (state full ARP), packet header analyzing and routing truce-based techniques and ML and DP learning-based techniques are discussed.

3.1 Centralized-based Techniques

The research in [7] proposed, by relying on the static characteristic of the ARP protocol, an efficient method to avoid ARP Spoofing attacks in local area networks. Two computer programs (server-client) were developed where a reliable ARP table is created by the server application, and distributed that table to client computers on the subnet. The ARP table is then automatically updated by the client application in a static mode on the computer. Despite the attacker continuing to send false response messages while the suggested strategy was in operation, the system prevented the attacker from acquiring any information. The result showed the system was efficient to preventing ARP Spoofing attacks in an inexpensive manner. However, this system does not detect insider threats because any user who first registers as a legal user is then considered an authorized user in the system. Also, it is not time-effective system, because it takes much time to detect attacks, and needs more work effort for the network administrator to register all users in the system plus every time a new user requests network access should be added first to the system. This is not suitable for large-scale network areas.

The study [11], proposed a new static entry approach for the detection and defense against ARP spoofing, poisoning, and impersonation attacks. In this approach, the client and server model were prepared with IP-MAC table to detect and prevent ARP spoofing. This approach was deployed on a proxy server and used ARP to automatically collect IP-MAC and register IP-MAC, then finally, identify and block ARP attacks using defined algorithm. The findings of this solutions indicate that both detection and prevention systems were the best security strategies. However, this method was designed for local area networks with static IP address configurations only. It does not work for DHCP configuration networks.

Furthermore, a scheme established on the ARP central server (ACS) is proposed by Sudhakar et al. [75]. Researchers can effectively use the IP associating configuration by combining the approval server with the ARPWATCH apparatus in this manner. This approval server, which functions somewhat like an ACS server, authorizes the entries in all of the network's hosts' ARP tables. Every client should maintain a long-term cache database using this method. In this manner, they can fix the ARPWATCH's IP aliasing issue. The configured ARPWATCH and centralized server are going to filter the data produced by the attackers. As a result, the issue of ARP contamination can be resolved. In this approach, the ACS will be connected to an ARPWATCH system that will be set up to inspect the MAC-IP mapping with the ACS's secondary cache whenever a false alarm is generated. The false alert won't be generated if the associations for that MAC-IP are already in the secondary cache. The ARPWATCH will listen to any system port at any time. If there is a problem with the IP connection between Macs and computers. Since the recommendation server has its own supplementary ARP backup records, it is going to reach it first when that occurs. The caution won't be generated if Macintosh's IP partner has a spot in that ARP cache database; otherwise, it will. To the secondary ARP cache database, ACS will transmit a reply message. In order to designate the user as a genuine user for additional upcoming infrastructures and to update its secondary ARP table. The problem with this work is that because it contains false warnings, it generates fictitious reports for administrators. Therefore, the administrator is disturbed by these numerous false alarms and they need to be managed. Additionally, the architecture of this technique is complicated and time consuming. This is because; the architecture of the proposed technique involves multiple components, such as the ARP central server (ACS), approval server, ARPWATCH apparatus, and secondary ARP cache database. The interactions and dependencies between these components may be intricate and require careful configuration and coordination. Additionally, setting up (configuring) this technique may involve various configuration steps and parameters. Each component needs to be properly configured and integrated into the network infrastructure. This configuration process can be time-consuming, especially when considering the need for coordinating and synchronizing the ACS, approval server, and ARPWATCH apparatus.

The research in [12], suggested a utility system that consists of techniques for authentication, detection, and prevention to stop all attacks in the dynamic IP configuration environment. The System has 3 components including: DHCP IP configuration using DHCP server, authentication of the user using radius server + MySQL database, and detection and prevention of ARP poisoning components. The utility system is implemented first and authenticates the username and password to the joining request, then by inspecting the IP-MAC cache among the primary and secondary IP-MAC cache by using ICMP protocol, and lastly, if there is no match, detect the attack and prevent access. This approach offers a safe and effective plan. However, it does not detect host impersonation attacks. Furthermore, in order to prevent ARP assaults, these methods necessitate explicit setups by the network administrator.

Detecting and preventing ARP poisoning by a centralized method was suggested by Sumit Kumar and Shashikala Tapaswi [76]. In this approach, an ARP Central Server (ACS) verifies each host's entry in the network's ARP tables. In this method, clients additionally keep a second long-term cache. This method, however, does not deal with the IP exhaustion issue that an attacker may bring about inside a network. Also, this method has a single point of failure and is centralized in nature.

According to the proposed detection mechanism in [77], the Central Server (CS) must first sniff network traffic. After sending a setup ICMP ping packet and examining the response in terms of an ICMP reply, CS efficiently identifies the attacker. Genuine CS is chosen through voting to prevent ARP poisoning on a centralized system. By checking and fixing IP-MAC pair entries that are present in host cache tables, CS effectively prevents ARP poisoning while maintaining system performance. This method, which is based on ICMP and voting, is recommended to identify and stop MITM-based ARP poisoning, an efficient way to overcome ARP's limitations. Backward compatibility, less cost, and ease of deployment are the main advantages of this approach. However, the host for which the static entry is not saved on the central server does not offer the MITM solution. Furthermore, it adds to the single point of failure issue if the central server crashed.

This section discusses various researches for implementing a central server that stores each host's IP-MAC binding and verifies each client in accordance with it. It is an easy solution to

implement, and there is no need for further network structure change. However, these approaches require much effort for network administrators to register new clients and require updates each time a host changes node. It is not a time-effective approach, because it takes much time to detect attacks, and have overhead to validate each host from a central server. Furthermore, this approach may have a single point of failure if the central server crashed.

3.2 Cryptographic-based Techniques

Ahmed et al.'s [78] method for identifying and treating ARP poisonousness is proposed. This proposed solution consists of three modules: the purpose of module one is to first grant authorization and store data in the record. As a protection safeguard, an MD5 hash is employed. To prevent internal ARP, use module two. Module three is used to determine whether an IP has two MACs or vice versa. This method contains a database with excellent features and support for storing data from ARP tables. To confirm that deviations in the network are taken into consideration, ARP table entries typically expire after a brief period of time. The effectiveness of the suggested mechanism was tested in experiments using Ettercap in a real-world network setting. The findings of the experiments demonstrate that the suggested method was successful in identifying and treating ARP poisoning. In particular whether an IP has two MACs or vice versa. The limitation of this method is, there is high communication overhead due to the use of an MD5 hash encryption.

To protect against ARP cache poisoning, Peta K. and Pulluru L.[79] suggest a novel technique that encrypts the MAC address. This technique aims to stop MITM attacks, particularly ARP poisoning, by using a switch as a third party to encrypt the host's MAC address and prevent interference from the perpetrator. The drawback of this method is communication overhead due to each host's MAC address encryption.

The first Secure-ARP (S-ARP) protocol was introduced by D. Bruchi et al. [80]. It uses public and private keys for verifying each ARP communication, as well as key distribution. The Authoritative Key Distributor (AKD), a reliable third party, distributes these keys by serving as a central server and distributing the public keys to many hosts. This technique substituted Secure-DHCP for standard DHCP and employed digital certificates to

authenticate each ARP reply within the network. The fact that the AKD is a single point of failure in the network is a disadvantage of this approach. Furthermore, this approach is difficult to integrate into the current ARP and lacks backward compatibility with the network infrastructure.

By allocating the centralized protected IP address and MAC address mapping proof, the ticket-based address resolution protocol (T-ARP) in [81] claims that it can prevent ARP spoofing. When a client approaches a network, a ticket, which serves as proof, is given to them. Unlike other public key techniques, this protocol can decrease the computational resources of S-ARP using an extra entity called Local Ticket Agent (LTA) and is backward compatible with ARP. This ticket's allocation is a duty of LTA. Nevertheless, it needs an independent entity known as a ticket agent.

In general, cryptographic methods are a strong solution to identify and prevent ARP attacks comparing with others mechanisms. The ARP protocol can be kept secure with the help of this method. However, there is a backward compatibility problem with the current network infrastructure, and the encryption and decryption processes require a lot of computational resources. More on this, Key management is a problem in cryptographic solutions, and the key-distributor server may be a single point of failure.

3.3 Modified Protocol-based Techniques (State full ARP)

The research in [82] tackles issues with scalability, virtual machine (VM) migration, and insecure communication that layer 2 protocols face. The suggested approach makes ARP state-based in order to avoid ARP cache poisoning attacks such as host impersonation, MITM and distributed denial of service. It also introduces two new types of communications alongside classic ARP. The parts of the suggested approach are to first decode the packets, update the incorrect entry the user made using the timestamp feature, and then introduce messages. The method has been tested and is capable of preventing broadcast storms and mitigating all ARP cache poisoning assaults without affecting virtualization. The drawback of this approach is that it does not detect ARP Scanning and not compatible with current standard ARP protocol and network infrastructure.

In the paper [83], an ARP enhancement (by making stateless ARP stateful ARP) is suggested to stop ARP spoofing attacks. The enhanced stateful ARP is to discard any unregistered reply to defend against MITM attacks and to send a request based on how many fake replies are received to stop DoS attacks. The proposed ARP enhancement is implemented using the Linux kernel and Ubuntu operating system. The outcome demonstrates that the proposed ARP improvement stops the ARP poisoning attack and, as a result, can withstand MITM attacks and denial-of-service attacks. The proposed ARP is faster because it uses fewer calls to update the ARP database during the attack. However, if the memory request becomes frequent, it lowers the system's performance.

A prevention method based on the use of a stateful ARP cache was suggested by the writers in [84]. To distinguish between legitimate and malicious ARP replies, this strategy employed a cutting-edge fuzzy logic technique. A constantly updated database used by the fuzzy logic controller adjusts to network changes. A stateful ARP cache entry with the status "Waiting" is generated when host A generates an ARP request to obtain host B's MAC address. For instance, an ARP reply is awaited by Host A within a predetermined delay. If an ARP reply is received, host A must wait for another timeout in order to gather any additional ARP replies that might have been sent by other sites connected to the network. It should be noted that if host A receives multiple ARP replies, it is probable that multiple hosts have responded. Therefore, only host B is an honorable host out of the group. The other hosts are most likely malicious ones who are attempting to poison host A's ARP cache. The limitation of this method is that it is not compatible with the original stateless ARP, means that it needs network infrastructure and ARP frame structure change. Because, the introduced stateful ARP cache method cannot be seamlessly integrated with existing systems or networks that rely on the original stateless nature of the ARP protocol. It requires modifications and updates to the ARP functionality to accommodate the stateful approach.

Making the ARP stateful protocol by registering and adding a time stamp for each ARP request made, which aids in identifying the attacker's reply messages, is a simple and better technique to recognize and stop ARP poisoning attempts. However, this type of stateful ARP has drawbacks since it requires modifications to the original ARP and is incompatible

with stateless ARP and existing network infrastructure. Because it needs network infrastructure and ARP frame structure change.

3.4 Packet Header Analyzing and Routing Trace-based Techniques

Time Stamp and Counter Based Approach (TSCBA), created by Prabadevi and Jeyanthi [8], is a method for identifying ARP cache poisoning attacks. For pre-processing and inspection of Ethernet header and ARP header MAC addresses, TSCBA makes use of a packet analyzer and a cross-layer checker. The ARP table is then used to verify these addresses for anomalies, and a time stamp is generated along with alert messages to be broadcast throughout the system. This detection procedure is very precise and secure. The main drawback of adopting TSCBA is high computational cost and is not time effective solution, because it takes much time to detect attacks.

To eliminate ARP spoofing attacks in networks and improve security, Moon et al, [85] suggested a Routing Trace-based Network Security System called RTNSS. The RTNSS system installs an agent on users' computers and keeps track of any alterations to the ARP cache tables. The anomalous hosts are identified based on these changes, and the routing traces are examined for any deviation from the accepted list of IP-MAC addresses. Thus, detected attacker hosts are removed from the routes and this approach does not require structural changes or traffic-increased encryption for security against ARP attacks. However, it has the limitations when it comes to detecting newly developed attack strategies as well as previously deployed ones.

The researchers of [86] suggest using gratuitous ARP request packets to combat the issue of ARP poisoning. They propose a system called the Gratuitous Decision Packet System (GDPS), which aims to accomplish two primary objectives: the first is the real-time detection of suspicious ARP packets through the use of received ARP packet analysis. The second involves transmitting a modified request message in place of the gratuitous ARP packets to distinguish between a trustworthy and malicious host. The tests also demonstrate the efficiency and accuracy of the proposed design, as well as the absence of any additional software or hardware requirements. The drawback of this technique is that there is a

communication overhead because of the broadcast of ARP request packets by the gratuitous packet algorithm.

In this section, different researchers propose a solution to detect and prevent ARP attacks using ARP header and Ethernet header packet analysis and routing trace-based techniques. These approaches are more effective and have better detection and prevention mechanisms compared to the previous related papers. However, these approaches are not time-effective solutions [8] and have limitations in their ability to detect all types of ARP attacks [85], and there is a communication overhead issue.

As discussed above, all methods are not effective solutions for ARP attack detection and prevention because most of the solutions require human involvement to manage and update the system. Also, there is a single point of failure due to a central server crash, low attack detection accuracy, high false positive and false negative rates, and compatibility issues with existing network infrastructure. Today, there is a greater need for more intelligent intrusion detection because a skilled attacker can get beyond these defenses [87]. In this field of cyber security, researchers are seeking to use machine learning approaches. Therefore, to tackle the above issue, and detect and defend against ARP attacks in local area networks; researchers have found and proposed other solutions that are more effective and intelligent through ML and DL techniques.

