

Chapter One

Introduction

1.1 Background to the Study

With the widespread and increasing use of technological advancement and the internet, computing resources have become more affordable, powerful, resourceful, and widely available than ever before. This technological advancement has enabled connectivity, communication, collaboration, and access to remote devices. Also, information storage and accessibility are now easily possible with the use of computer systems, mobile devices, and other electronic devices¹. Through the internet application and continuous development of network technology, in various fields of life, individuals from all over the world can communicate and exchange information^{2, 3}. Hence, individuals, business owners, firms, organizations, and governments are now embracing this fast-evolving digital infrastructure without considering the security issues, such as threats and vulnerabilities it exhibits and sharing their data and information without any thought of loss of privacy^{4, 5}.

However, this technological trend brings new security threats and challenges in terms of safety, availability, and reliability. Since cloud computing is part of the advancement of technology and is internet-based, chances of intrusion and malicious attacks are possible to exploit new vulnerabilities created by the shift from the traditional way of storing, processing, and accessing information, data, and communication to the new environment⁶. The traditional model involves downloading data from a central source, then storing and analyzing it locally, which could be a local computer or servers connected to the client computer using a local area network (LAN) or

wide area network (WAN). The data is stored on disks, which can be reformatted or reprovisioned as and when needed^{7, 8}.

Studies have shown that the innovation of many existing technologies, such as web services, web browsers, virtualization, and service orientated architecture (SOA) have contributed to the development of cloud-based systems. Hence, any intrusion, threat, or attack associated with these technologies also affect the cloud and could even have more effect⁹.

"Cloud computing technology," could therefore be referred to as an emerging area which involves the delivery of computing resources and services through the internet¹⁰. That is, an Internet-based platform for data storage, processing, and transmission, as well as resource sharing, that combines hardware, software, applications, and operational procedures¹¹. The definition of cloud computing, proposed by the National Institute of Standards and Technology in the special publication 800-145, simply referred to cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.

It is nearly impossible in the world of today to ponder the digital evolution of businesses and organizations without cloud computing. Cloud computing possesses some features that differentiate it from traditional web services. Some of these features are resource pooling, virtualization, on-demand self-service, elasticity, automatic or easy resource deployment, and metered billing. All these allow cloud computing to bring more cost savings, automation, and flexibility to users compared to using a traditional web service provider¹². Some of the technologies being rendered and maintained by cloud computing providers include Microsoft

Azure, Amazon Web Services (AWS), Google Cloud, and Rackspace Cloud. These technologies give individuals opportunities to effectively interact and communicate.

Furthermore, the providers of these services are referred to as "cloud service providers" which have been defined as "organizations that provide some form of IT infrastructure that is commercially distributed and sourced across several subscribers, typically businesses." Cloud providers deliver cloud solutions through on-demand, pay-as-you-go systems as a service to customers and end-users¹². Communication in cloud computing technology can be divided into two segments, namely the front end and the back end. The front end is the client, or the computer user, and the back end is the cloud division of the whole system. The personal computer needs access to cloud computing coordination, depending on the interface of the cloud services. The back-end system is made up of computers, a storage system, and servers that create the cloud services and dedicated servers needed for these applications¹³. The Figure 1.1 show the communication architecture in cloud computing.

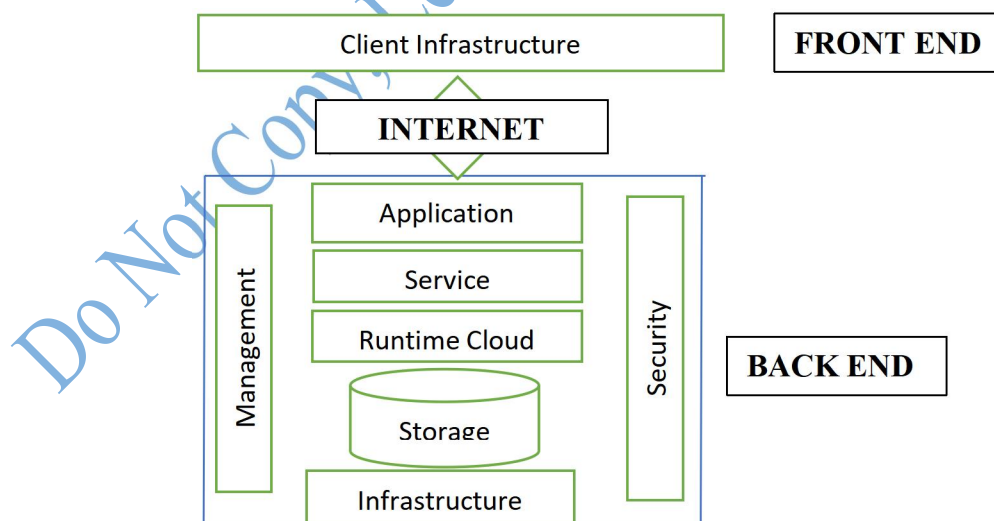


Figure 1.1. Communication Architecture in cloud computing⁴⁵

In cloud computing technology, computing resources are shared rather than having a local server or personal device to handle applications. That is, it enables clients and owners of businesses to access files and utilize software without having to install them on any computer or electronic device with internet access. Without being constrained by space, time, or distance, cloud computing can offer limitless services for computer users to access applications. Three service model types and four deployment model types are generally available. The delivery model of the service includes¹⁴.

- i. Infrastructure-as-a-Service (IaaS) is a service that lets users rent common IT resources, such as memory, computing power, storage, operating systems, and networks. Examples are, Amazon Web Services (AWS), Microsoft Azure, Google Cloud, Digital Ocean, and Linode.
- ii. Software-as-a-Service (SaaS) is a development of the idea of an ASP (Application Service Provider). As suggested by its name, SaaS makes it simple for customers to utilize software resources by subscribing to Examples: Office 365, Zoom, Drive, Dropbox, Cisco WebEx, and GoToMeeting.
- iii. Platform-as-a-Service (PaaS) is a service that offers pre-made modules that may be used to create an application, which naturally can only run on that platform. Examples: an operating system, a developer's platform, OpenShift by Google App Engine

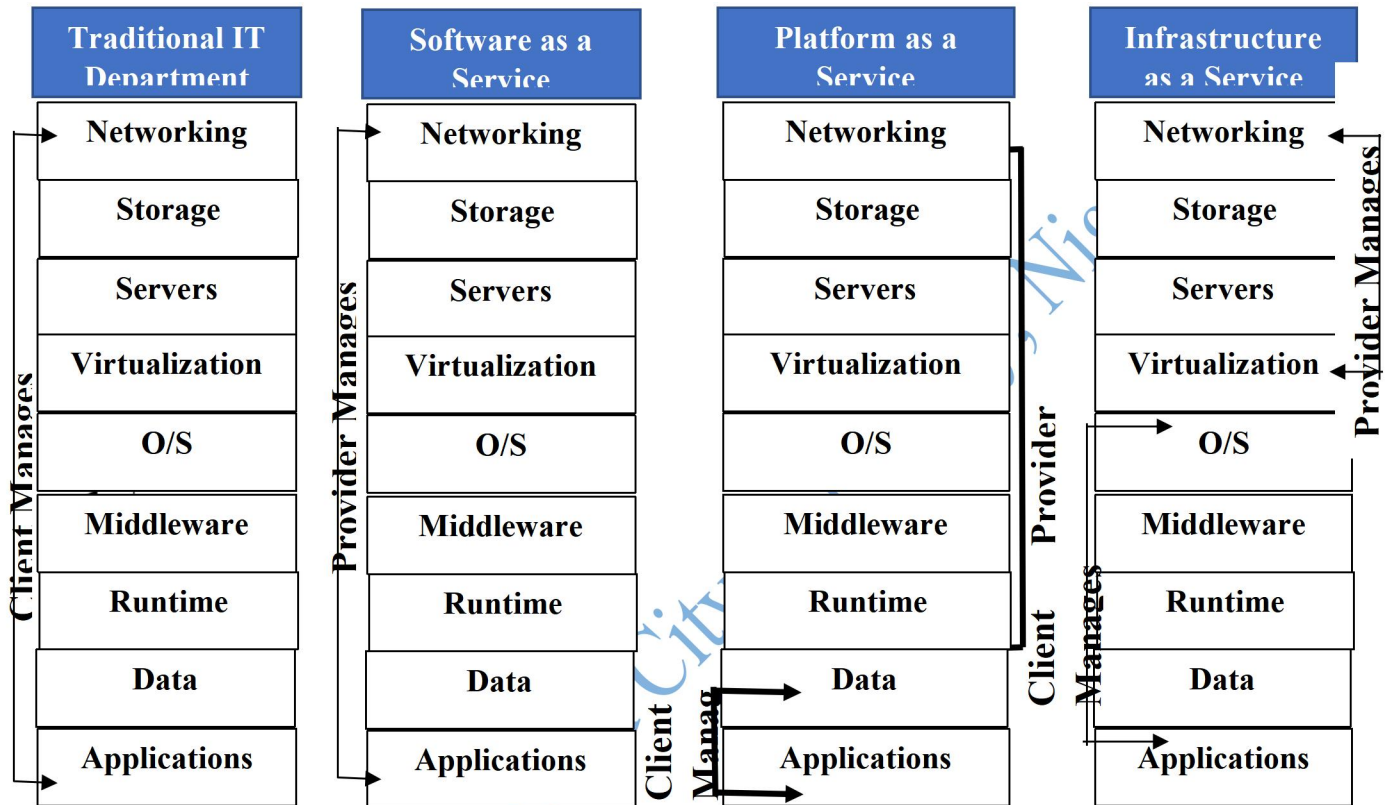


Figure 1.2: The traditional and Cloud Computing deployment models⁴⁶.

Deployment models include public, private, community, and hybrid. Each of these models has different features, and the deployment model depends on the cloud user objectives.

1. Public: The public has access to this infrastructure. As the name implies, the public cloud concept makes resources generally accessible to anybody, anywhere.
2. Private Model: This model was created for private entities like a single household and an organization, which can use it however they see fit. Not everyone uses this type of service.

3. Hybrid: Hybrid clouds combine public and private clouds on the same network. If a private cloud requires certain crucial services from the public cloud, such as information stored on private clouds that can be accessed on public clouds.

Community: Contrary to popular belief, a community cloud is not the same as a public cloud. In contrast to public clouds, where users do not share common interests, community clouds have resources available for a number of individuals or groups with shared interests. The computing infrastructure may be on-site or off-site. In contrast to a public cloud, where resources are owned and controlled by a single provider or owner, cloud resources are owned and managed by one or more of the collaborators in the community. Organizations adopting this cloud service share the same goals, rules for governance, and security standards. Therefore, before accepting a model deployment, the user is encouraged to review the security, reliability, and performance issues associated with the models^{15, 16}.

It is therefore not a surprise that many organizations and companies are increasing their investments in cybersecurity. This is as a result of a gradual increase in cloud computing usage and remote work, which is the common trend most organizations are adopting now¹⁷. Those trends make IT networks and systems, and the data they contain, more vulnerable to threats. attacks, and intrusions that can harm business operations, inflict substantial costs, and damage a company's reputation. Malicious attackers are increasingly focusing on unprotected web apps and systems connected to the internet. and secured, particularly as more people work from home because of the COVID-19 pandemic¹⁷. As a result, an intrusion is an important field of cybersecurity data analytics, defined as any activity that is designed to compromise security (confidentiality, integrity, or availability of a system or network), that is, any internet attack and

attempts by authorized users of the system to gain more privileges, or an authorized user who misuses their privileges. An intrusion may include any of the following¹⁸.

- i. malware or ransomware
- ii. Attempts to enter a system without authorization
- iii. Distributed denial-of-service (DDOS) attacks
- iv. Cyber-enabled equipment destruction
- v. Accidental employee security breaches (like moving a secure file into a shared folder)
- vi. Untrustworthy users are both team members and those outside of your organization.
- vii. Social engineering attacks such as phishing campaigns and other ways of tricking users with seemingly legitimate communication.

As a result, the cloud network security systems are essential, significant, and must not be compromised so that, the data and the information source can be secured from intruders, which could be any of the three types below⁴.

- Masquerader: This is an unauthorized outsider entering a system and exploiting a legitimate user's account.
- Misfeasor: A legitimate user, or an insider, who gains unauthorized access and abuses his privileges.
- Clandestine user: This could be either a person from the inside or outside that takes advantage of having direct or indirect access to the system, which they aim to attack unethically by stealing the data and information.

The cloud architecture technology is vast, and regardless of the type of cloud model to which you are connected, the intrusion process involves an intruder accessing digital content, typically through an open door on a network, regardless of the type and size of the deployment model, whether private, public, or hybrid ⁴. For example, just in the first quarter of the year 2020 (February 2020), Amazon Web Services (AWS) reported about a 2.3 Tbps Distributed Denial of Service (DDoS) attack¹⁹. Also, according to Kaspersky Lab, the number of incidents of intrusion and attacks has gone up in fourth quarter 2021, and the trend is increasing significantly²⁰. More recently, Google reported that one of their cloud customers was targeted with a series of HTTPS DDoS attacks that peaked at 46 million requests per second²¹. From the CISCO report, the recent attacks have targeted 68% of remote users, indicating that hackers are aware of which people are most vulnerable and take the easiest route possible when trying to get access to a network²².

According to the National Institute of Standards and Technology (NIST), an intruder affects or attacks the privacy, integrity, and availability of resources and services in a cloud network environment. From the research on intrusion, lots of security tools and methods have been proposed, used, and implemented to detect intrusion, like snort, encoding, firewalls, encryption, antivirus, and access control, but due to the high level of intelligence of the algorithms used by these hackers and intruders, it renders these security tools and methods incapable and undetectable by the intruders, which has led to data loss and theft. Implementing an intelligence system for detection intrusion in cloud environments for securing cloud network systems from those attacks and intruders is considered the best option to detect new and unknown threats and to protect and reduce any form of intrusion^{23,24}. Because the network is thought of as the foundation of the cloud, security in the cloud is directly impacted by network vulnerabilities.

1.1.1 Intrusion detection system

Intrusion detection is the practice of monitoring system or network activities, servers, workstations, and other IT assets to detect and identify any suspicious activity, malicious action, or policy violations²³. Hence, an intrusion detection system could either be a hardware device or a software application designed to track, monitor, analyze, and respond to malicious activity occurring in a computer system or network for signs of possible incidents of violations of security policies²⁴. The cloud IDS captures packets from the network, examines and analyzes them, and then sends the reports to the cloud administrator according to the analysis. The cloud service provider keeps a log list for storing malicious IP addresses after receiving the report from the cloud IDS. In cloud computing, IDS also issues an alert to the cloud administrator about vulnerable attacks²⁵. Figure 1.3 shows an example of IDS in cloud technology.

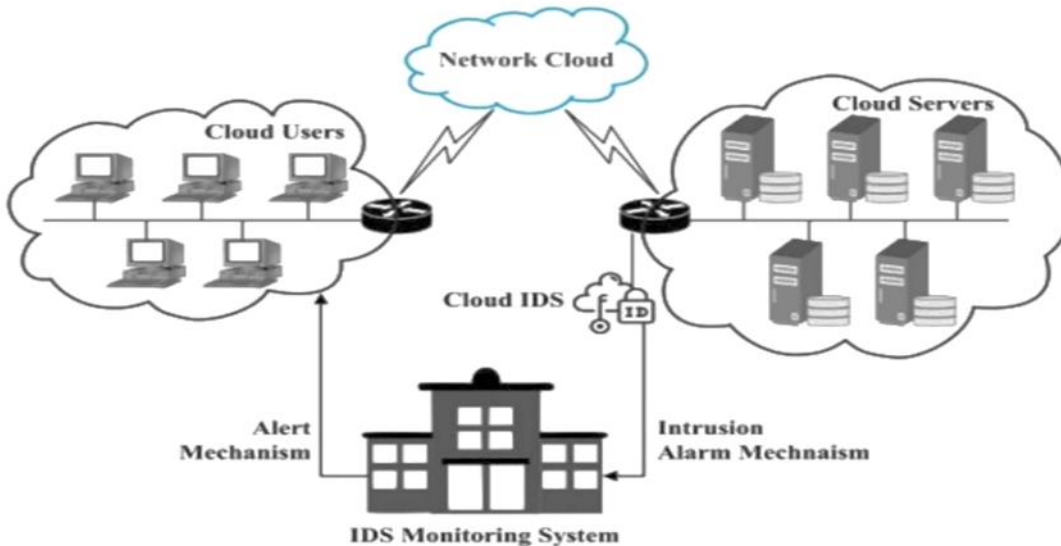


Figure 1.3: IDS in Cloud Environment⁴⁷

The Intrusion Detection System was introduced in the late 1980s by James Anderson's paper, "Computer Security Threat Monitoring and Surveillance". An intrusion detection system (IDS) has been described as an instrument, either a hardware or software device, that sounds an alarm when it detects hostile activity on a network. IDSs serve as the network's "watch-eye". It plays a significant role in the security architecture of contemporary networks. By enabling the early detection of threats, it provides the chance to counteract them. Additionally, they enable the detection of numerous threats, including Man in the Middle (MitM) and Denial of Service (DoS). IDSs have the ability to track and record any and all network activity as needed. Performance metrics are used to assess IDSs, which employ machine learning or deep learning techniques²⁶. There are several properties that are needed to develop an IDS in order to have optimized performance, maximize the detection rate, and minimize errors. These include²⁷.

- i. It runs continuously without human supervision.
- ii. It is fault-tolerant and can recover from crashes.
- iii. adapts to user or system behavioral changes over time.
- iv. It works in real-time.

To that end, due to the high level of intelligence of the algorithms involved, the field of soft computing is currently heavily incorporated in intrusion detection systems. Machine learning and deep learning algorithms are the most commonly used. Therefore, this has led to the development of an intrusion detection system. In order to identify attack patterns with the greatest degree of accuracy and the fewest number of false positives, the use of "deep learning," which is a subset of machine learning, has been the major focus of research.

The “algorithms” could therefore be referred to as rules set by humans, and all the machine can do is strictly implement these rules, while machine learning algorithms are essentially algorithms that summarize rules from data without rules. That is to say, the machine learning algorithm does not directly define the rules required by the machine to work but instead allows the machine to summarize the rules required for the work by itself, and in the face of different data, it will summarize different rules. It can be said that machine learning algorithms "teach a man to fish" rather than "give a man a fish"²⁸.

Machine learning could simply be referred to as a type of artificial intelligence, while deep learning is an especially complex part of machine learning, that is, a specialized subset of machine learning. In general, the learning process of these algorithms can either be supervised or unsupervised, depending on the data fed into the algorithms²⁹. Some of the application areas of ML and DL algorithms are intrusion detection, malware analysis, and spam detection³⁰. The intrusion detection model in the context of this research aims to discover illicit activities within a cloud network environment using a hybrid of deep learning algorithms, to analyze different intrusions and classify them into the appropriate detection family, and improve the detection process. Generally, figure 1.4 shows clearly how an intrusion detection system could be classified based on its deployment and detection techniques. In the cloud environment, the common deployment mode of IDS is usually based on host-based IDS (HIDS), hypervisor-based IDS (HyIDS), network-based IDS (NIDS), and distributed IDS (DIDS).

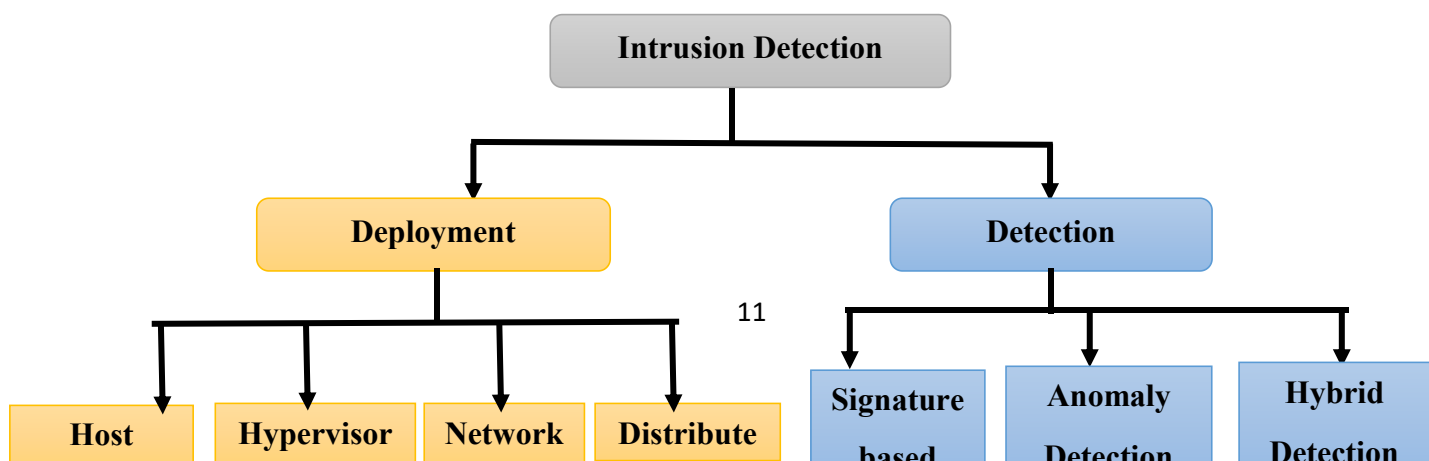


Figure 1.4: Intrusion detection system classification³

1.1.1.1 Deployment Techniques (IDS)

A host-based intrusion detection system (HIDS) is installed on a particular computer or server, known as the host, and it exclusively keeps an eye on activity there. Both signature-based (also known as misuse detection) and anomaly-based detection techniques can be used in host-based intrusion detection systems. When a hacker adds, updates, or deletes a file that is being monitored, HIDS keeps track of its state and alerts the user. Then, when one of the following changes takes place, the HIDS sends out an alert: new files are created, old files are removed, or file attributes are modified³¹.

Hypervisor Intrusion Detection System (HyIDS): This type of IDS runs at the hypervisor layer to monitor communication between VMs, the hypervisor, and the virtual network. VM introspection (VMI) is an example of a hypervisor-based IDS technique. It is different from traditional HIDS as it gives insight into hardware states, software states, and events in the host. The VMI-IDS has greater access to VMM, which allows it to determine changes in the VM state information and monitor events in VMs³².

Network-based Intrusion Detection System (NIDS): A key component of network defense is the use of NIDSs, which are used to identify malicious or unusual network traffic. This track packets through a network as they are sent and received, monitoring the traffic at various points with sensors that essentially keep track of the packets coming in or going out. An analysis of that traffic is then conducted to determine whether or not an intruder is present by examining the behavior of the packets. The analysis might be carried out within the sensors themselves or by transmitting data back to the management server, which analyzes everything^{33, 34}.

Distributed Intrusion Detection System (DIDS): Contains a number of IDS, such as NIDS and HIDS, that are deployed over the network to analyze the traffic for intrusive detection behavior. Each of these individual IDS has two components: a detection component and a correlation manager. The detection component examines the system's behavior and transmits the collected data in a standard format to the correlation manager. The correlation manager aggregates data from multiple IDS and generates high-level alerts that maintain a link to an attack. The analysis phase employs signature-based and anomaly-based detection techniques, allowing DIDS to detect both known and unknown attacks³⁵.

1.1.1.2 Detection Techniques (IDS)

It can be subdivided into signature-based intrusion detection (SIDS), anomaly detection-based intrusion detection (AIDS), and hybrid detection.

Signature-based intrusion detection (SIDS), further known as "misuse intrusion detection" or "knowledge-based intrusion detection," is built on the rules and concept of raises alarms when network events known to be connected with threats are observed in the network. For the purpose

of detecting attacks, data patterns are compared with the stored signatures in the signature database good example of this snort³⁶.

Anomaly detection-based intrusion detection (AIDS), also known as "behavior-based IDS is built around the concept of precisely defining a profile for typical activity. Any departure from this typical profile will be viewed as an abnormality or an aberrant action. Detecting new attacks and unknown is one of AIDS's main advantages. and the fact that the usual activity profile is tailored for various networks and applications³⁷.

Hybrid detection: Hybrid intrusion detection system (HIDS) incorporates signature-based detection techniques (SIDS) and anomaly detection techniques (AIDS) to detect both unknown and known attacks, overcoming the drawbacks of SIDS and AIDS³⁸.

1.1.2 Applications of Machine Learning and Deep Learning

Machine learning (ML) can simply be referred to as a means of empowering computer systems with the ability to "learn. That is, allow computers to learn on their own and produce precise predictions based on the data provided. It is a technique for algorithm training so that the algorithms can learn how to make a decision. Training in machine learning entails giving a lot of data to the algorithm and allowing it to learn more about the processed information. With machine learning, a system can be fed with lots of data so they can learn the difference and write their own algorithms based on the underlying human-driven programming to achieve the desired result. Algorithms are sets of rules that govern specific behaviour³⁹.

Three sorts of ML algorithms are used for wide classification: supervised, unsupervised, and reinforcement learning⁴⁰.

i. Supervised learning uses an algorithm that requires external help. The provided input database is separated into training and testing datasets. The output variable is predicted or classified from the training database. Algorithms attempt to learn some shapes while training the database and then apply these learned patterns to the testing database, yielding estimation results.

ii. Unsupervised learning is an algorithm that learns some characteristics of input information. After providing a new database, it utilizes formerly learned characteristics for the identification of the class of data. It is mostly preferred for feature reduction and clustering.

iii. Reinforcement learning is action-based learning based on decision-making concepts. In this learning, actions are taken based on decisions made so that the outcomes are more valuable at the desired output or favorable condition. However, the learner doesn't have prior information about data. After providing the situation, it learns to decide which action to take according to the given situation. The present and future situation are affected by the learner's decision, i.e., action taken. In nutshell, reinforcement learning exclusively relies on two conditions: delayed outcome and trial-and-error search⁴⁰.

Deep learning is part of a larger family of machine learning approaches that use artificial neural networks (ANNs). Deep learning provides a computational architecture by combining several processing layers, such as input, hidden, and output layers, to learn from data. The main advantage of deep learning over traditional machine learning methods is its better performance in several cases, particularly learning from large datasets⁴¹. Automated representation (abstraction) extraction from the data is the core idea behind deep learning algorithms. To automatically extract complicated representations, deep learning algorithms need a vast amount of unstructured input. These algorithms are mostly driven by the field of artificial intelligence, whose main

objective is to mimic the human brain's capacity for observation, analysis, learning, and decision-making, particularly in very difficult situations

Deep learning algorithms benefit from rising data generation, current improvements in computing capacity, and the expansion of artificial intelligence (AI) as a service. Deep learning enables machines to resolve complex problems even with widely variable, unstructured, and networked data collection. As they gain more knowledge, deep learning algorithms perform better. The objective is to learn a complicated and abstract representation of the data in a hierarchical manner by passing the data through multiple transformation layers. Stacking up the nonlinear transformation layers is the basic idea behind deep learning algorithms⁴². The more layers the data goes through in the deep architecture, the more complicated the nonlinear transformations that are constructed.

1.2 Statement of the Problem

With the increase in general acceptance and the use of cloud technologies across different fields of life and applications, cloud network security has been a major concern. Intrusions and attacks, which many may refer to as "cyberattacks," are growing exponentially, constantly evolving in size and variety, making them difficult to identify. There are various conventional and traditional security tools, methods, and technologies like firewalls, encryption, cryptographic methods, antimalware, antivirus, and access control mechanisms used in detecting and exploiting intrusions or attacks. But these attacks are diverse and are advancing with sophisticated algorithms, which makes them unnoticeable. With all of these traditional security tools, detecting unknown and known attacks, as well as insider intruders, results in a data breach²⁴.

Also, these tools and techniques were not effective in meeting the requirements of cloud network security. This is due to traditional intrusion detection systems' shortcomings, which include a low rate of attack detection, incorrect classification of network anomalies as attacks, and a high false-positive rate among detected attacks ²⁴. Recently, the application of machine learning and shallow learning techniques such as decision trees, Naive Bayes, and support vector machines has been used to improve intrusion detection. However, these methods are prone to error, struggle with identifying relevant data and patterns, and struggle with feature selection and engineering. They are unable to successfully address the enormous intrusion data classification issue, which leads to low recognition accuracy and a high false alarm rate^{43, 44}.

1.3 Aim and Objectives

This study aims to investigate the performance of Deep Belief Network (DBN) and Multilayer Perceptron (MLP) algorithm for detecting intrusion in cloud network environment.

Therefore, this research work is focusing on the use of deep learning techniques, which have been gaining recognition across security domains. The proposed hybrid intrusion detection model, which involves the combination of Deep Belief Network algorithms (DBN) and Multilayer Perceptron (MLP) algorithms, was made possible through a parallel integration technique. This integration led to the development of a singular, optimal model using the voting classifier method that is capable of accurately analyzing network traffic and will address the identified inadequacies, improving the security and detection accuracy of an intrusion in the cloud network environment.

In order to achieve the aim, the specific objectives of this study include the following:

1. develop a hybrid model consisting of deep belief network (DBN) and multilayer perceptron (MLP) for detecting intrusions.
2. optimize the model with high detection accuracy to reduce false positive rate in intrusion detection
3. evaluate the performance of the hybrid intrusion detection model against the conventional intrusion detection model

1.4 Research Questions

The research questions for this work are:

1. Why is it that existing technologies have not really solved the challenges of intrusion in the cloud environment using exiting approach?
2. How should deep learning techniques improve accuracy and other performance measures, such as precision in detecting intrusions in the cloud environment?
3. How should the effect of the hybridization of deep learning algorithms be on intrusion detection in cloud environments using conventional methods?

1.5 Significance of the Study

This study focuses on expanding previous research by combining multi-layer perceptron and deep belief network algorithms for intrusion detection in cloud network systems. It is believed that this will help to improve network security for clouds and that it will also make it difficult for attackers and intruders to circumvent and gain control of the cloud network, plant malware, and gain access to sensitive information with this implementation of the deep learning model.

1.6 Scope of the Study

This research focuses on the use of deep learning techniques to detect intrusion in a cloud network environment. UNSW datasets are employed to assess the performance of our intrusion detection model, and it will be the source of our data. The dataset will be preprocessed. Preprocessing of the data is done in two phases as follows: numeralization of symbolic features and normalization of numeric features. Finally, the datasets will be trained using the deep belief network and multi-layer perceptrons. The voting classifier will be applied to the model's output with aim of developing a high-performance model of intrusion detection

1.7 Limitation of the Study

This study employed the use of secondary dataset which might be directly or indirectly have some effect on the outcome of the performance of our model. The future work should consider the use of primary dataset.

1.8 Operational Definition of Terms

Some of the operational terms used include:

- **Remote device:** This is a piece of hardware that enables a user to control or access a device or information in another location.
- **Vulnerabilities:** These are flawing that cybercriminal can exploit to gain unauthorized access to a computer or network system.

- **Threats:** In cybersecurity, a threat is referred to as a malicious act that seeks to damage information, steal data, or disrupt digital life. This includes computer viruses, data breaches, denial-of-service (DoS) attacks, and other intrusion vectors.
- **Microsoft Azure:** Is a cloud computing platform and web portal that enables you to utilize and control Microsoft's cloud services and resources.
- **Amazon Web Services (AWS):** Is an Internet platform that offers cloud computing solutions that are scalable and affordable.
- **Firewall:** A network security device that is used to monitor and filter incoming and outgoing network traffic based on the security policies of an organization's
- **Encryption:** Is the method by which information is converted into a secret code that hides the information's true meaning.
- **Cryptography:** A method of protecting information and communications by using codes so that only those who are supposed to read and process the information can do so.
- **Cybersecurity:** Is the practice of defending computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks.
- **Confidentiality:** Is a measure that is designed to prevent sensitive information from being accessed by unauthorized parties.
- **Integrity:** Is the act of maintaining the consistency, accuracy, and trustworthiness of data over its entire lifecycle. That is, the data must not be changed in transit, and steps must be taken to ensure it cannot be altered by unauthorized people.
- **Availability:** This is a measure that is employed to make sure information is consistently and readily accessible for authorized parties.

- **Algorithms:** Are a procedure or set of rules used for solving a problem or performing a computation.
- **Network traffic:** This is the volume of data that is currently being transferred across a computer network. You may also call this "data traffic."
- **False alarms:** An alert that incorrectly indicates that a vulnerability is present
- **Hypervisors:** This is also known as a "virtual machine monitor," or VMM, and is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.
- **Signature-based intrusion detection systems:** These are typically used to detect known threats. It operates by using a pre-programmed list of known threats and their indicators of compromise.
- **Anomaly-based IDS:** This alerts you to suspicious behavior that is unknown. Instead of searching for known threats, an anomaly-based detection system utilizes machine learning to train the detection system to recognize a normalized baseline.
- **Datasets:** This contains information for creating the model.

Endnotes

1. J. Jang-Jaccard, S. Nepal & Y. J. Guo, “Cybersecurity Threats in Cloud Computing,” **Australian Journal of Telecommunications and the Digital Economy**, 1, no. 1, 2013, 4.1-4.17.
2. D. Srilatha & N. Thillaiarasu, “Implementation of Intrusion detection and prevention with Deep Learning in Cloud Computing,” **Journal of Information Technology Management**, 15, 2022, 1-18.
3. Z. Wang & X. Chen, “Intrusion Detection-Data Security Protection Scheme Based on Particle Swarm-BP Network Algorithm in Cloud Computing Environment.” **Journal of Electrical and Computer Engineering** 2023, 2022, 1-10.
4. N. S. Bhati, M. Khari, V. Garcia-Diaz, & E. Verdu. “A Review on Intrusion Detection Systems and Techniques.” **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems** 28, 2020, 65-91.
5. M. D. Abdul-Jabbar & Y. A. Alsahib-Saldeem. “State-of-the-Art in Data Integrity and Privacy-Preserving in Cloud Computing.” **Journal of Engineering** 29, no. 1, 2023.

6. V. B. Kosamkar. “*Intrusion Detection System in Cloud Computing: An Overview*,” **International Journal on Recent and Innovation Trends in Computing and Communication** 4, 2016. 164-167.
7. J. M. Dahlquist, S. C. Nelson, & S. M. Fullerton. “*Cloud-Based Biomedical Data Storage and Analysis for Genomic Research: Landscape Analysis of Data Governance in Emerging NIH-Supported Platforms*.” **Human Genetics and Genomics Advances** 4, no. 3, 2023.
8. RF Wireless World. “*Traditional storage vs cloud storage | difference between traditional storage and cloud storage*.” **RF Wireless World.com**. 2012. <https://www.rfwireless-world.com/Tutorials/traditional-storage-vs-cloud-storage.html>
9. K. Hashizume, D. G. Rosado, E. Fernández-Medina, & E. B. Fernandez. “*An Analysis of Security Issues for Cloud Computing*.” **Journal of Internet Services and Applications** 4, no. 1, 2013 5.
10. R. Islam, V. Patamsetti, A. Gadhi, R. M. Gondu, C. M. Bandaru, S. C. Kesani, & O. Abiona, “*The Future of Cloud Computing: Benefits and Challenges*,” **International Journal of Communications, Network and System Sciences** 16, no. 4, 2023. 53-65.
11. R. Amankwah, R. Asianoa, & B. Birago, “*Virtualization and Cloud Computing*,” **International Journal of Computer Applications** 176, no. 37, 2020.
12. W. T. Lai, H. Trancong, & S. Goh, “*Cloud Computing*,” in **A Fresh Graduate’s Guide to Software Development Tools and Technologies**, 2022. <https://www.studypool.com/documents/17457186/a-fresh-graduate-s-guide-to-software-development-tools-and-technologies-cloud-computing>
13. K. Thakur, L. Tao, T. Wang, & M. L. Ali, “*Cloud Computing and Its Security Issues*.” **Application and Theory of Computer Technology** 2, no. 1, 2017. 1-10
14. B. Baharuddin, D. Ampere, H. Febriasari, M. Aulia, R. Sembiring, & A. Hamid. “*Implementation of Cloud Computing System in Learning System Development in Engineering Education Study Program*,” **International Journal of Education in Mathematics, Science and Technology** 9, no. 4, 2021. 728-740.
15. H. B. Patel & N. Kansara, “*Cloud Computing Deployment Models: A Comparative Study*,” **International Journal of Innovative Research in Computer Science & Technology** 9, no. 2, 2021, 45-50.
16. R. Krishnan, “*Security and Privacy in Cloud Computing Security and Privacy in Cloud Computing*” **Western Michigan University**, 2017, 18-19.

17. S. Craig, “*The ultimate guide to cybersecurity planning for businesses*,” **TechTarget Security**, 2023. <https://www.techtarget.com/searchsecurity/The-ultimate-guide-to-cybersecurity-planning-for-businesses>
18. T. Geisle, “*What Is an Intrusion Detection System*,” **Datto A Kaseya company**, 2019. <https://www.datto.com/blog/what-is-an-intrusion-detection-system>
19. BBC News, “*Amazon 'thwarts largest ever DDoS cyber-attack'*,” **BBC News**, 2020. <https://www.bbc.com/news/technology-53093611>
20. Kaspersky, “*DDoS attacks hit a record high in Q4*” **Kaspersky**, 2021. https://www.kaspersky.com/about/press-releases/2022_ddos-attacks-hit-a-record-high-in-q4-2021
21. E. Kiner, S. Konduru, “*How Google Cloud blocked the largest Layer 7 DDoS attack at 46 million rps*,” **Google Cloud**, 2022. <https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps>
22. Secure security, “*Cisco Meraki provides Umbrella SD-WAN connector to enhance cloud security across the SD-WAN network*”, **Secure security**, 2022. <https://www.sourcesecurity.com/news/cisco-meraki-umbrella-sd-wan-connector-co-15001-ga-npr.1643261707.html>
23. A. Picotte, “*Best Intrusion Detection Techniques in Cloud Computing*,” **Uptycs**, 2020. <https://www.uptycs.com/blog/intrusion-detection-in-cloud-computing>
24. H. P. Chaitra, Archana, Khushi, P. Nandini, Sivaraman, & P. Honnavalli. “*Cloud-Based Network Intrusion Detection System Using Deep Learning*.” **ArabWIC 2021: The 7th Annual International Conference on Arab Women in Computing in Conjunction with the 2nd Forum of Women in Research, Sharjah, UAE** 2021, 1-6.
25. S. Manimurugan, P. Manimegalai, P. Valsalan, J. Krishnadas, & C. Narmatha, “*Retracted: Intrusion Detection in Cloud Environment Using Hybrid Genetic Algorithm and Back Propagation Neural Network*,” **International Journal of Communication Systems** 35, no. 16 2020. 1-15.
26. A. D. Khairkar, D. D. Kshirsagar, & S. Kumar. “*Ontology for Detection of Web Attacks*,” **International Conference on Communication Systems and Network Technologies** 2013.
27. M. J. Aljebreen, “*Towards Intelligent Intrusion Detection Systems for Cloud Computing*,” **Florida Institute of Technology**, 2018, 10-11.
28. Q. Deng, N. Yang, & N. Yang. “*Application of Deep Learning Algorithm in Web Page Advertising Design Style*.” **Security and Communication Networks**, 2022, 1-9.

29. A. Wolfewicz, “Deep Learning vs. Machine Learning -What’s The Difference,” **Levity**, 2023. <https://levity.ai/blog/difference-machine-learning-deep-learning>
30. G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, & M. Marchetti, “On the Effectiveness of Machine and Deep Learning for Cyber Security,” **10th International Conference on Cyber Conflict (CyCon)**, 2018, 371-390
31. M. Tiwari, R. Kuma, A. Bharti, & J. Kishan, “Intrusion Detection System,” **International Journal of Technical Research and Applications**, 5, 2017. 38-44.
32. V. Mahajan, & S. K. Peddoju, “Deployment of Intrusion Detection System in Cloud: A Performance-Based Study,” **IEEE Trustcom/BigDataSE/ICSS**, 2017.
33. T. Sommestad, H. Holm, & D. Steinvall. “Variables Influencing the Effectiveness of Signature-Based Network Intrusion Detection Systems,” **Information Security Journal: A Global Perspective**, 31 no. 6, 2021. 711-728,
34. A. Elsadai, J Ibrahim, F. Hajai, & P. Jakic “The Overview of Intrusion Detection System Methods and Techniques,” **Proceedings of the International Scientific Conference Sinteza**, 2019. 151-161
35. S. G. Kene, & D. P. Theng, “A Review on Intrusion Detection Techniques for Cloud Computing and Security Challenges,” **International Conference on Electronics and Communication Systems (ICECS)**, 2015, 227-232.
36. A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, & A. Alazab. “Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machin.” **Electronics** 9, no. 1, 2020. 173.
37. Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, & F. Ahmad. “Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches,” **Transactions on Emerging Telecommunications Technologies** 32, no. 1, 2020. doi: 10.1002/ett.4150.
38. A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, & A. Alazab, “Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine,” **Electronics** 9, no. 1, 2020.
39. H. D. Kasmaei, “Machine Learning Trends and Its Applications, “**International Webinar Baba Farid College-India**. IEEE, 2020. 1-89.
40. N. Sharma, R. Sharma, & N. Jindal, “Machine Learning and Deep Learning Applications-A Vision,” **Global Transitions Proceedings** 2, no. 1, 2021. 24-28

41. I. H. Sarker, "Machine Learning: Algorithms, *Real-World Applications and Research Directions*," **SN Computer Science** 2, no. 3, 2021. 1-21.
42. M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," **Computers** 12, no. 5, 2023.
43. N. Shone, T. N. Ngoc, V. D. Phai, & Q. Shi. "A Deep Learning Approach to Network Intrusion Detection," **IEEE Transactions on Emerging Topics in Computational Intelligence** 2, no. 1, 2018. 41-50
44. T. Su, H. Sun, J. Zhu, S. Wang, & Y. Li. "BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset," **IEEE Access** 8, 2020. 29575-29585,
45. R. Ara, M. A Rahim, S. Roy, Dr. U. K. Prodham. "Cloud Computing: Architecture, Services, Deployment Models, Storage, Benefits and Challenges," **International Journal of Trend in Scientific Research and Development (IJTSRD)** 4, 2020.
46. N. Tissir, S. E. Kafhali, & N. Aboutabit, "Cybersecurity Management in Cloud Computing: Semantic Literature Review and Conceptual Framework Proposal," **Journal of Reliable Intelligent Environments** 7, no. 2 2020, 69-84.
47. M. G. Raj, & Dr. S. K. Pani, "A Meta-Analytic Review of Intelligent Intrusion Detection Techniques in Cloud Computing Environment," **International Journal of Advanced Computer Science and Applications** 12, no. 10, 2021.

Chapter Two

Literature Review

2.1 Conceptual Review

In this twenty-first century, communication through digital processes has become easier, and therefore, every internet user now has access to the pool of information available and stored online. This growth in digital communication and advancements in the networking technology process enable the user to interact, collaborate, and communicate with others without thinking about the distance between them^{1, 2}. The research by Cybersecurity Ventures has shown that,

there will be six billion (6,000,000,000) Internet users by 2022, which amounts to 75 percent of the projected world population of eight billion (8,000,000,000), and more than 7.5 billion Internet users by 2030. From these figures and facts, it could be assumed that the internet has become part of the twenty-first century³.

With the rapid increase in the usage of the internet by professionals and unprofessionals for transactions, communicating, and processing information, and with the expansion of the network, security vulnerabilities have also surfaced. Cyberattacks pose a serious threat to the majority of online information and businesses, which shows that users may be exposed to malicious activities and may result in the loss of privacy, data, and any other information available online. Governments and enterprise business owners are putting in a lot of effort to stop sensitive data theft. In reality, organizations, service providers, and the government take proper care of their privacy and the availability of their information and services online by protecting networks from outside assaults, which has been one of the corporations' top priorities.⁴

The purpose of this chapter, the literature review, was to provide a general overview of the subject matter, the conceptual framework, a review of related works, and a summary of gaps identified in the literature.

2.1.1 Cloud Computing

In this new era, cloud computing technology has made it possible for apps, people, and organizations all over the world to connect to data and computer resources anywhere and at any time, bringing to reality the idea of storing and managing data on virtualized servers. Users of cloud computing can pool resources, including processing power, memory, bandwidth, and

software. Utilizing software or other IT services on demand through the Internet is known as cloud computing. On this platform, users can exchange processing power, bandwidth, memory, storage space, and software.⁵

The act of processing, accessing, storing, and handling data and information contents using web-based computer resources, as opposed to the conventional practice of processing and storing data and information contents in a physical (premise-based) computer system, is known as cloud computing⁶. Although the United States National Institute of Standards and Technology defined cloud computing as a model that enabled convenient, on-demand access to a shared pool of configurable computing resources on a network, Examples of this include networks, servers, storage, applications, and services. Figure 2.1 depicts the concept of cloud computing, in which end users, such as computers, phones, servers, and other electronic devices, can access a variety of services over the internet rather than locally.

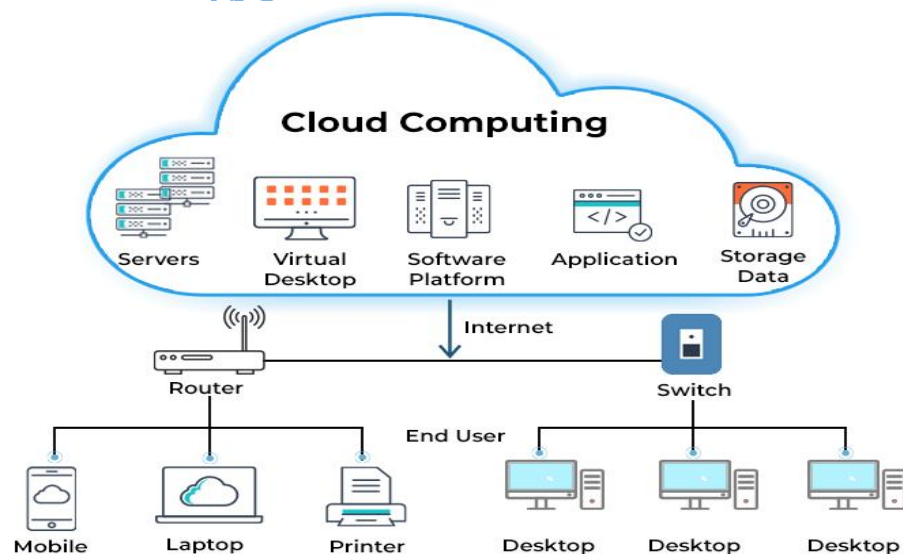


Figure 2.1: Cloud Computing Architecture⁶².

A model called "cloud computing" makes it possible to quickly provision and release a shared pool of reconfigurable computing resources (such as networks, servers, storage, applications, and services) over a network while requiring little management work or service provider interaction.⁶

According to the US National Institute of Standards and Technology, the key five features of cloud computing include;

- Utility-based pricing: Cloud computing makes use of the pay per-use pricing model. The schemes used in pricing differ from service to service. Let's say, for example, that an infrastructure as a service (IaaS) provider rents a virtual machine to a software as a service (SaaS) provider on a per-hour basis. Or, software as a service (SaaS) provides on-demand service to customer relationship management, billing the customers (CRM) based on the number of clients it serves.
- Multi-tenancy: In a cloud environment, it could simply be used to represent services owned by multiple providers that are co-located in a single data center. The management, performance, and issues of these services are shared among service providers and the infrastructure provider. The architecture of cloud computing also allows customers to share pooled computing resources in a public or private cloud.
- Service-oriented: Cloud computing embraces a service-driven operating model.
- Dynamic resource provisioning: One of the key characteristics of cloud computing is that computing resources can be obtained and released on the fly.
- Self-organizing: Since resources can be allocated, service providers are authorized to manage their resource consumption according to their own needs.

These above characteristics guarantee seamless and clear use of cloud services⁷.

Cloud computing comprises different components that make the technology more robust for deploying and delivering. The following are the components:

- Client Computers: This is the end-user device that makes interaction with
- the cloud possible using the client computers.
- Distributed Servers: This is an architecture where servers are distributed among different locations but act like they are working with each other.
- Data Centers: These are facilities that house all of the infrastructures associated with cloud computing⁸.

The Figure 2.2: shows the components as described.

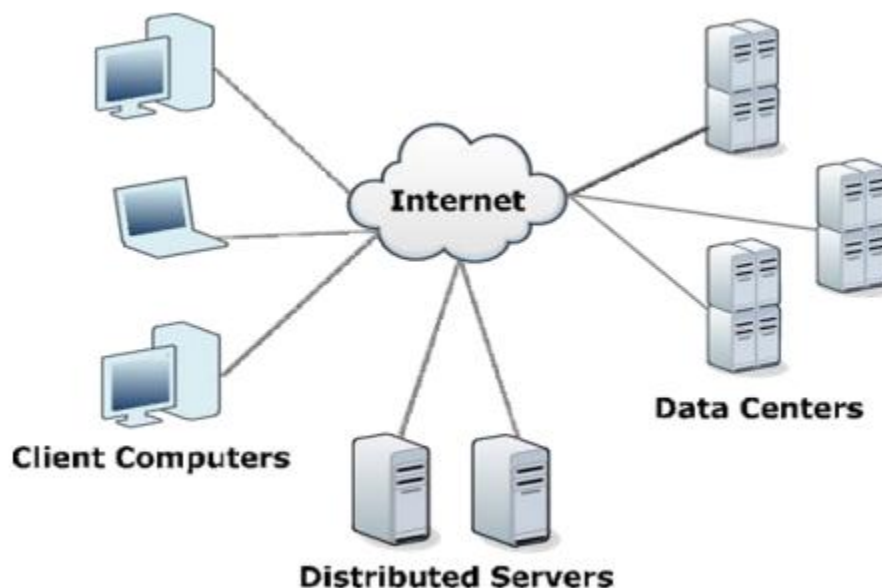


Figure 2.2: Cloud computing Components⁸.

The architecture of cloud computing comprises four main deployment models, which are: Private Cloud: The private cloud model is utilized by a particular company. This cloud architecture is

more secure than others because only authorized users are able to access the organization's system. For enterprises where security is a top concern, this architecture is appropriate. To manage it, either a third party or the organization itself may be liable. The key benefits of a private cloud architecture include cost effectiveness, high security and privacy, more control, and dependability

Public Cloud: It makes it simpler for the general public to have access to the systems and services. All physical resources are believed to be owned and managed by a third party, the cloud service provider. It offers various advantages, including adaptability, high scalability, location independence, and more. This is not the best option for larger businesses. The resources are made available on demand through a pay-per-use system; these clouds are suitable for both medium-sized and small-scale businesses. As all users have public access to all resources, it provides less security than other cloud models⁹.

Hybrid Cloud: This is a combination of clouds that are linked to one another by standardized technology to share data and applications according to ownership and location. It offers more flexibility and greater control over the application by bringing together the advantages of each other while addressing the limitations as well^{10,11}. Hybrid combines the private and public sectors.

Community Cloud: This cloud infrastructure is designed to be used by multiple organizations in a single community that has common interests. Everyone in the community cloud has free access to data and applications. Many other cloud deployment models are being developed because of different needs of different consumers¹⁰.

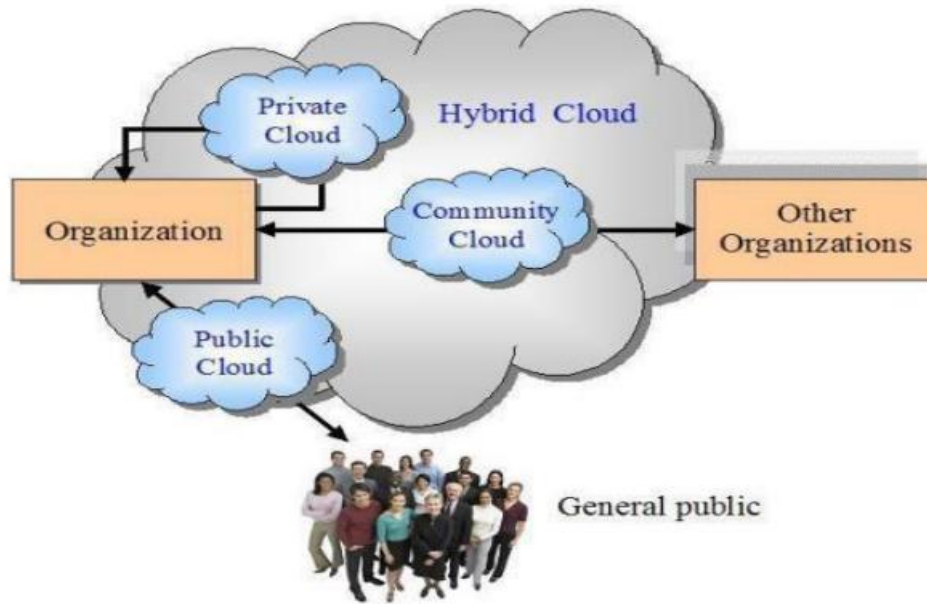


Figure 2.3 Deployment models of Cloud Computing¹⁰.

There are three basic cloud service delivery models, which are software, platform, and infrastructure.¹¹ Software as a Service (SaaS) In this model, the client can gain access to the provider's infrastructure through an interface. One of the main advantages of this model is that the consumer does not need to renew software licensing¹⁰.

In the Software as a Service (SaaS) model, customers have no privileges to manage the cloud infrastructure. Aside from being efficient, the model raises concerns because customers cannot see how data flows and is stored in the cloud. Because the service provider serves multiple clients, the security issues in this model are intended to ensure the security and privacy of client data and information. One of the most common security issues with this delivery model is data security.

Network security constitutes another security threat since the data flow occurs over the network. Hackers take advantage of such vulnerabilities to attack the model¹³.

Platform as a Service (PaaS): This is a development platform. That is, a web-based software application offered as a service for development and deployment by a programmer. PaaS supports a wide range of programming languages, including Java, Dot Net, and Python. Common security challenges associated with PaaS are host vulnerability, fault tolerance, host vulnerability, and lots more¹³. Infrastructure as a service: The basic infrastructure and computing abilities are being provided by the service provider to the customers, or clients. The networks, storage, and other computing resources are rented by the clients, who gain control over them. Although the client has control over some services, such as storage and the operating system, they do not have total control over the cloud infrastructure.

It is critical to gain a better understanding of how all three models work or relate to one another¹². The three models are built on the fundamental foundation of infrastructure as a service, platform as a service, and software as a service. Figure 2.4 shows how the three models relate, the subscriber, the service provider, and their functions. IaaS provides the customer with more security controls. The vulnerability in this model lies in the virtualization techniques. When working with IaaS¹⁰, virtualization security is critical.

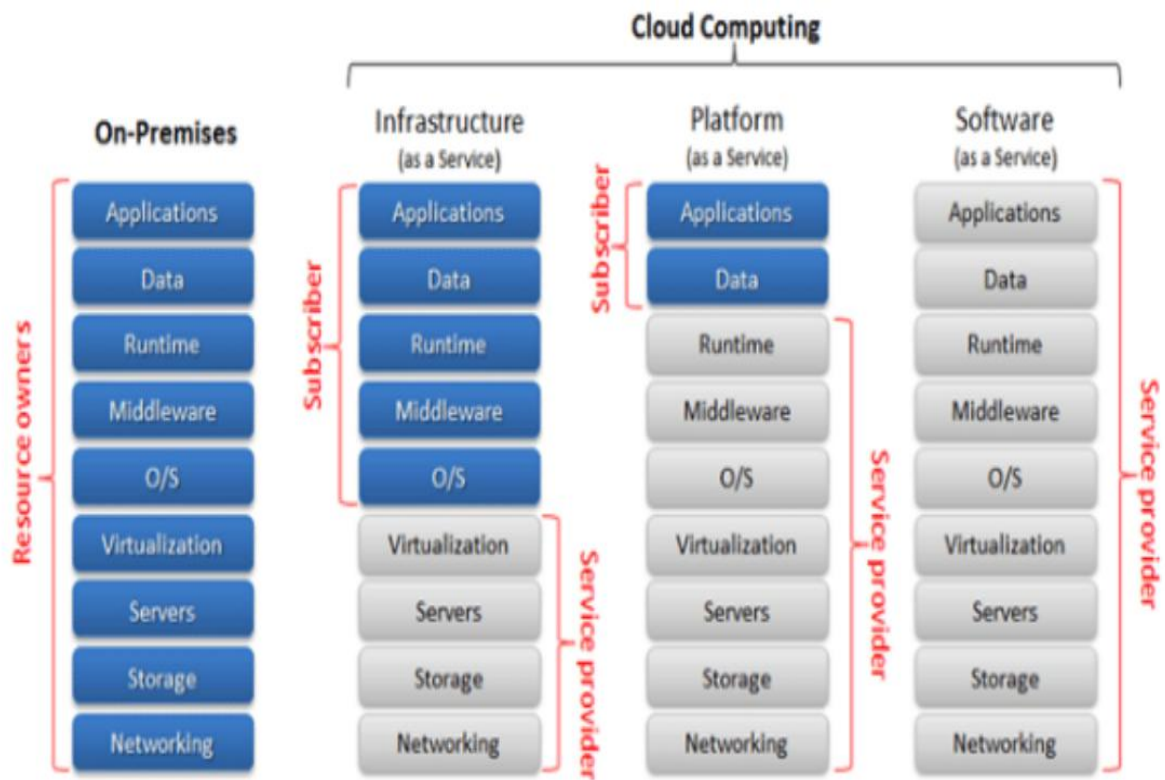


Figure 2.4 Cloud computing delivery models⁹.

2.1.1.1 Security Issues in Cloud Computing

Security and privacy are two major problems with cloud computing. Customers have access to processing power that exceeds what is available in their immediate physical environment in the world of cloud computing. To access this virtual environment, a user must send data through the cloud. There are many security threats as a result¹³. Some related security issues in cloud computing can be grouped into the following categories,

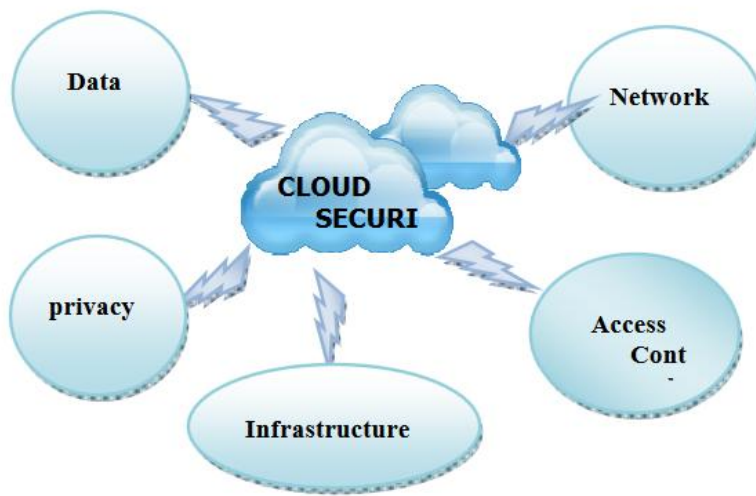


Figure 2.5 Cloud computing security categories¹⁴.

Network security: There are lots of security threats to deal with while considering how to ensure network level security in cloud computing. Proper access control, network integrity, and confidentiality, as well as security against external third-party threats, must all be taken into account.

Access Control: is the fundamental strategy utilized in cloud computing environments to achieve environmental security. Traditional access control primarily consists of a policy description language and an access control model. Role-based access control (RBAC), mandatory access control (MAC), and discretionary access control (DAC) are the three most commonly used access control types. The static environment is preferable for these three models. The dynamic nature of the business environment in cloud computing, particularly with regard to the security and integrity of the resource access subject environment, poses a major security risk to the system. For instance, static access control is used in the conventional token-based approach. The system issues the user a token after the user successfully authenticates their identity. Since each

token uniquely identifies a person, the system will no longer require user authentication for subsequent activities¹⁵.

Infrastructure Security: The different security challenges at various levels in the cloud computing environment, namely host level, application level, and network level, are not exactly caused by cloud computing, rather they are aggravated by its use¹⁴. multi-tenancy, insecure API interfaces, supplier dependability, security misconfiguration, quality of service, server location and backup, and sharing technical flaws are all security issues in cloud infrastructure¹⁶.

Cyber Attack: Cyberattacks are the main security issue with cloud computing. There are many various sorts of attacks on the data, including virus and ransomware attacks, as well as simple configuration errors or shoddy infrastructures. There are numerous difficulties, and we can observe that modern malware is quite versatile and simultaneously attacks several different routes. Understanding it requires taking a somewhat different approach to cloud security because it is built to spread itself so quickly¹⁷. Because the Internet is the major means of communication in cloud-based services, storing data in the cloud can be very risky.

Lack of Transparency: The precise information on how their data will be secured is not provided to a specific business when they purchase a cloud service from a provider as a public, private, or hybrid cloud solution. Because of this lack of service transparency, it is challenging for enterprises to determine whether the data being stored and processed is totally safe. People aren't even certain of the extent to which their data is safe¹⁷.

2.1.1.2 Threats in cloud computing

There are lots of threats in cloud computing. This section enumerates the major threats in cloud computing, which include:

Compromises in credentials and broken authentication: Sometimes organizations struggle with identity management in the process of trying to grant appropriate permissions to the user's job role. In most cases, they forget to delete user access when a job function changes or the user quits the organization. This kind of situation poses a threat to the environment¹⁸

APIs and hacking interfaces: APIs are now available on every cloud service and application technology. These interfaces and APIs are used by the information technology teams to communicate with and control cloud services, including those that provide cloud provisioning, management, and monitoring. Cloud service security and accessibility are reliant on API security. Organizations may need to provide additional services and credentials if third parties rely on APIs and build on these interfaces. Organizations may be exposed to security-related problems like confidentiality, accountability, and availability due to APIs and weak interfaces. Because they may be accessible from the public Internet, APIs and interfaces are the parts of the system that are most exposed¹⁸

Exploited system vulnerabilities: With the introduction of multitenancy in cloud computing, vulnerabilities in systems and exploitable bugs in programs have grown to be more of an issue. As businesses share memory, databases, and resources in close proximity, new attack surfaces are opened up. Compared to other IT expenses, the costs of reducing system vulnerabilities are quite low. Comparing the possible harm to the cost of implementing IT systems to identify and fix vulnerabilities, the potential harm is negligible¹⁸.

Account hijacking: Today, phishing, fraud, and software attacks are all quite common, and cloud services increase the threat because attackers can eavesdrop on activity, influence transactions, and change data. The cloud application may be used by attackers to conduct additional assaults. Organizations must forbid users and services from exchanging account information and, when possible, must allow multifactor authentication methods. Accounts must be kept track of so that each transaction may be linked to a certain person. Protecting account credentials against theft is essential¹⁸.

Permanent data loss: cloud data centers are just as susceptible to natural disasters as any other facility, and hackers have in the past permanently erased data to harm organizations. For enhanced security, cloud providers might advise dividing up apps and data across several zones. Data backup procedures that are adequate and disaster recovery plans are crucial. Utilizing cloud settings requires daily data backups and off-site storage, both of which are crucial. Both the data supplier and the cloud service provider are responsible for preventing data loss. Data can be encrypted by the client before being uploaded to the cloud, but the customer must take care to keep the encryption key safe. The data will also be lost if the key is lost. The length of time that organizations must keep records of audits and other documents is frequently outlined in compliance policies. Such sensitive data loss could have negative effects¹⁸.

Cloud service abuses: Services provided by the cloud can be used to enable actions like using cloud computing tools to crack an encryption key and launch an attack. DDoS attacks, spam emails, and phishing emails are examples of these attacks¹⁸.

To identify DDoS attacks, providers must be able to identify certain forms of abuse. They must also provide customers with tools to check on the health of their cloud environments. Customers

should confirm that service providers provide them with a way to report abuse. Although customers are not the obvious targets of malicious behavior, misuse of cloud services can result in service outages and data loss¹⁸.

DoS attacks: DoS attacks have been known for a while, but the rise of cloud computing has brought them back into the spotlight because they frequently disrupt availability. Systems may be slow or simply time out. These DDoS assaults use a lot of processing power, which the consumer can eventually be charged for. Organizations should be wary of asymmetric and application-level DoS assaults, which target Web server and database weaknesses. High-volume DDoS attacks are relatively common. DoS assaults are better handled by cloud providers than by their clients. The most important thing in this situation is to establish a plan to stop the attack before it happens so that administrators can access those resources when they're required¹⁸.

2.1.2 Intrusion Detection System

An intrusion detection system (IDS) is concerned with managing the events that occur across many networks and organizations and correlating them to identify potential threats, which is essential for guaranteeing network security. IDS monitors activity in a certain environment and classifies it as "malicious" (invasive) or "legitimate" ("normal") based on characteristics learned from network traffic data¹⁹.

An IDS (Intrusion Detection System) is a technique for spotting and tracking intrusive activities and reporting on any security breaches in an IT infrastructure or network. It also analyzes evidence of potential events, such as unpredictable or impending threats of breaching computer network security designs, acceptable implemented policies, or out-of-date safety provisions.

Both signatures and anomalies are used to underpin the two main categories of intrusion detection systems (IDS). While the second method makes use of a strategy for standardizing system operations and raises alarms for obvious abnormalities, the first method makes use of a repository that contains known malicious activity signatures along with generating an alert if communication over the network fits a specific signature²⁰.

An intrusion detection system (IDS) The practice of keeping an eye on and evaluating the activities taking place within a computer system or network for indications of potential incidents breaches or potential threats of violations of computer security policies is known as an intrusion detection system (IDS). IDS is used to detect various malicious behaviors that endanger computers and other electronic devices. Such acts consist of malware (viruses, Trojans, and worms) as well as network attacks against susceptible services, assaults against privilege escalation, unauthorized access to sensitive files, and so forth²¹.

The intrusion detection methods could rely on anomaly or signature detection. The network's packet flow is monitored via signature-based detection, in this method the incoming network pattern is compared to previously discovered, configured signatures of known threats. When compared with events that reveal a deviation from the normal user parameters, the anomaly detection technique can identify assaults as normal and abnormal user behavior^{22,24}.

The intrusion detection system (NIDS) can be network-based or host-based. Network administrators use host-based intrusion detection systems to track and assess activity on a specific computer. HIDS frequently outperforms NIDS in this regard, as encrypted information can be accessed while traveling over a network. Due to the necessity of establishing and managing information for each host, HIDS has the drawback of being highly challenging to

handle. Additionally, certain kinds of denial-of-service attacks can disable HIDS. On the other hand, NIDS are intelligently distributed intrusion detection systems that can be hardware- or software-based. They passively watch over network traffic through the devices they are installed on. NIDS have two interfaces: one for controlling and reporting, and another for listening in on network conversations. The benefit of NIDS is that it may only take a few well-suited NIDS to monitor a large network, and since most NIDS are undetectable to most attackers, they offer protection from attacks. However, NIDS have the disadvantage of having difficulty detecting an attack stroke during periods of high traffic^{23, 24}.

The initial intrusion detection systems were mostly signature-based, which means that the ability to detect malicious activity relies on the configuration of pre-defined and known signatures. signatures of recognized assaults, but this is a significant setback because the list of recognized attack signatures for such infiltration Since invaders are continually changing, detection systems must be updated. systematically seek out ways to exploit network activity^{24, 25}.

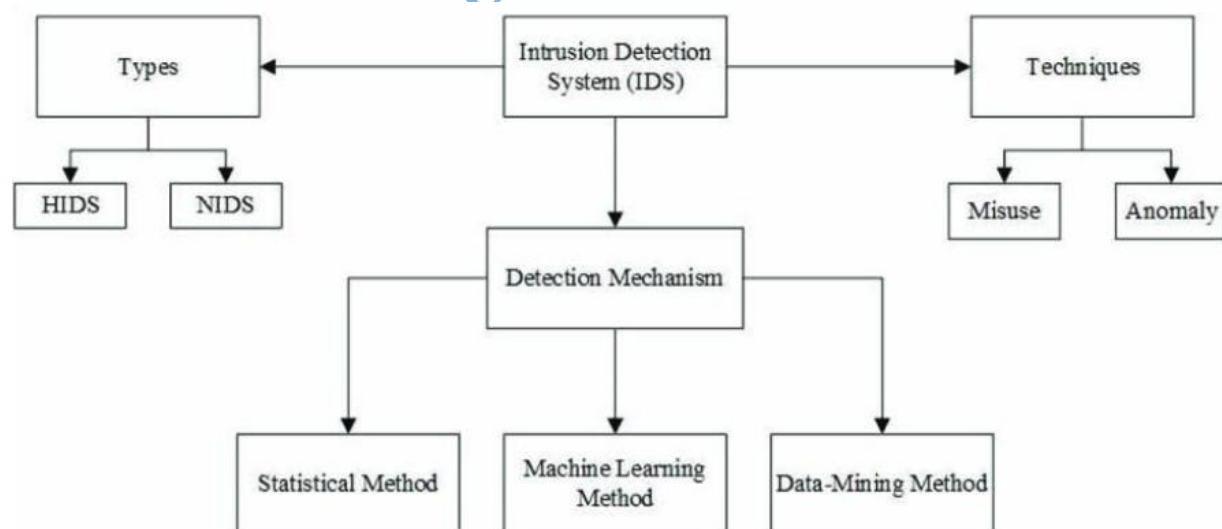


Figure 2.6 Intrusion detection system⁶³.