3.5 ML and DP Learning-based Techniques

In contrast to the earlier techniques, ML techniques do not require human assistance in the attack detection processes. Because it is able to detect attacks automatically without any human involvement once the algorithm is sufficiently trained with training input data. ML algorithm can learn to recognize trends and patterns from historical data to assess potential outcomes. The algorithm may replicate the process using current data and forecast the future using this as its base framework. This approach can lead to higher detection rates, decreased false alarm rates, and appropriate computing and communication costs in intrusion detection strategies. As a result, numerous researchers have suggested using machine learning approaches to improve ARP attack detection techniques.

In the study [13], a defense against ARP spoofing attacks based on Bayesian Support Vector Regression (BSVR-ARP) is developed. Based on previous information, this mechanism uses a probability prediction algorithm to detect the presence of an attack while taking into account host configuration changes and network transmission errors. This includes the creation of attack verification and recovery based on SDN controllers and BSVR-based attack probability prediction. Additionally, the routing trace and ARP cache table information were used in this method. The suggested BSVR-ARP provides better performance when it comes to identifying ARP attackers with increased accuracy of 84 %, and less time complexity when compared to previous techniques. However, because SVM is a traditional linear machine learning algorithm, the classification accuracy decreases as the input data size increases and also it was designed only for SDN technology.

On the other hand, a Bayes-based ARP attack detection algorithm created by the authors of [88] was implemented in the cloud center. Through the use of both normal and abnormal ARP features, they determine the likelihood that a host is an attacker. The host is labeled as an attacker if the probability number exceeds a predetermined threshold; otherwise, it is normal. The drawback of this detection algorithm is that it could not correctly identify an attack when it happens less frequently and when later features are added to normal host features, means that there is false negative results [13].

However, attack detection techniques using machine learning methods are subject to certain restrictions, like when the volume of the input data has increased the detection accuracy is decreased, due to scalability challenges, overfitting problem and increased complexity, the ability to represent and interpret features is limited, and the need for additional effort on feature engineering, means that it could not extract features automatically from input data like Autoencoder deep learning algorithm.

These days, technology is rapidly becoming more sophisticated and intelligent to solve complex problems through DL algorithms. In the realm of ML, it has been the most widely used statistical strategy. It has produced remarkable results on a range of challenging cognitive tasks, matching or even exceeding human performance. It is the most effective self-training algorithm and has the ability to learn large volumes of data and produce output. Because DL algorithms have more learning layers and a greater degree of abstraction than

regular ML algorithms, they are superior. The fact that all model elements directly benefit from data-driven learning is another factor contributing to this advantage [89].

The authors in [15] examined ARP traffic sequence data on a LAN based on the numerical count and degree of its packets. Additionally, a dynamic threshold is used to find hidden suspicious activity, which is then transformed into feature vectors and fed into an unsupervised Autoencoder neural network. Then, the authors use K-means clustering to divide the autoencoders latent features for suspicious activity into distinct patterns. In addition, researchers gather and use a real-world network traffic dataset from five distinct LANs to assess the performance. Finally, they were able to identify suspicious ARP patterns on the LANs that vary in size, duration, and regularity. The outcomes of this method demonstrate the optimal security measures for detecting ARP suspicious activity. But this method does not classify which ARP attack was detected in the experiment, but rather clusters suspicious ARP activity, and it needs further work to classify ARP attacks. Additionally, the unsupervised Autoencoder algorithm's nature requires a large amount of data in order to improve the accuracy of clustering or grouping. However, only about 28,318 rows of data were used by the authors, which is relatively insufficient to properly cluster suspicious ARP behaviors or have a high false alarm rate. We analyze the Authors' used a relatively small dataset size, which may not capture the full diversity and complexity of ARP traffic and potential attack scenarios. A larger dataset would provide more robust and reliable results, helping to ensure the method's effectiveness across various network environments. Also, the paper does not mention a comparison with other existing approaches or methods. Without such comparisons, it is challenging to evaluate whether the proposed method outperforms or offers advantages over alternative techniques.

According to the [14] study, the detection of ARP spoofing attacks using a convolutional neural network is proposed. And they use the routing trace method for determining network route changes. This neural network (CNN) can be trained to analyze TCP, UDP, and ARP traffic in the network and alert when ARP attacks are detected. A CNN is a powerful tool that they can use to process any kind of data where they can apply the convolution operation. In this paper, researchers evaluate the suggested CNN technique in MATLAB simulation software by using real-time network traffic data from the active host. Authors

took real-time network traffic from 100 normal hosts as a sample input data and generated 4 attacks for testing and evaluating their method. Then, after experimentation they compare the detection results with other previous papers (BVSR and RTNSS) in terms of detection time, attack detection accuracy, detection error, false detection probability, and packet drop rate. Based on this, the proposed method achieved better score like detection accuracy of 87% and other evaluation metrics except detection time (which has a medium value between BVSR and RTNSS). The limitation of this approach is that, first, the amount of input data is very small (take only 100 normal hosts traffic data and 4 attack samples are used) and not enough to train the CNN algorithm and attack classification, and this algorithm trained and able to detect only four attack samples. The second limitation is that, based on the experiment result of this approach, when the number of hosts is increased, the attack detection performance decreases in all evaluation metrics. Because, a small dataset size may not capture the full diversity and complexity of ARP traffic and potential attack scenarios. Additionally, the CNN algorithm needs a large dataset size to perform enough training for each data pattern and attack scenario, it should avoid overfitting and underfitting during the training to produce an effective method. Even if we select and use a good and powerful algorithm, there is overfitting and underfitting during the training, has poor detection and classification performance. This means the input dataset size and other parameters like hyperparameter selection play a great role in model building, training, and testing in deep learning algorithms to produce effective methods. So, the Authors used a powerful algorithm but there are weaknesses in their hyperparameter selection and dataset size for training and testing their proposed method. Therefore, further research and evaluation should be conducted to address this issue. So, we need to propose an effective solution to detect and classify ARP attacks.

3.6 Summary

In the above section, we try to discuss related works on traditional ARP attack detection and prevention mechanisms under five different categories. The first category is discussed in Section 3.1 which is various research papers that are used a centralized server to store the IP-MAC binding of each host and validate each client according to it. The second category is discussed in Section 3.2. We attempt to examine papers that use various cryptographic

methods to identify and prevent ARP attacks. In Section 3.3, we tried to explain different research works that modify stateless ARP to stateful ARP as a solution, which is the third category. Additionally, in Section 3.4, we discussed the fourth category, which tries to detect ARP attacks using ARP header and Ethernet header packet analysis and routing trace-based techniques. The last category is under Section 3.5 in two subcategories: first, different papers that use machine learning methods, and second, various papers that use deep learning methods.

As discussed in each section summary, all reviewed papers have different limitations, like not effective in terms of time and cost, having a single point of failure, having high communication overhead, needing more human involvement and effort, limited attack type detection, a high false alarm rate, and low detection accuracy. Therefore, we propose a hybrid deep learning-based ARP attack detection and classification technique that combines an Autoencoder for feature extraction and dimensionality reduction with a CNN for attack detection and classification algorithms in order to enhance detection accuracy and minimize false alarm rates.

Chapter 4: The Proposed Model Architecture

4.1 Overview

This Chapter presents the design of the hybrid deep learning-based ARP attack detection and classification model that has been proposed. The main objective of this work is to develop a model that makes it possible to identify and classify ARP attacks effectively. As a result, the architecture of the proposed model and the components that make up the model with the developed algorithms are presented in depth in this chapter. The proposed hybrid model consists of various components, which are described in detail in this chapter, along with their significance and the methods that should be applied when constructing them.

The hybrid ARP attack detection and classification method is built by combining two popular deep learning algorithms, which are the Autoencoder and CNN. To enhance the detection and classification performance of the model, it uses an unsupervised Autoencoder algorithm for feature extraction and dimensionality reduction and a supervised CNN for classifying the input data as attack or normal.

To design the model, the primary task is to find and select an appropriate dataset for model training and testing purposes. After selecting the dataset, the next step is the dataset preprocessing phase to make the data suitable for model training and testing later. So, the preprocessing task is performed in two phases, in the first phase, we perform data cleaning, data transformation and normalization. In the second phase, apply Autoencoder deep learning algorithms by using the first preprocessed dataset as input and the algorithm can reduce the dimension, extract the most significant feature of the input dataset, and finally produce this feature as output. Then, the model construction and training phases is started by taking the reduced output feature from the Autoencoder as input for the convolutional neural network classifier. Then, the CNN is well-trained using input data through a number of convolution layers and the ReLU activation function to perform classification.

Finally, the model should be evaluated by using testing dataset to determine whether it is able to correctly predict or not in better performance. The architecture of the proposed model is shown below in Figure 4.1. This architecture is adopted from the research in [90] and [91].

The Proposed Model Architecture

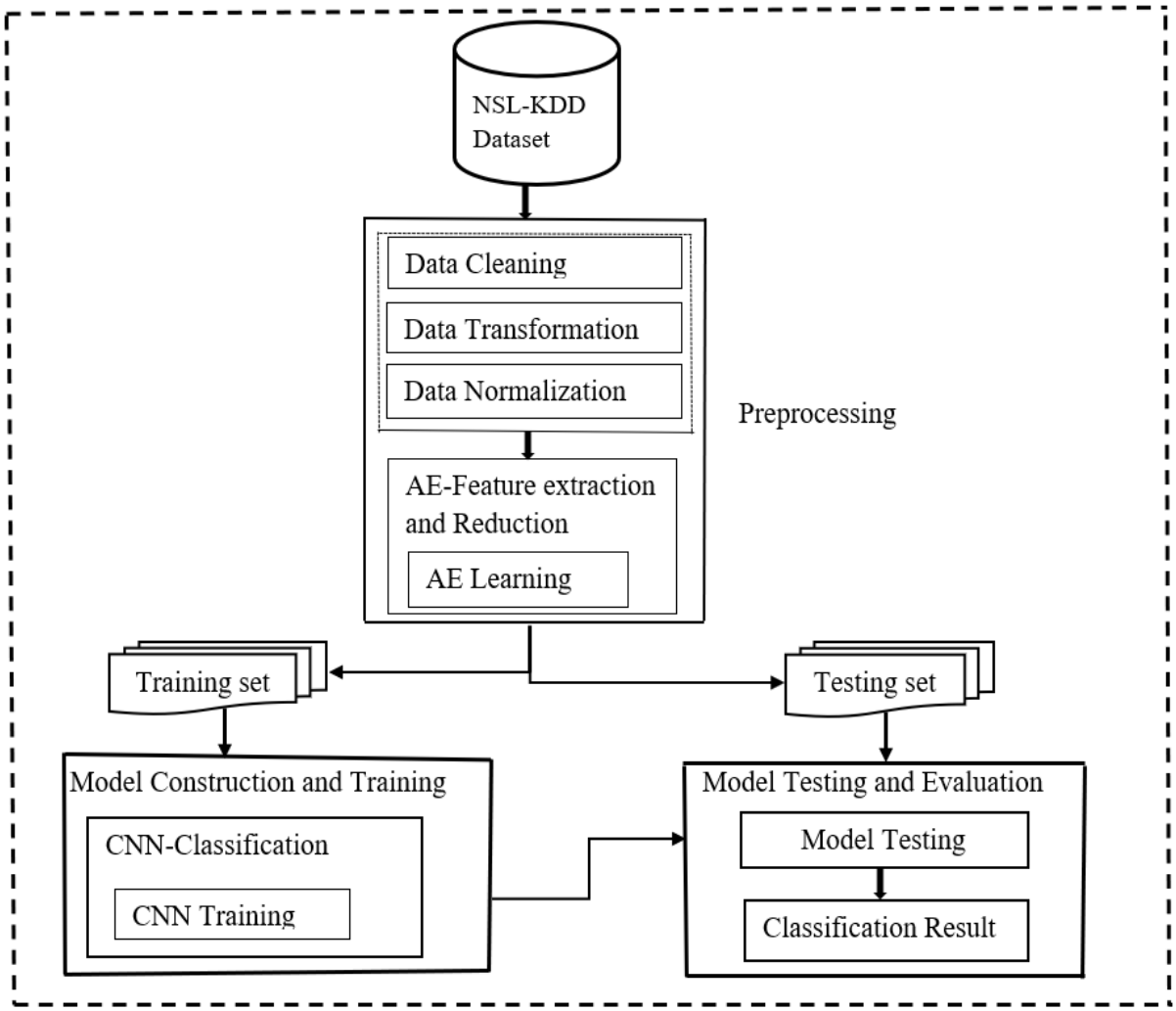


Figure 4.1 The proposed ARP attack detection and classification model architecture

4.2 Major Components of the Proposed Model

4.2.1 Data Preprocessing

The precision and content richness of the data play significant roles in data preprocessing, impacting the learning capability of deep learning algorithms. Therefore, it is essential to ensure that the chosen dataset is cleaned and processed before being utilized in a deep learning model. To avoid generating inaccurate statistics due to variables with varying magnitudes, data preprocessing involves several steps. It starts with exploring the data, followed by identifying and addressing missing data by estimating values for all columns. Additionally, feature scaling is performed to ensure all features are on the same scale, and

feature transformation may be applied to create new features. After this process, then perform feature extraction and dimensionality reduction on input dataset to produce important features.