Ongoing research is being done on novel technologies that can detect improper system usage automatically. Additionally, Denning announced the creation of an intrusion detecting model, which he said may serve as the framework for a multipurpose IDS²⁶. Since then, professionals have created and implemented a number of techniques for automating the network ID procedure. For this goal, they have also continuously sought out approaches that are more precise, quick, and scalable. By 2020, it is anticipated that there will be more than 26 billion linked devices as the "IoT" and "big data" eras arrive²⁶. The variety and quantity of cybersecurity challenges are anticipated to rise along with this trend. The difficulties with IDS are summarized in Figure 2.7. These issues are the false alarm rate, low detection rate, unbalanced datasets, and response time.

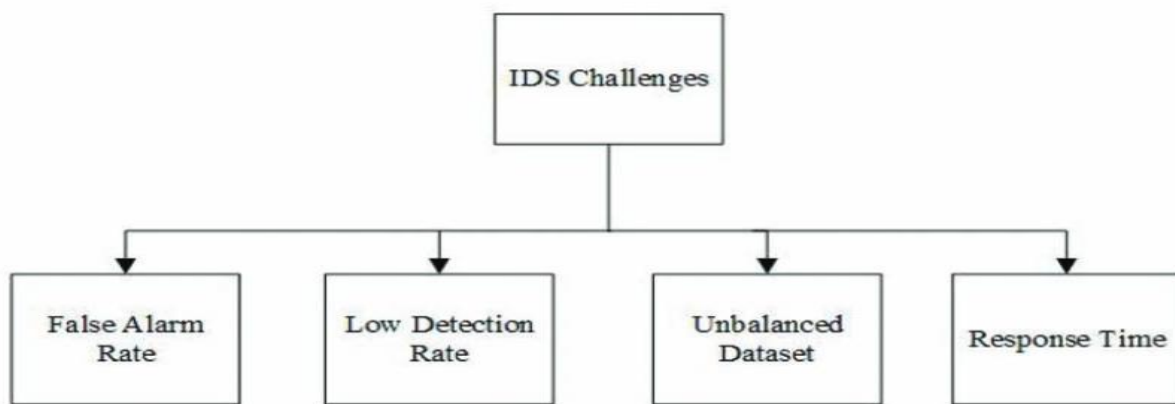


Figure 2.7 Intrusion detection system (IDS) challenges⁶³.

2.1.3 Deep learning

Due to advancements and the advent of deep learning, there has been a revolution in machine learning applications in recent years. Deep learning models are superior to traditional machine learning models because they include more learning layers and a greater level of abstraction. This benefit is further supported by the fact that all components of the model directly learn from the data. Traditional machine learning models have limitations due to the method they use to

process data, the increasing volume of data, and the increased desire to derive useful insights from the data. The increase in data volume has led to the development of more sophisticated learning algorithms that are quicker and more accurate. Every firm will undoubtedly utilize a model like this one that produces the most accurate predictions in order to stay ahead of the competition²⁷.

Deep learning, a type of machine learning algorithm that takes its cues from the biological function of neural networks, is gaining traction and outperforming more traditional machine learning methods²⁷. It is only possible because of their capacity to deliver quicker and more precise findings. It makes an effort to simulate high-level abstraction in data using multiple layers of neurons consisting of complex structures or non-linear transformations^{27, 28}.

All components of the model are directly learned from the data using deep learning approaches. Lowest level features that provide an appropriate representation of the data are used as a starting point. After that, it offers higher-level abstractions for each unique issue that it is applied to. The more training data there is, the more effective deep learning becomes. The advent of Deep Learning (DL) in particular has sparked an increased interest in using Machine Learning (ML) solutions in a number of fields, and the demand for sophisticated prediction services and applications has significantly increased^{27,29}.

Deep learning models employ numerous layers made up of both linear and non-linear transformations. Traditional machine learning algorithms have shown their limitations in analyzing data with increasing size or with innovations in the field of big data. In this analysis challenge, deep learning algorithms have produced improved outcomes. This method has been

hailed as a technological breakthrough because it distinguishes itself from traditional machine learning approaches²⁷.

In modeling the intricate relationship between data, it is helpful. It is predicated on learning data representations rather than working on task-specific algorithms. You have the option of semi-supervised, unsupervised, or supervised learning.

Feature extraction and transformation are carried out by numerous layers of nonlinear processing units in deep learning models. Every layer uses its input as the output of the layer below it. It is utilized in supervised classification challenges as well as unsupervised pattern analysis applications.

Feature extraction and transformation are carried out by numerous layers of nonlinear processing units in deep learning models. Every layer uses its input as the output of the layer below it. It is utilized in supervised classification challenges as well as unsupervised pattern analysis applications. A hierarchy of notions is created by the several levels that offer high-level abstraction. Artificial neural networks that are structured layer-by-layer in deep generative models are the foundation of many deep learning models. The production of observed data through the interaction of multiple components is the idea underlying this distributed representation. These stacked components enable high-level abstraction. By adjusting the number of layers and layer size, a variable level of abstraction can be achieved^{30,27}.

Researchers have created a variety of deep learning models that enable improved learning from the representation of vast amounts of unlabeled data. In the fields of computer vision and predictive analytics, some well-known deep learning architectures, such as convolutional neural

networks (CNN), deep neural networks (DNN), deep belief networks (DBN), and recurrent neural networks (RNN), are employed as predictive models. Deep learning models are demonstrating their abilities in the task of predictive analytics to solve data analysis and learning challenges caused by the increase in data quantity and the need to produce a quick and accurate result²⁷.

2.1.3.1 Deep Believe Network (DBN)

Artificial neural networks called Deep Belief Networks (DBNs) are utilized for both supervised and unsupervised learning tasks. It is a useful way for addressing issues from deep neural networks, such as poor velocity and the overfitting phenomenon in learning. They are made up of several layers of shallow neural networks called restricted Boltzmann machines (RBMs), which can be taught using unsupervised learning. Up until the final layer of the network is reached, the output of the RBMs is used as the input. A classifier that has been developed using supervised learning often makes up the DBN's last layer³¹. An efficient deep learning-based learning model is the deep belief network (DBN). DBN has found effective applications in computer vision and big data processing as a result of its excellent hierarchical feature extraction capacity. The gradual learning approach has also aided in the discovery and modeling of complex systems³². The RBMs, also referred to as restricted Boltzmann machines, make up the deep belief networks. Instead of employing a single layer of RBM, deep networks, or DBNs, stack multiple layers of RBM on top of one another, as shown in the figure 2.8.

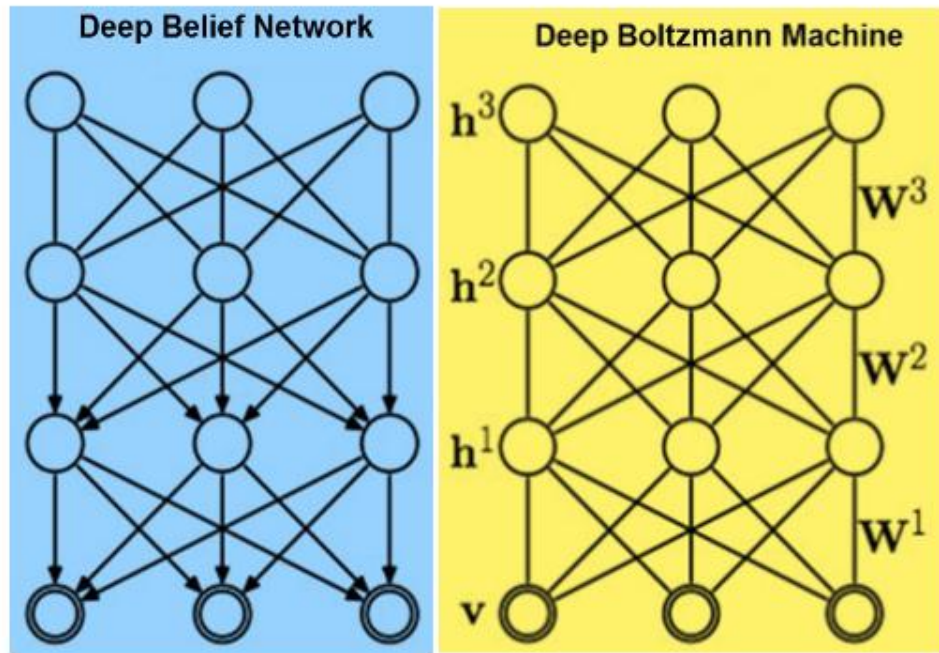


Figure 2.8: Mechanism of Deep Belief Network and Deep Boltzmann Machine³³.

Another kind of deep learning-based deterministic neural network is the DBN, which intelligently selects weights in accordance with the activation functions, as depicted in Figure 2.9.

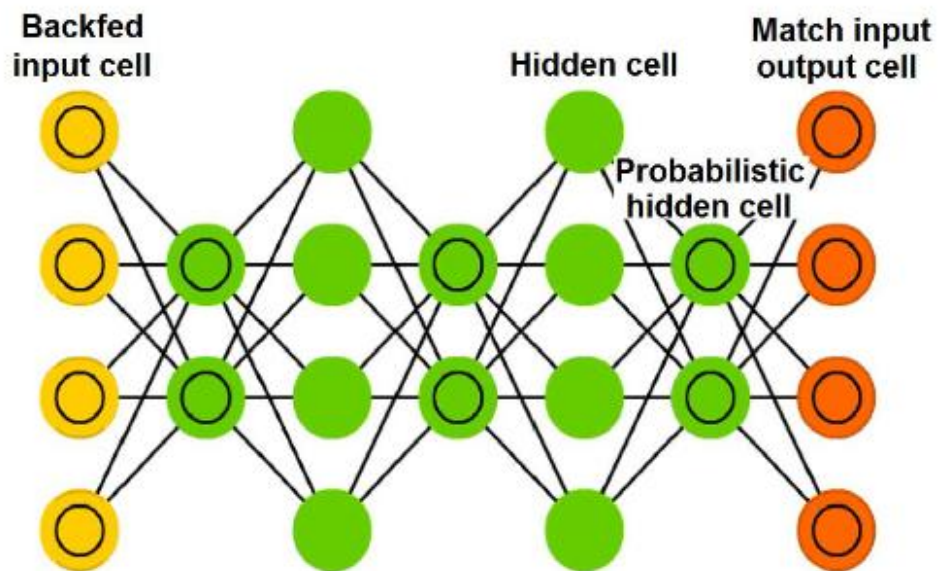


Figure 2.9: Architecture of Deep Belief Network (DBN)³³.

Deep learning is mostly used to obtain a hierarchical representation of the data for a specific issue, such as classification, regression, grouping, and ranking. Of course, one such method for putting deep learning systems into practice is a DBN. With the help of constrained Boltzmann machines built on top of one another, the layers of a DBN learn a hierarchical representation³³. Simply put, according to the DBN's characteristics, a constrained Boltzmann machine can be found at every level. A DBN is often trained using an unsupervised method, level by level. After the initial RDM has been trained, a second one is added on top, and the process is repeated. Similar to how auto-encoders are put together to train stacked auto-encoders, RBMs are put together to design DBNs. DBNs are capable of handling changing data distributions because they are generative probabilistic models that generate the latent representation at the current level by using the one from the previous level as their input³⁴.

2.1.3.2 Multilayer Perceptron (MLP)

A type of feed-forward artificial neural network is called a multi-layer perceptron (MLP). An input layer, a hidden layer, and an output layer are the three node layers that make up an MLP. Each node, with the exception of the input nodes, is a neuron that employs a nonlinear activation function. Back propagation is a supervised learning method used by MLP for training³⁵.

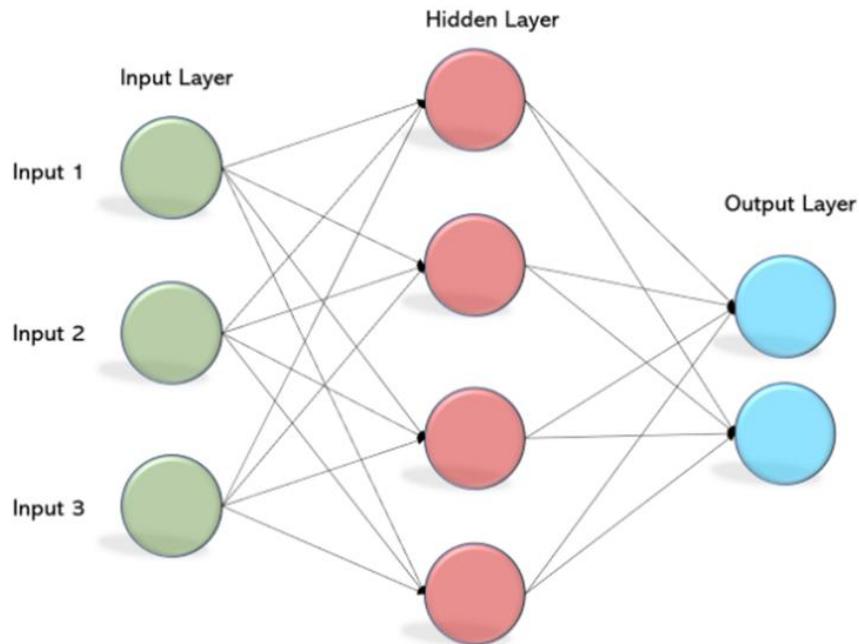
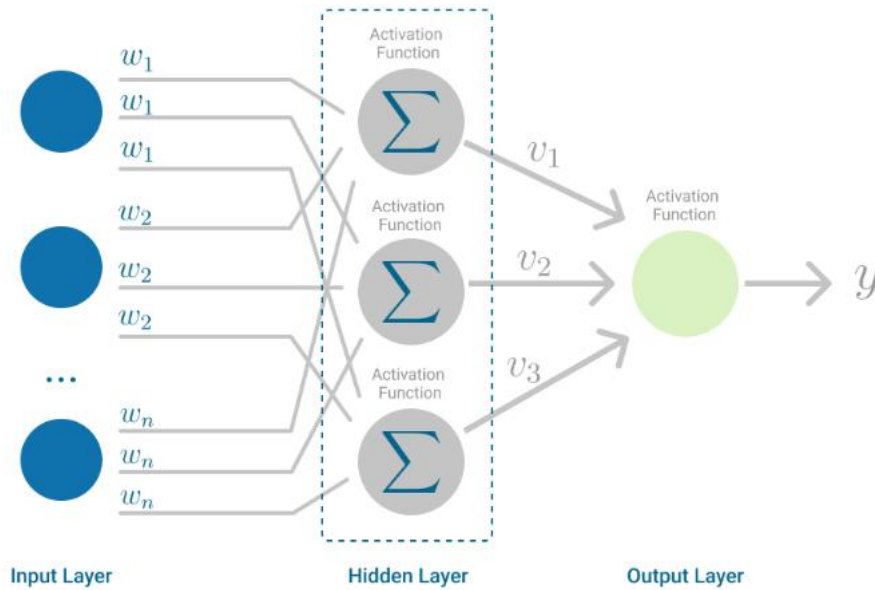


Figure 2.10. Schematic diagram of multi-layer perceptron (MLP)³⁵.

Although the concept has been around since the 1980s, deep learning as a methodology has sparked renewed interest in MLP because it consistently outperforms other leaning approaches such as logistic regression and decision trees on financial services data. In attempted to put this idea into practice using a genuine financial services data set in order to determine whether or not the MLP models can accurately categorize binary situations³⁵.

Multilayer Perceptron. It is a neural network with a non-linear input-to-output mapping. An input, an output, and one or more hidden layers each with several neurons layered together make up a multilayer perceptron. The neurons in a multilayer perceptron can employ any arbitrary activation function, in contrast to the neurons in a perceptron, which must have an activation function that enforces a threshold, such as ReLU or sigmoid³⁶.



F

figure 2.11 Multilayer Perceptron³⁶.

Given that inputs and initial weights are mixed in a weighted sum and are both subject to the activation function, multilayer perceptron's fall within the category of feedforward algorithms. However, the distinction is that each linear combination is carried over to the following layer.

The output of each layer's computation and internal representation of the data are fed to the layer below it. This passes through all hidden layers and ends at the output layer.

The method would not be able to discover the weights that minimize the cost function if it merely computed the weighted sums in each neuron, transmitted its findings to the output layer, and stopped there. If the algorithm only computed one iteration, there would be no genuine learning.

The multilayer perceptron's learning technique, backpropagation, enables the network's weights to be iteratively adjusted with the aim of reducing the cost function. Backpropagation must meet one hard criterion in order to function effectively. Both the threshold function, such as ReLU,

and the function that mixes inputs and weights in a neuron, such as the weighted sum, must be differentiable. Because gradient descent is frequently the optimization function used in multilayer perceptron's, these functions must have a limited derivative³⁶.

2.1.4 Ensemble technique

Therefore, to improve the accuracy (estimate) and performance of our model, ensemble learning techniques are developed. Ensemble is a concept used by machine and deep learning in which several models are trained using algorithms. It combines low performing classifiers (also called weak learners or base learners) and individual model predictions for the final prediction. Speaking about base learners, ensemble techniques can be classified as either homogeneous or heterogeneous ensemble techniques. If the base learners are the same, then it is a homogeneous ensemble technique. If the base learners are different, then it is a heterogeneous ensemble technique³⁷.

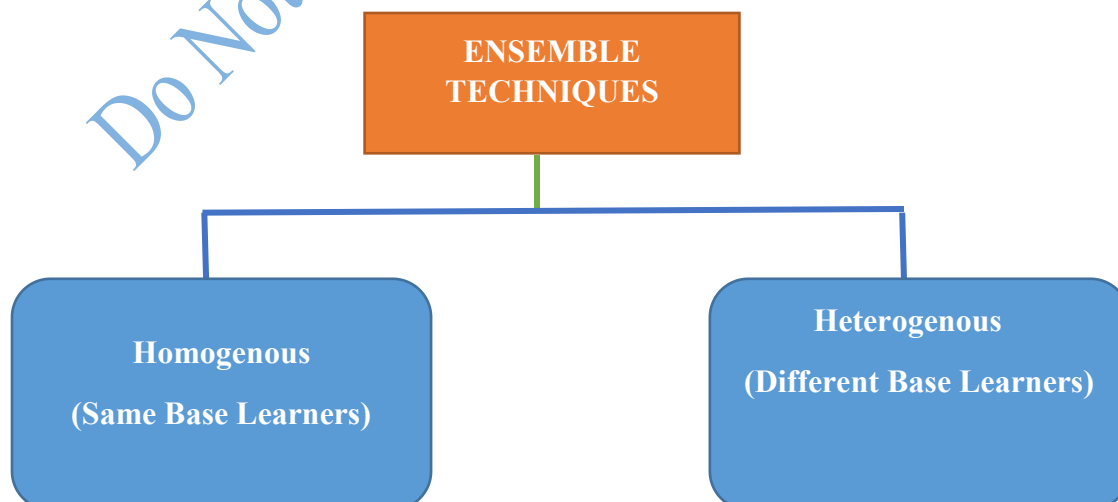


Figure 2.12. Categories of Ensemble Method³⁷

The learning techniques in ensemble algorithms combine the predictions from two or more models. That is an approach used by machine learning and deep learning that seeks to improve predictive performance by combining the predictions from multiple models³⁷. The most prominent techniques that are frequently utilized are described below.

- **Bagging Ensemble Learning:** The bootstrap aggregating (bagging) was developed to enhance the classification performance of ML models by combining the predictions from randomly generated training sets. Basically, the bagging method involves splitting the training data for each base learner using random sampling to generate different subsets used to train b base learners. The b base learners are then combined using majority voting to obtain a strong classifier. Bagging focuses more on reducing the variance in the ensemble members than the bias. Therefore, bagging performs optimally when the ensemble members have high variance and low bias. A major advantage of bagging is that it efficiently decreases the variance without increasing bias. Other advantages of bagging include its ability to introduce diversity in the input data because of the bootstrapping approach³⁸.

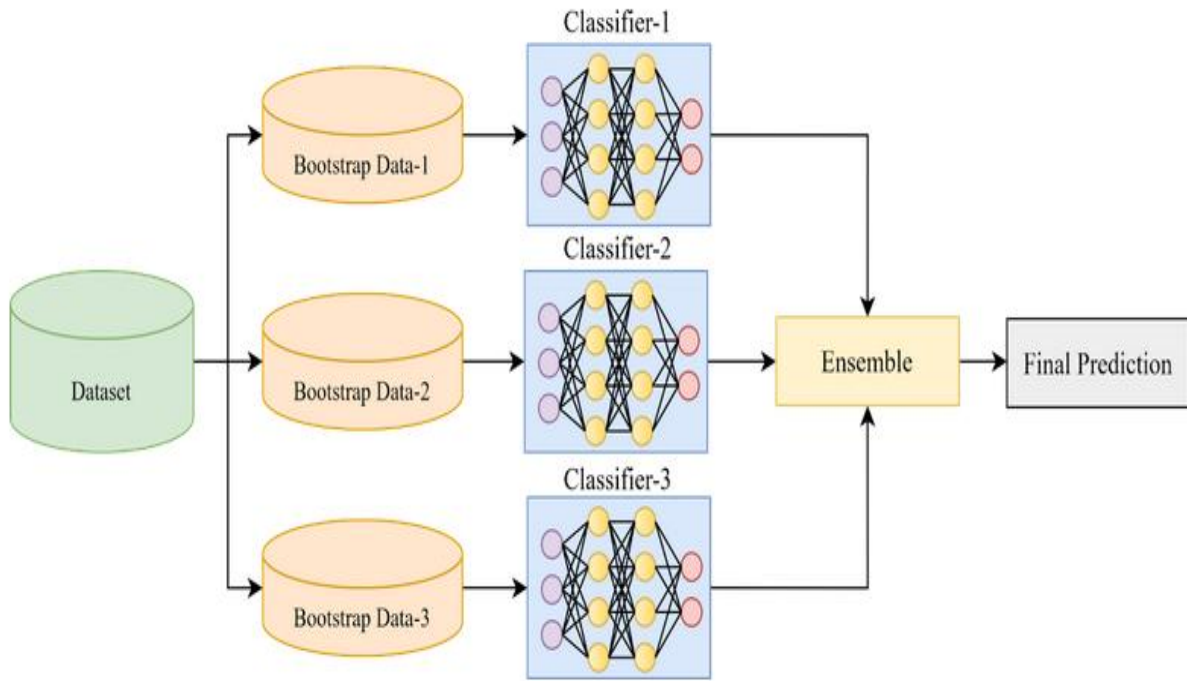


Figure 2.13. Bagging Ensemble Learning⁴¹

- Boosting Ensemble Learning:** Boosting is a machine learning technique capable of converting weak learners into a strong classifier. It is a type of ensemble meta-algorithm used for reducing bias and variance. Meanwhile, a weak learner is a classifier that performs a bit better than random guessing, while strong learners are those that attain good accuracy and they are the core of which the boosting ensemble algorithms are built. The main idea behind boosting involves iteratively applying the base learning algorithm to adjusted versions of the input data in particular, boosting techniques use the input data to train a weak learner, compute the predictions from the weak learner, select misclassified training samples, and train the subsequent weak learner with an adjusted training set comprising the misclassified instances from the previous training round. The iterative learning process continues until a predefined number of base learners is obtained,

and the base learners are weighted together. Boosting focuses more on reducing bias than variance³⁸. Therefore, it enhances base learners with a high bias and low variance, such as decision stumps (a decision tree with one internal node)

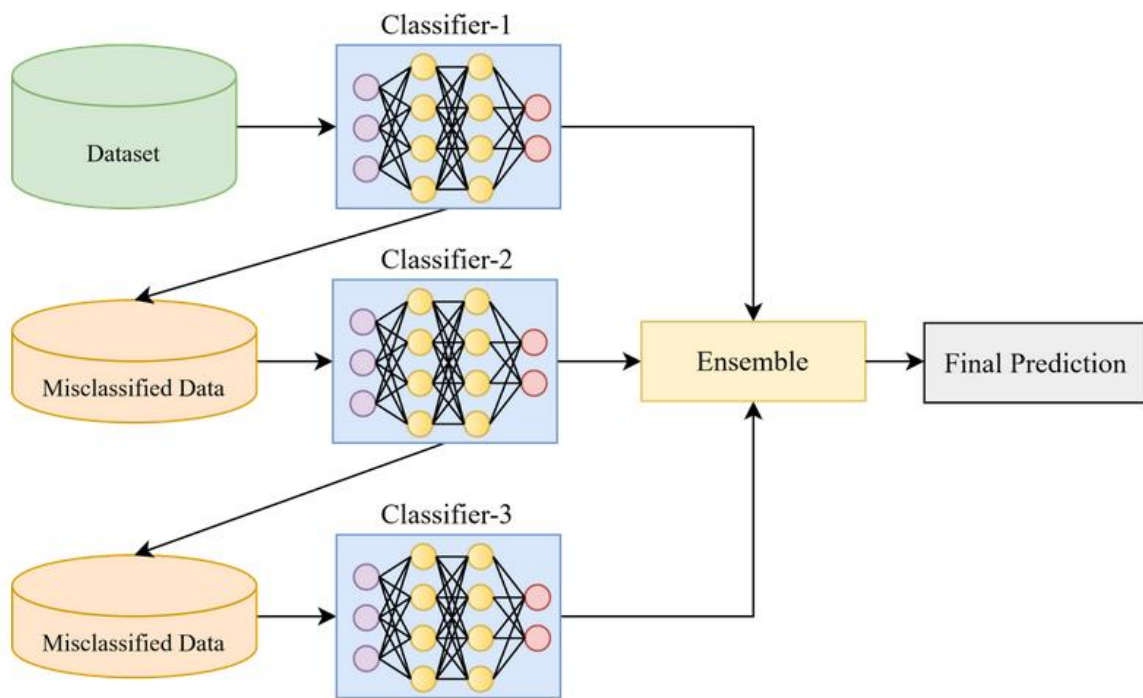


Figure 2.14. Boosting Ensemble Learning⁴¹

- Stacking Ensemble Learning: (Stacking) is an ensemble learning framework that trains a separate ML algorithm to combine the predictions from two or more ensemble members. It was introduced in 1992 by Wolpert to reduce the generalization error in machine learning problems. Stacking is useful in situations where several ML models are uniquely skillful on a particular task; then, the stacking approach would use a separate ML model to learn when to use the predictions from the various models. The stacking framework is depicted in Figure 3. Specifically, it involves building models using multiple base algorithms, called level-0 models, and a meta-learning algorithm that trains

another model to combine the predictions from the base models. The core idea in stacking is that the level base learners are trained using the training dataset and are provided with out-of-sample or unseen data; their predicted target labels on the unseen data, together with the actual labels, form the input and output pairs of a new dataset used to train the meta-learner. Meta-learning is the part of machine learning where algorithms are trained using the output of other ML algorithms and make more accurate predictions given the predictions made by the other base classifiers³⁸.

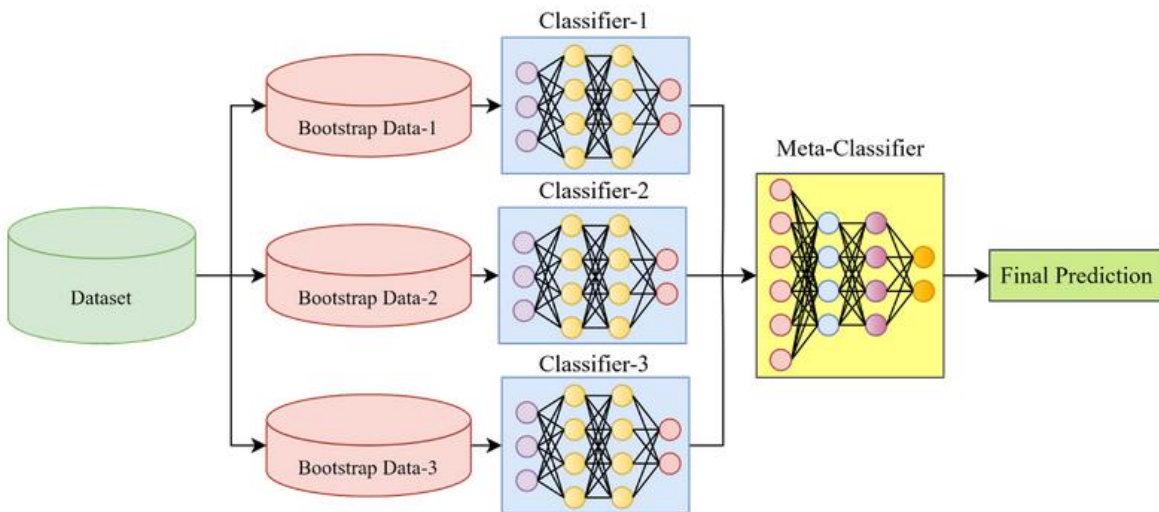


Figure 2.15. Stacking Ensemble Learning⁴¹

- **Voting Ensemble Learning:** Voting is arguably the easiest ensemble algorithm and is frequently very efficient. It can be applied to categorization i.e., classification or regression issues. The process of voting involves building two or more sub-models. Each sub model makes predictions that are combined in some manner, such as by calculating the mean or mode of the predictions, giving each sub model to vote on what the final result should be³⁹. The diagram below presents the workflow of the Voting Ensemble:

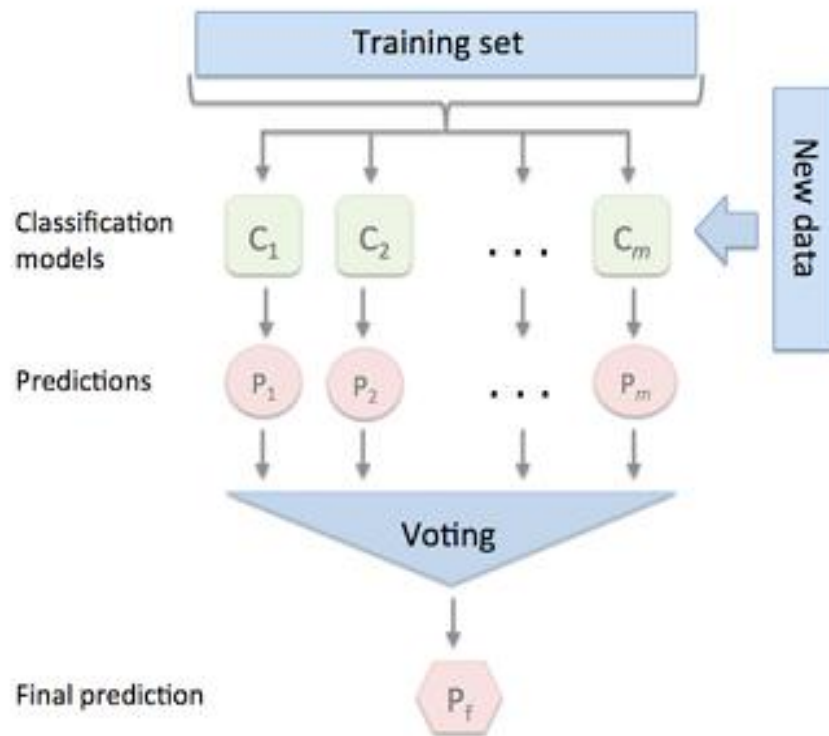


Figure 2.16: Voting Ensemble Learning⁴²

The voting classifier can be used in multiple ways, which include:

1. **Hard Voting (Majority Voting):** In the hard voting classifier technique, models make predictions separately and note the ones that receive the most votes. For example, in a model that involved three classifiers (X, Y, and Z) that predict three classes (1, 2, and 3), after the training, classifier X predicts as 1, classifier Y predicts as 2, and the Z classifier predicts as class 2. By the technique used in voting, the final output will be 2, because two out of the three models predicted it⁴⁰.

2. **Soft Voting:** soft voting is the technique in which multiple models make the predictions, and the average of all the predicted values is considered the final predicted value. For example, if three models predict class probabilities of 20%, 40%, and 60%, then the average probability will

be $(20+40+60)/3 = 60$. So, 60 is the average predicted value. Therefore, voting technique is an approach where multiple classifiers are combined to provide a better result.

2.2 Conceptual Framework

A successful IDS that makes use of fuzzy clustering, the ABC algorithm, and a multilayer perceptron (MLP) network was used by Bahram Hajimirzaei and Nima Jafari Navimipour. The training subsets are created using the fuzzy clustering method. The MLP distinguishes between regular and anomalous packets in network traffic. The NSL-KDD 99 dataset's characteristics are mostly responsible for the MLP's structure. Additionally, the ABC method trains the MLP by maximizing the linkage weights and bias values. An artificial neural network (ANN) is a computational system with artificial intelligence that was modeled after the form and operation of biological neural networks. The main applications of ANNs are prediction, classification, and pattern recognition in machine learning. The three levels of an ANN structure are the input, hidden, and output layers⁴³.

Each layer contains a certain number of nodes that are determined by the type of problem. Each node can connect to nodes in the following layer with the aid of the linkage weights. The bias weights also assist a bias node in connecting every node on that specific layer. Using one of the optimization strategies, the linkage and bias weight values between the layers of an ANN structure are adjusted. Each node's output in each layer is influenced by the weighted inputs at the node and the activation function being employed.

The three steps in the proposed method are training, validation, and testing. The suggested IDS architecture extracts and optimizes the fuzzy rules used in the training phase. High accuracy on the test data is made possible by using the training data. In the validation phase, the system's performance is precisely assessed. This method is used to detect faults on the validation dataset,

and system training is finished when errors start to increase. During testing, intrusion detection is performed by running the test data through the trained model, intrusion detection is carried out during testing.

Three subsets of the dataset are identified as training (TR), validation (VA), and testing (TE). The primary process of clustering based on similarity measures is the discovery of natural groups or clusters in multidimensional data; the primary objective of the fuzzy cluster is the classification of a given collection of data into clusters. The fuzzy cluster requires characteristics like homogeneity within the clusters, which refers to data within the same cluster, and heterogeneity between the clusters, where data belonging to various clusters should be as varied as feasible.

The proposed method was performed with a Core i7 2700 CPU and 16 GB of DDR3 RAM, and the evaluation of the suggested method was carried out using CloudSim version 4. For their modeling and simulation, cloud computing infrastructures and services need the flexible and adaptable simulation framework offered by CloudSim. As new cloud computing infrastructures and application services emerge, CloudSim offers unified modeling, simulation, and testing of such systems. As a result, it is the ideal option for emulating the suggested strategy.