Several vital data preprocessing techniques contribute to the development of effective deep learning models. In this study, we perform data processing in two phases, in the first phase: we employ data cleaning, data transformation, and data normalization. After that, in the second phase, we perform feature extraction and dimensionality reduction by using Sparse Autoencoder to produce latent features at bottleneck layer and then fed it to the next CNN model as input for training and classification.

A, **Data cleaning**

Data cleaning involves the tasks of filling in missing data, addressing gaps, and removing inaccurate or redundant information from a dataset. It is the most crucial preprocessing stage to improve the completeness, accuracy, and consistency of the dataset and it will make sure that our data is prepared for our subsequent needs. Therefore, we will perform necessary data cleaning tasks on the dataset in order to obtain a neat and organized dataset for the next step.

B, **Data Transformation**

When talking about preprocessing, data transformation is the process of transforming the original information into a format appropriate for modeling training and testing. In this phase, the main task is transforming the symbolic character column (categorical value) of the data to a numeric (non-categorical value). The NSL-KDD dataset contains three symbolic features: protocol type, service, and flag. The protocol type feature consists of three categories: tcp, udp, and icmp. The service feature consists of 70 categories, including ftp_data, ssh, http, and others. Lastly, the flag feature comprises 11 categories, such as SF, S0, SH, and more. Therefore, we have to label or transform this value into a numeric value for each feature.

C, **Data Normalization**

Normalization is the process of scaling individual samples so that they have a consistent range. This involves adjusting the numerical properties based on their original range. In the

NSL-KDD dataset, there are several features that have large ranges between their minimum and maximum values. For example, the duration feature has minimum and maximum values of 0 and 58329, respectively. These values are incompatible and unsuitable for direct use. Therefore, it is necessary to normalize these features to improve the learning performance of the model and achieve optimal results. We used the min-max normalization technique, which scales all attribute feature values between 0 and 1, following the equation (18)[92].

$$X_{Norm} = \frac{X - Xmin}{Xmax - Xmin} \tag{18}$$

Where, X represents the current value of the entry, X_{max} the maximum and X_{min} the minimum value of the entry and X_{Norm} the normalized value of the entry.

D, AE- Feature Extraction and Reduction

DL techniques utilize feature learning methods to improve the performance and detection rate of supervised learning. By employing these methods, the training and testing time can be reduced, resulting in enhanced performance. The idea of feature extraction and dimensionality reduction from the input data served as the starting point for our proposed model's first phase. The Autoencoder algorithm, which our suggested method employs, is a popular methodology in deep learning for feature extraction and dimensionality reduction at this moment. So, Unsupervised Autoencoders are effective feature extraction algorithm that learns the best features to duplicate its output as closely as possible to its input data. In other words, dimensionality reduction is implied by autoencoder algorithms since they transform data into a new representation while keeping the most important features before performing a reconstruction phase. The extracted features minimize the dataset's dimensionality by removing extraneous characteristics. Due to the reduced dimensionality of the dataset, model training and testing times are lowered, which improves attack detection and classification performance.

As shown in Figure 4.1, the second phase preprocessing task starts by using the first phase preprocessed dataset as input and implementing dimensionality reduction and feature learning techniques using a single-layer sparse Autoencoder. The sparse Autoencoder can automatically extract relevant and effective low-dimensional features for the classification process by learning useful low-dimensional features from the input training and testing

dataset. As discussed in Chapter 2 Section 2.6.2, Sparse Autoencoder involves two steps, which are encoder and decoder for feature extraction and dimensionality reduction process. The encoding step maps the input data into the hidden units' representations, which are processed by a nonlinear activation function. Then, the decoding step maps the hidden units' representations into the reconstructed data, which is output data that is processed by the output vector activation function. We used a sigmoid activation function for both hidden and output layers, which, is a very common choice for feed-forward neural networks. Sparse Autoencoder applies a backpropagation algorithm to obtain the optimal values for its weight matrices and bias vectors, which attempt to learn and reconstruct its output values to be equal to its inputs and also to minimize the loss function. Sparse Autoencoders operate by regularizing and penalizing the activation of a certain number of neurons in hidden layers, and we use the KL divergence sparsity constraint in addition to the mean squared error loss function. Through these fourth and back processes, Sparse Autoencoders could learn and contract the new data representation, which is the most important and lower dimension of features at the bottleneck layer. This feature contains the significant nature of the dataset and is used to detect and classify the exact label of the dataset, whether it is normal or an attack. These features are used as input in the next phase, which is the CNN model for classification purposes. Algorithm 4.1 shows the AE feature extraction and dimensionality reduction algorithm. This algorithm and others for CNN classification and Model testing algorithms are adopted from the research in [90] and [93].

```
Input: NSL-KDD dataset X= (x1, x2, x3…,xi)
Output: Latent Features
Begin:
Initialize: weight, bias, Encoder E, Decoder D
EP= number of epochs, BS=Bach size i=0
#AE learning for feature extraction and dimensionality
reduction
while (i<EP) do:
For (z=0, z≤ BS; z++)
     E=h ←f(WX+b1)    // encoding the input dataset and get
     latent representation
     D=z←g(VH+b2)    // decoding the latent representation
     features to output
```

```
      Calculate loss function L after reconstruction
      By using mean squared error and KL divergence sparsity
      constraint
      L ← calculate by using Equation (11)
      Update weight and bias
      EP++
      If x_i=x̂_i or its difference is very close to zero
       return stable AE model with reduced and most important
      features contained(Latent Features)
      Else
      Perform back propagation and sparsity constraint
      To calculate the above reconstruction error
End
```

Algorithm 4.1 AE feature extraction and reduction algorithm

AE Learning

The Autoencoder learns by minimizing the loss function to approximate the output value very close to or similar to the input value. The learning process starts at the encoder stage by assigning weight and bias to each input vector. Then, perform the activation function through the sigmoid function to reduce the dimensionality of the input data and extract the most important features in the hidden layer, which is the latent space representation of the data. The sparsity penalization also applies for hidden layers, whether the neuron is active or not for regularizing, and to the most effective feature on the low-dimensional layer. After constructing the hidden layer, which is the new representation of the most crucial features, the learning process continues in the decoding phase to reconstruct the input data into output data by applying the reconstruction activation function.

Autoencoder creates an encoder model through self-supervised learning technique that makes the input and output is the same without a label. In this study, we used labeled NSL-KDD dataset, but the AE does not use the label for learning process, first drop the labeling from the dataset (handled by Y variable),which later used for CNN algorithm training and only the features (handled by X variable) are fed to the AE learning process to extract latent features.

4.2.2 CNN- Classification

We adopt CNN in our proposed hybrid approach to detect and categorize the multi-class attacks in the ARP attack detection and classification method. After the Sparse Autoencoders most crucial features have been extracted, these low-dimensional features are feed into the CNN classification algorithm for training and testing, which improves attack detection and classification accuracy while minimizing the false alarm rate.

A deep Neural Network called a CNN is known to be successful in numerous studies and applications, particularly in the area of image categorization. There are a number of layers in CNN that make it a more effective and powerful algorithm for classification problems. So, that's why we chose this algorithm for classification purposes. These layers are categorized into three main parts, named input layer, hidden layer, and output layer. And also, the hidden layers include the convolutional layer, the ReLU activation function layer, the pooling layer, and the fully connected layer. The number of layers employed determines the CNN's depth, and the more layers we use, the deeper the CNN becomes.

In this algorithm, the convolutional layer was able to extract the features from the data fed from the input layer (that are coming from Autoencoder algorithm), by employing various filters, based on the ReLU activation function. ReLU is frequently employed as one layer or component in CNN architecture and used as the default activation function. The pooling layer was then, responsible for reducing the matrix size by utilizing max pooling techniques to speed up learning and prevent overfitting problems after computing the ReLU function. The maximum weighted features from the corrected feature map are chosen or pooled in this situation as part of the Max pooling technique to create a reduced pooled feature map containing the most crucial features. The data was moved from the final pooling layer to the fully connected layer after completing several Max pooling for all rectified feature maps. This directed the data in the fatten layer and produced a one-dimensional array that represented the classes (normal, attack). In order to finish the categorization task, we apply the SoftMax activation function. Here, the output layer is a five-class SoftMax layer (one class for each attack type plus the normal class). For each character class, SoftMax generates a probability-like prediction.

CNN Training

During CNN training phase, supervised learning techniques are used to carry out the essential training process as shown in algorithm 4.2. In this case, the training process is very easy and takes a short period of time because of it used reduced size, and only the most important features are taken as input from extracted the Autoencoder bottleneck layer. Because, extraneous or unnecessary features for attack detection are removed in the Sparse Autoencoder feature extraction phase, it reduces training time for CNN model. Thus, reduced model complexity and improved the learning process. The model can easily generalize and form similar patterns through the knowledge it gains from training. As a result, the classifier has strong capability for classification prediction in the testing and evaluation stages once it has been trained in a well-organized manner.

```
# CNN classification training Cγ
Input: NSL-KDD dataset X= (x1, x2, x3… xi)
      Labeled training set Y= (y1, y2, y3…yi)
Output: Trained CNN model
Begin:
    γ ← Initialize CNN network connection parameters for
                        classification
For each (x, y) input sample
     do
     h ←E(x)    # To get latent presentations from trained AE
     CNN = Cγ((h), y)  #To train CNN
     γ ← Update CNN parameters using Adam optimization
     a ← average Reconstruction loss on Training data set
     Return Trained CNN model
End training phase
```

Algorithm 4.2 CNN classification and training algorithm

4.2.3 Model Testing

In this stage, the model has gone through training on the training set, learning what is normal and what an attack is, in the previous subcomponent CNN training. Therefore, now we perform the model testing task to evaluate the proposed approach by feeding the testing set into the trained model and determining if it can accurately predict target values for

normal as normal and attack as attack, as shown in algorithm 4.3. We employed the K-fold cross-validation parameter during the testing phase. In this stage, we also perform model evaluation by using performance metrics like the confusion matrix to measure the accuracy of the classification result.

4.2.4 Classification Result

This component displays the classification result, whether it is normal or an attack. And also, the attacks have four different categories, similar to what the input dataset has. These are the Dos, Probe, R2L, and U2R attack categories.

```
# Testing phase
Input: NSL-KDD Testing set X`= (x`1, x`2, x`3,…,x`i)
       Labeled testing set Y`= (y`1, y`2, y`3,…,y`i)
Output: Classification Result R= Output (ŷ`|y`)
Begin:
For each (x`, y`) sample in Testing dataset
     do
    h`←E(x`)  # for AE feature extraction
    R= Cγ((h`), y`)
    Return R
   Calculate reconstruction Loss
End
```

Algorithm 4.3 CNN model testing and classification algorithm

4.3 Summary

We introduce the structure and algorithm of the hybrid deep learning model proposed for the identification and categorization of ARP attacks in a LAN, in this Chapter. The model combines two widely used deep learning algorithms, namely the Autoencoder and Convolutional Neural Network, to achieve its objectives. In order to increase the detection performance of the model, it uses an unsupervised Sparse Autoencoder algorithm for feature extraction and dimensionality reduction, and a supervised Convolutional Neural Network for classifying the input data as attack or normal.

The suggested model's architecture is made up of four essential components: data preprocessing, CNN classification, model testing and classification result. The model takes

preprocessed data, which is a primarily performed activity by feeding the NSL-KDD dataset, as input and then uses Sparse Autoencoders to produce reduced size and the most important features at the bottleneck layer through encoding and decoding steps. In this stage, we compute a number of forward and backward propagations to calculate the reconstruction error and also apply the sparsity function for penalizing and regularizing on the weights and biases of neurons, plus the active or inactive state of each neuron, to optimize the average parameter. The next stage is to train the CNN classification algorithm by taking bottleneck layer features from AE and training it through its compressive layers, including the convolutional layer, ReLU activation function layer, max pooling, and fully connected layers, to learn the nature and structures of the input data to predict whether it is normal behavior or anomalous network activity and further categorize it. Once the model training is finished, testing and evaluation are performed using the testing dataset to measure the accuracy of the proposed model and display the classification result at the final stage.

Chapter 5: Implementation and Evaluation

5.1 Overview

This chapter presents the simulation environment and its evaluation for the suggested ARP attack detection and classification approach. Development tools, the experimental setup, data collection and preparation, model construction, training and testing, performance assessments, evaluation criteria, and a discussion of the findings are presented in the following parts.

5.2 Development Tools

For the development of proposed model simulation environment, we have used various tools and packages. Following are some important frameworks, packages or tools that we used [94].

- **TensorFlow: -** which is an open-source framework designed by researchers at Google for executing workloads related to statistical and predictive analytics, deep learning, and machine learning. It models ML methods as computational graphs. The vertices or nodes of a computational graph are operations, and the edges between them are the data that flows between these operations [94]. All of this is made available to ML experts by TensorFlow's Python API. We used Tensor flow platform for our proposed system.
- **Python: -** is a popular computer programming language used for creating websites and applications, process automation, and data analysis. In recent years, it has become one of the most frequently used object-oriented programming languages globally.
- **Keras**: - is a Python interface-based open-source software library for artificial neural networks. TensorFlow is accessed through the Keras user interface.
- **Pandas:** - is a Python data analysis library. It offers a range of tools, such as data frames, series, and time-series, for working with structured data. Pandas can be used for a variety of activities, such as segmenting and separating the data based on preference or filtering the data based on specific criteria.