The studies employ the NSL-KDD dataset, an improved version of the KDD'99 dataset, for training and testing. The collection has five million connection records because training data and nearly test data both contain two million connection records. A total of 41 features are available in the dataset, one for each relationship. The dataset additionally contains labels that categorize connection status reports as either a typical attack type or a specialized attack type. These

characteristics are divided into four divisions and are represented by various continuous, discrete, and symbolic variables:

1. The first category provides information on the key characteristics of individual TCP connections, such as the connection's duration, the kind of protocol used (TCP, UDP, etc.), and the network service used (HTTP, telnet, etc.).
2. Features recommended by domain knowledge are used to evaluate the payload of the original TCP packets, such as the number of unsuccessful login attempts.
3. For host characteristics that have the same destination host as the current connection, examine the connections that were launched in the last two seconds. Through the launched connections, statistics pertaining to protocol behavior, service, and other topics are calculated.
4. The same-service features look at connections that have the same service as the current connection and were made during the previous two seconds⁴³.

2.2.1 IDS model with deep learning and optimization

Many industries use cloud computing as a storage medium, which results in the storage of many packets. This presents a variety of problems, with security problems like integrity, availability, and confidentiality, as well as cyberattacks and intruders, being among the more significant ones. The environment for cloud computing contains a number of security challenges, including those related to virtualization, applications, networks, physical security, and so forth. Among them, the recommended works solely took network-level security problems in the cloud environment into account. Due to the fact that cloud services are made available through web browsers, this occurs.

Denial of service attacks, user-to-root assaults, remote-to-user attacks, and probing are examples of network-level intrusions. As a result, IDS is proposed in this study in combination with the Ant optimization method⁴⁴.

Originally, the attack classification was carried out using the LSTM network, a form of RNN, together with CNN and the Ant lion optimization (ALO) algorithm. The meta-heuristic ALO method is used to optimize this process since huge volumes of packets cause the deep neural network to train with a high error rate, resulting in lower error rates and higher classification rates. When fresh data is received, the IDS scans it for intrusions, attacks, and other policy violations before responding with an attack categorization. In Figure 2.17, the proposed process is illustrated.

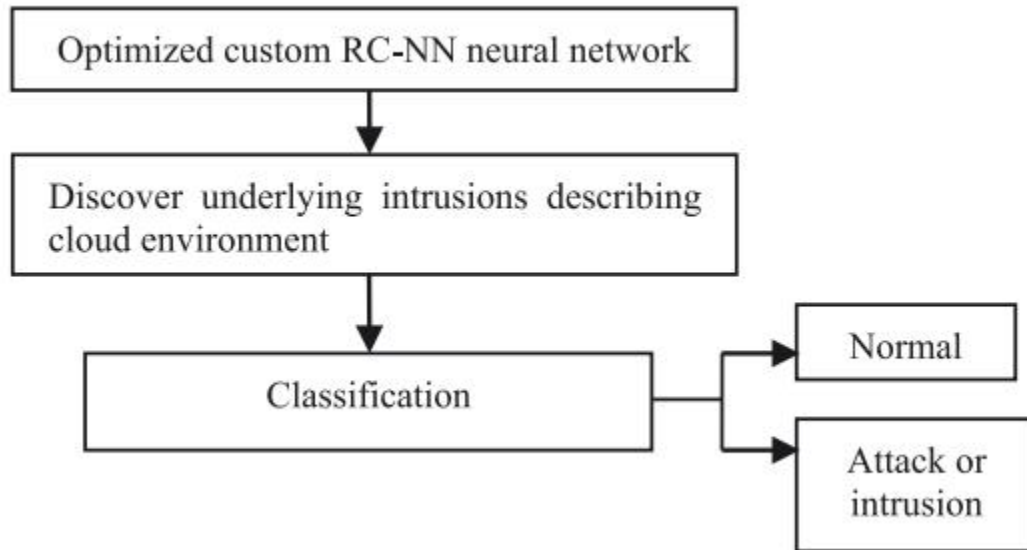


Figure 2.17. Flow diagram for proposed methodology⁴⁴.

With the aid of the Defense Advanced Research Projects Agency (DARPA) dataset and the Communications Security Establishment (CSE) and The Canadian Institute for Cybersecurity

(CIC) (CSE-CIC-IDS2018) dataset, the proposed IDS is put out and tested. These datasets include both literary and mathematical content.

The lowercase texts are transformed, and the punctuation is removed, in order to deal with this text data initially. As an alternative, to make them all the same length, the document is padded and truncated. Feature analysis is carried out to extract features prior to supplying the input to the customized RC-NN network that has been optimized. The DARPA dataset contains 43 features, including protocol type, service, scythes, ds bytes, attack, and more. These data packets contain a variety of properties with various features. Not all 43 features work with IDS. A number of features are unnecessary and irrelevant, lengthening the detection process and degrading performance. A small number of sporadic data points are eliminated as unnecessary data and sorted into a dataset of manageable size. To reduce computation complexity and improve learning accuracy, it is vital to select characteristics that protect a data set's most valuable data⁴⁴.

2.2.2 Hybrid-detection Model Using Machine Learning

The proposed hybrid detection model and the created methodologies will both be covered in detail in this part. Hybrid strategies perform better than conventional ones in intrusion detection systems. Due to this, the study's main objective is to apply hybrid methodologies to analyze the contextual relationship between data flow in the network. More specifically, the emphasis is on using the K-means clustering algorithm to automatically generate labels and using SVM to create a learning model that could be used to assess the incoming data questions. By utilizing several parameters to enhance the results of the clustering algorithm and feature extraction techniques to only choose the more useful features, this study differs from others⁴⁵.

The suggested framework for this investigation is shown in Figure 2.18. The following factors contribute to the high false positive rates in anomaly detection for intrusion detection systems:

- (i) The amount of data samples used to train the model is typically insufficient to ensure its accuracy for deployment in a production environment.
- (ii) An intrusion detection system's false positive rate even a small increase in it can have a significant negative impact on the system's performance in a production environment.
- (iii) It is challenging to establish a universal standard for what constitutes normal or abnormal traffic because the definition of an anomaly varies from organization to organization.
- (iv) Network traffic is highly variable due to the diversity of network applications. A new hybrid detection approach is suggested for network-based anomaly detection in a cloud computing network, keeping these problems in mind.

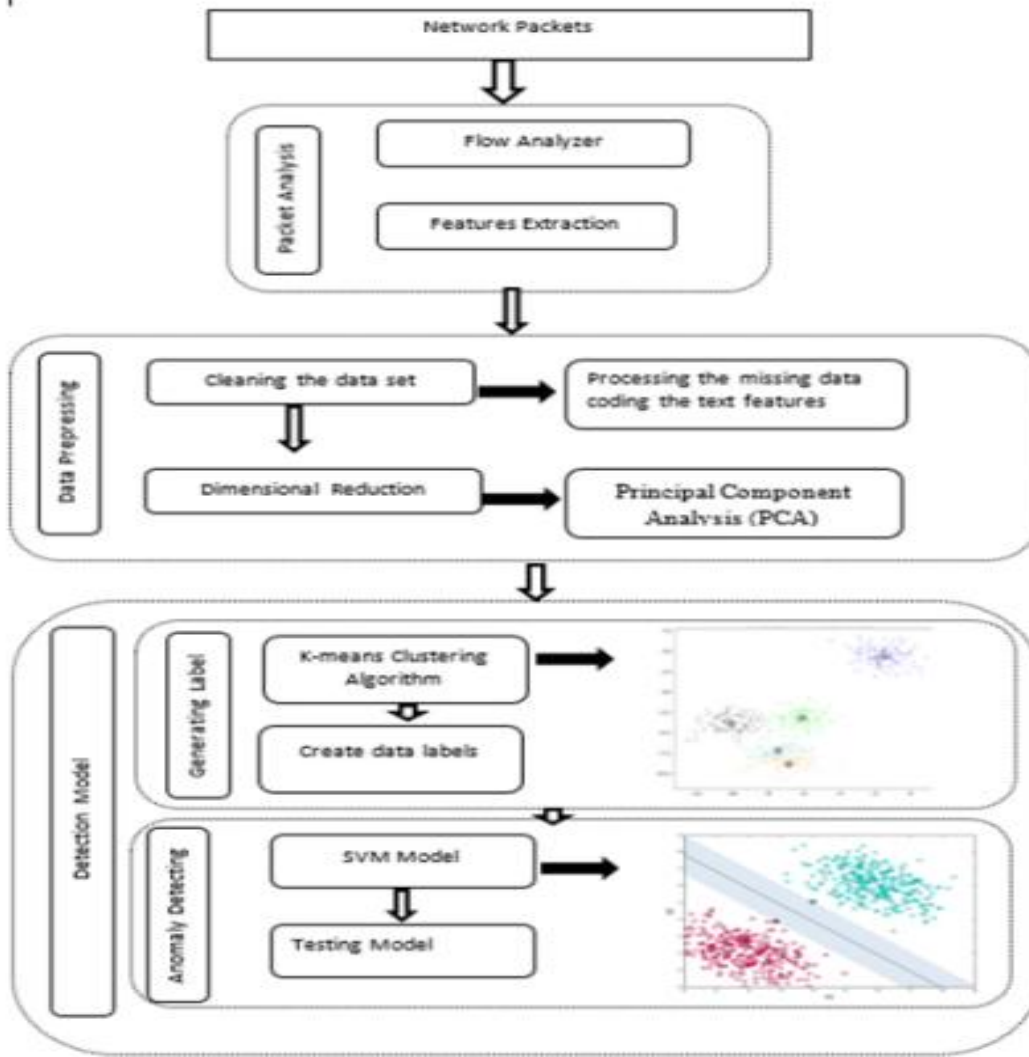


Figure. 2.18. Hybrid-detection model Framework Using Machine Learning⁴⁵.

2.2.3 Hybrid Deep Learning Model for Intrusion Detection

The proposed model was referring to as a "deep CNN-WDLSTM model," since it combines a deep convolutional neural network (CNN) with weight-dropped long short-term memory (WDLSTM) to classify intrusions in real-time data flow (see Figure 2.3). Two 1D convolutional layers, one 1D maximum pooling layer, one 1D WDLSTM layer, and one fully connected layer make up the deep CNN-WDLSTM model. The Rectified Linear Unit (ReLU) function, which

can be calculated as the activation function used to train the model in the two convolutional layers.

$$\sigma(x) = \max(0, x) \quad 2.1$$

The 1D WDLSTM layer receives the output of the maximum-pooling layer. After that, the 1D WDLSTM network learns how the extracted features are related while arbitrarily disregarding some weights to avoid overfitting. The fully connected layer receives the output of the 1D WDLSTM layer and uses a SoftMax activation function to classify incursions before detecting them⁴⁶.

$$p_i = \frac{e^{x_{ij}}}{\sum_{j=1}^n e^{x_j}} \quad 2.2$$

in which x represents the input. P produces output values that range from 0 to 1, with the sum of the outputs being one. The number of CNN filters, epochs, learning rate, WDLSTM hidden unit count, drop-connect ratio, batch size, and maximum-pooling length are the hybrid model's key variables. During the training phase, all of these parameters are learned by trial and error.

The CNN-WDLSTM model's performance is measured by the four most popular validation metrics: accuracy, precision, recall, and F1-score. This is how they are calculated:

$$Accuracy = \frac{(TP+TN)}{TP+FP+TN+FN} \quad 2.3$$

$$Precision = \frac{TP}{TP+FP} \quad 2.4$$

$$Recall = \frac{TP}{TP+FN} \quad 2.5$$

$$2.6$$

$$F1 - Score = 2 \times \frac{(Precision \times Recall)}{Precision + Recall}$$

where TP represents the number of abnormal samples that test positive (correct detection), TN represents the number of normal samples that test negative (correct detection), FP represents the number of normal samples that test positive (incorrect detection), and FN represents the number of abnormal samples that test negative (false detection) (incorrect detection).

Accuracy is defined as the proportion of samples that were correctly categorized compared to all other samples in the testing set. Precision is defined as the percentage of samples that are correctly classified for all of the TP and FP in the testing set. The ratio of TP samples to the total number of TP and FN samples is known as the "recall." The F1-score is then calculated using the weighted average of precision and recall⁴⁷.

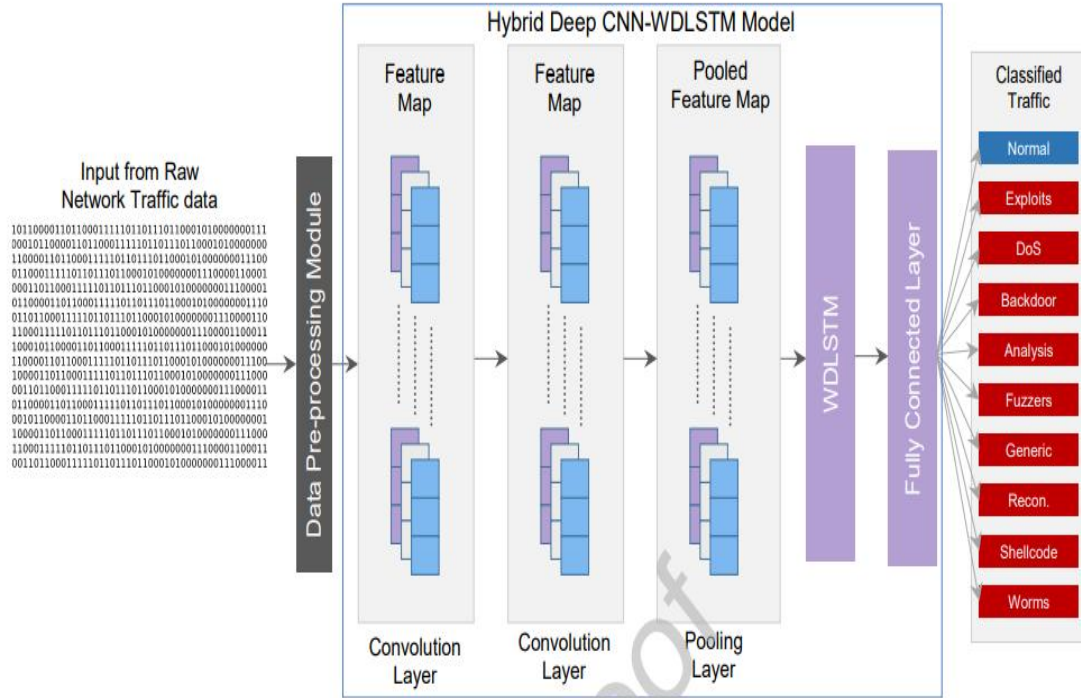


Figure. 2.19. The Architecture of Proposed Deep CNN-WDLSTM Model⁴⁷.

2.2.4. Network Intrusion Detection System Using Voting Ensemble Machine Learning

Different types of network attacks and intrusions are increasing rapidly with the increase in network traffic. It is challenging work to identify and protect the network from those threats. However, the majority of detection systems rely on malware detection techniques based on signatures, which are primarily capable of detecting known assaults. As a result, they are unable to identify novel attacks. To get around these limitations, researchers are utilizing ensemble machine learning models that can operate in several domains. Figure 2.20 shows the suggested methodology for the research in offline analysis.

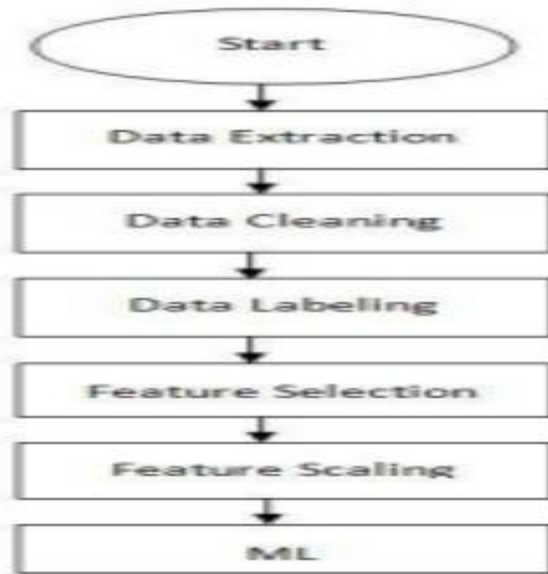


Figure 2.20: Offline Analysis of Proposed Methodology⁴⁸.

2.2.4.1 Data Extraction

The data that used for this study was taken from NSLKDD, a Canadian network traffic dataset. By addressing many shortcomings in the KDD CUP 99 dataset, this dataset has been enhanced. There are four files in the NSL-KDD dataset: KDDTest+, KDDTrain+, KDDTest-21, and KDDTest-20. The KDDTrain+ dataset has 125,973 network traffic samples; the KDDTest+ dataset contains 22,554 traffic samples; and the KDDTest-21 dataset contains 11,850 network traffic samples with 43 characteristics.

Because it is challenging to identify using a class label when the dataset is unbalanced, they can be divided into 5 categories of network attack: normal, probe, R2L, U2R, and doS.

It is more realistic because some attacks are present in the test set but not in the training set. The following includes discussions of DoS, Remote to Local (R2L), Probe, and User to Root (U2R).

When a network device is under a denial-of-service attack, it is unable to manage valid requests and access. An attacker who uses a probe gathers information about the target system that may be useful for future attacks. The attacker uses R2L to attempt to gain unauthorized access to the target system. With the use of U2R, the attacker gains access to a system that is not permitted for them, and they use this method to try to get root access⁴⁸.

Data cleaning, which was done in this study using Python programming, came first after the dataset was extracted. We obtain 43 characteristics from the Canadian NSL-KDD dataset. We eliminate two of them because they mostly comprise continuous value information rather than categorical information. We also eliminate another functionality that uses constant values. The dataset was cleaned, and we eventually got 40 relevant characteristics. One feature out of 40 features is chosen to serve as the target feature. Table 2.1 lists the 39 features' data types.

Table 2.1 Features and their data types in NSL-KDD⁴⁸

Feature	Feature type
2,3,4	Normal
7,12,14,15,21,22	Binary
1,5,6,8,9,10,11,13,16,17,18,19,20,23,24,25	Numeric
26, 27,28,29,30,31,32,33,34,35,36,37,38,39	

2.2.4.2 Data Labeling

There are three non-numeric qualities among the 40 features. They are "protocol type," "class," and "flag." For instance, the feature protocol type, which has the attributes tcp, udp, and icmp, is labeled 0, 1, and 2 correspondingly. In a similar vein, the attributes for the features "class" and

"flag" each have five different types. Following labeling, all non-numerical values are converted into numerical values in the same way, and our prediction goal is divided into five categorization groups⁴⁸.

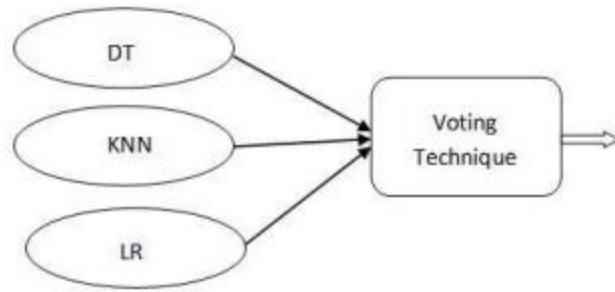
2.2.4.3 Feature Selection

The filter method is the fastest of the feature selection strategies. There are various filtering techniques available. Using the Scikit-Learn library, we take into account the p-value of features in our research.

The p-value is used to find the physical characteristics or features that can classify normal and anomalous network traffic. Important aspects that are statistically significant are chosen using this method. The parameters are deemed to be significant if the p-value is low.

2.2.4.4 Classification using Ensemble Machine Learning

In order to categorize network attacks, we have employed several arrangements of basic classes in the voting technique. In Fig. 2.21, we use a voting ensemble machine learning algorithm employing three basic classes: DT (Decision Tree), KNN (K Nearest Neighbor), and LR (Logistic Regression). Table 2.2 lists the five voting methods and the underlying classifiers that were suggested for our system.



Figure, 2.21 Voting Ensemble Techniques⁴⁸.

Table 2.2 voting arrangements⁴⁸.

Methodology	Base classifier-1	Base Classifier-2	Base Classifier-3	Voting type
Voting-1	DT	KNN	LR	Hard
Voting-2	DT	SVC(Kennel=poly	LR	Hard
Voting-3	DT	SVC(Kernnel=RBF)	LR	Hard
Voting-4	DT	SVC(Kernel=linear)	LR	Hard
Voting-5	AdaBoost	SVC(Kernel=RBF)	GradientBootsing	Hard

The network assault can be categorized using ensemble voting. The voting algorithm has two stages: the generation of votes and the prediction of votes based on the voting classifier produced in the first stage. Below is the voting prediction pseudocode:

1. Take the test features into account, then utilize each base classifier's rules to forecast the result and store the projected target.
2. You must determine the votes for each forecasted target.
3. The maximum number of votes will be taken into account for the final outcome.
4. Produce a forecast using the voting system.

Following are the procedures we took to establish the voting process:

1. Load the dataset first.
2. Make the base classifiers initial.
3. Use basic classifiers to vote using the voting technique.
4. Predict the target based on the base classifiers' majority vote.
5. Calculate the detection rate using the confusion matrix's value.

2.2.4.5 Evaluation Metric

The detection rate is one of the most crucial performance measures in the intrusion detection model. It is the proportion of anomaly records that were correctly identified as anomalous when compared to the overall number of anomaly records. the equation

$$DR = TPR = TP / (TP + FN) \quad 2.7$$

Where FP stands for false positive, FN for false negative, TN for true negative, and TP for true positive⁴⁸.

2.2.5. Genetic algorithm and back propagation neural network

The GA and BPNN classifier algorithms are utilized in this suggested model to detect intrusions and attacks in the cloud computing environment. The dataset is initially provided as input to the system and preprocessed, as indicated in the suggested model in Figure 2.22 of the diagram. The CICIDS 2017 dataset is used to find assaults and intrusions. The preprocessed data are sent to a GA for initial anomaly identification, and the GA's output is supplied to the BPNN classifier. The GA results are used to identify attack categories found in the dataset in order to identify

potential intrusions and attacks. The attack labels that are contained in the dataset are used to describe such harmful data. The different sorts of attacks are categorized using the BPNN classifier⁴⁹.

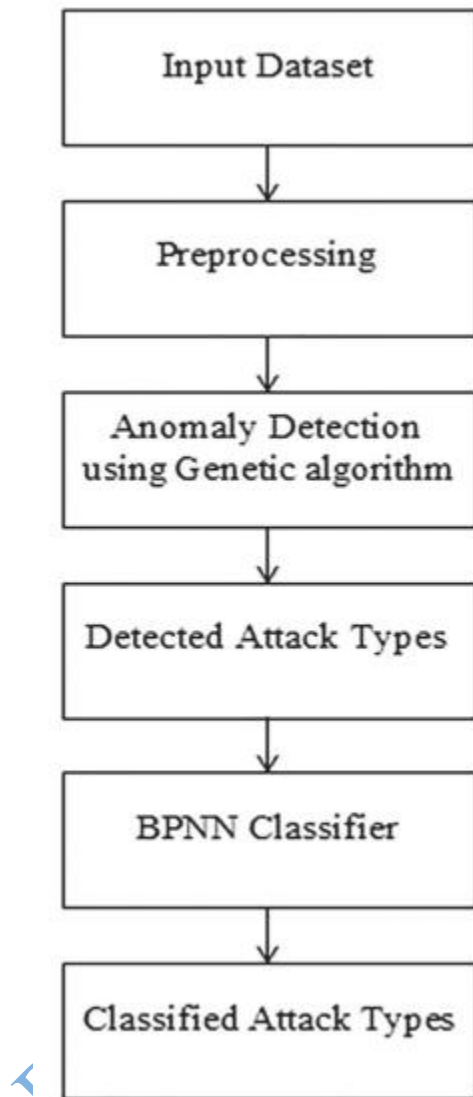


Figure 2.22. Genetic algorithm and back propagation neural network model ⁴⁹.

2.2.5.1. Genetic algorithm

The GA updates the population of individuals iteratively. Each iteration of the procedure includes an evaluation of the individuals using the fitness function. By probabilistically selecting fitter people, the present population generation is used to create a new population generation.

Some of the people were permitted to procreate unaltered. Others are referred to as genetic operators that create new offspring, such as crossover and mutation. Z is the population proportion that will be replaced by a crossover in each iteration, g is the current generation, n is the total number of people in the population, and μ is the mutation rate⁴⁹.

//Initialize generation

$g=0$;

$P_g = n$ randomly produced individuals' population;

Calculate P_g

Calculate fitness (f) for every $f \in P_g$;

Do

{ //Create generation $g+1$

//1. Copy

Choose $Z \times n$ individuals of P_g ; pair them up; generate offspring; insert offspring into P_{g+1} ;

//3. Mutate

Choose $\mu \times n$ individuals of P_g ; invert a randomly chosen bit in each;

//Calculate P_{g+1}

Calculate fitness (f) for every $f \in P_g$;

//Increment

$g= g+1$;

}

While the fitness of the fittest individual in P_g is not enough high;

Return the fittest individual from P_g ;

2.2.5.2 Neural network with back propagation

The BPNN is uncomplicated, quick, and straightforward to program. Aside from the quantity of inputs, it has no parameters that can be adjusted. Given that it doesn't require knowledge of the network beforehand, it is a flexible technique. It is a tried-and-true method that typically works

well. No precise cues about the characteristics of the function's features are required for learning. "Backward propagation of mistakes" is referred to as "back propagation" informally. It is a typical method of ANN training^{49,50}. As illustrated in Figure 2.23, the BPNN has an input, hidden, and output layer, which is a typical multilayer network governed by complete interconnection within layers but independent of integration in the same cell layer. Both forward and backward propagation are parts of the learning process. The connection layer nodes' aberrant connection weight and threshold may be the cause of the error, which may be determined by measuring the connecting node's error value and adjusting it in accordance with the connection weight⁵⁰.

By deleting weighted links that have little or no impact on the trained network, back propagation cleans up the network topology. It is a method for changing the weight of the neural network based on the error value obtained in the previous iteration. By increasing the system's generalization, proper weight tuning makes it possible to reduce mistake rates and increase dependability. The pre-connected path is where the inputs (X) come from. The weight W was used to model the input. Typically, the weights are chosen at random. From the input layer to the hidden and output layers, all neurons' outputs are computed. Error = Actual Output is the formula. The desired output is used to calculate the output error.

The weights can then be adjusted from the output layer to the hidden layer, lowering the inaccuracy. The process is carried out repeatedly until the desired result is achieved. Figure 2.23 shows the BPNN design. There are three layers to it. The weighted input total for node j in the hidden layer was established when the input was fed with a certain training pattern by

$$Net_j = \sum w_{ij}x_i + \Delta \Theta_j \quad 2.8$$

The total input to the neuron was calculated using equation 4. The weighted value of the j expression from a bias node consistently displayed output as 1. This bias node served as the "pseudo input" for each neuron in the hidden and output layers and was also used to address problems in situations where the input pattern is 0. The neural network cannot be trained if one of the input patterns receives a value of zero without a bias node⁴⁹.

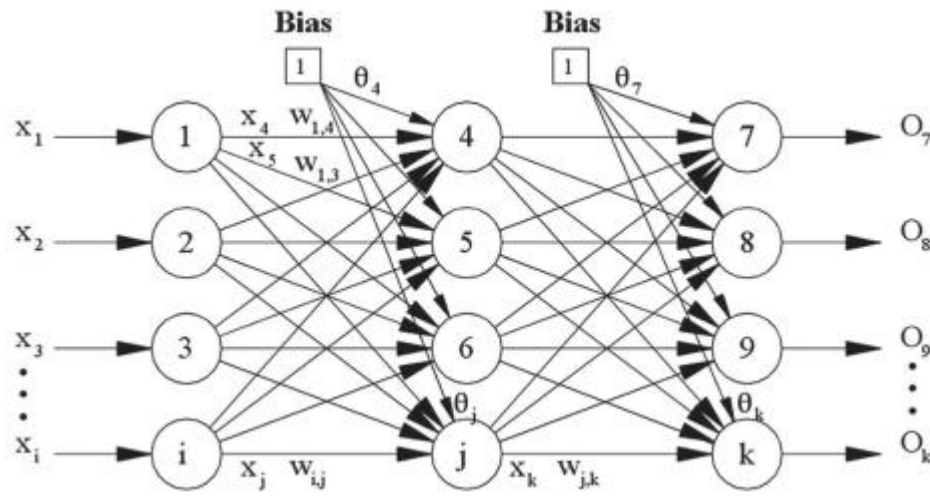


Figure 2.23. BPNN Architecture⁴⁹.

The "net" term, also known as the action potential, was sent to a suitable actuation mechanism to decide whether the neuron needed to fire. The output value of the activation function determines the actions of neurons and serves as an input to neurons in related layers below. As a result, the BP method requires a differentiable activation function. Equation 2.9 expresses the sigmoid function as a common activation function.

$$O_j = x_k = \frac{1}{1 + e^{-Net_j}} \quad 2.9$$

Similar to that, criteria 1 and 2 were used to determine the output value for node k in the output layer.

2.2.5.3 Output layer

If the intended output for node k was t_k and the actual activation value of the output node, k, is O_k , the difference between the real and the anticipated output is shown in Equation 6:

$$O_j = \frac{x_k = 1}{1 + e^{-Net_j}} \quad 2.10$$

Equation 4 is used to calculate the error signal in the output layer for node k.

$$\delta_k = \Delta_k O_k (1 - O_k) \quad 2.11$$

Or

$$\delta_k = (t_k - O_k) O_k (1 - O_k) \quad 2.12$$

The derivative of the sigmoid function is $O_k (1 - O_k)$. The j actuation in relation to the k error determines the weight correlation difference between nodes k and j using the delta rule. The following equations (5 and 6) can be used to modify the weights, $w_{j, k}$, between the j and k:

$$\Delta w_{i k} = l_r \delta_k x_k \quad 2.13$$

$$w_{i k} + \Delta w_{j, k}, \quad 2.14$$

where l_r is the learning rate and Δw_{jk} is the change in weight between k and j . The relevant changes in weight are represented by the smaller constant l_r , which is often smaller. It should be noted that in condition 5, x_k was both the input value for k and a corresponding output value from j . The following modification was made to condition 5 in order to improve the process for updating weights:

$$\Delta w_{i,k}^n = l_r \delta_k x_k + \Delta w_{i,k}^{(n-1)} \mu \quad 2.15$$

Equation 7 describes how weight is updated during the n th epoch by multiplying the (μ) momentum term by $w_{j,k}$ $(n-1)$ th epoch.

For j , the error signal in this layer is computed in Equation 2.13:

$$\delta_k = (t_k - O_k) O_k \sum (w_{j,k} \delta_j) \quad 2.16$$

The following equations can be used to alter $w_{i,j}$ inside the j and I

$$\Delta w_{i,k}^n = l_r \delta_j x_j + \Delta w_{i,k}^{(n-1)} \mu \quad 2.17$$

$$W_{i,j} = w_{i,j} + \Delta w_{i,j} \quad 2.18$$

2.2.5.4 Global Error

In order to reduce the output node error across the patterns added to the neural network, the BP was finally acquired via assumption. For each pattern, the error function E was calculated using the accompanying condition 14.

$$E = \frac{1}{2} \sum (\sum (t_k - O_k)^2) \quad 2.19$$

The error function technically needs to be zero when the neural network has been properly trained.

```

Allot every network input and output
Initialize each weight with lower numbers randomly, generally among -1 and 1 repeat
  For each pattern in the training set
    Present the network pattern
  // Propagate the input forward through the network:
  For every layer in the network
    For each node in the layer
      1. Evaluate the input's weighted sum for the node
      2. Add the threshold to the sum
      3. Evaluate the activation for the node
    end
  end

  //Propagate the errors backward within the network
  for each output layer node
    Evaluate the error signal
  end
  for every hidden layer
    for each layer node
      1. Evaluate the error signal of the node
      2. Update weight for every node in the network
    end
  end
//Compute Global Error
  Compute the Error Function
end

While ((Maximum number of iterations < than determined) AND
      ((Error Function is >than determined))

```

The above is the BPNN algorithm. The GA is used to identify attacks, and the BPNN classifier is used to categorize them. The tests were conducted with a 32 GB memory and an 8-core Amazon Cloud Platform. The performance analysis for the proposed model was assessed using the dataset CICIDS 2017⁴⁹.

2.3. Related Works

In recent times, artificial intelligence approaches have gained widespread use to develop solutions in the security environment, owing to their effective automated feature extraction methodologies. A lot of researchers have employed machine learning and deep learning techniques in the past couple of years to detect network intrusions. Numerous techniques based on learning have been put forth to enhance the performance of intrusion detection system efficacy. The most recent algorithms and techniques that have been researched to find areas where efficient network intrusion detection systems performance could be improved in terms of accuracy, precision, detection rate, and low false-negatives are reviewed in depth in this part.

Network security has been effectively achieved by the use of intrusion detection, which can detect unknown threats from network traffic. These days, the majority of network anomaly detection techniques are based on conventional machine learning models like KNN, SVM, etc. While these techniques can yield some exceptional results, they have a low accuracy rate and mainly rely on human traffic feature design, which is out of date in the big data era. A traffic anomaly detection model, called BAT, is presented to address the issues of low accuracy and feature engineering in intrusion detection systems. Bidirectional long-short-term memory (BLSTM) and the attention mechanism are combined in the BAT model. The network flow vector, made up of packet vectors produced by the BLSTM model, is screened using an attention mechanism in order to extract the essential characteristics needed to classify network traffic.

Furthermore, utilize several convolutional layers to extract regional characteristics from traffic information. BAT model was referred to as BAT-MC since several convolutional layers are used to process data samples. Network traffic is classified using the Softmax classifier. The suggested end-to-end model can automatically identify the salient features of the hierarchy and does not require any expertise in feature engineering. It can greatly enhance anomaly detection and provide a clear description of network traffic behavior. The model was evaluated using a publicly available benchmark dataset, and the experimental findings show that the model outperforms alternative comparison techniques.⁵²

Another paper proposes a hybrid and layered intrusion detection system (IDS) that provides high-performance intrusion detection across various attack types by combining various machine learning and feature selection techniques. The NSL-KDD dataset is first subjected to data preprocessing in the proposed system. Subsequently, the size of the dataset is minimized through the application of several feature selection algorithms. There are now two fresh methods for the feature selection process. By selecting the proper machine learning algorithms based on the type of attack, the layered architecture is developed. The NSL-KDD dataset is used to assess the suggested system's performance in terms of accuracy, DR, TP Rate, FP Rate, F-Measure, MCC, and time. The performance of the suggested method is evaluated and compared with research found in the literature in order to show its effectiveness. The suggested system has been demonstrated to have excellent accuracy and minimal false positive rates across all attack types⁵³.