- **NumPy:** - is a Python scientific computing library. It offers tools to work with arrays and a high-performance multidimensional array object.
- **Matplotlib:** - is a Python data plotting library. It offers both 2D and 3D data graphing functions in a variety of formats.
- **Scikit-learn**: - is a very useful ML library for Python. It provides a range of powerful techniques for statistical modeling and machine learning, including dimensionality reduction, clustering, regression, and classification, through a Python consistency interface.
- **Google Collaboratory (Colab):** - With its notebook interface, users may write and run code, view output, and produce visualizations. With Google Colab, multiple users may work on the same notebook at once and collaborate in real-time. Additionally, it has Google Drive connection, which makes it simple to share notebooks with others. It establishes a connection to strong Google Cloud Platform runtime resources and boosts runtime efficiency by accelerating computation utilizing a free GPU.
- **Google Drive:** - is a service created by Google for file syncing and storage. It enables users to share files, synchronize files between devices, and store files in the cloud. For datasets and model implementation codes, we used Google Drive.

5.3 Experimental Setting

Every experiment was conducted on a Dell Latitude laptop running Windows 10 with an Intel(R) Core (TM) i7 6600U CPU running at 2.60GHz and 2.81GHz, 8GB of RAM, and a 500GB hard drive for storage. Google drive for dataset and implementation source code storage, online browser-based Google Colab notebook with python 3 and connected it to cloud platform for accessing free resources like RAM, CPU and GPU to speed up learning and testing runtime of the implementation. TensorFlow framework with other libraries like Keras, pandas, NumPy, Matplotlib, Scikit-learn are integrated with google colab notebook and easy to use without any installation and configuration setup on our computer.

5.4 Data Collection and Preprocessing

5.4.1 NSL-KDD Dataset

A huge amount of data is the main concern for machine learning-based systems to train the algorithm. So, dataset quantity and quality are important components of the machine learning environment. In this thesis, we tried to find a publicly available ARP attack dataset, but there is no dataset that is directly collected from an ARP packet and labeled for the purpose of ARP attack machine learning model evaluation. Most of the research papers regarding this attack's methods are evaluated using private datasets and real-life traffic data. On the other hand, collecting real-life network traffic with attack information from a LAN is difficult due to the privacy and security issues of that LAN organization. Additionally, generating this traffic using network traffic generation tools is possible, but obtaining a large amount of data with proportional normal and attack traffic is a challenge, and proper labeling of this data is also a difficult and time-consuming task.

Therefore, due to the above reason, we enforced the selection of the publicly available dataset, which is more appropriate, and assumed it represents ARP attacks and normal ARP packets, which is the NSL-KDD dataset. This dataset is collected from the Air Force LAN environment, and it contains similar attack categories, with ARP attack types like DOS attacks exactly the same, and Probe, R2L, and U2R attack types included under MITM attack. So, we selected and used the NSL-KDD dataset to train, test and evaluate the proposed model.

The NSL-KDD dataset has two sets of data: KDDTrain, which contains 22 different types of attacks, and KDDTest, which contains an additional 17 undiscovered types of attacks. As seen in table 5.1[21], any sort of attack can be precisely classified into one of four attack categories that share certain characteristics. The table's bolded terms indicate that new attack kinds have been introduced to the test set.

Table 5.1 NSL-KDD trains and test set attack types with their categories

Attack Categories	ARP Attack types
Dos	**apache2**, back, land, **mailbomb**, Neptune, Pod, **processable**, smurf, teardrop, **udpstorm**
Probe	ipsweep, **mscan**, nmap, portsweep, **saint**, satan
R2L	Spy, warezclient, ftp_write, guesspasswd, **httptunnel**, imap, multihop, **named**, phf, **sendmail**, **snmpgetattack**, warezmaster, **xlock**, **xsnoop**
U2R	Bufferoverflow, loadmodule, perl, **ps**, rootkit, **snmpguess**, **sqlattack**, **worm**, **xterm**

Within the NSL-KDD dataset, there are 125,973 training patterns and 22,544 testing patterns. The attack classes and the number of patterns under each class are displayed in Table 5.2 [21].

Table 5.2 Attack class and number of patterns under each class

Training data set		Testing data set	
Class	Patterns	Class	Patterns
Normal	67343	Normal	9711
Dos	45927	Dos	7458
Probe	11656	Probe	2421
R2L	995	R2L	2754
U2R	52	U2R	200
Total	**125,973**	**Total**	**22,544**

The forty-one features present in the NSL-KDD dataset, according to [95], can be divided into three categories:

1. **Basic features:** this cluster compresses all the features that can be extracted from a TCP/IP connection and derived from packet headers. Ten of the 41 features in the NSL-KDD dataset are classified as basic features.

2. **Content features:** refer to all of the suspicious data contained in a TCP packet's payload. The amount of failed login attempts, for example, is considered as content feature.
3. **Traffic features:** This category includes features that are computed with respect to a window time interval. This feature can be also divided into two groups:

   a) The same service features are those that are examined across only the previous two connections that have the same service as the present connections.

   b) The same host features are features (behavior, services of protocol, etc.) that are evaluated across those connections in the past 2 seconds, which contain the same host as current connections hold. The details are shown in appendix A.

5.4.2 Preprocessing

One important stage in the ML model is preprocessing the dataset. The data needs to be examined, purified, and made model-appropriate before it is fed into an algorithm. Data preparation and quality are critical since poor data will cause even the most accurate predictor to perform poorly. In this study, data preprocessing is performed by calling a preprocessing library from the sklearn packages, as shown below.

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder,
MinMaxScaler
```

After importing other necessary libraries like the above sample, we perform the following preprocessing activities:

1. To load the dataset into the Panda data frame, first we need to mount the Google Drive and then load both the training and testing datasets as shown in Figure 5.1 for train set loading.

```
[3] df = pd.read_csv('/content/drive/My Drive/Dataset/NSL-KDD/KDDTrain+.csv')
```

```
df.head()
```

	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	...	Column34	Column35
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	0.17	0.03
1	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60
2	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05
3	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00
4	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00

5 rows × 43 columns

Figure 5.1 Loading Train set into Panda data frame

2. Assigning or labeling each column with proper feature naming as shown in Figure 5.2 for train set.

```
[7] df.columns =["duration","protocol_type","service","flag","src_bytes","dst_bytes","land",
                 "wrong_fragment","urgent","hot", "num_failed_logins","logged_in","num_compromised",
                 "root_shell","su_attempted","num_root","num_file_creations","num_shells","num_access_files",
                 "num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_rate",
                 "srv_serror_rate", "rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate",
                 "srv_diff_host_rate","dst_host_count","dst_host_srv_count","dst_host_same_srv_rate",
                 "dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate",
                 "dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate",
                 "dst_host_srv_rerror_rate","label","difficulty_level"]
```

```
df.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...
1	0	udp	other	SF	146	0	0	0	0	0	...
2	0	tcp	private	S0	0	0	0	0	0	0	...
3	0	tcp	http	SF	232	8153	0	0	0	0	...
4	0	tcp	http	SF	199	420	0	0	0	0	...

5 rows × 43 columns

Figure 5.2 Labeling encoding for each feature naming

3. Then, we need to check if a null value exists or not from both train and test set as shown in Figure 5.3.

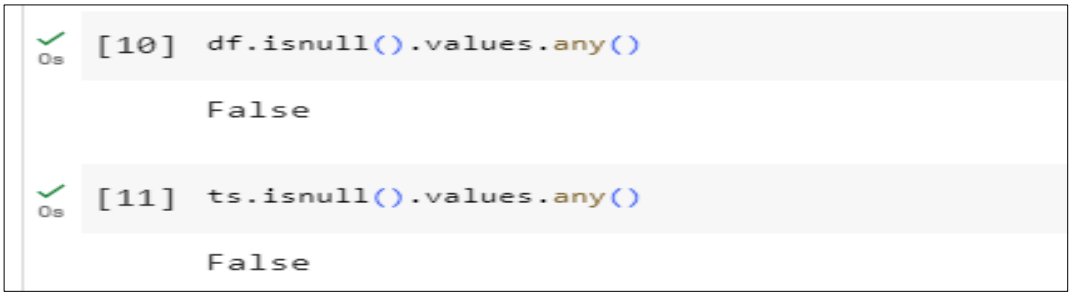


Figure 5.3 checking any null value for train and test set

4. Remove unnecessary features that have no role in attack detection, like the *difficulty-level* feature as shown in Figure 5.4.

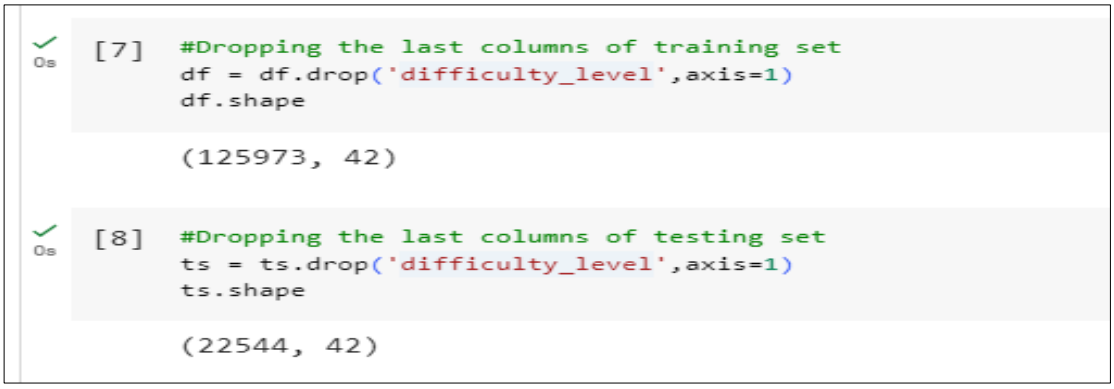


Figure 5.4 Removing difficulty- level feature from both train and test set

5. Data transformation, which means labeling categorical features like *protocol type*, *service*, and *flag* to a numerical value using one hot encoder function, after all the transformations, 41-dimensional features were converted into 122-dimensional features.

```
cols= ['protocol_type','service','flag']
def one_hot(df, cols):
  for each in cols:
        dummies = pd.get_dummies(df[each], prefix=each, drop_first=False)
        df = pd.concat([df, dummies], axis=1)
        df= df.drop(each, axis=1)
  return df
train_data=one_hot(df,cols)
train_data
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...
0	0	491	0	0	0	0	0	0	0	0	...
1	0	146	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	232	8153	0	0	0	0	0	1	0	...
4	0	199	420	0	0	0	0	0	1	0	...
...	...	...	...	...	...	...	...	...	...	...	...
125968	0	0	0	0	0	0	0	0	0	0	...
125969	8	105	145	0	0	0	0	0	0	0	...
125970	0	2231	384	0	0	0	0	0	1	0	...
125971	0	0	0	0	0	0	0	0	0	0	...
125972	0	151	0	0	0	0	0	0	1	0	...

125973 rows × 123 columns

Figure 5.5 One hot encoding for training set categorical features

6. Categorizing each attack type into four respective classes, which are Dos, Probe, R2L and U2R as shown in Figure 5.6. And labeling each class with the numerical values 0, 1, 2, 3, and 4.

```
[17] def change_label(df):
       df.label.replace(['apache2','back','land','neptune','mailbomb','pod',
                         'processtable','smurf','teardrop','udpstorm','worm'],'Dos',inplace=True)
       df.label.replace(['ftp_write','guess_passwd','httptunnel','imap','multihop','named','phf',
                         'sendmail','snmpgetattack','snmpguess','spy','warezclient','warezmaster',
                         'xlock','xsnoop'],'R2L',inplace=True)
       df.label.replace(['ipsweep','mscan','nmap','portsweep','saint','satan'],'Probe',inplace=True)
       df.label.replace(['buffer_overflow','loadmodule','perl','ps','rootkit',
                         'sqlattack','xterm'],'U2R',inplace=True)

[18] change_label(df)
     df.label.value_counts()

     normal    67343
     Dos       45927
     Probe     11656
     R2L         995
     U2R          52
     Name: label, dtype: int64
```

Figure 5.6 Attack labeling for one of four categories for training set

7. Apply Min-Max scalar normalization to normalize values that have a wide gap between the highest and minimum values as shown in Figure 5.7 the training set.

```
[108] t = MinMaxScaler()
      sacled_data=t.fit(train_data)
      X_train_scaled = sacled_data.transform(train_data)
```

```
X_train_scaled
```

```
array([[0.00000000e+00, 3.55806412e-07, 0.00000000e+00, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 1.05799870e-07, 0.00000000e+00, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 1.61670897e-06, 2.93143779e-07, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 1.09423153e-07, 0.00000000e+00, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

Figure 5.7 Train data normalization using Min max scalar

5.5 Implementation of Attack Detection and Classification Model

5.5.1 Feature Extraction

In this phase, all the preprocessing activities are done for the training and testing datasets, and are ready to be fed as inputs for model training and testing. Then, we should select optimized hyperparameters to control the learning process and have a significant impact on the model's performance. As shown in Table 5.3, we have chosen hyperparameters for both AE and CNN implementations to build an optimized model. Then, first, we implement a sparse Autoencoder using 122, 61, and 122 neurons for the input layer, hidden layer, and output layer, respectively. So, the encoder accepts the input data and compresses it into hidden layers, which is the latent space or bottleneck layer, and then the decoder reconstructs the latent space to the output layer using the sigmoid activation function and mean-squared error

(MSE) for the loss function plus Kl-divergence for sparsity constraint regularization. The MSE loss function will encourage the model to accurately reconstruct the input data and the KL-divergence regularizer will encourage the activations of the hidden layer to be sparse. This will help to prevent overfitting and make the model more interpretable. Then, we train the Autoencoder with the training and testing sets for feature reduction and better feature representation.

Table 5.3 Hyper parameters used for model building

Hyper parameter	**For AE**		**For CNN**	
	Parameter's value	Description		Description
Number of neurons	(122, 61,122)	For input hidden layer and output layers	(61,30, 5)	For input hidden layer and output layers
Activation function	Sigmoid	Activation function for hidden and output layers	ReLU and SoftMax	ReLU for hidden layers and SoftMax for output layer Classification purpose
Loss function	MSE and KL-divergence	MSE for reconstruction error and KL-divergence for Regularizer	categorical_crossentropy and Dropout	categorical_crossentropy for reconstruction error and dropout for Regularizer
Optimizer	Adam optimizer	For model optimization	Adam optimizer	For model optimization
Bache size	32	It represents how much the data partions during the training phase	32	It represents how much the data partions during the training phase
Epoch	10	Number of iterations during the training	10	Number of iterations during the training

Finally, extract latent features from the bottleneck layer as shown in Figure 5.8, latent feature representation, using the t-SNE visualization method, and use them as input in the CNN model. A sample code for Autoencoder model construction and feature extraction is shown in Appendix B.

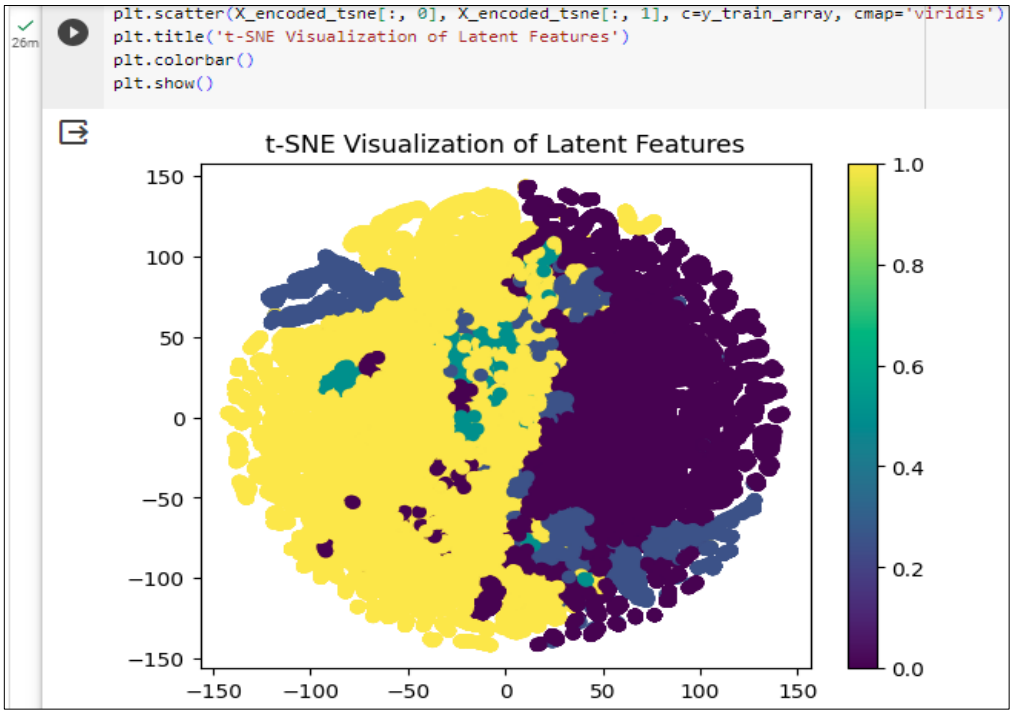


Figure 5.8 Latent features representation after encoder using t-SNE Visualization

5.5.2 CNN Classification

In this phase, we build the CNN model by taking the latent features as input with 61 neurons and using the ReLU activation function for hidden layers and the SoftMax function for multiclass classification tasks at the output layer. It also uses a dropout layer for regularization, which randomly drops the fraction of neurons that are set to zero, to prevent overfitting in the model training. CNN also makes use of the optimizer Adam, an adaptive learning rate optimization method, for CNN model training, the default learning rate defined in Keras, which is 0.001, and the loss function categorical cross-entropy, a traditional loss used in multi-classification applications. After model building, we train the CNN model using the input data for ten epochs of iteration time in thirty-two Bache size normalization. Finally, we evaluate the model using a testing dataset to predict the output, and display the classification result in a confusion matrix. A sample code for CNN model construction is shown in Appendix B.

5.6 Evaluation based on Test Data

In order to implement this model and provide a more precise estimate of the model's performance, the 10-fold cross-validation technique is utilized for model evaluation. In 10-fold cross-validation, the data is divided into ten folds; the model is trained on nine of the folds, and it is validated on the remaining one. To estimate the model's performance, this process is carried out ten times, with the results averaged. Typically, training uses up 90% of the entire NSL-KDD dataset, with the remaining 10% going toward model validation and testing the suggested model. From 90% of the training sets, 69,331 are normal network traffic and 64,335 are attack traffic data, sum of both is 133,666. And for testing, 10% of the remaining 14,851 (7,723 are normal network traffic and 7,128 are attack traffic data) is used, as shown in Figure 5.9 test data distribution across each class.

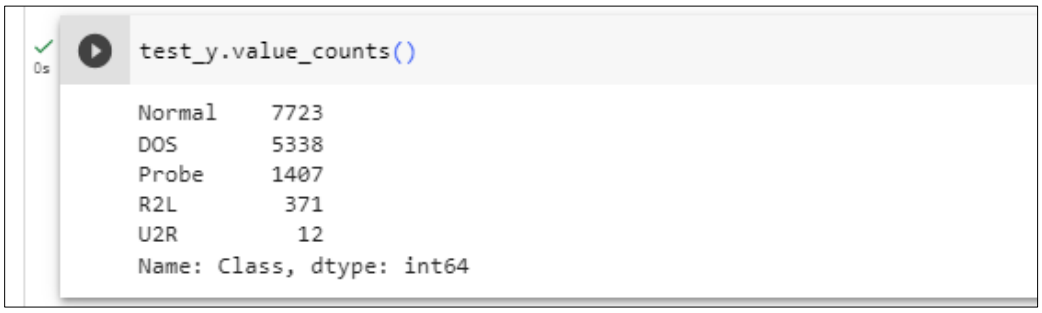


Figure 5.9 Test dataset distribution value across each class.

In this phase, we test and evaluate the model in two scenarios: the first is using the proposed approach of AE-CNN model testing and evaluation, and the second is testing only the CNN model by feeding test data directly into it. Therefore, we compare the performance metrics of our proposed model, AE-CNN, with those of a single CNN model to assess the superiority of our approach.

a) AE-CNN Classification results

The testing results are presented in this section using a confusion matrix to evaluate all classification results. Figure 5.10 shows the AE-CNN classification result using the testing dataset displayed in a 5x5 confusion matrix.

```
[47] from sklearn.metrics import confusion_matrix
     confussion_matrix=confusion_matrix(y_eval, pred, labels=[0,1,2,3,4])
     confussion_matrix

     array([[5328,   10,    0,    0,    0],
            [   3, 7687,   13,   20,    0],
            [   0,   16, 1390,    1,    0],
            [   0,   41,    0,  330,    0],
            [   0,    6,    0,    0,    6]])
```

Figure 5.10 The 5x5 confusion matrix for AE-CNN classification results

This confusion matrix, which is derived by classifying the test samples into four categories (TP, TN, FP, and FN) as described in Chapter two in Section 2.6, shows how many accurate and inaccurate predictions the classification model produced in relation to the target values in the data. After testing on the test data set, it displays the model's TN, FN, TP, and FP results. Class **Attack** indicates **positive** in this confusion matrix, whereas Class **Normal** indicates **negative**.

Figure 5.11 provides a more detailed illustration of the classification performance result for the AE-CNN model utilizing the confusion matrix on the five-class. The matrix below provides the TP, TN, FP, and FN counts for each class (Normal, DoS, Probe, R2L, and U2R).

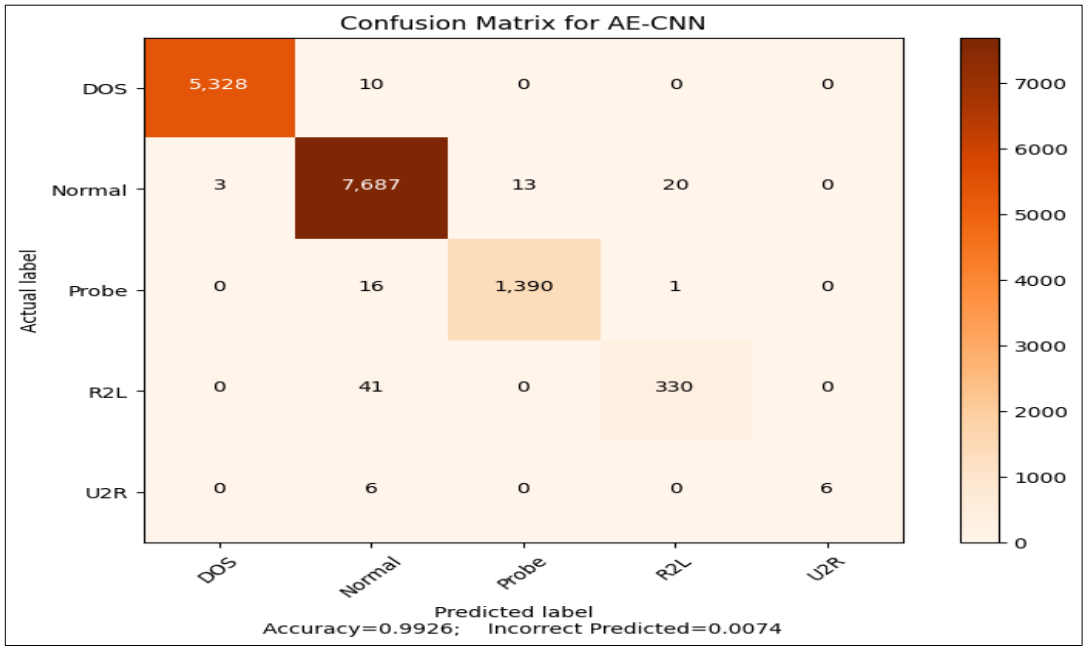


Figure 5.11 AE-CNN Classification results in confusion matrix with each class

Based on the confusion matrix result, the proposed model is able to predict the output with high accuracy = 99.26%, error rate = 0.74%, precision = 99.47%, recall = 98.97%, F1-score = 99.21, and false positive rate = 0.46%, as shown in Table 5.5. And also, Figure 5.12 shows the performance evaluation metrics results for each attack class in a bar graph. From this, DoS and probe attack detection scored the highest results for precision and recall. Despite this, U2R attack detection scores a very small recall, which is 50%. The detailed output of classification results in precision, recall, and F1-score evaluation metrics for each category is shown in Appendix C.

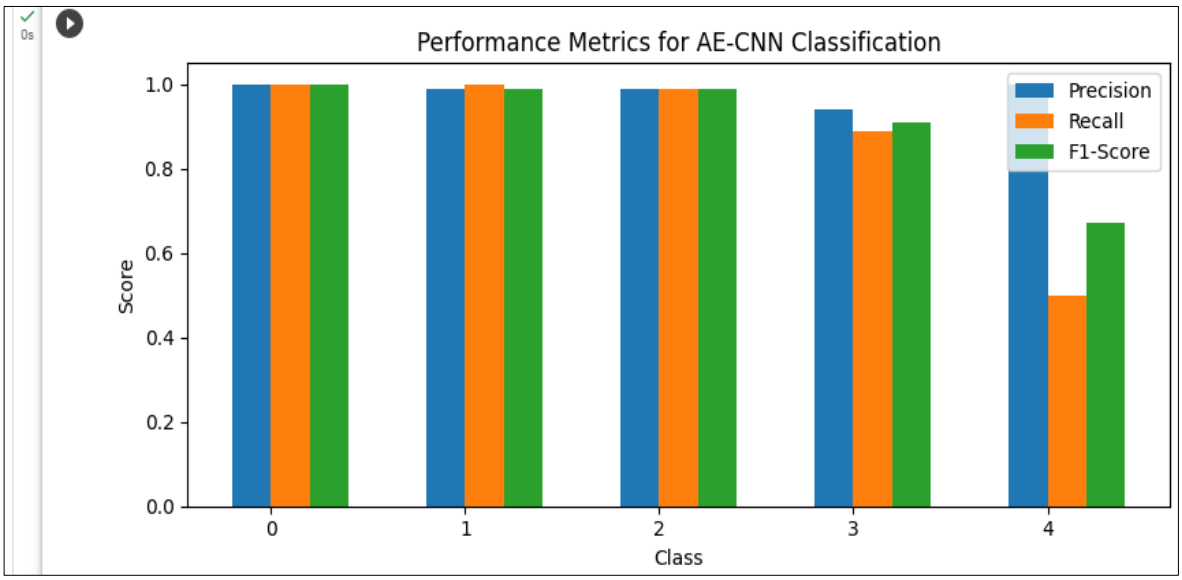


Figure 5.12 AE-CNN classification results evaluation for each attack class in bar graph

b) CNN Classification results

When we perform testing on the CNN model only, the output is less than the AE-CNN in all performance metrics. In Appendix C shows the CNN classification result using the testing dataset displayed in a 5x5 confusion matrix. Also, the result shows in Figure 5.13 more clear way, in the confusion matrix graph. So, based on the confusion matrix result, the CNN model can predict the output with accuracy = 97.18%, error rate = 2.82%, precision = 96.85%, recall = 97.67%, F1-score = 97.25%, and false positive rate = 2.91%, as shown in Table 5.4. Also, Figure 5.14 shows the performance evaluation metrics results for each attack class in a bar graph. From this, DoS and probe attack detection scored better results

for precision and recall. But, U2R attack detection scores a very small recall and relatively small precision, which are 25% and 75%, respectively. The detailed output of classification results in precision, recall, and F1-score evaluation metrics for each category is shown in Appendix C.