When it comes to sharing resources and information, cloud computing is essential. Protecting cloud services from various hackers is difficult. In addition to being used to monitor network traffic, intrusion detection systems (IDS) are essential for identifying intruder attacks. The goal

of the study is to manage the attacks by integrating the artificial bee colony (ABC) known as Group-ABC (G-ABC) with machine learning (ML) techniques. The G-ABC has been used to determine further simulation results after the implementation of the IDS detector. Accuracy, recall, precision, and the F1 measure have all been used in the review process. Numerous attacks have been identified, including viruses, backdoors, denial-of-service (DoS) assaults, user to root (U2R), probing, and root to local (R2L). Two datasets the NSL-KDD dataset and the UNSW-NB15 dataset are used for the simulation analysis, and a comparison is made with earlier work to demonstrate the viability of the suggested IDS. The work's goal is to identify the invader attacker system by applying deep learning techniques⁵⁴.

The authors in this research suggests a new intrusion detection system (IDS) that combines fuzzy clustering, artificial bee colonies (ABC), and multilayer perceptron (MLP) networks. The MLP classifies network traffic packets as normal or abnormal, and the ABC algorithm optimizes the linkage weight and bias values to train the MLP. The suggested approach is validated using the NSL-KDD dataset and the CloudSim simulator. The kappa statistic, mean absolute error (MAE), and root mean square error (RMSE) are regarded as evaluation criteria. The findings collected have demonstrated the superiority of the suggested procedure when compared to cutting-edge techniques⁵⁵.

Information security relies heavily on intrusion detection, and the technology that is used to accurately categorize various types of attacks is also crucial. Within the context of cloud networks, intrusion detection systems (IDS) are recognized as critical security concerns. The author(s) in this research proposes a novel customized RC-NN (Recurrent Convolutional Neural Network) for intrusion detection using the Ant Lion optimization algorithm. An original

approach is used to provide IDS. Convolutional neural networks (CNNs) and long short-term memory (LSTMs) can be hybridized using this technique. Consequently, every assault linked to the cloud's network layer is effectively categorized. The experiment's findings, accurately depict how the IDS classification model is presented, increasing the detection rate or error rate. As a result, the customized RC-NN-IDS model that was optimized produced a 94% classification accuracy increase and a 0.0012 error rate reduction. Performance measurements also include precision, true positive rate, and true negative rate. The CSE-CIC-IDS2018 dataset and the DARPA IDS evaluation data sets are used to assess the suggested method and compare it to a few other methods that are currently in use⁵⁶.

Deep learning models have demonstrated impressive performance when used for large-scale big data analysis, but they haven't been tested for intrusion detection in a big data setting. In order to effectively detect network intrusions, this study suggests a hybrid deep learning model that combines a weight-dropped long short-term memory (WDLSTM) network with a convolutional neural network (CNN). In order to avoid overfitting on recurrent connections, the author employ the deep CNN to extract significant features from the massive IDS data and the WDLSTM to preserve long-term dependencies among the derived features. The suggested hybrid method's adequate performance was shown by comparing it with more conventional methods using a dataset that was made available to the public⁵⁷.

A hybrid method that involves the combination of the K-means clustering algorithm and the SVM classification algorithm for a network-based anomaly detection system at the cloud hypervisor level was introduced. This hybrid approach is intended to increase the anomaly detection system's accuracy. The proposed approach is assessed using data from the UNSW-

NB15 project, and the results are contrasted with those from earlier research. The K-means clustering model has marginally higher accuracy than others. The SVM model's accuracy is still below average for supervised methods. In the final analysis, the false alarm rate is still very high for such a system as a computer network. for the reason that if an attack goes undetected, that may affect the whole system. Thus, the result of the model is still not practical in a production system⁴⁵.

Security concerns have severely harmed the cloud computing environment, which has a negative impact on the healthy and long-term growth of cloud computing. One of the tools for preventing harmful assaults on the cloud computing environment is intrusion detection. However, because of the huge scale, high dimensionality, and high redundancy of network traffic in the cloud computing environment, the development of cloud intrusion detection systems faces significant difficulties. The potential of deep learning technology for intrusion detection is significant. Therefore, the goal of this study is to effectively achieve high detection performance by using deep learning to automatically extract key feature representations. For unsupervised feature extraction, the stacked contractive autoencoder (SCAE) approach is proposed. Better and more reliable low-dimensional features can be automatically learned from raw network traffic using the SCAE approach. Based on the SCAE and SVM classification methods, a unique cloud intrusion detection system is created. Deep and shallow learning approaches are used in the SCAE+SVM strategy, which fully utilizes their benefits to drastically minimize the analytical overhead. Experiments on two well-known intrusion detection evaluation datasets, KDD Cup 99 and NSL-KDD, demonstrate that the proposed SCAE+SVM technique yields greater detection performance in comparison to three other state-of-the-art methods⁵⁸.

Because cloud computing is based on a distributed architecture and offers storage and computing resources as a service across an open environment (the Internet), it raises a number of security and privacy-related issues. Therefore, protecting network accessible cloud resources and services from different types of risks and attacks is very important. In order to solve this problem, a powerful and effective Network Intrusion System (NIDS) that can identify insider and outsider intrusions in the cloud environment with high detection precision must be developed. In this work, the authors propose to employ an Improved Genetic Algorithm (IGA) to optimize Back Propagation Neural Networks (BPNNs), a very popular soft computing technique that is commonly used for intrusion detection. Optimization techniques like parallel computing and fitness value hashing, which shorten execution and convergence times and conserve computing power, are used to improve genetic algorithms (GAs). For simulation, DARPA's KDD Cup datasets from 1999 and the CloudSim simulator 4.0 are utilized. Using CloudSim 4.0 and KDD CUP's 99 datasets, experimental results show that the IDS, called ANIDS BPNN-IGA, performs better than the original ANIDS BPNN. Furthermore, GA has benefited from performance enhancement techniques that have shortened execution and convergence times and conserved computing resources. The authors decide to install the suggested system, "ANIDS BPNN-IGA," on the front and back ends of the cloud in order to quickly identify and thwart assaults that could jeopardize the cloud datacenter's security. The next task is to integrate the new IDS with Suricata IDS, Snort, to create a hybrid IDS that is more productive and efficient⁵⁹.

To reduce false positive rate, a light-structure NBAD algorithm that utilizes DBN and LSTM network was suggested by the authors in which deep belief network's goal is to simultaneously decrease the dimension of raw data and efficiently extract features from high-dimensional network behavior data. With the feature-extracted data, there are fewer hidden layers, smaller

dimensions, and fewer redundant features. The pattern of network behavior must be learned by hidden nodes, which can significantly lower computation costs. An LSTM system is used in order to identify patterns in network behavior and get the temporal sequence data, and the outcomes of the experiment demonstrate that. With great accuracy in NBAD, the LSTM network may efficiently learn the general pattern of network behavior. Furthermore, compared to the existing methods, this methodology requires less time and may readily update the model in the event of new attacks, in addition to obtaining a model quickly. On the other hand, the suggested model's performance is not optimal for categories where the datasets contain a small number of records. Furthermore, the suggested approach requires two models to train independently to further decrease the false positive rate to make NBAD more reliable⁶⁰,

To protect the data source from intrusions and attacks, cloud network security is crucial. The best course of action for protecting against those intrusions and attacks is to implement an intrusion detection system (IDS). Currently, a lot of IDS models are built on various methodologies and algorithms, such as deep learning and machine learning. An intrusion detection system for the cloud computing environment was recommended for attack detection and classification, in this model, the genetic algorithm (GA) and back propagation neural network (BPNN) are used. The Canadian Institute for Cybersecurity's CIC-IDS 2017 dataset is used to assess performance. The attack was initially discovered after preprocessing the dataset's data using a genetic algorithm. The BPNN classifier is used to categorize the detected attacks in order to determine the different sorts of attacks. The findings of the performance study were obtained, and they were compared to those of the current machine learning-based classifiers, such as FC-ANN, NB-RF, KDBN, and FCM-SVM approaches. In every performance indicator, including accuracy, precision, recall, and detection rate, the suggested GA-BPNN model outperforms the competition. Overall,

according to the performance analysis, the detection rate for brute force attacks is the best, at 97.89%, and the classification accuracy for web attacks is the greatest, at 97.90%⁴⁹.

Security concerns have severely harmed the cloud computing environment, which has a negative impact on the healthy and long-term growth of cloud computing. One of the tools for preventing harmful assaults on the cloud computing environment is intrusion detection. However, because of the huge scale, high dimensionality, and high redundancy of network traffic in the cloud computing environment, the development of cloud intrusion detection systems faces significant difficulties. The potential of deep learning technology for intrusion detection is significant. Therefore, the goal of this study is to effectively achieve high detection performance by using deep learning to automatically extract key feature representations. For unsupervised feature extraction, the stacked contractive autoencoder (SCAE) approach is proposed. Better and more reliable low-dimensional features can be automatically learned from raw network traffic using the SCAE approach. Based on the SCAE and SVM classification methods, a unique cloud intrusion detection system is created. Deep and shallow learning approaches are used in the SCAE+SVM strategy, which fully utilizes their benefits to drastically minimize the analytical overhead. Experiments on two well-known intrusion detection evaluation datasets, KDD Cup 99 and NSL-KDD, demonstrate that the proposed SCAE+SVM technique yields greater detection performance in comparison to three other state-of-the-art methods but the SVM classifier cannot effectively detect new attacks. The classifier also needs to be optimized⁶¹.

In addition to intrusion detection for cloud computing using neural networks and artificial bee colony optimization, another study proposed a synthesis of fuzzy clustering methods, artificial bee colonies, and artificial neural networks (ANNs). An ANN can work in an IDS by itself, but

an IDS is more effective when it combines fuzzy clustering, fuzzy ABC, and an ANN. The authors propose a novel artificial bee colony (ABC), fuzzy clustering, and multilayer perceptron (MLP) network-based intrusion detection system (IDS). The MLP distinguishes between normal and abnormal network traffic packets, and the ABC algorithm performs the MLP training by maximizing the linkage weights and biases. The suggested approach is tested using the NSL-KDD dataset and the CloudSim simulator. The kappa statistic, mean absolute error, and root mean square error are all taken into consideration as evaluation criteria. The findings collected have shown that the proposed method is superior to state-of-the-art methods⁶².

Another strategy was presented due to the fact that network intrusion attacks have increased significantly, which is quite worrying from a privacy and security perspective. The complexity of cyber-security threats is increasing due to technological improvements, making it impossible to address this problem with the current detection systems. It is therefore claimed that, it is essential to design a smart and reliable network intrusion detection system to address this issue. In this study, the authors build an intelligent detection system that can recognize various network intrusions using deep learning approaches, specifically convolutional neural networks (CNN) and recurrent neural networks (RNN). Additionally, in order to choose the optimum model for the network intrusion detection system, we compare the outcomes of our suggested solution and evaluate the performance of the proposed solution using several evaluation matrices⁵¹.

Information security is greatly aided by intrusion detection, and the ability to correctly classify various threats is a key technique. One significant security concern in the cloud network environment is the intrusion detection system (IDS). Therefore, IDS is presented based on a novel, customized, and optimized RC-NN (recurrent convolutional neural network) that is

suggested for intrusion detection, coupled with the Ant Lion optimization technique. This technique combines LSTM and CNN (convolutional neural network) (long short-term memory). As a result, all assaults detected by the cloud's network layer are efficiently categorized. The experiment's findings, which are displayed below, accurately depict how the IDS classification model is presented, increasing the detection rate or error rate. This resulted in an increase in classification accuracy of 94% and a reduction in error rate of 0.0012 for the customized, optimized RC-NN-IDS model. Furthermore, accuracy, true positive rate, and true negative rate are taken into account as performance measurements. The suggested method is assessed using the CSE-CIC-IDS-2018 dataset and DARPA IDS evaluation data sets, and it is contrasted with various other methods already in use⁶³.

According to further research into network intrusion detection methods combining CNN and BiLSTM in a cloud computing environment, the KDD CUP 99 data set is preprocessed using a data extraction and a data mining algorithm. The data set is cleaned, extracted, and mapped into an image data set. Next, CNN and BiLSTM are used to extract parallel local features of attribute information and long-distance-dependent information, respectively. This fully accounts for the influence of front and back attribute information, and an attention mechanism is added to increase classification accuracy. In order to avoid the design feature selection step and utilize a deep learning model directly to learn the representational features of high-dimensional data, the C5.0 decision tree and CNN BiLSTM deep learning model are coupled. According to experimental findings, this method's network intrusion detection impact is superior to that of approaches based on AE-AlexNet and SGM-CNN. Its average accuracy can be improved to 95.50%, and its false-positive rate can be decreased to 4.24% and 6.66%⁶⁴.

Furthermore, concerns regarding security and privacy are a barrier to the adoption of cloud computing. Network intrusion detection systems (NIDS) are widely used, yet false positives are still a problem. In addition, very little research has looked at the intrusion detection problem as a time-series problem that calls for time-series modeling. With time series data, this study demonstrated a collaborative feature selection approach for effective intrusion detection in cloud networks. To distinguish between correlation and causation, the method combines anomaly detection and causality tests. The suggested strategy reduced the number of input predictors and saved training time and resources while increasing forecast accuracy. The trials were run on the CSE-CIC-IDS2018 dataset, which is a sizable and recent dataset. The experiments have shown encouraging results, including an increase in forecast accuracy, a decrease in the number of input predictors, a reduction in complexity, and a shorter prediction time. In this work, the Granger causality metric was utilized; however, it has been expanded to take into account dissimilarity metrics. By adjusting hyperparameters and including more training data, the proposed model can produce respectable results⁶⁵.

An intrusion detection-data security protection scheme based on the particle swarm-BP network algorithm in a cloud computing environment is proposed in order to address the issues of the low detection rate and high false detection rate of intrusion detection algorithms in the traditional cloud computing environment. First, the general framework of the intrusion detection model is built by constructing matching functions based on the four modules of data collection, data preprocessing, feature selection, and intrusion detection. The model's data processing speed is then increased after the decision tree algorithm is added, and on this foundation, the feature selection is carried out using the "gain rate" optimization method, which eliminates the redundant information in the feature vector. Finally, the BP neural network is improved based on

the momentum factor and adaptive learning rate, and the high detection rate and low false detection rate are realized by incorporating the Particle Swarm Optimization (PSO) algorithm into the optimization of the initial weights and thresholds of the BP neural network. The suggested intrusion detection approach and the other three methods are evaluated and compared through simulation tests under identical circumstances. The results demonstrate that under four different types of sample data, the detection rate and false detection rate of the approach suggested in this study are the best; the highest detection rate reaches 95.72%, while the lowest false detection rate falls to 2.03%. The suggested technique outperforms the previous two comparison algorithms in terms of performance ⁶⁶.

This study designs an integrated deep intrusion detection model based on SDAE-ELM to overcome the long training time and low classification accuracy of existing deep neural network models, and to achieve timely response to intrusion behavior. For host intrusion detection, an integrated deep intrusion detection model based on DBN-Softmax is constructed, which effectively improves the detection accuracy of host intrusion data. At the same time, in order to improve the training efficiency and detection performance of the SDAE-ELM and DBN-Softmax models, a small batch gradient descent method is used for network training and optimization. Experiments on the KDD Cup99, NSL-KDD, UNSW-NB15, CIDDS-001, and ADFA-LD datasets show that SDAE-ELM and DBN-Softmax integrated deep inspection models have better performance Compared with the traditional machine learning models. However, the data mining ability of SDAE-ELM model is effective, and the detection effect of small datasets is poor. In addition, the DBN-Softmax model has the disadvantage of long training time for large datasets, and cannot realize real-time detection of intrusions. In the future, the study will consider using hybrid feature extraction technique to reduce the dimensionality of the dataset and reduce the

training time of the model under the premise of ensuring the accuracy of the intrusion detection target. In addition, due to the complex structure of the deep intrusion detection model and the large number of parameters, it is necessary to consider improving the model neurons and calculation methods, simplify the network structure, and improve the model efficiency⁶⁷

Massive information has been transmitted through complicated network connections around the world. Thus, providing a protected information system has fully consideration of many private and governmental institutes to prevent the attackers. The attackers block the users to access a particular network service by sending a large amount of fake traffics. Therefore, this work demonstrates two-classification models for accurate intrusion detection system (IDS). The first model develops the artificial neural network (ANN) of multilayer perceptron (MLP) with one hidden layer (MLP1) based on distributed denial of service (DDoS). The MLP1 has 38 input nodes, 11 hidden nodes, and 5 output nodes. The training of the MLP1 model is implemented with NSL-KDD dataset that has 38 features and five types of requests. The MLP1 achieves detection accuracy of 95.6%. The second model MLP2 has two hidden layers. The improved MLP2 model with the same setup achieves an accuracy of 2.2% higher than the MLP1 model. The study shows that the MLP2 model provides high classification accuracy of different request types, but variation in the dataset have some effect on the performance of the system⁶⁸

Summary of Gaps in Literature

A review of literature has been done on different articles, journals, conference papers, and even online articles based on the security of a network and the host devices connected to it, both on premise and in a cloud environment. The use of intrusion detection systems has been emphasized as one of the major security measures that can protect information and the entire network system

from any cybercrime. Many of the researchers in the literature use both convolutional machine learning and deep learning algorithms to create a model to solve the issues of intrusion and attacks on a network. Some used supervised and unsupervised learning algorithms, creating both single and multi-classifiers with different datasets and having different results from the experiment carried out. Some of the output of the results led to better performance of the model in terms of accuracy and precision, while others led to enormous intrusion data classification issues, which led to low recognition accuracy and a high false alarm rate. Also having shortcomings, which include a low rate of attack detection, incorrect classification of network anomalies as attacks, and a high false-positive rate among detected attacks. The use of an unbalanced dataset also has an impact on the performance of a model. In fact, large variations in the dataset and a large number of different features caused a longer training time for a model. Also, many methods used by different authors are prone to error, struggle with identifying relevant data and patterns, and struggle with feature selection and engineering, which leads to low recognition accuracy and a high false alarm rate. **To fill the gap in the literature, there is a need to** create a unique classifier that performs better than individual classifiers. Combining many classifiers is one way to decrease false positives and increase accuracy. Therefore, a hybrid intrusion detection approach using parallel learning with deep learning algorithms for optimized network intrusion detection in a cloud environment is developed by utilizing one of the most frequently used network intrusion datasets to evaluate the approach. As compared to the recent studies, the proposed method is simple, computationally efficient, works on multi-class classification, and does not require high-end computational resources. The popular deep learning algorithms, such as deep belief networks (DBN) and multilayer perceptron (MLP), were used.

The performance of the proposed model was compared against the conventional intrusion detection model.

Endnote

1. A. Saeed. *“Cloud Intrusion Detection Systems: Fuzzy Logic and Classifications,”* **University of Nottingham repository**, 2017.
2. C. Kanini, *“Cloud Computing Communication Environment Security and Performance: Challenges and Probable Solutions,”* **World Journal of Advanced Engineering Technology and Sciences** 8, no. 1 2023. 110-127.
3. S. Morgan, *“Humans on The Internet Will Triple From 2015 To 2022 And Hit 6 billion,”* **Cybercrime Magazine**, 2019, <https://cybersecurityventures.com/how-many-internet-users-will-the-world-have-in-2022-and-in-2030/>
4. M. Talwar, *“Security Issues in Internet of Things,”* **International Journal on Emerging Technologies** 6, no. 2, 2015. 364-367.
5. G. Murigi, & M. Mutuku, *“Security Issues and Challenges in Cloud Computing among Public Institutions in Africa,”* **Journal of Business and Management Sciences**, 10, no. 3, 2022, 131-137
6. H. N. Elekwachi, A. S. Dick, & P. O. Omunakwe, *“Cloud Computing Capability and Organizational Communication of Public Tertiary Institutions in Rivers State,”* **Nigerian Journal of Management Sciences**, 24, 2023.
7. L. Savu, *“Cloud Computing: Deployment Models, Delivery Models, Risks and Research Challenges,”* **“International Conference on Computer and Management (CAMAN) 2011**
8. S. S. Dadpe, P. P. Patil, & V. M. Sultanpure, *“Trends in Cloud Computing,”* **International Journal of Scientific Research in Science, Engineering and Technology** 8, no. 2, 2021. 402-405.

9. R. Ara, M. A. Rahim, S. Roy, & U. K. Prodhan, “*Cloud Computing: Architecture, Services, Deployment Models, Storage, Benefits and Challenges*,” **International Journal of Trend in Scientific Research and Development (IJTSRD)** 4, 2020. 2456 – 6470
10. R. k. Sharma, & N. K. Joshi, “*Security and Privacy Problems in Cloud Computing*,” **International Journal of Computer & Organization Trends (IJCOT)** 9, no. 4 2019. 31-32
11. S. O. Kuyoro, F. Ibikunle, & O. Awodele,” *Cloud Computing Security Issues and Challenges*,” **International Journal of Computer Networks (IJCN)** 3, 2011: 247-254
12. K. Hashizume, D. G. Rosado, E. Fernandez-Medina, & E. B. Fernandez, “*An Analysis of Security Issues for Cloud Computing*,” **Journal of Internet Services and Applications** 4, no. 1 2013. 5.
13. J. Ogala, & S. E. Mughele, “*A Review of Security Issues in Cloud Computing*.” **Creative Research Publishers** 2022. (book)
14. B. S. Al-Attab, & H. S. Fadewar, “*Security Issues and Challenges in Cloud Computing*,” **International Journal of Emerging Science and Engineering (IJESE)** 2, no.7, 2014.
15. N. Alharbe, A. Aljohani, M. A. Rakrouki, & M. Khayyat, “*An Access Control Model Based on System Security Risk for Dynamic Sensitive Data Storage in the Cloud*,” **Applied Science**, 13, no. 5, 2023, 3187.
16. I. Khalil, A. Khreishah, & M. Azeem, “*Cloud Computing Security: A Survey*,” **Computers** 3, no. 1, 2014. 1-35.
17. N. R. Tadapaneni, “*Cloud Computing Security Challenges*,” **International Journal of Innovations in Engineering Research and Technology [IJIERT]** 7, 2020
18. R. Jathanna, & D. Jagli, “*Cloud Computing and Security Issues*,” **International Journal of Engineering Research and Applications** 7, no. 6, 2017 31-38.
19. A. S. Jaradat, M. M. Barhoush, & R. B. Easa, “*Network intrusion detection system: machine learning approach*.” **Indonesian Journal of Electrical Engineering and Computer Science** 23, no. 2, 2022. 1151-1158.
20. S. S. Tripathy, & B. Behera, “*Performance Evaluation of Machine Learning Algorithms for Intrusion Detection System*.” **Journal of Biomechanical Science and Engineering** 2023.
21. S. H. Abbas, W. A. Khuder-Naser, & A. A. Kadhim, “*Intrusion Detection System (IDS) and Intrusion Prevention System (IPS)*,” **Global Journal of Engineering and Technology Advances**, 14, no. 02, 2023, 155-158.

22. W. Lin, S. Ke, & C. Tsai, "CANN: An Intrusion Detection System Based on Combining Cluster Centers and Nearest Neighbors," **Knowledge-Based Systems** 78 2015. 13-21.
23. S. Thapa, & A. D. Mailewa, "The Role of Intrusion Detection/Prevention Systems in Modern Computer Networks: A Review" **Conference of Midwest Instruction and Computing Symposium (MICS)**. 2020. 1-14.
24. U. S. Musa, M. Chhabra, A. Ali, & M. Kaur, "Intrusion Detection System Using Machine Learning Techniques: A Review." **In International Conference on Smart Electronics and Communication (ICOSEC)**, (2020). 149-150.
25. D. Perez, M. A. Astor, D. P. Abreu, & E. Scalise, "Intrusion Detection in Computer Networks Using Hybrid Machine Learning Techniques," **XLIII Latin American Computer Conference (CLEI)** 2017.
26. M. Aljanabi, M. A. Ismail, & A. H. Ali, "Intrusion Detection Systems, Issues, Challenges, and Needs," **International Journal of Computational Intelligence Systems** 14, no. 1 2021. 560.
27. V. Kumar, & M. L. Garg, "Deep Learning as a Frontier of Machine Learning: A Review," **International Journal of Computer Applications** 182, no. 1 2018. 22-30.
28. L. Deng, & D. Li, "Deep Learning: Methods and Applications," **Foundations and Trends in Signal Processing** 7, no. 3, 2014. 197-387.
29. R. Nisbet, G. Miner, & K. Yale, "Deep Learning," **Handbook of Statistical Analysis and Data Mining Applications**, 2018. 741-751
30. M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, & E. Muharemagic, "Deep Learning Applications and Challenges in Big Data Analytics," **Journal of Big Data** 2, no. 1 2015.
31. N. V. Otten, "Deep Belief Network Explanation, Application & How to Get Started in TensorFlow." **Machine Learning, Natural Language Processing**, 2023.
<https://spotintelligence.com/2023/02/10/deep-belief-network/>
32. G. Wang, J. Qiao, C. Liu, & Z. Shen." *How Deep Is Deep Enough for Deep Belief Network for Approximating Model Predictive Control Law.*" **IEEE Transactions on Automation Science and Engineering** 2021.1-12
33. A. P. Kale, R. M. Wahul, A. D. Patange, R. Soman, & W. Ostachowicz. "Development of Deep Belief Network for Tool Faults Recognition." **Sensors** 23, 2023.

34. W. Deng, H. Liu, J. I. Xu, H. Zhao. & Y. Song, “An Improved Quantum-Inspired Differential Evolution Algorithm for Deep Belief Network,” **IEEE Transactions on Instrumentation and Measurement** 69, no. 10, 2020. 7319-7327
35. A. Mohanty, “Multi-layer Perceptron (MLP) Models on Real World Banking Data.” **Becoming Human: Artificial Intelligence Magazine**, 2019, <https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f>.
36. C. Bento, “Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis,” **Towards Data Science**, 2021, <https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141>
37. A. Inamdar, “Ensemble Learning Techniques in Machine Learning,” **Fire blaze AI School**, 2021, <https://www.fireblazeaischool.in/blogs/ensemble-learning-techniques-in-machine-learning/>
38. I. D. Mienye, & Y. Sun, “A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects,” **IEEE Access** 10, 2022. 99129-99149.
39. L. G. Kabari, “Comparison of Bagging and Voting Ensemble Machine Learning Algorithm as a Classifier,” **International Journal of Computer Science and Software Engineering**, 9, 2019. 19-23
40. M. Raihan-Al-Masud & H. A. Mustafa, “Network Intrusion Detection System Using Voting Ensemble Machine Learning,” **IEEE International Conference on Telecommunications and Photonics (ICTP)**. 2019.
41. R. Kundu, “The Complete Guide to Ensemble Learning,” **V7**, 2022, <https://www.v7labs.com/blog/ensemble-learning>.
42. R. P. Dharmaraj, & J. B. Patil, “Malicious URLs Detection Using Decision Tree Classifiers and Majority Voting Technique,” **Cybernetics and Information Technologies** 18, no. 1 (March 2018): 11-29.
43. B. Hajimirzaei, & N. J. Navimipour, “Intrusion Detection for Cloud Computing Using Neural Networks and Artificial Bee Colony Optimization Algorithm,” **ICT Express** 5, no. 1 2019 56-59.
44. T. Thilagam, & R. Aruna, “Intrusion Detection for Network Based Cloud Computing by Custom RC-NN and Optimization,” **ICT Express** 7, no. 4 2021. 512-520,
45. I. Aljamal, A. Tekeoğlu, K. Bekiroğlu, & S. Sengupta, “Hybrid Intrusion Detection System Using Machine Learning Techniques in Cloud Computing Environments,” **IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)**, 2019. 84-89.

46. D. A. Clevert, T. Unterthiner, & S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUS)." **International Conference on Learning Representations**, 2016. 1-14
47. M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, & G. Fortino, "A Hybrid Deep Learning Model for Efficient Intrusion Detection in Big Data Environment," **Information Sciences** 513 2020. 386-396.
48. M. Raihan-Al-Masud. & H. A. Mustafa, "Network Intrusion Detection System Using Voting Ensemble Machine Learning," **IEEE International Conference on Telecommunications and Photonics (ICTP)**, 2019.
49. S. Manimurugan, P. Manimegalai, P. P. Valsalan, J. Krishanadas, & C. Narmatha. "Intrusion Detection in Cloud Environment Using Hybrid Genetic Algorithm and Back Propagation Neural Network," **International Journal of Communication Systems** 35, no. 16, 2020.
50. M. S. Taj, "Enhancing Anomaly Based Intrusion Detection Techniques for Virtualization in Cloud Computing Using Machine Learning," **International Journal of Computer Science and Information Security (IJCSIS)** 18, no. 5, 2020. 60-78
51. S. Al-Emadi, A. Al-Mohannadi, & F. Al-Senaid, "Using Deep Learning Techniques for Network Intrusion Detection," **IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)**, 2020. 171-176
52. T. Su, H. Sun, J. Zhu, S. Wang, & Y. Li. "BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset," **IEEE Access** 8, 2020. 29575-29585,
53. Ü. Çavuşoğlu, "A new hybrid approach for intrusion detection using machine learning methods," **The International Journal of Research on Intelligent Systems for Real Life Complex Problems** 4, no. 2 2019
- 54.. N. Gulia, K. Solanki, S. Dalal, A. Dhankhar, O. Dahiya, & N. U. Salmaan, "Intrusion Detection System Using the G-ABC with Deep Neural Network in Cloud Environment", **Scientific Programming** 2023
55. B. Hajimirzaei, & N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm," **Korean Institute of Communications and Information Sciences (KICS)** 2018
56. T. Thilagam, R. Aruna, "Intrusion detection for network-based cloud computing by custom RC-NN and optimization," **The Korean Institute of Communications and Information Sciences (KICS)**, 2021

57. M. M. Hassan, A. Gumaiei, A. Alsanad, M. Alrubaian, & G. Fortino, “A Hybrid Deep Learning Model for Efficient Intrusion Detection in Big Data Environment,” **Information Sciences** 2019.
59. Z. Chiba, N. Abghour, K. Moussaid, A. El omri & M. Rida, “New Anomaly Network Intrusion Detection System in Cloud Environment Based on Optimized Back Propagation Neural Network Using Improved Genetic Algorithm,” **International Journal of Communication Networks and Information Security (IJCNIS)** 11, no. 1, 2019.
60. 55. A. Chen, Y. Fu, X. Zheng, & G. Lu, “An efficient network behavior anomaly detection using a hybrid DBN-LSTM network,” **Computers & Security** 114, 2022.
61. W. Wang, X. Du, D. Shan, R. Qin, & N. Wang, “Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine,” **IEEE Transactions on Cloud Computing** 10, no. 3, 2020. 1634-1646.
62. B. Hajimirzaei, & N. J. Navimipour, “Intrusion Detection for Cloud Computing Using Neural Networks and Artificial Bee Colony Optimization Algorithm,” **ICT Express** 5, no. 1, 2019. 56-59.
63. T. Thilagam, & R. Aruna, “Intrusion Detection for Network Based Cloud Computing by Custom RC-NN and Optimization,” **ICT Express** 7 no. 4, 2021. 512-520
64. J. Gao, “Network Intrusion Detection Method Combining CNN and BiLSTM in Cloud Computing Environment,” **Computational Intelligence and Neuroscience** 2022. 2022, 1-11
65. A. R. Al-Ghuwairi¹, Y. Sharrah, D. Al-Fraihat, M. AlElaimat¹, A. Alsarhan, & A. Algarni, “Intrusion detection in cloud computing based on time series anomalies utilizing machine learning,” **Journal of Cloud Computing**, 12, 2023.
66. Z. Wang, & X. Chen, “Intrusion Detection-Data Security Protection Scheme Based on Particle Swarm-BP Network Algorithm in Cloud Computing Environment,” **Journal of Electrical and Computer Engineering**, 2023, 2023, 1-10.
67. Z. Wang, Y. Liu, D. He, S. Chan, “Intrusion detection methods based on integrated deep learning model,” **Computers & Security** 103, 2021.
68. A. J. Mohammed, M. H. Arif, & A. A. Ali, “A multilayer perceptron artificial neural network approach for improving the accuracy of intrusion detection systems,” **IAES International Journal of Artificial Intelligence (IJ-AI)** 9, no. 4, 2020, 609-615.

Chapter Three

Methodology

3.1 Research Approach

This chapter focuses on the approaches and techniques used in this research work. Research involves a systematic process of developing, investigating, and understanding knowledge in a particular field and even addressing a specific problem¹. Therefore, the approach to research could either be qualitative or quantitative.

Qualitative research offers understanding and insights into the context of the problem. It is an exploratory, unstructured research strategy that examines extremely complicated phenomena that quantitative research cannot explain. Nevertheless, it inspires concepts or hypotheses for the next round of quantitative studies. On the basis of observation and interpretation, qualitative research aims to achieve a comprehensive understanding of human behavior, experience, attitudes, intentions, and motives. In this type of research, the participant's opinions are given more weight

by the researcher. Qualitative research methods include case studies, grounded theory, ethnography, historical research, and phenomenology².

Quantitative methods are used in the natural sciences to obtain hard data and numbers. It seeks to establish causality between two variables by employing computational, statistical, and mathematical techniques. Due to its ability to be properly assessed, the research is also referred to as "empirical research." The information gathered by the researcher can be categorized, ranked, or quantified in terms of measurement units. With the use of quantitative research, graphs and tables of raw data may be created, making it simpler for the researcher to analyze the findings. In contrast to the qualitative research approach, quantitative research uses procedures including questionnaires, systematic observations, and experiments². Studies about artificial neural networks are highly based on numbers and figures; therefore, a quantitative research approach is used in this study.

Therefore, the intrusion detection performance in cloud network environment applies a neural network to a dataset that contains a series of scenarios and realistically distributes normal and attack traffic using a hybrid of Deep Belief Network (DBN) and a Multi-layer Perceptron (MLP) is a quantitative study, and the workflow consists of a six-step procedure. The hybrid model begins by taking input from the network. The second stage involves packet classification and observation. During this phase, packet data is divided into packet headers and payloads. While the header contains source information, the payload contains exchanged data. The third stage does the pre-processing and numerical encoding. At this stage, symbolic features are mapped to ordered numbers. The fourth stage is where training and classification are done using a deep belief network and a multi-layer perceptron. In the fifth stage, the voting classifier will be used to

combine the deep belief network and multilayer perceptron models' output to form a single optimal model with higher precision, increased consistency, and reduced bias. The final stage is the performance, which is the evaluation stage to measure the quality of the intrusion detection model to know whether it has a high detection rate and a low false positive rate. Therefore, with these approaches, we have created a model that use deep learning techniques to identify anomalous network traffic. To evaluate the performance, various metrics were used which includes accuracy, recall, precision and false positive rate.

3.2 Requirements Specification

3.2.1 Software Requirements

This study utilized a laptop equipped with the necessary software and programming packages to facilitate efficient implementation and documentation. The laptop operated on Windows 10 with a 64-bit architecture. Microsoft Word will be used for typing, and Microsoft Excel will also be used to create graphical representations of the study results. Additionally, various functions and libraries from the Python programming language will be utilized to aid in the implementation process. The study used several software and programming packages, such as: is a web-based IDE

- Anaconda is open-source software that provides a toolkit that is tailored for the purpose of research and distribution of R and Python programming languages and simplifies package management and deployment. It is commonly used for scientific computing applications such as data science, predictive analytics, machine learning, and deep learning.

- Jupyter Notebook: an open-source, free IDE (application) used to create and share documents that contain live code, equations, visualizations, and narrative text. It is used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and more.
- Matplotlib: a visualization library in the Python programming language used for two-dimensional plots of arrays. It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of visualization is that it allows high-dimensionality data to be easily understandable. It consists of several plots like line, bar, scatter (Scatter plot), histogram, and more.
- NumPy: the fundamental package for scientific computing in the Python programming language. It contains a powerful N-dimensional array object, a sophisticated broadcasting function, and is useful in linear algebra, Fourier transforms, and random number capabilities.
- Pandas: the most popular Python programming language package used for data science. It offers powerful, flexible data structures that make data manipulation and analysis easy.
- Scikit-Learn: a free machine learning library for the Python programming language designed to interoperate with Python numerical NumPy and scientific libraries SciPy. It can be used for classification, regression, and clustering algorithms, including support vector machine, linear and logistic regression, random forests, gradient boosting, decision tree, K-means, and density-based spatial clustering of applications with noise.

3.2.2 Hardware requirement

For the study, the hardware requirements included the following specifications which were carefully selected to ensure that the necessary computations and analysis could be performed efficiently and effectively:

- HP Folio 9480 Computer with a reliable performance track record
- Intel Core i7 8565U Processor for high-speed processing
- 14-inch Full HD screen for clear and detailed display of results
- UHD Graphics 620 Graphics card for enhanced graphics performance
- 8GB RAM to handle complex computations and data analysis
- 256GB SSD for faster read and write speeds and better storage capacity.

These specifications were carefully evaluated and found to be optimal for the study, ensuring that the necessary computations and analysis could be performed smoothly and efficiently.

3.3 System Design

Researchers have already proposed numerous network intrusion detection models in order to reduce false positives and improve detection rates and accuracy. To further increase high-quality performance in terms of accuracy, precision, detection rate, and low false-negative intrusion, a hybrid model was developed. The figure 3.1 below shows the proposed model developed and its workflow for cloud network intrusion detection based on the use of UNSW-NB15 detection datasets. Multilayer perceptron (MLP) and deep belief network (DBN) algorithm techniques were combined together to form a single model through the use of the voting classifier. The system-developed model consists of four stages, including datasets, data preprocessing, data partitioning, and model development and model evaluation

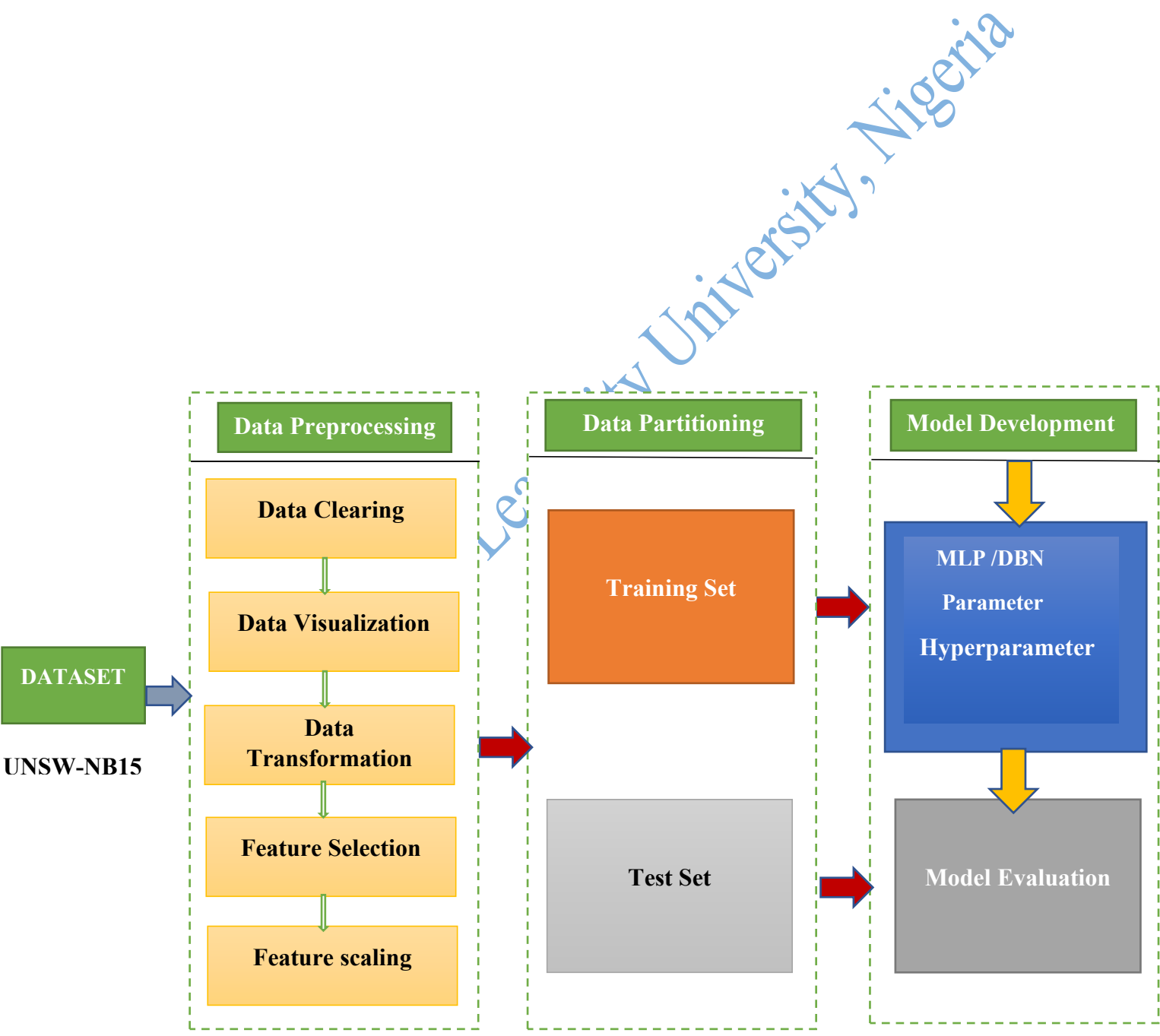


Figure 3.1. Proposed model. (Researcher: Awodele, S. O 2023)

3.3.1 Dataset

Building an intrusion detection system by training a model with the most recent, realistic dataset plays a significant role in an effective solution when taking emerging cyberattacks into account. As a result, in the context of this thesis, a dataset is the first item needed to develop a deep learning model because data is the basis for all deep learning models. Datasets basically involve collection of different kinds of data that are kept in digital form. In the fields of machine learning, deep learning, and artificial intelligence, a dataset is a group of data that is used to develop and evaluate algorithms and models. As they offer the necessary input and output data for the algorithms to learn from, these datasets are essential to the development and success of deep learning and AI systems. These records may be structured or unstructured. Prior to preprocessing, the data is particularly delicate and contains missing values and inconsistencies. The results of deep learning algorithms are influenced by the quality of the data. Raw data is preprocessed to improve data quality and efficiency. The most crucial stage in creating a deep learning model is data preprocessing³.

During the review of literature, it was discovered the UNSW-NB15 dataset by the Australian Center for Cyber Security (ACCS) in 2015 by the research group on cyber security. The UNSW-NB15 dataset promotes research on the development of intrusion detection systems using data-driven methods. In contrast to other benchmark datasets, like DARPA98 (LABORATORY, 1998), KDDCUP (99), and NSL-KDD, among others, this dataset has a more sophisticated structure. In order to assess the current network intrusion detection systems in a more accurate manner, the UNSW-NB15 is now more thorough⁴. UNSW-NB15 contains approximately 2.5

million records. The dataset was generated using two diverse simulation periods of 15 hours and 16 hours, respectively. This dataset consists of both simulated attack activities and real modern normal activities, as well as network traffic. TCP dump is employed to capture 100 GB of the raw traffic. The table 3.1 shows the ten categories of traffic featured in the dataset: normal, fuzzy, analysis, backdoor, DOS, exploits, generic, reconnaissance, and worms¹². The detailed distribution of these categories in terms of number of records per attack as well as training and testing set distribution are illustrated in Table 3.2

Table 3.1 Description of Attack Categories of the UNSW-NB15 Dataset¹².

Traffic Type	Description
Normal	This is the type of traffic that contains no threat
Fuzzing	An attack where the attacker tries to find security flaws in the operating system, program, or network and suspends these resources for a while or even crashes them.
Analysis	Certain intrusions, such as those that send spam emails, scan ports, and otherwise infiltrate web applications,
Backdoor	A technique by which attackers can bypass the usual authentication and get unauthorized remote access to a system.
DOS	An intrusion in which the attacker tries to disrupt the computing resources, by making them extremely busy in order

	to prevent authorized access to the resources.
Exploits	Code that takes advantage of a software vulnerability or security flaw. Often incorporated into malware, this allows for fairly easy and fast propagation.
Generic	This attack acts against a cryptographic system and tries to break the key of the security system.
Reconnaissance	It can be defined as a probe; an attack that gathers information about the target computer network in order to bypass its security controls.
Shellcode	Set of instructions or statements that are injected into and executed by a flawed program. Directly manipulates registers and the functions of a program
Worms	Malware that replicates itself and spreads to other computers by using the network to spread the attack, depending on the security failures on the target computer that it wants to access.

Table 3.2 Training and testing set record distribution¹².

CATEGORY	TRAINING SET	TESTING SET
Normal	56,000	37,000
Analysis	2,000	677
Backdoor	1,746	583
DOS	12,264	4,089
Exploits	33,393	11,132

Fuzzing	18,184	6,062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1,133	378
Worms	1130	44
Total Records	175,341	82,332

The characteristics of the UNSW-NB15 dataset are classified into the following six groups¹³.

1. Flow features: These features have identifier attributes between hosts, such as client-to-server or server-to-client.
2. Basic features: These features include the attributes that represent protocol connections.
3. Content features: These features contain the attributes of TCP/IP and some attributes of HTTP services.
4. Time features: This group contains the attributes of time, for example, arrival time between packets, start/end packet time, and round-trip time of the TCP protocol.
5. Additional generated features: This group can be further divided into two groups: (i) General purpose features, where each feature has its own purpose, in order to protect the service of

protocols. (ii) Connection features are built from the flow of 100 record connections based on the sequential order of the last time feature.

6. Labeled Features: This category represents the label of each record

This dataset served as the source of the training and testing data used in this study. Therefore, before importing the datasets into the system model, they have to go through some purification processes. This is to enable us to get the best from the datasets. That is, removing the undesirable properties of the data. This process led us to the next stage, which is the data preprocessing.

3.3.2 Data preprocessing

In reality, datasets of today are quite prone to noisy, missing, inconsistent data, and some rows may be duplicated, resulting in over-fitting issues. Some columns have dirty values such as spaces, nulls, or various types of data because of their often-enormous size and because they most likely came from a variety of different sources. Poor data will produce poor model outcomes. Therefore, since the aim of this research is to investigate the performance of deep belief network (DBN) and Multilayer perceptron (MLP) algorithm for detecting intrusion in cloud network environment and, therefore, to develop operational data analysis and create a dataset that is intelligible, the dataset is first preprocessed so that it can be converted into a format that the algorithm can use. Typically, this phase entails normalization and coding. Sometimes the dataset needs to be cleaned, which is done during this step. Duplicate entries and entries with

missing data and outliers are removed⁵. Data preprocessing involves a series of steps to ready the dataset for deep learning models. Here's a breakdown of the process in this study.

Data Loading and Initial Exploration:

- The dataset, sourced from 'UNSW.csv', is loaded.
- The dataset's shape is checked, and the initial rows are examined.

Handling Missing Values:

- The dataset is verified for missing values; none are found, requiring no further action.

Feature Selection and Clamping Extreme Values:

- Unnecessary columns ('id' and 'attack_cat') are removed to focus on pertinent features.
- Numeric features with extreme values are clamped, ensuring values beyond the 95th percentile is set to the 95th percentile value, reducing the influence of outliers.

Log Transformation for Numeric Features:

- Numeric features with over 50 unique values undergo log transformation. This process is valuable for wide-ranging features, enhancing their suitability for modelling.

Handling Categorical Features:

- Categorical features are transformed using one-hot encoding, converting them into binary vectors conducive to machine learning algorithms.
- Certain categorical features' unique values are limited, reducing dimensionality, and enhancing their manageability for modeling purposes.

3.3.2 Data Partitioning

This is the third stage of our system-developed model, and it involves the techniques used to randomly divide the preprocessed data into two parts: one is used for training the model (training sets), and the second part is used for evaluation (test sets). This is an essential step in the development of any model, whether it is a machine learning, deep learning, and statistical model. The training set will be utilized to modify the models' parameters, and the test set will be used to evaluate how accurate the selected model operates⁹. The primary goal of data partitioning is to

ensure that the model's performance is evaluated on independent data, which is essential to determine how well the model will generalize to new data.

Data partitioning, or splitting the dataset into training and testing sets, was achieved using the `'train_test_split'` function from the `'sklearn.model_selection'` module. This function divides the data into subsets, enabling the model to be trained on one portion and tested on another. A common practice is an 80-20 split, where 80% of the data is utilized for training and 20% for testing. The `'random_state'` parameter ensures reproducibility, as it sets the random seed. This ensures consistent results across multiple runs of the program. Partitioning the data allows for the evaluation of the model's performance on unseen data which is essential for assessing its effectiveness and generalizability.

3.3.4. Model development

The model development workflow consists of a two-step:

1. Parallel Learning stage
2. Model integration and evaluation stage.

This is shown in the figure 3.2.

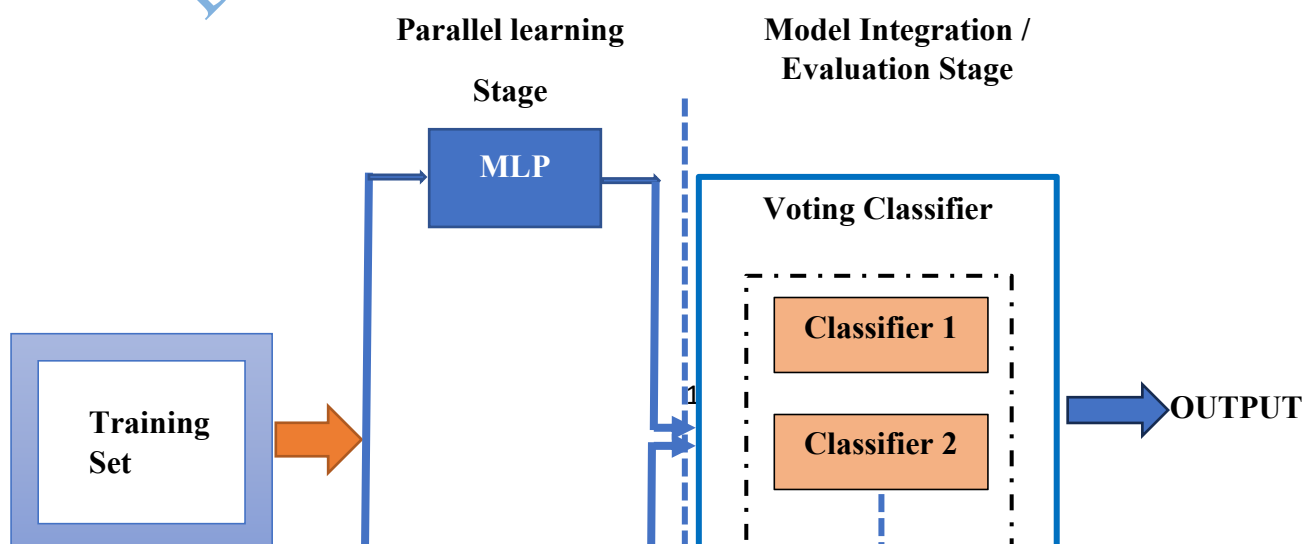


Figure 3.2 Model development architecture. (Researcher: Awodele S. O. 2023)

The parallel learning stage involves the training of pre-processed data set using the multi-layer perceptron and the deep belief network. MLP is the most commonly used deep learning classifier; it is a feed-forward artificial neural network with one or more layers between the input and output layers. Consider a network of different layers. The first layer will be the input layer, and the last will be the output layer, while the middle layers are the hidden layers of the network. However, MLP mapped the input data into output data by adjusting the weight among its internal nodes. Using the back propagation learning technique. An example of a deep learning technique that solves the issues with conventional neural networks is deep belief networks (DBNs). The network, which is made up of layers of stochastic latent variables, is used to accomplish this. These feature detectors and hidden units, also known as the binary latent variables, are stochastic in nature and can potentially take on any value within a given range. Dataset are trained using the MLP and DBN Models and they are classified accordingly.

In deep learning model development, parameters and hyperparameters play a crucial role in optimizing the performance of a model. Parameters are the values that are learned by the model during training, whereas hyperparameters are the values that are set before training begins and

control the behavior of the learning algorithm. In the context of building a multilayer perceptron (MLP) classifier, the parameter settings include `max_iter`, which sets the maximum number of iterations for the MLP solver to converge. The hyperparameters, which are tuned during grid search, include `hidden_layer_sizes`, `activation`, `solver`, `alpha`, and `learning_rate`. These hyperparameters control the number of nodes in each hidden layer, the activation function used in the hidden layers, the optimizer used to minimize the loss function, the regularization parameter used to prevent overfitting, and the learning rate used by the optimizer.

Additionally, building a deep belief network (DBN) involves setting hyperparameters such as `n_components`, `learning_rate`, `n_iter`, and `verbose`. These hyperparameters control the number of components in the restricted Boltzmann machines (RBMs) used in the DBN, the learning rate used during training, the number of iterations used during training, and whether or not to print progress messages during training. By tuning these hyperparameters, it is possible to optimize the performance of the models and achieve better accuracy in classification tasks. Grid search is a common technique used to find the best combination of hyperparameters for a given model. Overall, understanding the role of parameters and hyperparameters is essential for building and optimizing deep learning models.

The parameter settings used in the code are related to building a multilayer perceptron (MLP) classifier. The MLP classifier is a type of artificial neural network that can be used for classification tasks.

The first parameter used is `max_iter`, which sets the maximum number of iterations for the MLP solver to converge. The default value is 200, but in this case, it has been set to 100.

The parameter space dictionary contains the hyperparameters that will be tuned during grid search. Grid search is a technique used to find the best combination of hyperparameters for a

given model. The hyperparameters in the parameter space dictionary are as follows:

hidden_layer_sizes: A tuple that specifies the number of nodes in each hidden layer of the MLP.

In this case, two values have been provided - (10, 30, 10) and (20,). The first tuple specifies a MLP with 3 hidden layers, while the second specifies a MLP with 1 hidden layer activation: The activation function used in the hidden layers. Two options are provided - tanh and relu. tanh stands for hyperbolic tangent function, while relu stands for Rectified Linear Unit.

solver: The optimizer used to minimize the loss function. Two options are provided - SGD (stochastic gradient descent) and adam (adaptive moment estimation).

alpha: The regularization parameter used to prevent overfitting. Two values are provided - 0.0001 and 0.05.

learning_rate: The learning rate used by the optimizer. Two options are provided - constant and adaptive. Constant means the learning rate remains fixed throughout training, while adaptive means the learning rate decreases as training progresses.

The next set of parameters is used for building a deep belief network (DBN). A DBN is a type of deep learning model that consists of multiple layers of restricted Boltzmann machines (RBMs).

The parameters used in this case are:

n_components: The number of components (or nodes) in the RBMs. In this case, it is set to 256.

learning_rate: The learning rate used during training. In this case, it is set to 0.1.

n_iter: The number of iterations used during training. In this case, it is set to 30.

verbose: A boolean value that determines whether or not to print progress messages during training.

In deep learning model development, parameters and hyperparameters play a crucial role in optimizing the performance of a model. Parameters are the values that are learned by the model

during training, whereas hyperparameters are the values that are set before training begins and control the behavior of the learning algorithm. In the context of building a multilayer perceptron (MLP) classifier, the parameter settings include `max_iter`, which sets the maximum number of iterations for the MLP solver to converge. The hyperparameters, which are tuned during grid search, include `hidden_layer_sizes`, `activation`, `solver`, `alpha`, and `learning_rate`. These hyperparameters control the number of nodes in each hidden layer, the activation function used in the hidden layers, the optimizer used to minimize the loss function, the regularization parameter used to prevent overfitting, and the learning rate used by the optimizer.

Additionally, building a deep belief network (DBN) involves setting hyperparameters such as `n_components`, `learning_rate`, `n_iter`, and `verbose`. These hyperparameters control the number of components in the restricted Boltzmann machines (RBMs) used in the DBN, the learning rate used during training, the number of iterations used during training, and whether or not to print progress messages during training. By tuning these hyperparameters, it is possible to optimize the performance of the models and achieve better accuracy in classification tasks. Grid search is a common technique used to find the best combination of hyperparameters for a given model. Overall, understanding the role of parameters and hyperparameters is essential for building and optimizing machine learning models.

Model Integration and Evaluation Stage: The stage gives an informed decision about a possible attack based on the data set made available in the training phase and also based on the model integration. The integration and evaluation stage is mainly characterized by the voting classifier which combines a number of basis estimators to create model with superior prediction capabilities. The primary theory guiding this is the construction of strong learner from base

learners for increased accuracy. The voting classifier gives an output based on the classifications given by the MLP model and the DBN model to make an informed decision.

3.3.4.1 Deep Belief Network (DBN)

The DBN model needs to be trained in two stages: unsupervised pre-training and supervised fine-tuning. Each RBM is trained to reconstruct its input during the first phase by changing its weights and providing input to the following RBM's input layer. Every RBM layer is pre-trained after repeating this approach (greedy learning). Back-propagation and stochastic gradient descent are used to maximize all of the weights in supervised fine-tuning¹⁴. DBN is a probability generation model that consists of one layer of visible units and many levels of hidden units. There are connections between the layers, but not between the units in the same layer. A DBN can be thought of as being made up of unsupervised, straightforward networks like limited Boltzmann machines (RBMs). The probability distribution over the raw features can be learned by a stochastic artificial neural network called an RBM. When training a DBN, the taught hidden units from the prior RBM are used as the pre-training initialization. A DBN employs a greedy layer-wise training strategy, taking the output of the previous trained layer as its input¹⁵. Figure 3.7 illustrates the DBN and the learning process using stacked RBMs

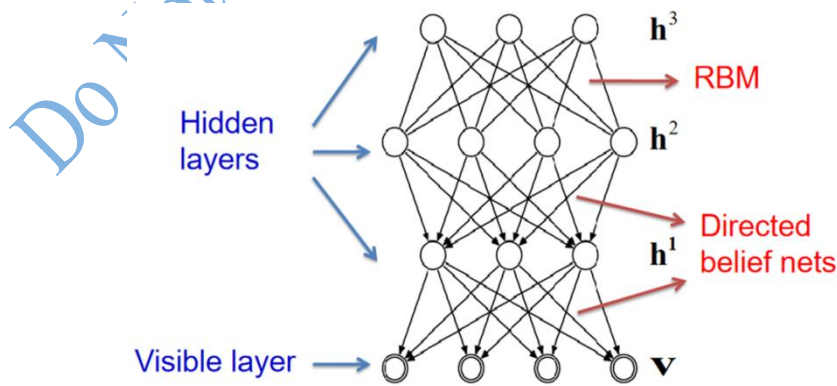


Figure 3.3. DBN structure¹⁶

$$P(v, h^1, h^2, \dots, h^l) = P(v|h^1) P(h^1|h^2) \dots P(h^{l-2}|h^{l-1}) P(h^{l-1}, h^l) \quad 3.1$$

The training processes involved in the deep belief network (DBN) are below

Step One:

- Construct an RBM with an input layer v , a hidden layer h and weights w
- Training the RBM

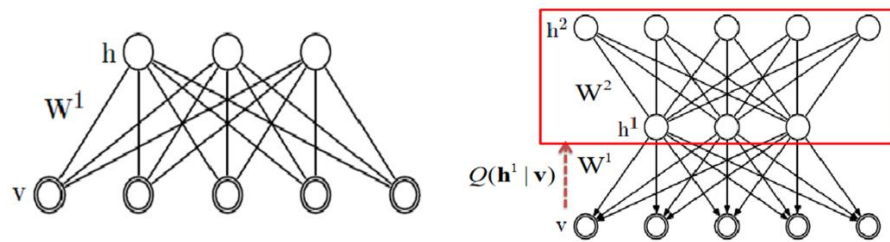


Figure 3.4. learning procedure of DBN¹⁷

Step Two

- Create a new RBM, by stacking a hidden layer on top of the existing one.
- Take W^1 , as a sample h^1 from $Q(h^1 | v)$ as the input. Then train W^2 as RBM.

Step Three

- Keep adding layers on top of the network; train it as in the previous step using samples taken from $Q(h^2 | h^1)$
- And so on.

Algorithm for Training DBN

Input: Dataset

Output: Trained network

Step 1: Train the first layer as RBM that models the input $a = h(0)$ as its visible layer.

Step 2: By using the first layer obtain representation of the input that will be used as input for the next layer

$$P(h^{(1)}=1 | h^{(0)}) \text{ or } P(h^{(1)} | h^{(0)})$$

3.2

Step 3: Train the second layer as an RBM

Step 4: Repeat step 2 and step 3 for all the number of layers

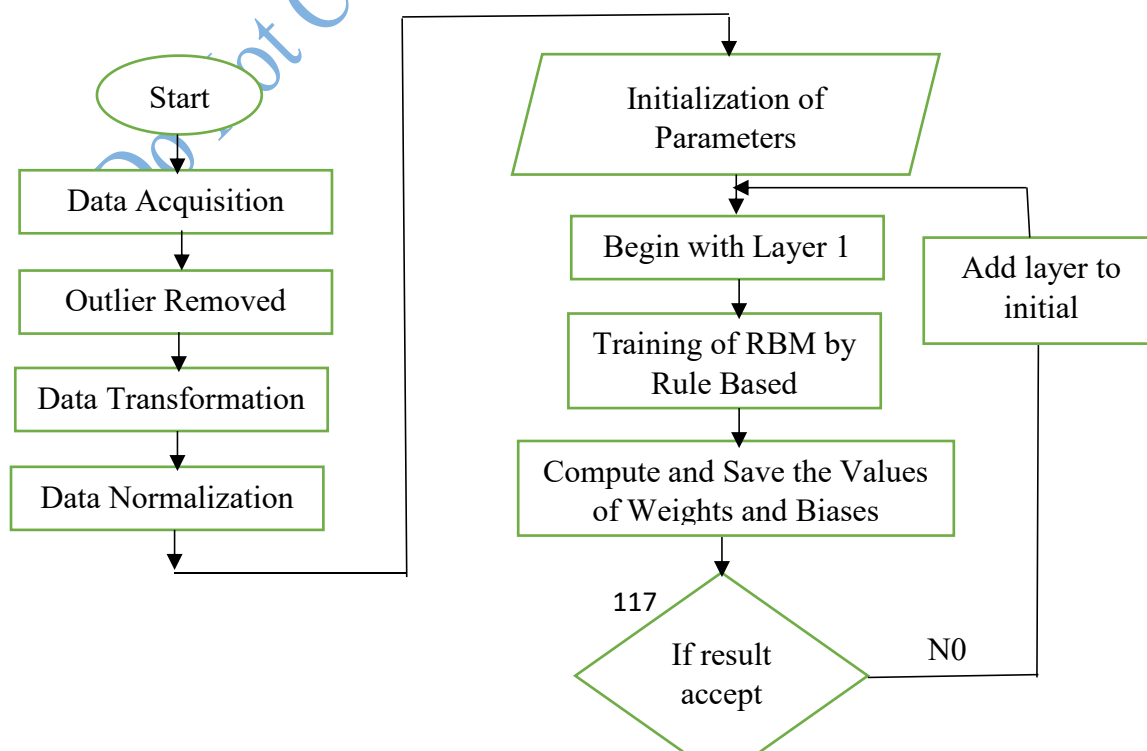


Figure 3.5 DBN Flowchart. (Researcher: Awodele S. O. 2023)

3.3.4.2 Multilayer Perceptron (MLP)

Neural networks are one of the areas of artificial intelligence study centered on attempts to replicate the human nervous system, notably the nervous system's capacity to learn from mistakes and rectify them, which should allow the work of the human brain to be roughly simulated. Neurons make up the neural network. Figure 3.14 depicts the neuron's block diagram.

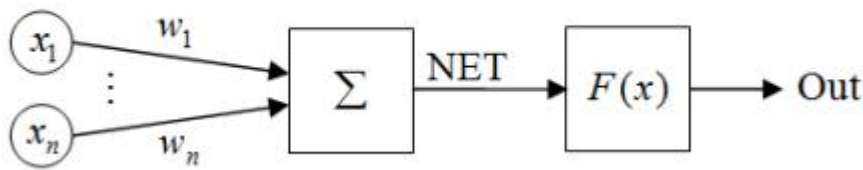


Figure 3.6: Structural scheme of the neuron¹⁸

The structure of the neuron from the above represented:

1. Input signals.
2. Weighting factors.

3. Composer and its output NET.
4. The activation function of the neuron $F(x)$.
5. Output signal.

However, MLP transformed the input data into the output data by varying the weights of its internal nodes. A function $f(x)$ is learned using the back propagation learning technique. $f(x): R^i \rightarrow R^o$ learns using MLP by training on a data set where the dimensions of the inputs and outputs, $I, o \in Z^+$, are computed as follows¹⁹.

$$y = \phi(\sum_{i=1}^n w_i X + b) \quad 3.3$$

$$y = \phi(W^T X + b) \quad 3.4$$

Where ϕ is the activation function, w is the weights, X is the input data and b are the bias.

A multi-layered perceptron is made up of interconnected neurons that communicate with one another, much like the human brain. It is possible to value each neuron. There are three main layers to the network.

- Input Layer: This helps to distribute data over the network and does not perform any calculations. The outputs of this layer transmit signals to the inputs of the next layer (hidden or output).
- Hidden layer(s): These are layers of normal neurons that digest information obtained from the previous layer and transmit signals from the input to the output. Their input is the output of the previous layer, and the output is the input of the next layer.
- Output layer: Typically, one neuron (or more) provides the result of the neural network's calculations.

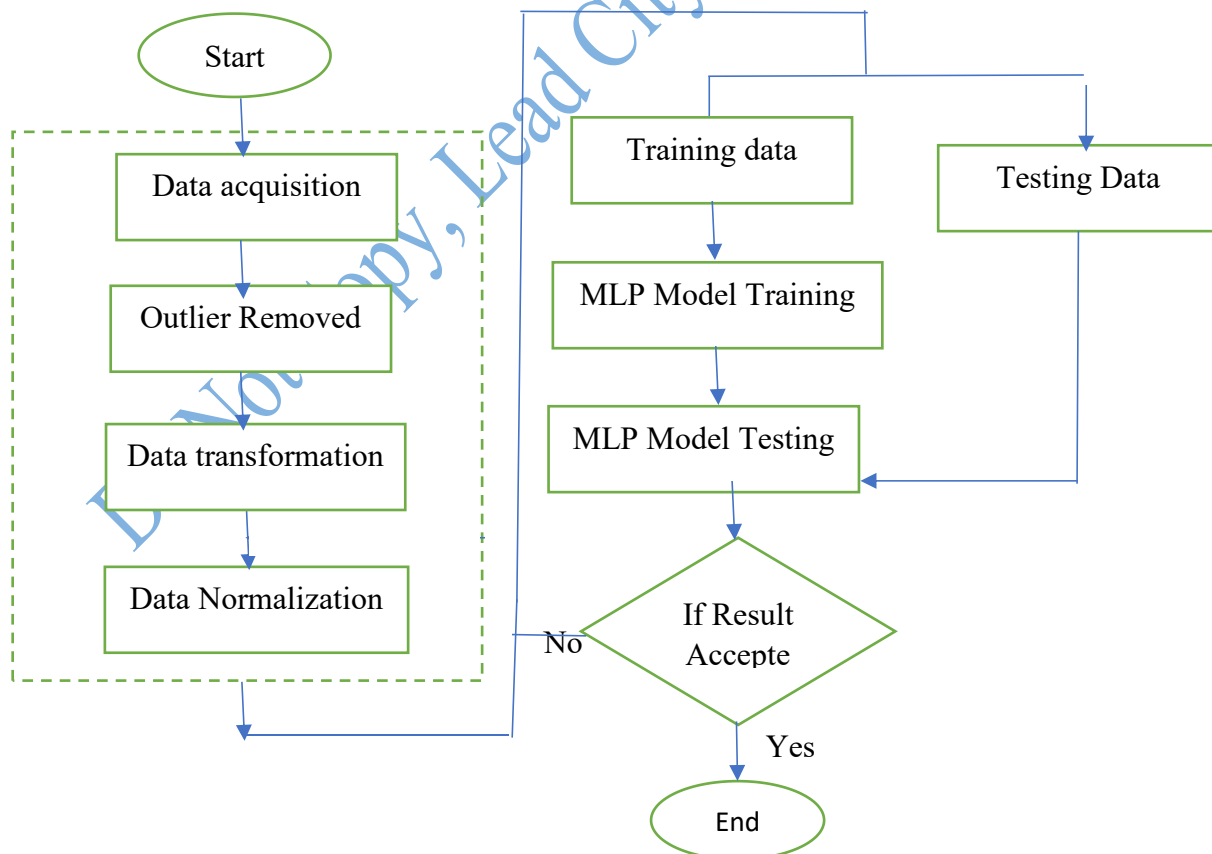


Figure 3.7; MLP Flowchart. (Researcher: Awodele S. O. 2023)

3.4 Research Methods

In this research work, we investigate the performance of deep belief network (DBN) and Multilayer perceptron (MLP) algorithm for detecting intrusion in cloud network environment. The two different deep learning algorithms have been combined together to reduce their weaknesses and take advantage of the positive features of both algorithms to optimize the model with high detection accuracy to reduce the false positive rate in intrusion detection and to evaluate the performance of the hybrid model against the conventional model of intrusion detection system. Figure 3.1 illustrates the proposed model approach. A quantitative approach was used for this research. We consulted a variety of research articles and publications during the literature review. In this first stage of the literature study, we discovered research had been done on intrusion detection systems (IDS) using different methods and techniques, which include the traditional methods of detecting attacks and intrusions as well as the use of soft computing in the likes of fuzzy logic, genetic algorithms, artificial neural networks, expert systems, machine learning, and deep learning algorithm techniques.