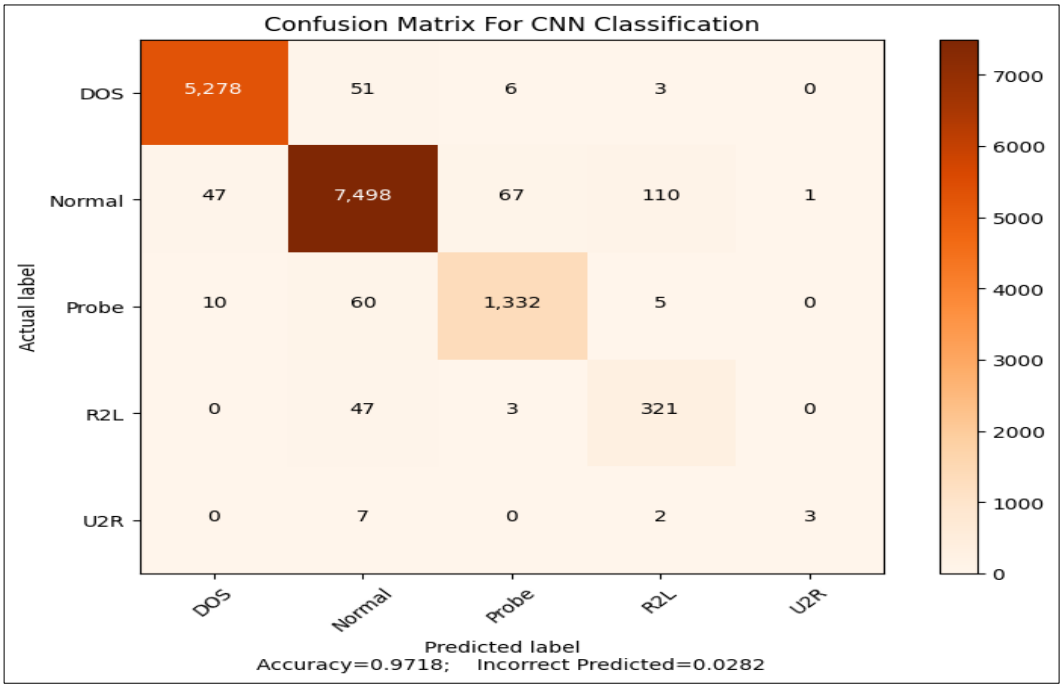


Figure 5.13 CNN Classification results in a confusion matrix with each class

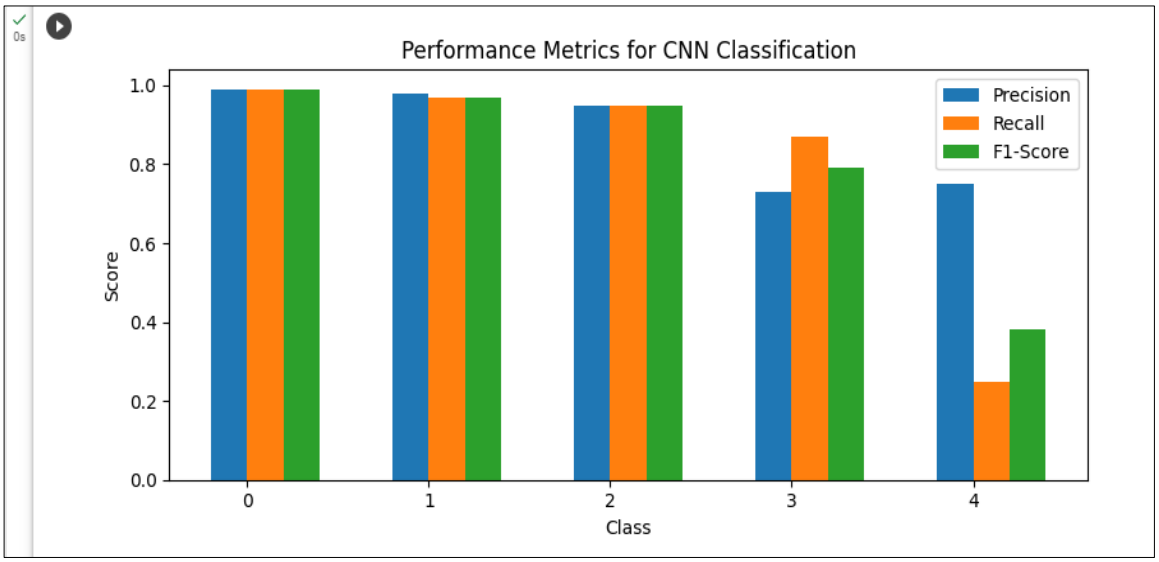


Figure 5.14 CNN classification results evaluation for each attack class in bar graph

c) Evaluation Result Comparison AE-CNN with CNN

Based on the testing result in both approaches, the result shows that the proposed model AE-CNN performs the highest score for all performance evaluation metrics: attack detection accuracy, precision, recall and f1-score with the minimum error rate and false alarm rate compared to a single CNN model as illustrated in Table 5.4 comparison.

Table 5.4 Performance evaluation results for proposed model AE-CNN and CNN

Approaches		**Performance evaluation metrics in percent (%)**				
	Accuracy	Precision	Recall (DR)	F1-score	Error Rate	False Alarm Rate (FPR)
CNN	97.18	96.85	97.67	97.25	2.82	2.91
Proposed AE-CNN	**99.26**	**99.47**	**98.97**	**99.21**	**0.74**	**0.46**

The comparison for these two approaches is also shown in Figure 5.15 in the bar graph, which represents the accuracy of both approaches and the macro average of precision, recall, and f1-score of all five classes (Normal, DoS, Probe, R2L, and U2R). So, the graph shows the superiority of the proposed approach to that of a single CNN approach in all performance metrics.

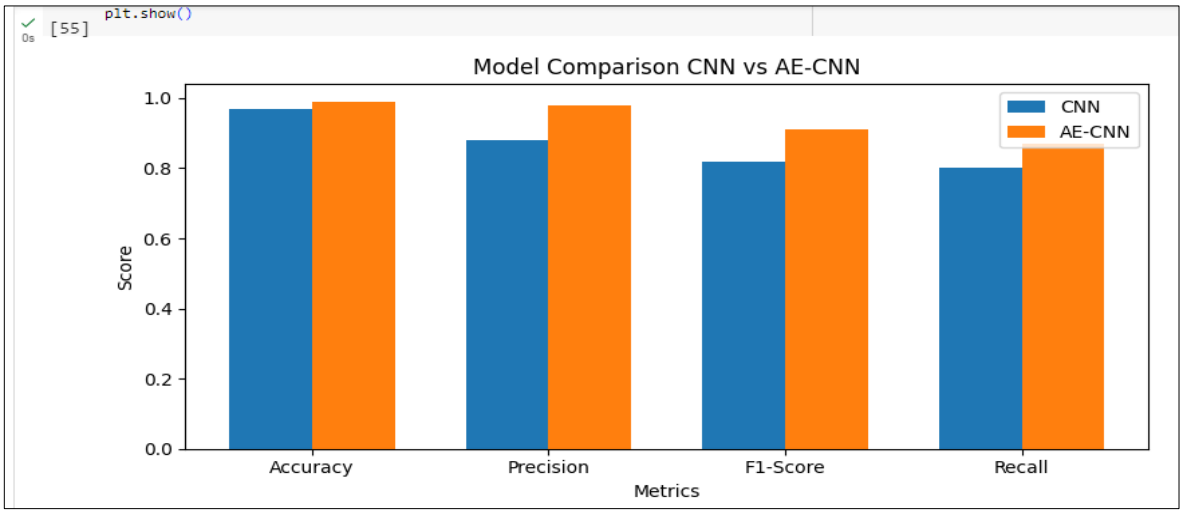


Figure 5.15 Evaluation results comparison AE-CNN with CNN in bar graph

The evaluation's overall findings indicate that to improve classification accuracy and other performance metrics like precision, recall, and f1-score while lowering false alarm and detection error rates, it is important and plays a great role in reducing the dimensionality of

the input dataset and extract significant features from it by using the energetic Sparse Autoencoder algorithm for this task.

5.7 Summary

In this chapter, we perform an experimentation and evaluation of the proposed method to examine whether it achieves its initial objectives. In the experimentation, we used the TesnorFlow framework, Python programming language, and other libraries integrated with the Google collaborative notebook environment to construct, train, and test our model. Also, we used an open-source NSL-KDD dataset, which is improved from the previous version and has a balanced and well-distributed training and testing set. So, this plays an important role in detection and classification performance improvement. By using this dataset as input, first perform data preprocessing like data cleaning, data transformation, and normalization. Then we choose appropriate hyperparameters for model construction, learning, training, and testing to produce an effective model. After that, extract important features using the Sparse Autoencoder algorithm, and then feed this latent feature into the CNN model for supervised training.

We perform model testing and evaluation by using a testing set in two scenarios. The first scenario is our proposed model AE-CNN testing, and the result is displayed in a 5x5 confusion matrix. It achieves the highest score, accuracy of 99.26%, error rate of 0.74%, precision of 99.47%, recall of 98.97%, F1-score of 99.21, and false positive rate of 0.46%. The second scenario involves using a single CNN model for testing and evaluation. The result shows that the output is less than the first scenario in all evaluation metrics, which are accuracy of 97.18%, error rate of 2.82%, precision of 96.85%, recall of 97.67%, F1-score of 97.25%, and false positive rate of 2.91%. Based on the testing results of both approaches, the result shows that the proposed model, AE-CNN, performs the highest score for all performance evaluation metrics: attack detection accuracy, precision, recall, and f1-score, with the minimum error rate and false alarm rate compared to a single CNN model.

Therefore, Algorithm selection plays a great role in detection and classification performance. Hence, we select and used a Sparse Autoencoder that is capable of feature extraction and dimensionality reduction by using a sparsity function or regularizer like KL

divergence in our case. In addition to this, we used different hyperparameters for model performance improvement, like the number of neurons per layer, the number of epochs, the bache size, the optimizer, and others are playing a great role on model performance. We assumed that it would increase the detection performance of our model, and our result proves this expectation.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

Local area networks are easily exploited and may lose data integrity, confidentiality, and availability of an organization's information due to improper security control of the data link layer. This means that ARP requests and replay messages, as they are known in Ethernet communication terminology, can trigger ARP attacks, which are one type and a major contributor to security vulnerabilities in LAN environments. Any attack can readily exploit the security flaw in the ARP protocol's design and harm the network. Such a problem has been addressed by several efforts, but as attack techniques progress over time, certain limits still need to be improved.

In this thesis work, we have tried to design a hybrid ARP attack detection and classification model that tries to solve the shortcomings of the previous research in terms of improving classification accuracy and attack detection rate while minimizing false alarm rates. To design this work, we used two powerful DL algorithms, which are unsupervised AE and supervised CNN, for different purposes. In the first AE algorithm, we used Sparse Autoencoders for feature extraction and dimensionality reduction at the encoder layer, and then the extracted and more important latent features are ready for input for the second algorithm, which is the CNN algorithm. So, the CNN algorithm accepts these extracted features as input and trains the model using this data for model optimization and prediction.

In this work, we used the NSL-KDD dataset for model training and testing to evaluate model performance. Therefore, after model training, we perform model evaluation by using testing data for detection and classification. We evaluate our model in two ways: first, an AE-CNN evaluation comparison with a single CNN model, and it achieved a high score in all evaluation metrics. Second, comparing it with other related work, it also has high classification accuracy and a minimum false alarm rate, as discussed in Chapter 5.

6.2 Contribution

This thesis study makes three primary contributions.

- ✓ The main contribution of this work is creating a new model by optimizing CNN deep learning algorithms with AE, which can handle various types of ARP attacks. We combine state-of-the-art deep learning methods, known as Sparse Autoencoder algorithms, for CNN model optimization by offering reduced and extracted important features for classification tasks, in contrast to previous research described in the related paper. When we compared the classification to previous research, this increased its accuracy.

- ✓ By providing an initial model that may be used as a starting point in ARP attack detection and classification to ensure businesses' data confidentiality, availability, and integrity, this work will also significantly improve the security of LAN environments.