Therefore, our second phase of the literature review was mainly focused on IDS using traditional methods, machine learning, and deep learning techniques. During this stage, we tried to look and study the implementation processes, the results and limitations attached to the methods, and the techniques and ways to improve them. Many various machine learning and deep learning techniques were used on different intrusion datasets. From our findings, we decide to go into further detail on the idea of modeling an intrusion detection system for a deep learning

framework using both supervised and unsupervised deep learning algorithms on the UNSW-NB15 dataset, and to develop an optimal model from the output of parallel learning of the MLP and DBN stage to form a hybridized model of intrusion detection in a cloud network environment that could aid in enhancing the accuracy, detection rate, precision, false negative alarms, and true positive alarms of an IDS. During the initial stage, the model used multilayer perceptrons (MLP) and deep belief networks (DBN) on the UNSW-NB15 dataset, which is available for free on the internet. Figure 3.6 shows the flow sequences diagram of a network intrusion detection system model.

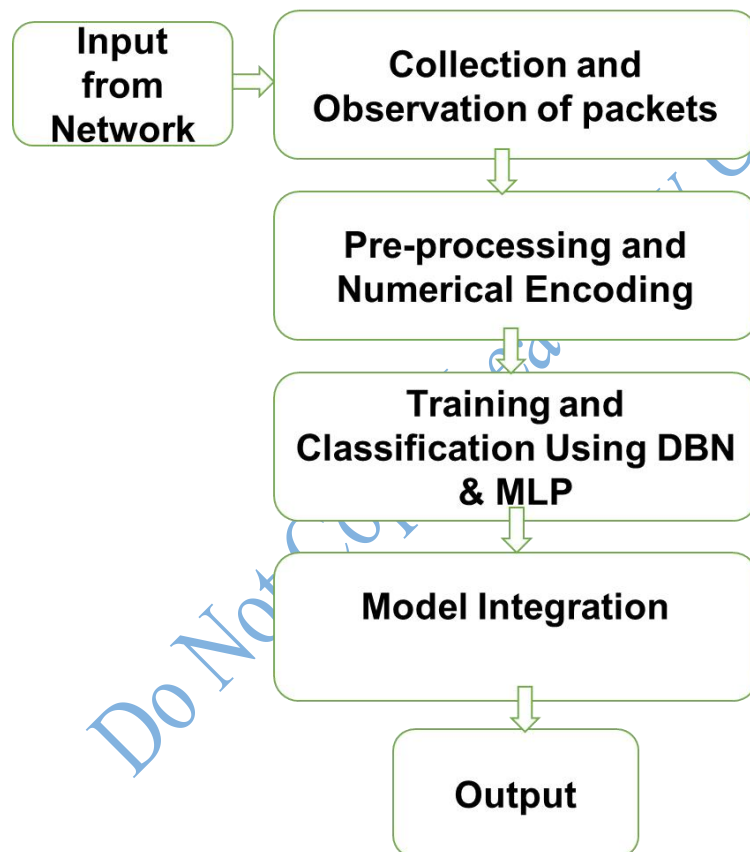


Figure 3.8 Flow sequences of a network intrusion detection system model. (Researcher: Awodele S. O. 2023)

3.4.1 Hybrid of Deep Belief Network (DBN) & Multilayer Perceptron (MLP)

Developing a unique classifier that combines many classifiers is one way to decrease false positives and increase accuracy rate of a model this method outperforms using individual classifiers. The unique classifier in our case is the hybridization of deep belief network (DBN) and multilayer perceptron (MLP). The strength of DBNs excel in capturing complex pattern and hierarchical features from raw data, making them valuable for unsupervised tasks and generative modeling. While MLPs are versatile in handling diverse tasks and intricate non-linear relationships efficiently. Therefore, developing and training hybrid model require careful consideration of architectures to effectively combine the strengths of both components. In the field of intrusion detection, the challenge lies in accurately identifying and mitigating unauthorized access and malicious activities within a network. Traditional methods often face limitations in handling complex, evolving threats which leads to low recognition accuracy and a high false alarm rate. To address this challenge, a pioneering approach involving the development of a hybrid model was explored. This research delved into the integration of a deep belief network (DBN) and a multilayer perceptron (MLP) using the parallel integration technique, with the aim of harnessing the synergistic strengths of both algorithms. The objective was to optimize the hybrid model's performance and accuracy, enabling it to adeptly identify intrusion attempts and ensure the security of network systems.

To achieve this, an optimization process was initiated. The hyperparameters and parameter functions of the DBN network architecture were fine-tuned using the Gridsearch technique. This method systematically explored the hyperparameter space, selecting functions that would effectively optimize the DBN model. Subsequently, the multilayer perceptron architecture underwent a similar rigorous optimization process to ensure an optimal configuration for the hybridization process.

Integrating the multilayer perceptron into the DBN network architecture was imperative to enhance the model's capabilities. Unlike DBN, primarily known as a clustering algorithm, the complete DBN model was transformed into a classification model through seamless integration with the MLP network. DBNs possess versatility, applicable to both unsupervised and semi-supervised learning tasks. While primarily used for unsupervised tasks such as dimensionality reduction and feature learning, they can be adapted for supervised learning. One prevalent approach involves pre-training a DBN in an unsupervised manner to extract valuable features from the data. Subsequently, the network is fine-tuned using supervised learning techniques like backpropagation, enabling it to perform specific supervised tasks such as classification or regression. This process, referred to as pre-training, combines unsupervised pre-training and supervised fine-tuning, empowering DBNs to effectively handle supervised learning tasks. The integration of DBN with MLP to make it a single model was done using Scikit-learn's pipeline functionality, ensuring a seamless connection between the optimized DBN model and the constructed MLP architecture. Furthermore, the completed DBN model with the MLP model was hybridized with another distinct MLP model architecture with the aid of a voting classifier in Scikit-Learn.

Algorithm 3.1: Proposed Algorithm for hybrid model

1. Start
2. Data Preprocessing:
 - Handle Missing Values if Any.
 - Drop Unnecessary Columns (e.g., 'id', 'attack_cat') from the Dataset.
3. Feature Scaling:
 - Scale Numerical Features to a Standard Scale (e.g., using StandardScaler).
4. Feature Selection:
 - Use Feature Selection Techniques (e.g., SelectKBest and chi-squared statistics) to Choose Relevant Features.

5. Data Splitting:
 - Split the Preprocessed Data into Training and Testing Sets.
6. Initialize Models:
 - Initialize the DBN Model with:
 - Number of Hidden Layers: 2
 - Learning Rate: 0.1
 - Number of Epochs: 100
 - Initialize MLP Model with:
 - Number of Layers: 3
 - Activation Function: ReLU
 - Learning Rate: 0.01
 - Number of Epochs: 50
7. Train Models:
 - Train DBN Model using the Training Data.
 - Train MLP Model using the Training Data.
8. Generate Predictions:
 - Obtain DBN Predictions for both Training and Testing Sets.
 - Obtain MLP Predictions for both Training and Testing Sets.
9. Hybridize Model:
 - Create an Hybridize Model using Soft Voting:
 - Combine DBN and MLP Predictions using Soft Voting.
10. Evaluation:
 - Evaluate the Hybrid Model on the Testing Data.
 - Calculate and Analyze Performance Metrics:
 - Accuracy, ROC Curves, Confusion Matrix, etc.
11. Result Analysis:
 - Interpret the Results to Understand the Hybrid Model's Performance.
 - Visualize the Results using Plots and Charts.
12. Output:
 - Best Solution (Hybrid Model)
13. End

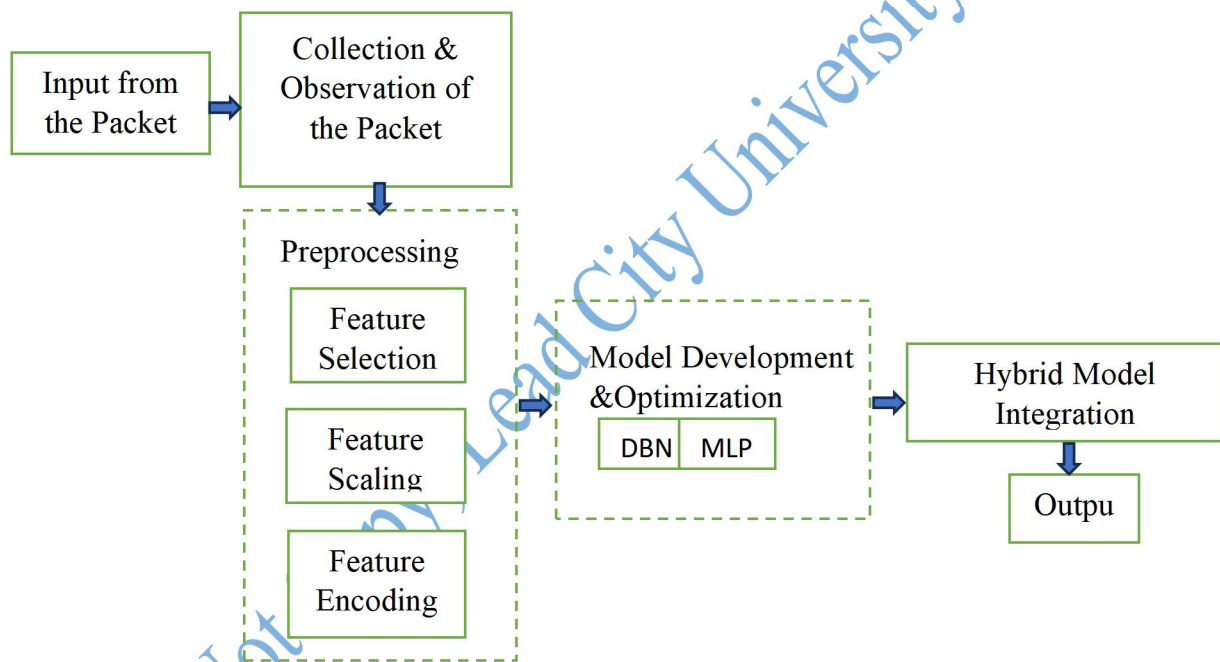


Figure 3.9 Flow Sequence of Hybrid DBN and MLP. (Researcher: Awodele S. O. 2023)

3.4.2 Optimization of the hybrid model

The process of model optimization and training is outlined. Firstly, a hyperparameter space is defined, encompassing various configurations for hidden layer sizes, activation functions, solver

options, alpha values, and learning rate options. Next, an MLP (Multi-Layer Perceptron) classifier is initialized with a maximum of 100 iterations. Grid search, employing GridSearchCV, explores the defined hyperparameter space to find the optimal configuration for the MLP classifier. Also, a Deep Belief Network (DBN) pipeline is created, incorporating Restricted Boltzmann Machines (RBMs) for feature learning and an MLP model. The DBN pipeline is trained on the provided training data. After training, the model's performance is evaluated using test data, and if the accuracy falls short of expectations, further refinements are considered. These steps illustrate a systematic approach to finding the best model configuration through hyperparameter tuning and potentially fine-tuning strategies for improved accuracy and reliability in intrusion detection tasks.

Algorithm 3.2: Optimization of Hybrid Model

1. Define Hyperparameter Space:

- Set `hidden_layer_sizes_range = [(10, 30, 10), (20,)]`
- Set `activation_functions = ['tanh', 'relu']`
- Set `solver_options = ['sgd', 'adam']`
- Set `alpha_values = [0.0001, 0.05]`
- Set `learning_rate_options = ['constant', 'adaptive']`

2. Initialize MLP Classifier:

- Initialize `mlp_classifier` with `max_iter=100`

3. Perform Grid Search:

- Set `parameter_space = {`
 - `'Hidden_layer_sizes': hidden_layer_sizes_range,`
 - `'activation': activation_functions,`
 - `'solver': solver_options,`
 - `'alpha': alpha_values,`
 - `'learning_rate': learning_rate_options``}`
- Perform `grid_search` using `GridSearchCV` with `mlp_classifier`, `parameter_space`, `cv=5`, and `n_jobs=-1`

4. Create DBN Pipeline:

- Create `dbn_pipeline` with steps:

- ('rbm', BernoulliRBM(n_components=256, learning_rate=0.1, n_iter=20, verbose=True))
- ('mlp', clone(grid_search))

5. Train and Optimize:

- Fit dbn_pipeline on training data: dbn_pipeline.fit(X_train, y_train)

6. Evaluate and Fine-Tune (Optional):

- Calculate test_accuracy using dbn_pipeline.score(X_test, y_test)
- If performance not satisfactory:
 - Perform further hyperparameter tuning or explore other optimization techniques.

3.4.3 Performance Metric

The Receiver Operating Characteristic Curve (ROC) curves are valuable tools for assessing model performance in relation to the false positive and true positive rates. The ROC curve plots the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis, and the area under the curve (AUC) represents the degree of separation between the classes.

Additionally, a Confusion Matrix is a useful visual tool for evaluating the performance of a model in a tabular format. It is constructed by comparing the predicted class labels to the actual class labels. The matrix displays four outcomes: true positives, false negatives, true negatives, and false positives.

- True positive is when abnormal network is correctly classified as abnormal.
- False negative is when abnormal network is incorrectly classified as normal.
- True negative is when normal network is correctly classified as normal.
- False positive is when normal network is incorrectly classified as abnormal.

To select the best model, the followings metrics are used:

Precision

Precision is a metric that measures the ability of a model to make accurate predictions. It is a measure of the exactness of the model and is calculated as the number of true positive predictions divided by the number of true positive predictions plus the number of false positive predictions.

$$Precision = \frac{TP}{TP+FP} \quad 3.5$$

Recall

In the context of intrusion detection, recall is also known as sensitivity or True Positive Rate (TPR). It is a measure of the model's ability to correctly classify normal network. It is calculated by dividing the number of true positive (correctly classified normal networks) by the number of true positive plus the number of false negative (normal networks that are incorrectly classified as abnormal networks). A high recall value indicates that the model can correctly identify most of the normal networks, while a low recall value means that the model is missing many normal networks and likely classifying them as abnormal.

$$Recall = \frac{TP}{TP+FN} \quad 3.6$$

F-Measure

F-measure, also known as F1 Score, is a composite measure of the accuracy of a classification model, which is calculated as the harmonic mean of precision and recall. It is a trade-off between the precision and recall values, considering both the number of false positives and false negatives. It can be used as a single-number summary of a model's performance. F-measure is particularly useful when you want to seek a balance between precision and recall, as it will be close to the best value of precision and recall only if both have similar values.

$$F = \frac{2 * Precision * Recall}{Precision + Recall} \quad 3.7$$

Accuracy

Accuracy is a commonly used metric for evaluating the performance of a classification model. It is the proportion of total classifications that were correctly made by the model. It is calculated as the ratio of correctly predicted observations to the total number of observations. In other words, it is the number of correct predictions made by the model divided by the total number of predictions made by the model.

$$ACC = \frac{TP+TN}{FP+FN+TP+TN} \quad 3.8$$

Endnote

1. M. N. Saunders, P. Lewis, & A. Thornhill, *Research Methods for Business Students (Ninth edition)*, **Pearson**, 2023.
2. S. Ahmad, S. Wasim, S. Irfan, S. Gogoi, A. Srivastava, Z. Farheen, “Qualitative v/s. Quantitative Research- A Summarized Review,” **Journal of Evidence Based Medicine and Healthcare** 6, no. 43, 2019, 2828-2832.
3. P. S. Bravin, “A review on Data Preprocessing Techniques in Data Mining,” **International Journal of Advances in Engineering and Management (IJAEM)** 4, no. 5, 2022, 1206-1209
4. Z. Zoghi, & G. Serpen, “UNSW-NB15 Computer Security Dataset: Analysis through Visualization.” **Electrical Engineering & Computer Science, University of Toledo, Ohio, USA**, 2021.
6. A. Kumar, “Feature Scaling in Machine Learning: Python Examples - Data,” **Data Analytics**, 2022, <https://vitalflux.com/python-improve-model-performance-using-feature-scaling/>.
5. S. Meftah, T. Rachidi, & N. Assem, “Network Based Intrusion Detection Using the UNSW-NB15 Dataset,” **International Journal of Computing and Digital Systems** 8, no. 5. 2019, 477–87.
- 7.. A. R. Sonule, M. Kalla, A. Jain, & D. S. Chouhan, “Unsw-Nb15 Dataset and Machine Learning Based Intrusion Detection Systems,” **International Journal of Engineering and Advanced Technology** 9, no. 3 2020. 2638–2648.

8. O. Belarbi, A. Khan, P. Carnelli, & T. Spyridopoulos, “*An Intrusion Detection System Based on Deep Belief Networks*,” **International Conference on Science of Cyber Security**, 2022, 377–92.
9. C. Qiao, B. Gao, & Y. Shi, “*SRS-DNN: A Deep Neural Network with Strengthening Response Sparsity*,” **Neural Computing and Applications** 32, no. 12, 2019. 8127-8142.
10. U. Rishabh, “*Review of Accent Classification Using Deep Belief Network*,” **Conceicao Rodrigues College of Engineering, Bandra, Mumbai**. 2017.
11. Cherifi Imane, “*Deep Belief Network*,” **Open Genus IQ: Computing Expertise & Legacy**, 2022, <https://iq.opengenus.org/deep-belief-network/>
- 12.. M. Karpinski, M. Karpinski, A. Shaikhanova, O. Veselska, & A. Azatov, “*Detection of Intrusion Attacks Using Neural Networks*,” **International Scientific and Practical Conference “Information Security and Information Technologies**, 2021.
13. A. Ali; S. Shaukat, M. Tayyab, M. A. Khan, J. S. Khan. & A. Jawa, “*Network Intrusion Detection Leveraging Machine Learning and Feature Selection*,” **International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)**, 2020, 49-53.

Chapter Four

Results and Discussion

In this study, a hybrid model for detecting intrusions in cloud network environments using a combination of deep belief network (DBN) and multi-layer perceptron (MLP) algorithms was implemented.

4.1 Results

This section presents the results obtained from hybridizing the Multi-Layer Perceptron (MLP) and Deep Belief Network (DBN) using Python on Jupyter Notebook. Based on the analyses, conclusions are drawn on the effectiveness of this model for intrusion detection and how it can be utilized based on its performance evaluation. Python was chosen as the programming language due to its robustness and suitability for data analysis and machine learning-oriented projects. The intrusion detection dataset used was in Comma Separated Values (CSV) format.

The UNSW dataset is a network intrusion detection dataset that provides various types of network traffic data that simulate a realistic network environment. It was developed to aid in the

development and evaluation of intrusion detection systems. The dataset contains several network traffic features, including basic features such as duration, source and destination IPs and ports, protocol, flags, and more advanced features such as flow statistics (e.g., the number of packets and bytes per flow), packet statistics (e.g., size, inter-arrival times, and payloads), and time-based traffic statistics (e.g., number of connections and packets per time interval).

The dataset is labelled, meaning that each traffic record is classified as either benign or malicious. The malicious traffic is further categorized into several attack categories such as DoS, probing, and infiltration attacks. The UNSW dataset is widely used in academic research and industry to develop and evaluate network intrusion detection systems. It is also publicly available for download and can be used for educational and research purposes.

4.1.1 Result of the Hybrid Model of Deep Belief Network and Multilayer Perceptron

The implementation of the model involved several steps to prepare the dataset for analysis and develop the multiplayer perceptron (MLP) and deep belief network (DBN) algorithms. These steps included pre-processing, data partitioning, and performance evaluation.

The pre-processing step involved data cleaning, visualization, and transformation to prepare the dataset for analysis. The dataset did not have any missing values, and it was visualized using several Python libraries such as Numpy, Pandas, Scit-learn, Seaborn, and Matplotlib. Since MLP and DBN algorithms only work with numerical data, categorical data were converted to numerical using the Label Encoder library. The purpose of visualization was to show the distribution of variables (features) in the entire dataset and gain insights from the data.

The next step was data partitioning, which involved dividing the dataset into training and testing sets. The training set was used to train the model, while the testing set was used to evaluate its

performance. Feature scaling was also performed on the dataset to ensure that all features have the same scale and prevent certain features from dominating others.

The final step was developing and evaluating the MLP and DBN algorithms. The performance of the algorithms was evaluated using various metrics such as accuracy, precision, recall, F1-score, and receiver operating characteristic (ROC). The purpose of this evaluation was to determine the effectiveness of the model in detecting intrusions.

```

In [84]: import os
os.chdir('C:/Users/user/Documents/Project/intrusion detection')

In [85]: import warnings
warnings.filterwarnings('ignore')

In [86]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.graph_objects as go

In [87]: df=pd.read_csv('UNSW.csv')

In [88]: #dataset shape
df.shape

Out[88]: (82332, 45)

In [89]: df.head(5)

Out[89]:

```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftw_cmd	ct_ftw_http_mth
0	1	0.000011	udp	-	INT	2	0	496	0	90909.0902	...	1	2	0	0	
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.0003	...	1	2	0	0	
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.0051	...	1	3	0	0	
3	4	0.000006	udp	-	INT	2	0	900	0	166666.6608	...	1	3	0	0	
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.0025	...	1	3	0	0	

Figure 4.1 Path navigation, Python Libraries and Loading Dataset. (Researcher: Awodele S. O. 2023)

Figure 4.1 illustrated how the system path was navigated and shows the python libraries used for reading both dataset and feature names files. The python library used to read the files is called

Pandas literally means python for data analysis. The dataset used contains 82332 rows and 45 features.

NOTE: To read open the jupyter notebook, the anaconda prompt was navigated on the laptop by searching for the menu followed by typing the jupyter lab on the anaconda prompt.

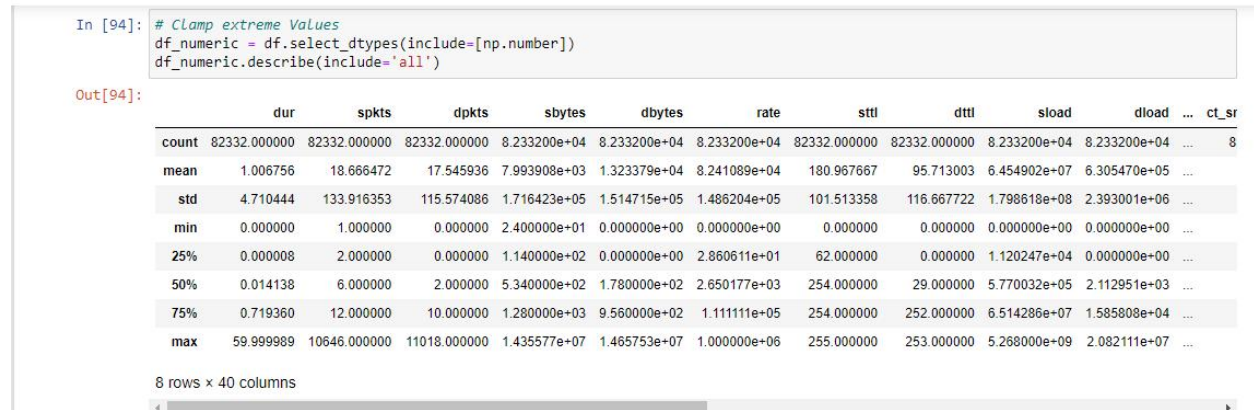


Figure 4.2 Data Preprocessing. (Researcher: Awodele S. O. 2023)

The code provided performs a data transformation on a pandas DataFrame named "df".

First, the code selects only the numeric columns from the DataFrame using the "select_dtypes" method from pandas. This method filters columns based on their data type, in this case selecting only columns that are numeric. The "include" parameter specifies the data types to include, which is set to "[np.number]" here. This requires the numpy library to be imported as "np".

Next, the "describe" method is called on the numeric columns of the DataFrame to generate summary statistics of the data. The "include" parameter is set to "all" to include summary statistics for both numeric and non-numeric columns.

The purpose of these operations is to clamp extreme values in the DataFrame. The "describe" method provides summary statistics such as the minimum and maximum values for each numeric

column, which can be used to identify extreme values. These values can be clamped or truncated to a specific range to remove any potential outliers or anomalies in the data.

It's worth noting that this code does not actually clamp or transform any values in the DataFrame; rather, it selects and provides summary statistics of the numeric columns, which can be used to inform subsequent data transformations.

```
In [95]: DEBUG = 0

for feature in df_numeric.columns:
    if DEBUG == 1:
        print(feature)
        print('max = ' + str(df_numeric[feature].max()))
        print('75th = ' + str(df_numeric[feature].quantile(0.95)))
        print('median = ' + str(df_numeric[feature].median()))
        print(df_numeric[feature].max() > 10 * df_numeric[feature].median())
        print('-----')
    if df_numeric[feature].max() > 10 * df_numeric[feature].median() and df_numeric[feature].max() > 10 :
        df[feature] = np.where(df[feature] < df[feature].quantile(0.95), df[feature], df[feature].quantile(0.95))

In [96]: df_numeric = df.select_dtypes(include=[np.number])
df_numeric.describe(include='all')
```

Out[96]:

	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	...	ct_sr
count	82332.000000	82332.000000	82332.000000	82332.000000	82332.000000	82332.000000	82332.000000	82332.000000	8.233200e+04	8.233200e+04	...	8.
mean	0.445016	11.84186	9.178424	1580.566135	2866.918367	71576.702810	180.967667	95.713003	4.649418e+07	3.105380e+05	...	
std	0.672222	15.66461	14.504212	2948.850472	7525.606738	102631.946851	101.513358	116.667722	7.417784e+07	8.918691e+05	...	
min	0.000000	1.000000	0.000000	24.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	...	
25%	0.000008	2.000000	0.000000	114.000000	0.000000	28.606114	62.000000	0.000000	1.120247e+04	0.000000e+00	...	
50%	0.014138	6.000000	2.000000	534.000000	178.000000	2650.176667	254.000000	29.000000	5.770032e+05	2.112951e+03	...	
75%	0.719360	12.000000	10.000000	1280.000000	956.000000	111111.107200	254.000000	252.000000	6.514286e+07	1.585808e+04	...	
max	2.403792	60.000000	54.000000	12472.000000	30622.000000	333333.321500	255.000000	253.000000	2.666667e+08	3.741446e+06	...	

8 rows x 40 columns

Figure 4.3 Data Pre-processing. (Researcher: Awodele S. O. 2023)

This code is used to identify and clamp extreme values in the numeric columns of a pandas DataFrame named "df".

The first line sets a variable named "DEBUG" equal to 0, which appears to be a flag to toggle whether or not to print debugging information for each feature in the DataFrame. If "DEBUG" is set to 1, the code will print information about each feature, including the maximum value, 75th percentile, median, and whether the maximum value is greater than 10 times the median.

Next, the code loops over each feature in the numeric columns of the DataFrame using a for loop. For each feature, the code checks whether the maximum value is greater than 10 times the median and whether the maximum value is greater than 10. If both conditions are true, the code replaces any values in the feature that are less than the 95th percentile with the 95th percentile value using the numpy "where" function.

This process effectively "clamps" or truncates extreme values in the DataFrame to a specific range, which can help to remove outliers or anomalies in the data. The specific threshold for identifying extreme values is based on the maximum value and median of each feature, as well as a threshold of 10. The code is designed to be configurable by setting the "DEBUG" variable to 0 or 1, which controls whether or not to print debugging information. Overall, this code is a useful tool for data cleaning and pre-processing, particularly when dealing with large datasets with many features and potential outliers.

```
In [97]: df_numeric = df.select_dtypes(include=[np.number])
df_before = df_numeric.copy()
DEBUG = 0
for feature in df_numeric.columns:
    if DEBUG == 1:
        print(feature)
        print('unique = ' + str(df_numeric[feature].nunique()))
        print(df_numeric[feature].nunique() > 50)
        print('-----')
    if df_numeric[feature].nunique() > 50:
        if df_numeric[feature].min() == 0:
            df[feature] = np.log(df[feature] + 1)
        else:
            df[feature] = np.log(df[feature])
df_numeric = df.select_dtypes(include=[np.number])

In [98]: df_cat = df.select_dtypes(exclude=[np.number])
df_cat.describe(include='all')
```

Out[98]:

	proto	service	state
count	82332	82332	82332
unique	131	13	7
top	tcp	-	FIN
freq	43095	47153	39339

Figure 4.4 Data Preprocessing. (Researcher: Awodele S. O. 2023)

This code performs some data preprocessing steps on a pandas DataFrame in Python. Here is a step-by-step explanation of what the code is doing:

`df_numeric = df.select_dtypes(include=[np.number])` - This line selects all columns in the DataFrame `df` that have a data type of `np. number`, which includes numeric data types such as integers and floats. The selected columns are stored in a new DataFrame called `df_numeric`.

`df_before = df_numeric.Copy()` - This line creates a copy of the `df_numeric` DataFrame for comparison purposes.

`DEBUG = 0` - This line sets a debugging flag to 0, which means that the print statements in the following loop will not be executed.

`for the feature in df_numeric.columns:` - This line starts a loop that iterates through each column in the `df_numeric` DataFrame.

`if DEBUG == 1:` - This line checks if the debugging flag is set to 1.

`if df_numeric[feature].nunique()>50:` - This line checks if the number of unique values in the current column is greater than 50.

`if df_numeric[feature].min()==0:` - This line checks if the minimum value in the current column is equal to 0.

`df[feature] = np.log(df[feature]+1)` - This line replaces the values in the current column of the original DataFrame `df` with their natural logarithm plus 1. This is done to prevent taking the logarithm of 0 values.

`else:` - This line executes if the minimum value in the current column is not equal to 0.

`df[feature] = np.log(df[feature])` - This line replaces the values in the current column of the original DataFrame `df` with their natural logarithm.

`df_numeric = df.select_dtypes(include=[np.number])` - This line selects all columns in the modified DataFrame `df` that have a data type of `np.number`, which includes numeric data types such as integers and floats. The selected columns are stored in a new DataFrame called `df_numeric`. This code performs some exploratory data analysis on a pandas DataFrame in Python. Here is a step-by-step explanation of what the code is doing:

`df_cat = df.select_dtypes(exclude=[np.number])` - This line selects all columns in the DataFrame `df` that do not have a data type of `np.number`, which includes non-numeric data types such as strings and categorical variables. The selected columns are stored in a new DataFrame called `df_cat`.

`df_cat.describe(include='all')` - This line generates descriptive statistics for the `df_cat` DataFrame, including count, unique, top, and frequency. The `include='all'` argument ensures that statistics are generated for both numeric and non-numeric columns.

By default, the `describe()` method only generates statistics for numeric columns, so specifying `include='all'` in this case includes non-numeric columns as well. The resulting output provides insights into the distribution of values in the non-numeric columns, including the number of unique values, the most common value, and the frequency of the most common value. This information can be useful in understanding the data and identifying potential data quality issues, such as missing values or inconsistent values.


```

In [99]: DEBUG = 0
for feature in df_cat.columns:
    if DEBUG == 1:
        print(feature)
        print('nunique = ' + str(df_cat[feature].nunique()))
        print(df_cat[feature].nunique() > 6)
        print(sum(df[feature].isin(df[feature].value_counts().head().index)))
        print('-----')

    if df_cat[feature].nunique() > 6:
        df[feature] = np.where(df[feature].isin(df[feature].value_counts().head().index), df[feature], '-')

In [100]: df_cat = df.select_dtypes(exclude=[np.number])
df_cat.describe(include='all')

Out[100]:

```

	proto	service	state
count	82332	82332	82332
unique	6	5	6
top	tcp	-	FIN
freq	43095	49275	39339

```

In [101]: df['proto'].value_counts().head().index
Out[101]: Index(['tcp', 'udp', '-', 'unas', 'arp'], dtype='object')

In [102]: df['proto'].value_counts().index
Out[102]: Index(['tcp', 'udp', '-', 'unas', 'arp', 'ospf'], dtype='object')

```

Figure 4.5. Data Preprocessing. (Researcher: Awodele S. O. 2023)

This code performs some data preprocessing steps on a pandas DataFrame in Python. Here is a step-by-step explanation of what the code is doing:

DEBUG = 0 - This line sets a debugging flag to 0, which means that the print statements in the following loop will not be executed.

for feature in df_cat.columns: - This line starts a loop that iterates through each column in the df_cat DataFrame, which contains non-numeric columns.

if DEBUG == 1: - This line checks if the debugging flag is set to 1.

if df_cat[feature].nunique() > 6: - This line checks if the number of unique values in the current column is greater than 6.

df[feature] = np.where(df[feature].isin(df[feature].value_counts().head().index), df[feature], '-') -

This line replaces the values in the current column of the original DataFrame df with the original

value if it is in the top 5 most common values in the column, or with a dash (-) otherwise. This is done to reduce the number of unique values in the column and prevent overfitting in models that may be trained on this data. The `np.where()` function is used to perform the conditional replacement.

The `if DEBUG == 1:` block of code contains several print statements that provide additional information about the column being processed. Specifically, it prints the name of the column, the number of unique values in the column, whether the number of unique values is greater than 6, the number of values in the column that are in the top 5 most common values, and a separator line.