- ✓ It will function as a point of reference for forthcoming investigations and furnish insights for specialists seeking to enhance their areas of expertise in the security of networks.

6.3 Future Works

Lastly, we intend to carry out further work on the following issues:

- ➢ The proposed model can be implemented to analyze and classify real-time network traffic.
- ➢ To further enhance the performance, the proposed model can be implemented using various model hyperparameters and datasets.
- ➢ A comprehensive solution can be achieved by implementing an attack prevention system following the detection and classification results obtained from the proposed model.

References

[1] M. Anathi, K. Vijayakumar, "An Intelligent Approach for Dynamic Network Traffic Restriction Using MAC Address Verification", Published by Elsevier B.V. Computer Communications, Vol. 154, pp. 559–564, 2020.

[2] McLennan M., "The Global Risks Report 2022 17th Edition". Retrieved from, https://tatsigroup.com/fa/wp-content/uploads/2022/02/the-global-risks-report-2022.pdf, Last accessed on June 10, 2023.

[3] Positive Technologies**, "**Business in the crosshairs: analyzing attack scenarios", retrieve from https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Business-at-gunpoint- analyzing-attack-scenarios-eng.pdf**,** Last accessed on June 10, 2023.

[4] Rajwinder Kaur, Er. Gurjot Singh, Suman Khurana, "A Security Approach to Prevent ARP Poisoning and Defensive tools", International Journal of Computer and Communication System Engineering (IJCCSE), Vol. 2 (3), pp. 431-437, 2015.

[5] Eric Vyncke, Christopher Paggen, "LAN Switch Security What Hackers Know About Your Switches", Cisco Press, USA, 2007.

[6] Abdel Salam AM, Elkilani WS, Amin KM, "An automated approach for preventing ARP spoofing attack using static ARP entries", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 5, No. 1, 2014.

[7] Mohamad Fakher, Mohammad Samir, "A Proposed Technique to Prevent ARP spoofing in Local Area Networks", International Journal of Computer Applications, Vol. 175, No. 32, pp. 0975 – 8887, 2020.

[8] B. Prabadevi, N. Jeyanthi, "TSCBA-A Mitigation System for ARP Cache Poisoning Attacks", Cybernetics and Information Technologies, Vol. 18, No. 4, pp. 75-93, 2018.

[9] B. Prabadevi, N. Jeyanthi, "Distributed Denial of Service Attacks and Its Effects on Cloud Environment – A Survey", IEEE 2014 International Symposium on Networks, Computers and Communications, pp. 1-5, 2014.

[10] Abdel Salam AM, Elkilani WS, Amin KM, "An automated approach for preventing ARP spoofing attack using static ARP entries", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 5, No. 1, 2014.

[11] S. Hijazi, M. S. Obaidat, “A New Detection and Prevention System for ARP Attacks Using Static Entry,” IEEE Systems Journal, pp. 1-7, 2018.

[12] D. R. Rupal, D. Satasiya, H. Kumar, A. Agrawal, "Detection and prevention of ARP poisoning in dynamic IP configuration", IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), pp. 1240-1244, 2016.

[13] C. Divya, D Francis Xavier Christopher, “Security against ARP Spoofing Attacks using Bayesian Support Vector Regression”, International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 8, 2019.

[14] G. Madhusri and C.V.G. Rao, “Security against ARP Spoofing Attacks using Convolution Neural Networks”, International Journal of Mechanical Engineering, Vol. 7 No. 1, 2022.

[15] Yuwei Sun, Hideya Ochiai, and Hiroshi Esaki, “Suspicious ARP Activity Detection and Clustering Based on Autoencoder Neural Networks”, Unpublished Master’s Thesis, Graduate School of Information Science and Technology University of Tokyo, 2022.

[16] Yuksel Arslan, “A Solution for ARP Spoofing: Layer-2 MAC and protocol filtering and ARPSERVER” Article, Ankara Science University, August 2017.

[17] Security, “Check Point Research Reports”, https://blog.checkpoint.com/2023/01/05/38-increase-in-2022-global-cyberattacks/ , Last accessed on June 10, 2023.

[18] Roumani, Y.,” Detection time of data breaches”, Computers & Security, Vol. 112, 2022.

[19] Prajapati, S., Noorani, Z. “A Survey on ARP Poisoning and Techniques for Detection and Prevention”, Int. J. Adv. Res. Innov. Ideas Educ., Vol. 3, pp. 594–60, 2017.

[20] Tripathi N, Mehtre BM, “Analysis of various ARP poisoning mitigation techniques: A comparison”, International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014.

[21] Bhupendra Ingre and Anamika Yadav, “Performance Analysis of NSL-KDD dataset using ANN”, Dept of ECE, K L UNIVERSITY, 2015.

[22] ARP Spoofing, https://www.imperva.com/learn/application-security/arp-spoofing, Last accessed on February 22, 2023.

[23] ARP Poisoning: Definition, Techniques, Defense & Prevention, https://www.okta.com/identity-101/arp-poisoning , last accessed on February 22, 2023.

[24] D. Srinath, S. Panimalar, A. J. Simla, and J. Deepa, "Detection and Prevention of ARP spoofing using Centralized Server", International Journal of Computer Applications (0975 – 8887), Vol. 113, No. 19, March 2015.

[25] Request For Comments: 826 (RFC826): "An Ethernet Address Resolution Protocol", November 1982, retrieved from, https://www.rfc-editor.org/rfc/pdfrfc/rfc826.txt.pdf., Last accessed on March 20, 2023.

[26] Walter Goralski, "The Illustrated Network (Second Edition)", Elsevier Inc, 2017.

[27] Singh, J., Dhariwal, S., & Kumar, R., "A detailed survey of ARP poisoning detection and mitigation techniques", International Journal of Computer Technology and Applications, 9(41), 131-137, 2016.

[28] Trabelsi Z. and El-Hajj W. "On investigating ARP spoofing security solutions". International Journal of Internet Protocol Technology, Vol. 5, Nos. 1/2, April 2010.

[29] Trabelsi Z. and Shuaib K., "A novel man-in-the-middle intrusion detection scheme for switched LANs", The International Journal of Computers and Application, ACTA Press, Vol. 3, No. 3. 2008.

[30] ARP entry types: https://techhub.hpe.com/eginfolib/networking/docs/switches/5130ei/5200-3942_l3-ip-svcs_cg/content/483572226.htm, Last accessed on April 07, 2023.

[31] Configuring Static ARP, https://support.huawei.com/enterprise/en/doc/EDOC1100127057/f34f42a1/configuring-static-arp, Last accessed on April 07, 2023.

[32] ARP Configuration, https://www.h3c.com/en/d_201211/761864_294551_0.htm, Last accessed on April 07, 2023.

[33] Bruce Hartpence, "Packet Guide to Core Network Protocols", O'Reilly Media, Inc, June 2011.

[34] Darwesh, G., Vorobeva, A. A., & Korzhuk, V. M., “An efficient mechanism to detect and mitigate an ARP spoofing attack in software-defined networks”, *21*(3), 401-409, 2021.

[35] G. Agrawal and V. Chennai, “Detection and Prevention of ARP-Spoofing Attacks,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 10, 2019.

[36] Abad, C. L., & Bonilla, R. I., “An analysis on the schemes for detecting and preventing ARP cache poisoning attacks”, In *27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07)*, pp. 60-60, 2001.

[37] S. G. Bhirud and V. Katkar, "Light weight approach for IP-ARP spoofing detection and prevention", *Second Asian Himalayas International Conference on Internet (AH-ICI)*, Kathmundu, Nepal, pp. 1-5, 2011.

[38] R. R. Saini and H. Gupta, "A security framework against ARP spoofing," *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, Noida, India, pp. 1-6, 2015.

[39] Ashraf, R. (2022, April 8). “*A list of ARP spoofing tools*. Root Install”, Retrieved from, https://rootinstall.com/tutorial/a-list-of-arp-spoofing-tools/, Last accessed on April 07, 2023.

[40] Tutorialspoint.com. Retrieved from https://www.tutorialspoint.com/arpwatch-tool-to-monitor-ethernet-activity-in-linux, Last accessed on April 24, 2023.

[41] ARP Spoofing attacks- protection from ARP Spoofing with ARP Guard. (n.d.). 3mfuture.com. Retrieved from http://www.3mfuture.com/network_security/arp-guard-arp-spoofing.htm, Last accessed on April 24, 2023.

[42] Alternativeto.net. Retrieved from https://alternativeto.net/software/xarp/about/, Last accessed on April 24, 2023.

[43] Softpedia.com. Retrieved from https://www.softpedia.com/get/Security/Firewall/ARP-AntiSpoofer.shtml , Last accessed on April 24, 2023.

[44] J. Francis and J. Amitha, “A Review on ARP Spoofing Detection and Prevention”, International Conference on Intellectual Property Rights, Idukki, Kerala, India, February, 2021.

[45] Singh, J., Kaur, G., & Malhotra, J.,"A Comprehensive Survey of Current Trends and Challenges to mitigate ARP attacks", In *2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO)* ,pp. 1-6, January,2015.

[46] Grimmick, R., "ARP Poisoning: What it is & How to Prevent ARP Spoofing Attacks", *Varonis.com*. (2021, April 27), Retrieved from https://www.varonis.com/blog/arp-poisonin, Last accessed on April 24, 2023.

[47] Jinhua, G., & Kejian, X., "ARP spoofing detection algorithm using ICMP protocol", In *2013 International Conference on Computer Communication and Informatics* (pp. 1-6), January, 2013.

[48] Fortinet, "*SNORT:* network intrusion detection and prevention system.", Retrieved from https://www.fortinet.com/resources/cyberglossary/snort , Last accessed on April 25, 2023.

[49] Boughrara, A.; Mammar, S., "Implementation of a SNORT's output Plug-In in reaction to ARP Spoofing's attack," 2012 6th International Conference on Sciences of Electronics Technologies of Information and Telecommunications (SETIT), pp.643,647, 21-24, March 2012.

[50] Mohammed, M., Khan, M. B., & Bashier, E. B. M., "Machine learning: algorithms and applications," Crc Press., 2016.

[51] B. G. Becker, "Visualizing decision table classifiers," Proceedings IEEE Symposium on Information Visualization (Cat. No.98TB100258), Research Triangle, CA, USA, pp. 102-105, 1998.

[52] Meseret Kebede, "K-Means Clustering and Random Forest Based Hybrid Intrusion Detection Algorithm", Unpublished Thesis Paper, School of Electrical and Computer Engineering, Addis Ababa University, Ethiopia, 2017.

[53] "A guide to the types of machine learning algorithms." (2023, March 27). Sas.com. Retrieved from https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html, Last accessed on April 25, 2023.

[54] Roza Dastres and Mohsen Soori," Artificial Neural network system", International Journal of Imaging and Robotics, Volume 21; Issue No. 2; Year 2021.

[55] D. Roza, and S. Mohsen, “Artificial Neural Network Systems.” Article *in* International Journal of Imaging and Robotics · March 2021.

[56] “Activation functions in neural networks.” (2018, January 29). GeeksforGeeks, Retrieved from https://www.geeksforgeeks.org/activation-functions-neural-networks/, Last accessed on September 25, 2023.

[57] V7 Labs, “Activation Functions in Neural Networks” retrieved from https://www.v7labs.com/blog/neural-networks-activation-functions, Last accessed on May 10 2023.

[58] Sharma, S., Sharma, S., & Athaiya, A., “Activation functions in neural networks.” *Towards Data Sci*, *6*(12), 310-316, 2017.

[59] Krizhevsky, A., & Hinton, G.,”Convolutional deep belief networks on cifar-10.” *Unpublished manuscript*, *40*(7), 1-9, 2010.

[60] Lecture Note: “Neural Networks: Forward and Backpropagation" Nikhil Sardana, October 2017, Retrieved from https://tjmachinelearning.com/lectures/1718/nn2/nn2.pdf , Last accessed on September 17, 2023.

[61] Shah, D. (2023, April 21). “The essential guide to PyTorch loss functions. “V7labs.com, Retrieved from https://www.v7labs.com/blog/pytorch-loss-functions, Last accessed on September 17, 2023.

[62] Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., & Chavez-Urbiola, E. A., “Loss Functions and Metrics in Deep Learning.” A Review. arXiv preprint arXiv:2307.02694, 2023.

[63] Csiszar, “I-Divergence Geometry of Probability Distributions and Minimization Problems,” The Annals of Probability, vol. 3, no. 1, pp. 146 – 158, 1975.

[64] T. N. N. V. D. P. a. Q. S. N. Shone, ""A deep learning approach to network intrusion detection," IEEE transactions on emerging topics in computational intelligence," vol. 2, pp. 41-50, 2018.

[65] Techtarget.com. Retrieved from https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network, Last accessed on April 26, 2023.

[66] V7 Labs,"Autoencoders in Deep Learnings: Tutorials & use cases." Retrieved from https://www.v7labs.com/blog/autoencoders-guide, Last accessed on April 26, 2023.

[67] Analytics Vidhya, "A Gentle Introduction to Autoencoders for Data Science Enthusiasts", Retrieved from https://www.analyticsvidhya.com/blog/2021/06/autoencoders-a-gentle-introduction/, Last accessed on April 27, 2023.

[68] Sublime, J., & Kalinicheva, E., "Automatic post-disaster damage mapping using deep-learning techniques for change detection: Case study of the Tohoku tsunami." Remote Sensing, 11(9), 1123, 2019.

[69] Michelucci, U., "An introduction to autoencoders." arXiv preprint arXiv:2201.03898, 2022.

[70] Ng, Andrew. "Cs294a lecture notes: Sparse autoencoder." *Lecture Notes: Sparse Autoencoder, Standford, CA, USA: Stanford Univ. Press*, 2010.

[71] Smplilearn, "Convolutional Neural Network Tutorial", Retrieved from https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network, Last accessed on April 27, 2023.

[72] Analyticsvidhya, "Basics of CNN in Deep Learning", Retrieved from https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/, Last accessed on April 27, 2023.

[73] Nsl-kdd. (n.d.). Unb.Ca. Retrieved from https://www.unb.ca/cic/datasets/nsl.html , Last accessed on April 15, 2023.

[74] Kelem Birhane, "Comparison of Machine Learning Technique for Intrusion Detection System", Unpublished Master's Thesis, Telecommunication Engineering Graduate Program, Addis Ababa University, 2018.

[75] Sudhakar, & Aggarwal, R. K.," A Security Approach and Prevention Technique against ARP Poisoning." In Information and Communication Technology for Intelligent Systems (ICTIS 2017)-Volume 1 2 (pp. 39-49). Springer International Publishing, 2018.

[76] Kumar, S., & Tapaswi, S., "A centralized detection and prevention technique against ARP poisoning." In Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec) (pp. 259-264), June 2012.

[77] Arote, P., & Arya, K. V., "Detection and prevention against ARP poisoning attack using modified ICMP and voting." In 2015 International conference on computational intelligence and networks, pp. 136-141, January 2015.

[78] Galal, A. A., Ghalwash, A. Z., & Nasr, M., "A New Approach for Detecting and Mitigating Address Resolution Protocol (ARP) Poisoning." International Journal of Advanced Computer Science and Applications, Vol. 13, No. 6, 2022.

[79] Kaleemsha, P., & Likhitha, P.," Avoiding Man in the Middle Attack Based on ARP Spoofing in the LAN." International Journal of Computer Applications Technology and Research, Vol. 5, No. 5, pp. 249 - 252, ISSN: - 2319–8656, 2016.

[80] Bruschi, D., Ornaghi, A., & Rosti, E., "S-ARP: a secure address resolution protocol." In *19th Annual* Computer *Security Applications Conference, Proceedings*, pp. 66-74, 2003.

[81] Lootah, W., Enck, W., & McDaniel, P., "TARP: Ticket-based address resolution protocol. *Computer networks*," Vol.*5*, *No*. 15, pp. 4322-4337, 2007.

[82] Prabadevi, B., & Jeyanthi, N., "Security solution for ARP cache poisoning attacks in large data centre networks." *Cybernetics and Information Technologies*, Vol.*17*, No. 4, pp. 69-86, 2017.

[83] Hussain, M. A., Jin, H., Hussien, Z. A., Abduljabbar, Z. A., Abbdal, S. H., & Ibrahim, A., "ARP Enhancement to Stateful Protocol by Registering ARP Request." In *2016 International Conference on Network and Information Systems for Computers (ICNISC)*, pp. 31-35, 2016.

[84] Trabelsi, Z., & El-Hajj, W., "Preventing ARP attacks using a fuzzy-based stateful ARP cache." In 2007 IEEE international conference on communications, pp.1355-1360, 2007.

[85] D. Moon, J. D. Lee, Y. S. Jeong, J. H. Park, "RTNSS: a routing trace-based network security system for preventing ARP spoofing attacks", The Journal of Supercomputing, vol. 72, No. 5, pp. 1740-1756, 2016.

[86] Salim, H., Li, Z., Tu, H., & Guo, Z., "Preventing ARP spoofing attacks through gratuitous decision packet." In 2012 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, pp. 295-300, 2012.

[87] Alsallal, M. (2017, January 17), Security Intelligence: "Applying machine learning to improve your intrusion detection system." Retrieved from https://securityintelligence.com/applying-machine-learning-to-improve-your intrusion-detection-system, Last accessed date on September 06, 2023.

[88] Ma, H., Ding, H., Yang, Y., Mi, Z., Yang, J. Y., & Xiong, Z., "Bayes-based ARP attack detection algorithm for cloud centers." *Tsinghua science and technology*, Vol. 21, *No.* 1, pp. 17-28, 2016.

[89] Taye, M. M., "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions." *Computers*, Vol. *12*, No. 5, 91, 2023.

[90] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "AE-MLP: A hybrid deep learning approach for DDoS detection and classification," IEEE Access, vol. 9, pp. 146810–146821, 2021.

[91] M. Radhi Hadi and A. Saher Mohammed, "A novel approach to Network Intrusion Detection System using Deep Learning for SDN: Futuristic approach," in Machine Learning & Applications, 2022.

[92] Ü. Çavuşoğlu, "A new hybrid approach for intrusion detection using machine learning methods," *Appl. Intell.*, vol. 49, no. 7, pp. 2735–2761, 2019.

[93] A. Dawoud, "Deep Learning: Enhancing the Security of Software-Defined Networks", Unpublished PhD dissertation, School of Computing, Engineering and Mathematics, Western Sydney University, June 2019.

[94] Girum Berhanu, "Network Intrusion Detection and classification using Stacked Autoencoder with Random Forest Technique", Unpublished Master's Thesis, Faculty of Computing, Bahir Dar University, 2022.

[95] Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., ... & Zissman, M. A., "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation." In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, Vol. 2, pp. 12-26, 2000.

Appendix A

NSL-KDD dataset: 41 Features Description

No.	Feature Name	Description	Type
Basic features of individual TCP connection			
1	Duration	Length(number of seconds) of the connection	Numeric
2	protocol_Type	Types of the protocol, e.g. ,tcp, udp,etc	Categorical
3	Service	Network service on the destination, e.g., http, telnet, etc.	Categorical
4	src_bytes	Number of data bytes from source to destination	Numeric
5	dst_bytes	Number of data bytes from destination to source	Numeric
6	Falg	Normal or error status of the connection	Categorical
7	Land	1 if connection is from /to the same host/port; 0 otherwise	Binary
8	Wrong_fragment	Number of "wrong" fragments	Numeric
9	Urgent	Number of urgent packets	Numeric
Content features within a connection suggested by domain knowledge.			
10	Hot	Number of "hot" indicators	Numeric
11	num_failed_logins	Number of failed login attempts	Numeric
12	logged_in	1 if successfully logged in; 0 otherwise	Binary
13	num_compromised	Number of "compromised" conditions	Numeric
14	root_shell	1 if root shell is obtained; 0otherwise	Binary
15	su_attempted	1 if "su root" command attempted ; 0 otherwise	Binary
16	num_root	Number of "root" accesses	Numeric
17	num_file_creations	Number of file creation operations	Numeric
18	num_shells	Number of shell prompts	Numeric
19	num_access_files	Number of operations on access control files	Numeric
20	num_outbound_cmds	Number of outbound commands in an ftp session	Numeric
21	is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	Binary
22	is_guest_login	1 if the login is a "guest" login; 0 otherwise	Binary
Traffic features computed using a two-second time window			
23	Count	Number of connections to the same-host as the current connection in the past two seconds	Numeric

24	sev_count	no. of connections to the same service as the current connection in the past two seconds	Numeric
25	serror_rate	% of connections that have "SYN" errors	Numeric
26	sev_serror_rate	% of connections that have “SYN” errors	Numeric
27	rerror_rate	% of connections that have "REJ" errors	Numeric
28	srv_rerror_rate	% of connections that have “REJ” errors	Numeric
29	same_srv_rate	% of connections to the same service	Numeric
30	diff_srv_rate	% of connections to the different services	Numeric
31	srv_diff_host_rate	% of connections to different hosts	Numeric
32	Dst_host_count	count of connections having the same destination host	Numeric
33	Dst_host_srv_count	Count of connections having the same destination host and using the same service	Numeric
34	Dst_host_same_srv_rate	% of connections having the same destination host and using the same service	Numeric
35	Dst_host_diff_srv_rate	% of different services on the current hos	Numeric
36	Dst_host_same_src_port_rate	rate % of connections to the current host having the same src port	Numeric
37	Dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts	Numeric
38	Dst_host_server_rate	% of connections to the current host that have an S0 error	Numeric
39	Dst_host_srv_serror_rate	% of connections to the current host and specified service that have an S0 error	Numeric
40	Dst_host_rerror_rate	% of connections to the current host that have an RST	Numeric
41	Dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error	Numeric

Appendix B

Sample Codes for Model building

Importing Necessary libraries

```
[1]  import math
     import tensorflow as tf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from google.colab import drive
     from tensorflow import keras
     from keras import Model
     from keras.utils import plot_model
     from keras import Sequential
     from keras.layers import Dense, Conv1D, MaxPooling1D, Flatten, Dropout,BatchNormalization
     from keras.losses import MeanSquaredLogarithmicError
     from sklearn.preprocessing import OneHotEncoder, LabelEncoder, MinMaxScaler
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score, f1_score, precision_score,recall_score
     from sklearn.model_selection import StratifiedKFold
     from sklearn.metrics import classification_report
     from sklearn.utils import class_weight
     from keras.optimizers import Adam
     from keras import backend as K
     from keras.metrics import categorical_accuracy
```

AE model for NSL-KDD dataset Feature extraction and dimensionality reduction

```
[41] #AE model for feature extraction and dimentionality reducction
     input_dim = X_train.shape[1]
     encoding_dim = 122
     autoencoder = keras.models.Sequential([
         keras.layers.Dense(encoding_dim, input_shape=(input_dim,), activation='sigmoid'),
         keras.layers.Dense(61, activation='sigmoid'),
         keras.layers.Dense(input_dim, activation='sigmoid') ])
     # Define the MSE and KL Divergence regularization
     from keras.losses import mean_squared_error, kullback_leibler_divergence
     def sparse_autoencoder_loss(y_true, y_pred):
         mse_loss = mean_squared_error(y_true, y_pred)
         average_activation = K.mean(K.sigmoid(y_pred), axis=0)
         # Compute the KL Divergence loss
         kl_div_loss = kullback_leibler_divergence(0.01, average_activation)
         total_loss = mse_loss + kl_div_loss
         return total_loss
     # Train the Autoencoder
     autoencoder.compile(optimizer='adam', loss=sparse_autoencoder_loss)
     autoencoder.fit(X_train, X_train, epochs=10, batch_size=32, shuffle=True, validation_split=0.1)
```

Extract latent features using the trained Autoencoder

```
[43] # Extract latent features using the trained Autoencoder
     encoder = keras.models.Model(autoencoder.input, autoencoder.layers[0].output)
     X_train_encoded = encoder.predict(X_train)
     X_test_encoded = encoder.predict(X_test)
```

```
4178/4178 [==============================] - 5s 1ms/step
465/465 [==============================] - 1s 1ms/step
```

CNN Model for Attack classification

```
# Define the CNN model for attack classification
input_dim_encoded= X_train_encoded.shape[1]
num_classes = len(np.unique(y_train))
from tensorflow.keras.utils import to_categorical
# Convert the target data to one-hot encoded format
y_train_encoded = to_categorical(y_train, num_classes=5)
y_test_encoded = to_categorical(y_test, num_classes=5)

model = Sequential()
model.add(Conv1D(61, padding="same",kernel_size=122, activation="relu",input_shape=(input_dim_encoded, 1)))
model.add(MaxPooling1D(pool_size=(4)))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics= ['accuracy'])
model.fit(X_train_encoded, y_train_encoded, epochs=10, batch_size=32, shuffle=True, validation_split=0.1)
# Get the predicted probabilities for each class
from sklearn import metrics
pred = model.predict(X_test_encoded)
pred = np.argmax(pred,axis=1)
y_eval = np.argmax(y_test_encoded,axis=1)
score = metrics.accuracy_score(y_eval, pred)
```

Appendix C

Some Testing Outputs

Classification results In Performance metrics for AE-CNN

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5338
1	0.99	1.00	0.99	7723
2	0.99	0.99	0.99	1407
3	0.94	0.89	0.91	371
4	1.00	0.50	0.67	12
accuracy			0.99	14851
macro avg	0.98	0.87	0.91	14851
weighted avg	0.99	0.99	0.99	14851

Classification results in confusion matrix for CNN

```
array([[5278,   51,    6,    3,    0],
       [  47, 7498,   67,  110,    1],
       [  10,   60, 1332,    5,    0],
       [   0,   47,    3,  321,    0],
       [   0,    7,    0,    2,    3]])
```

Classification results In Performance metrics for CNN

```
plt.show()
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	5338
1	0.98	0.97	0.97	7723
2	0.95	0.95	0.95	1407
3	0.73	0.87	0.79	371
4	0.75	0.25	0.38	12
accuracy			0.97	14851
macro avg	0.88	0.80	0.82	14851
weighted avg	0.97	0.97	0.97	14851

Declaration

I, Undersigned, declare that this thesis is my original work and has not been presented for degree in any other university, and that all sources of material used for the thesis have been acknowledged.

Declared by: Name: Yeshareg Muluneh

Signature: _____

Date _____

Confirmed by advisor: Name: Solomon Gizaw (PhD)

Signature: _____

Date _____

Place and date of submission: Addis Ababa University, December 19, 2023