Overall, this code performs a value reduction technique on non-numeric columns with more than 6 unique values. It replaces values that are not in the top 5 most common values with a dash (-). The resulting DataFrame is modified in place. The debugging code can be used to understand the impact of this data preprocessing step on the non-numeric columns.

```

In [103]: # Feature Selection
from sklearn.feature_selection import SelectKBest, chi2

best_features = SelectKBest(score_func=chi2,k='all')

X = df.iloc[:,4:-2]
y = df.iloc[:,-1]
fit = best_features.fit(X,y)

df_scores=pd.DataFrame(fit.scores_)
df_col=pd.DataFrame(X.columns)

feature_score=pd.concat([df_col,df_scores],axis=1)
feature_score.columns=['feature','score']
feature_score.sort_values(by=['score'],ascending=True,inplace=True)

fig = go.Figure(go.Bar(
    x=feature_score['score'][0:21],
    y=feature_score['feature'][0:21],
    orientation='h'))

fig.update_layout(title="Top 20 Features",
    height=1200,
    showlegend=False,
)

fig.show()

```

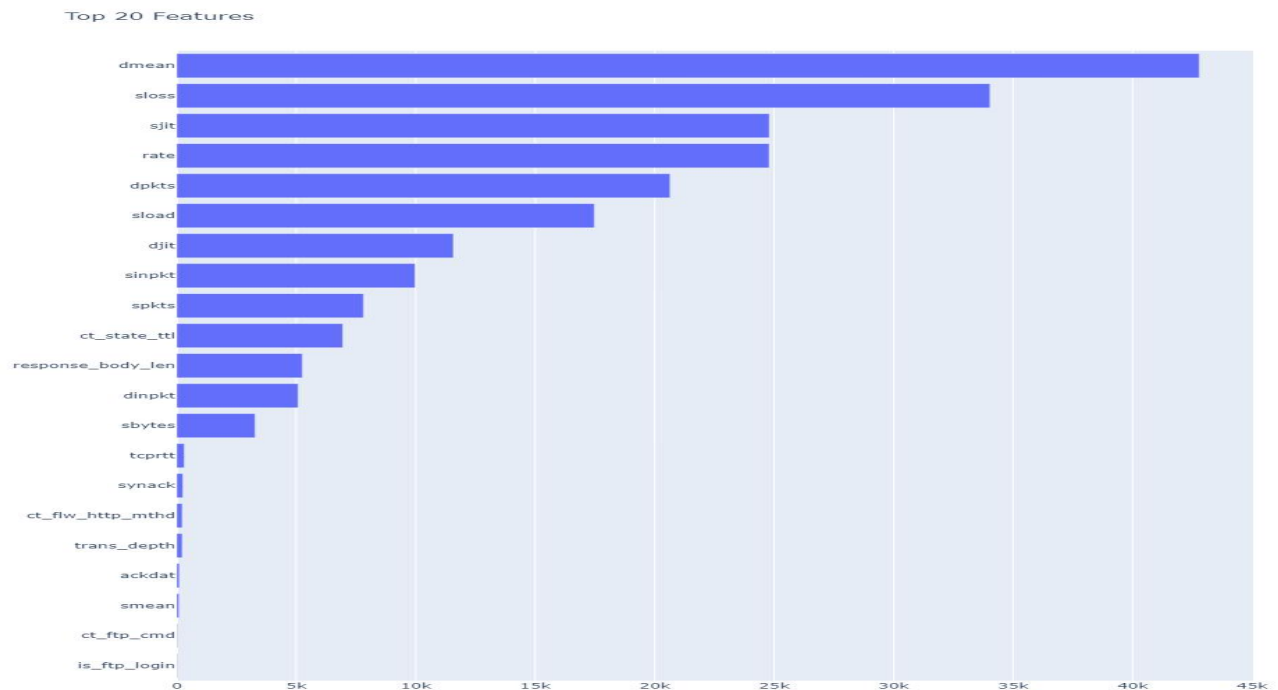


Figure 4. 6 Feature Selection. (Researcher: Awodele S. O. 2023)

This code performs feature selection using the chi-squared test (chi2) for categorical variables. The SelectKBest method from scikit-learn is used to select the best k features (in this case, 'all') based on the chi-squared test scores.

The input data is assumed to be stored in a pandas DataFrame called 'df', where the X variable contains the feature matrix (all columns except for the last two), and the y variable contains the target variable (last column).

After fitting the SelectKBest method to the data, the chi-squared test scores for each feature are stored in a DataFrame called 'feature_score'. This DataFrame is then sorted in ascending order based on the scores.

Finally, the top 20 features with the highest chi-squared test scores are plotted in a horizontal bar chart using the Plotly package. The resulting plot shows the names of the top 20 features on the y-axis and their corresponding scores on the x-axis.

```
In [106]: from sklearn.compose import ColumnTransformer
          from sklearn.preprocessing import OneHotEncoder
          ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1,2,3])], remainder='passthrough')
          X = np.array(ct.fit_transform(X))

In [107]: np.shape(X)
Out[107]: (82332, 56)

In [108]: df_cat.describe(include='all')
Out[108]:
```

	proto	service	state
count	82332	82332	82332
unique	6	5	6
top	tcp	-	FIN
freq	43095	49275	39339

```
In [109]: len(feature_names)
Out[109]: 42
```

Figure 4.7. Data Transformation. (Researcher: Awodele S. O. 2023)

This code utilizes the ColumnTransformer class from the scikit-learn library to transform and preprocess data. Specifically, it applies one-hot encoding to three columns (1, 2, and 3) of a dataset and passes through the remaining columns unmodified.

Here is a step-by-step explanation of what is happening:

The first line imports the necessary classes from the scikit-learn library: `ColumnTransformer` and `OneHotEncoder`.

The second line creates a `ColumnTransformer` object called `ct`. The `"transformers"` argument specifies that we want to apply the `OneHotEncoder` to the columns with indices 1, 2, and 3 of our dataset (presumably stored in a variable called `"X"`). The `"remainder"` argument specifies that we want to pass through the remaining columns unchanged.

The third line applies the transformation to our dataset. We first call the `"fit_transform"` method of the `ColumnTransformer` object `ct` on our dataset `"X"`. This applies the transformations specified by the `ct` object to the data and returns the transformed data. We then convert the transformed data to a NumPy array and store it back in the `"X"` variable.

Overall, this code is useful for preprocessing data that have categorical variables that need to be encoded before they can be used in a machine-learning model. The `ColumnTransformer` object allows for applying different transformations to different columns of the dataset, making it a flexible and powerful tool for data preprocessing.

```

In [110]: for label in list(df_cat['state'].value_counts().index[::-1][1:]):
           feature_names.insert(0,label)

           for label in list(df_cat['service'].value_counts().index[::-1][1:]):
               feature_names.insert(0,label)

           for label in list(df_cat['proto'].value_counts().index[::-1][1:]):
               feature_names.insert(0,label)

In [111]: len(feature_names)
Out[111]: 56

In [112]: from sklearn.model_selection import train_test_split
           X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                             test_size = 0.2,
                                                             random_state = 0,
                                                             stratify=y)

In [113]: df_cat.describe(include='all')
Out[113]:

```

	proto	service	state
count	82332	82332	82332
unique	6	5	6
top	tcp	-	FIN
freq	43095	49275	39339

Figure 4.8 Data Splitting. (Researcher: Awodele S. O. 2023)

This code is performing several tasks related to feature engineering and data splitting in the context of machine learning. Here's a breakdown of what each section does:

The first section loops through the values of the 'state' column in a pandas DataFrame called 'df_cat'. It counts the number of occurrences of each value using the 'value_counts()' method, then selects all but the first value (with index [::-1][1:]) in the resulting list of counts. For each of these selected values, the code inserts the label at the beginning of a list called 'feature_names'. This effectively creates a new categorical feature for each state value, with a binary value of 0 or 1 indicating whether a given data point belongs to that state.

The second section is similar to the first, but instead loops through the values of the 'service' column in 'df_cat', creating new binary features for each service value.

The third section is again similar, but loops through the values of the 'proto' column in 'df_cat', creating new binary features for each protocol value.

The final section imports the 'train_test_split' function from scikit-learn and uses it to split the feature matrix 'X' and target vector 'y' into training and test sets. The 'test_size' parameter specifies that the test set should be 20% of the total data, while 'random_state' ensures reproducibility of the split. The 'stratify' parameter is set to 'y', which means that the class proportions in 'y' will be preserved in the train-test split. This is useful in cases where the target variable is imbalanced, and we want to ensure that the distribution of classes is similar in the training and test sets.

```
In [114]: # 6 + 5 + 6 unique = 17, therefore the first 17 rows will be the categories that have been encoded, start scaling from row 18 on
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, 18:] = sc.fit_transform(X_train[:, 18:])
X_test[:, 18:] = sc.transform(X_test[:, 18:])

In [115]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n=====")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print(" ")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print(" ")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test Result:\n=====")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print(" ")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print(" ")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
```

Figure 4.9 Data Normalization. (Researcher: Awodele S. O. 2023)

This code performs three different tasks. Here is an overview explanation of each section:

The first section is calculating the number of unique categories for three different groups of categorical features ($6 + 5 + 6 = 17$). It then specifies that the StandardScaler should only start scaling the data from the 18th row, which corresponds to the first row of non-categorical data. This is because the categorical data has already been one-hot encoded, so it doesn't need to be scaled.

The second section imports the necessary modules to evaluate the performance of a machine learning model. The 'accuracy_score' function calculates the accuracy of the model, while the 'classification_report' function generates a report that includes metrics such as precision, recall, and F1 score for each class. The 'confusion_matrix' function generates a confusion matrix that shows the number of correct and incorrect predictions for each class.

The third section defines a function called 'print_score' that takes a classifier, training and test data, and a 'train' argument that specifies whether to print results for the training or test set. If 'train' is True, the function calculates and prints the accuracy score, classification report, and confusion matrix for the training set. If 'train' is False, the function does the same thing for the test set.

4.1.2 Result of the Model Optimization to Reduce False Positive Rate

```
In [116]: from sklearn.neural_network import BernoulliRBM
          from sklearn.pipeline import Pipeline
          from sklearn.base import clone
          from sklearn.model_selection import GridSearchCV
          from sklearn.neural_network import MLPClassifier
          from sklearn.ensemble import VotingClassifier

In [117]: mlp1 = MLPClassifier(max_iter=100)
          parameter_space = {
              'hidden_layer_sizes': [(10,30,10),(20,)],
              'activation': ['tanh', 'relu'],
              'solver': ['sgd', 'adam'],
              'alpha': [0.0001, 0.05],
              'learning_rate': ['constant', 'adaptive'],
          }

In [118]: mlp=GridSearchCV(mlp1, parameter_space, n_jobs=-1, cv=5)

In [119]: # Train the MLP model
          mlp.fit(X_train, y_train)
          prediction_MLP = mlp.predict(X_test)

In [120]: dbn = Pipeline(steps=[('rbm', BernoulliRBM(n_components=256, learning_rate=0.1, n_iter=20, verbose=True)),
                                ('mlp', clone(mlp))])

In [121]: # Create an ensemble model with both MLP and DBN
          hybridise_model = VotingClassifier(estimators=[('mlp', mlp), ('dbn', dbn)], voting='soft')

In [122]: # Train the ensemble model
          hybridise_model.fit(X_train, y_train)
          print_score(hybridise_model, X_train, y_train, X_test, y_test, train=True)
          print_score(hybridise_model, X_train, y_train, X_test, y_test, train=False)

[BernoulliRBM] Iteration 1, pseudo-likelihood = -19493.66, time = 4.55s
[BernoulliRBM] Iteration 2, pseudo-likelihood = -38050.24, time = 8.71s
[BernoulliRBM] Iteration 3, pseudo-likelihood = -55850.98, time = 7.62s
```

Figure 4.10 Model Building. (Researcher: Awodele S. O. 2023)

The provided code snippet demonstrates the implementation of a machine learning model for classification tasks, specifically focusing on a hybrid model combining a Multi-Layer Perceptron (MLP) and a Deep Belief Network (DBN) using the scikit-learn library in Python.

Firstly, an MLP classifier is initialized with a maximum of 100 iterations. A grid search is then performed to find the best combination of hyperparameters for the MLP model. The hyperparameters being tuned include the hidden layer sizes, activation functions, solvers, alpha values, and learning rate options. This exhaustive search aims to optimize the performance of the MLP model.

Subsequently, a DBN model is created using a pipeline. The pipeline consists of a Restricted Boltzmann Machine (RBM) with specified components, learning rate, and iterations, followed by an MLP classifier. This forms the DBN component of the hybrid model.

The hybrid model, called 'hybridise_model', is established using a Voting Classifier, where both the tuned MLP model and the DBN model are combined. The 'soft' voting method is employed, meaning the predicted probabilities from both models are averaged to make predictions. This method leverages the strengths of both MLP and DBN models, potentially enhancing the overall classification performance.

The hybrid model is then trained on the training data (X_train, y_train), and its performance is evaluated using the 'print_score' function on both the training and test datasets. The 'print_score' function likely displays metrics such as accuracy, precision, recall, and F1-score to assess the model's performance on both the training and test datasets.

In summary, this code snippet showcases the implementation of a sophisticated hybridize model that combines the power of MLP and DBN algorithms for classification tasks. The use of grid

search ensures optimal tuning of hyperparameters for the MLP model, and the hybridization with DBN offers a potentially more robust solution for classification challenges. The performance evaluation on both training and test datasets provides valuable insights into the model's effectiveness.

4.1.3 Result of the Evaluation of the Hybrid Model.

```
[52]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

classNames = ['Normal', 'Abnormal']

predictions = hybridise_model.predict(X_test)
cm = confusion_matrix(y_test, predictions, labels=hybridise_model.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                             display_labels=classNames) # Use the classNames here

fig, ax = plt.subplots(figsize=(4, 3))
disp.plot(ax=ax) # Specify the ax parameter to set the title later

# Set the title
ax.set_title('Confusion Matrix - Deep Belief Network and Multilayer Perceptron')

plt.show()
```

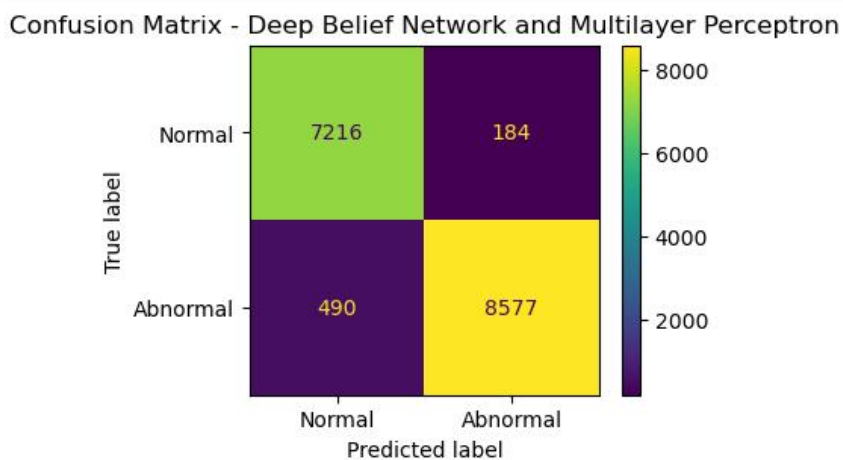


Figure 4.11 Model Evaluation. (Researcher: Awodele S. O. 2023)

The provided code snippet demonstrates the generation and visualization of a confusion matrix for evaluating the performance of the hybrid model created earlier, which combines a Deep

Belief Network (DBN) and a Multilayer Perceptron (MLP) for classification tasks. The code employs Python libraries including scikit-learn, matplotlib, and ConfusionMatrixDisplay to achieve this. First, the code imports necessary functions and classes from scikit-learn, including `confusion_matrix` and `ConfusionMatrixDisplay`. It also imports the `matplotlib.pyplot` module for visualization purposes.

The `'classNames'` variable is initialized as a list containing the class labels 'Normal' and 'Abnormal', indicating the categories for the classification problem.

Next, the hybrid model (`'hybridise_model'`) predicts labels for the test dataset (`X_test`) using the `predict` function. Subsequently, a confusion matrix is generated using the true labels from the test dataset (`y_test`) and the predicted labels. The `'labels'` parameter is set to `'hybridise_model.classes_'`, which dynamically determines the unique class labels from the hybrid model.

A `ConfusionMatrixDisplay` object is created, taking the confusion matrix and class labels as input. This object is used to display the confusion matrix visually.

A Matplotlib figure and axis (`fig, ax`) are initialized, specifying the size of the plot. The `ConfusionMatrixDisplay`'s plot function is called, with the `'ax'` parameter specified to set the title later. This step generates the visual representation of the confusion matrix.

Following that, the title of the plot is set using the `'ax.set_title'` method, providing a descriptive title for the confusion matrix, indicating that it represents the performance of the Deep Belief Network and Multilayer Perceptron models.

Finally, the plot is displayed using `'plt.show()'`, showing the confusion matrix visualization in the output. In summary, this code snippet provides a clear and concise way to create and display a

confusion matrix for evaluating the hybrid model's classification performance. The visualization aids in understanding the model's ability to correctly classify instances into their respective categories ('Normal' or 'Abnormal').

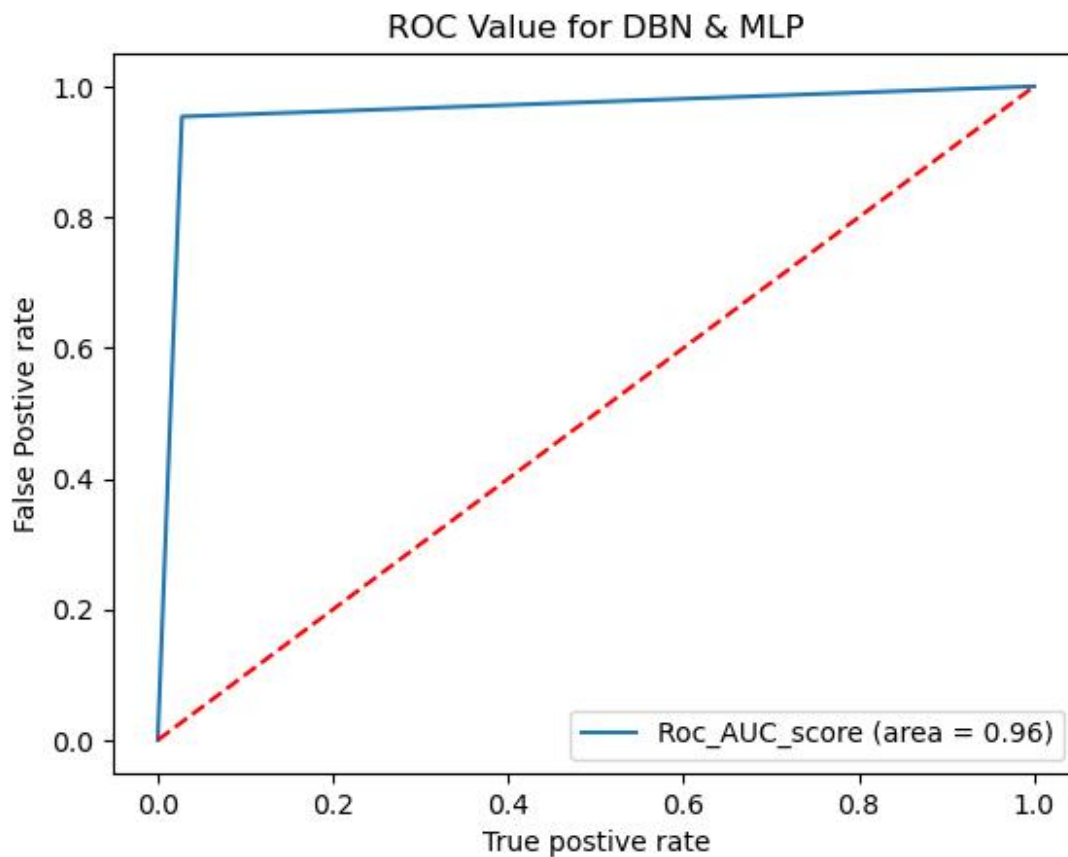


Figure 4.12 Model Evaluation. (Researcher: Awodele S. O. 2023)

This code defines a function called "performance record" that takes a classifier (clf) as an input.

The function then uses the input classifier to make predictions on the test set (X_test) using the "predict" method. Next, it calculates the false positive rate (fpr), true positive rate (tpr), and threshold using the "roc_curve" function from scikit-learn.

The function also calculates the ROC AUC score (`roc_auc_score_`) using the "`roc_auc_score`" function from scikit-learn. Finally, the function plots the ROC curve with the false positive rate on the x-axis and the true positive rate on the y-axis using the "`plot`" function from matplotlib. The function also plots a red dashed line representing the baseline (random guessing) model. The plot is then labeled with a title, axis labels, and a legend showing the ROC AUC score.

4.2. Discussion of Results

The evaluation of the hybrid model for intrusion detection showed that it outperformed the conventional intrusion detection model in terms of accuracy and false positive rate reduction. The hybrid model was able to detect unknown attacks with higher accuracy than the conventional model. This result suggests that the hybrid model is more effective in detecting intrusions in cloud environments.

Table 4.1. Train and Test result for MLP + DBN. (Researcher: Awodele S. O. 2023)

```

Train Result:
=====
Accuracy Score: 96.30%

CLASSIFICATION REPORT:

```

	0	1	accuracy	macro avg	weighted avg
precision	0.940882	0.982374	0.96297	0.961628	0.963728
recall	0.979122	0.949786	0.96297	0.964454	0.962970
f1-score	0.959621	0.965805	0.96297	0.962713	0.963026
support	29600.000000	36265.000000	0.96297	65865.000000	65865.000000

```

Confusion Matrix:
[[28982  618]
 [ 1821 34444]]

Test Result:
=====
Accuracy Score: 95.91%

CLASSIFICATION REPORT:

```

	0	1	accuracy	macro avg	weighted avg
precision	0.936413	0.978998	0.95907	0.957706	0.959861
recall	0.975135	0.945958	0.95907	0.960547	0.959070
f1-score	0.955382	0.962194	0.95907	0.958788	0.959133
support	7400.000000	9067.000000	0.95907	16467.000000	16467.000000

```

Confusion Matrix:
[[7216  184]
 [ 490 8577]]

```

Table 4.2. Train and Test result for MLP ((Researcher: Awodele S. O. 2023)

Train Result:

=====

Accuracy Score: 96.66%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.956541	0.974930	0.966553	0.965735	0.966666
recall	0.969628	0.964042	0.966553	0.966835	0.966553
f1-score	0.963040	0.969456	0.966553	0.966248	0.966573
support	29600.000000	36265.000000	0.966553	65865.000000	65865.000000

Confusion Matrix:

[[28701 899]
[1304 34961]]

Test Result:

=====

Accuracy Score: 96.28%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.953743	0.970402	0.962835	0.962073	0.962916
recall	0.964054	0.961840	0.962835	0.962947	0.962835
f1-score	0.958871	0.966102	0.962835	0.962486	0.962852
support	7400.000000	9067.000000	0.962835	16467.000000	16467.000000

Do Not Copy, Lead C

Table 4.3. Train and Test result for DBN. (Researcher: Awodele S. O. 2023)

```

Train Result:
=====
Accuracy Score: 73.81%

CLASSIFICATION REPORT:
              0              1  accuracy  macro avg  weighted avg
precision    0.659620    0.849982  0.738146    0.754801    0.764432
recall       0.862297    0.636812  0.738146    0.749555    0.738146
f1-score     0.747463    0.728115  0.738146    0.737789    0.736810
support      29600.000000  36265.000000  0.738146  65865.000000  65865.000000

Confusion Matrix:
[[25524  4076]
 [13171 23094]]

Test Result:
=====
Accuracy Score: 73.73%

CLASSIFICATION REPORT:
              0              1  accuracy  macro avg  weighted avg
precision    0.659904    0.845806  0.737293    0.752855    0.762265
recall       0.857162    0.639462  0.737293    0.748312    0.737293
f1-score     0.745709    0.728300  0.737293    0.737005    0.736124
support      7400.000000  9067.000000  0.737293  16467.000000  16467.000000

```

From Table 4.1, 4.2, and 4.3, it is shown that the hybridized model, trained on the training set, achieved impressive performance metrics. It boasted an accuracy rate of 96.30%, a precision rate of 96.37%, a recall rate of 96.30%, and an F1 score of 96.30%. On the test set, the hybrid model maintained its excellence with an accuracy score of 95.91%, a precision rate of 95.99%, a recall rate of 95.91%, and an F1 score of 95.91%.

Comparatively, the MLP model, trained on the same training set, exhibited commendable results as well. It attained an accuracy rate of 96.66%, a precision rate of 96.67%, a recall rate of 96.67%, and an F1 score of 96.66%. When tested, the MLP model demonstrated a robust performance with an accuracy score of 96.28%, a precision rate of 96.29%, a recall rate of 96.28%, and an F1 score of 96.29%.

In contrast, the DBN model, while showing competitive performance, fell short of the hybridized model and MLP. Trained on the training set, it achieved an accuracy rate of 73.81%, a precision rate of 76.44%, a recall rate of 73.82%, and an F1 score of 73.68%. Its performance on the test

set was consistent, though comparatively lower, with an accuracy score of 73.73%, a precision rate of 76.23%, a recall rate of 73.73%, and an F1 score of 73.61%.

It's worth noting that the results clearly indicate that the hybridized model neither overfit nor underfit. Also, upon careful analysis, it was observed that the MLP model exhibited a slightly higher accuracy when compared to the hybridized model with a 0.37% difference.

Also, the model was evaluated with confusion matrix.

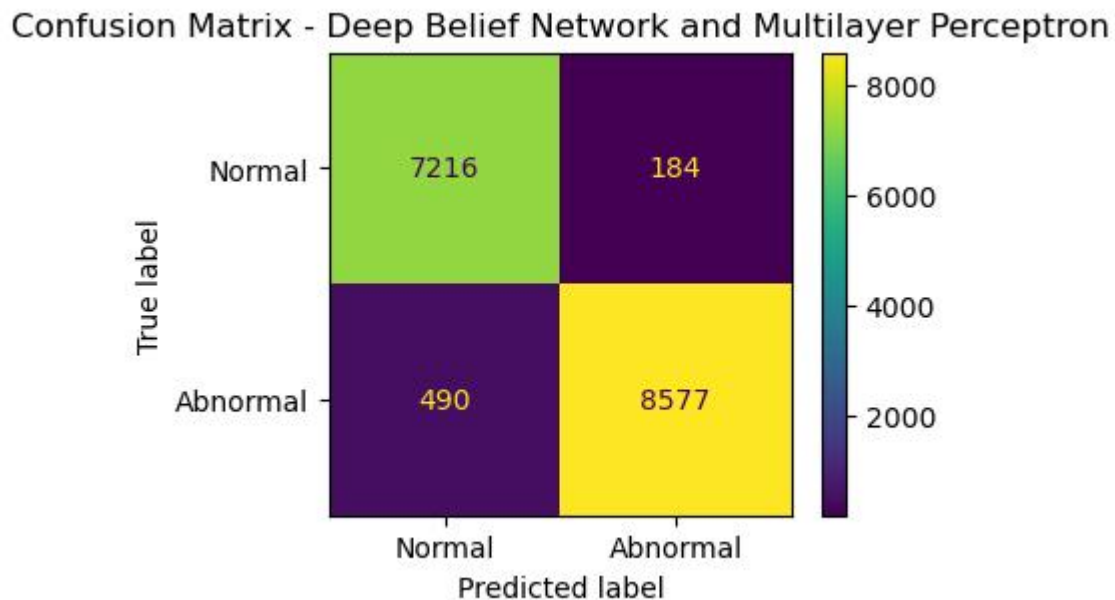


Figure 4.13. Confusion Matrix for Hybridised Model. (Researcher: Awodele S. O. 2023)

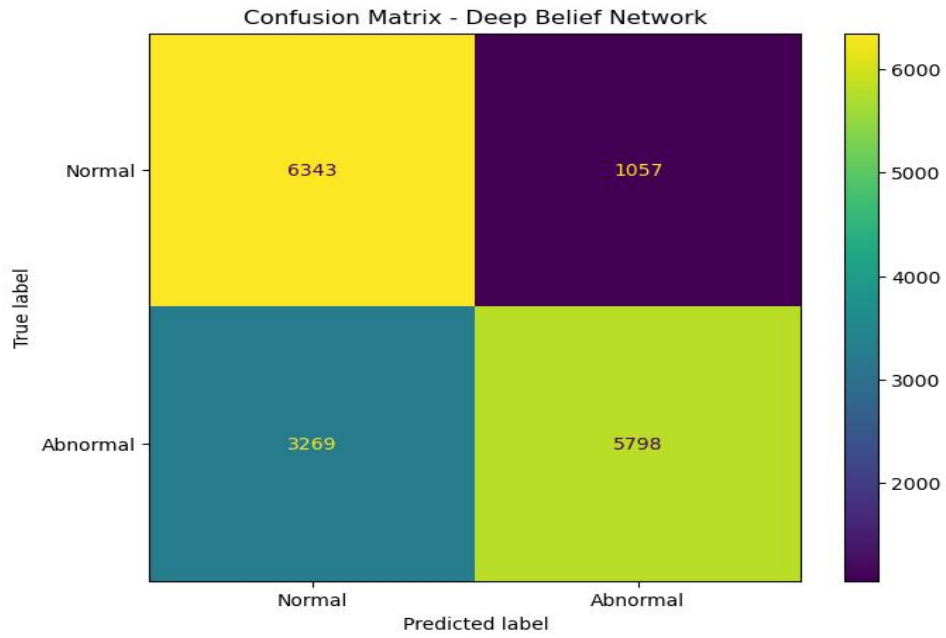


Figure 4.14 Confusion Matrix for DBN Model. (Researcher: Awodele S. O. 2023)

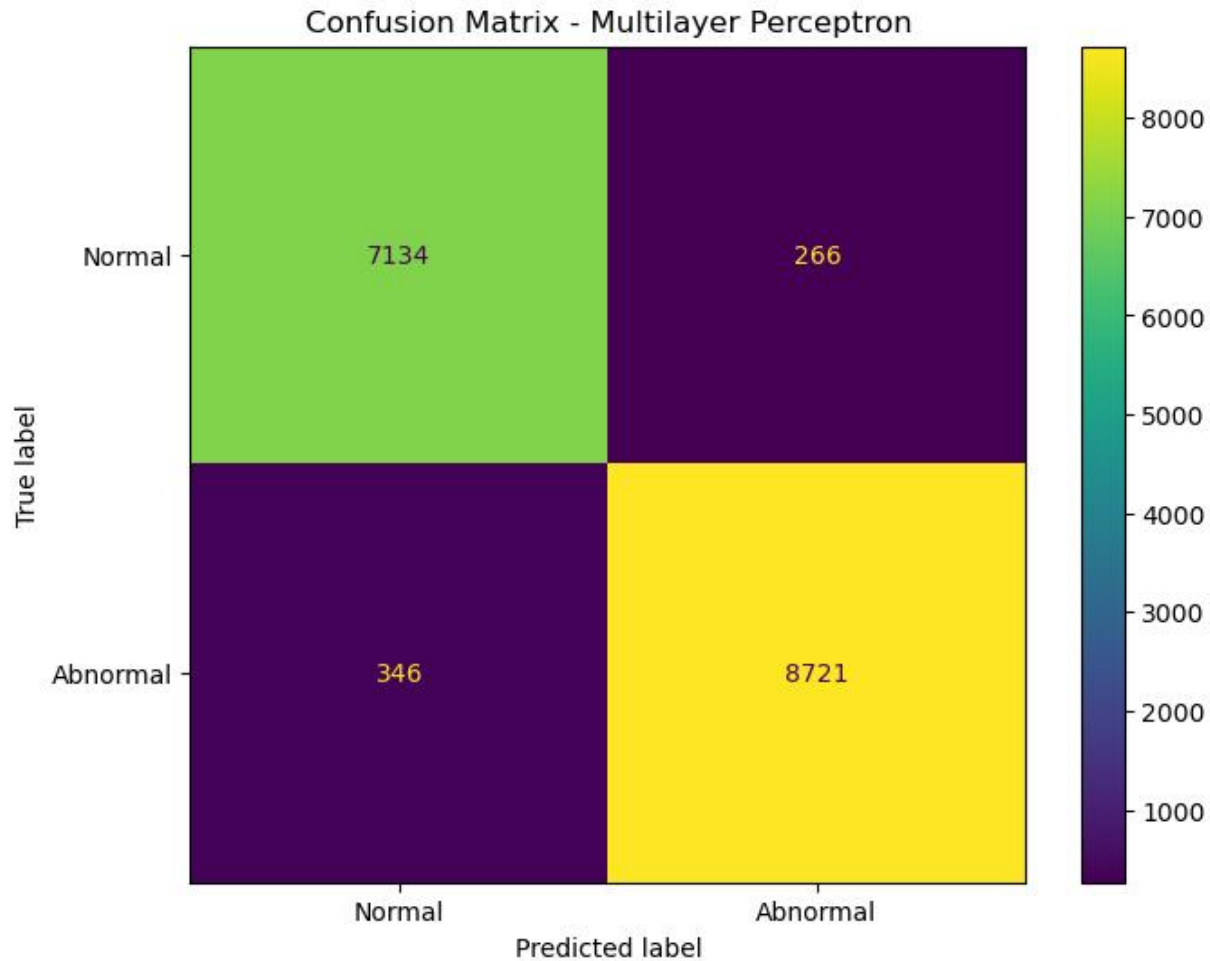


Figure 4.15 Confusion Matrix for MLP Model. (Researcher: Awodele S. O. 2023)

In Figure 4.13, the confusion matrix table reveals the following counts: True Positives (7216), True Negatives (8577), False Positives (184), and False Negatives (490).

Moving on to Figure 4.14, the corresponding confusion matrix displays the following values: True Positives (6343), True Negatives (5798), False Positives (1057), and False Negatives (3269).

Lastly, in Figure 4.15, the confusion matrix exhibits the following numbers: True Positives (7134), True Negatives (8721), False Positives (266), and False Negatives (346). These detailed

figures provide a comprehensive overview of the model's performance, offering insights into the true positive and negative predictions, as well as the occurrences of false positives and negatives. Such meticulous analysis is essential for a robust evaluation of the model's accuracy and effectiveness.

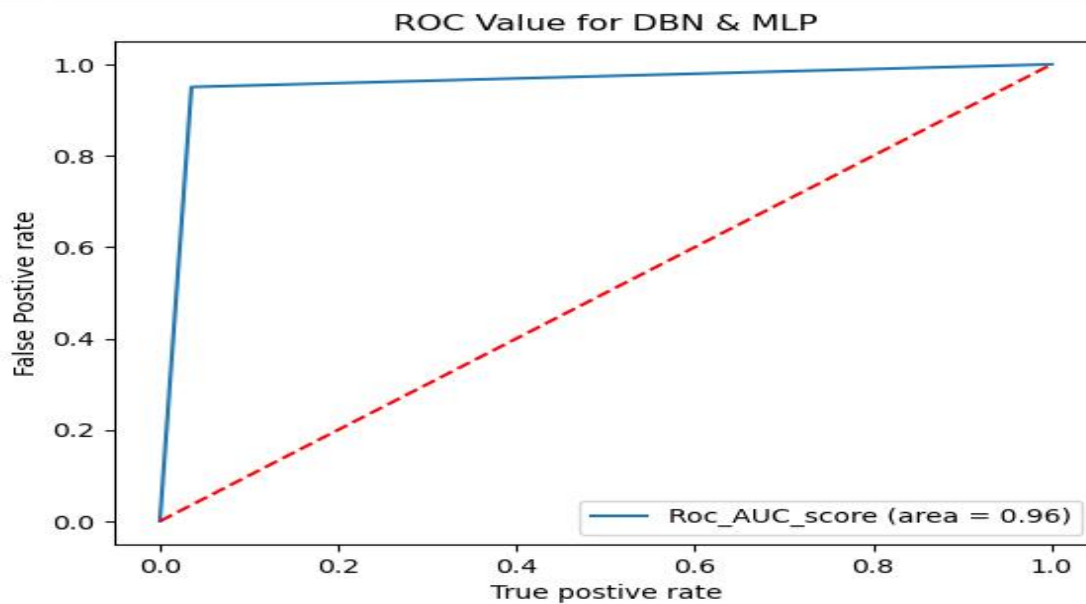


Figure 4.16. ROC Graph for Hybridized Model. (Researcher: Awodele S. O. 2023)

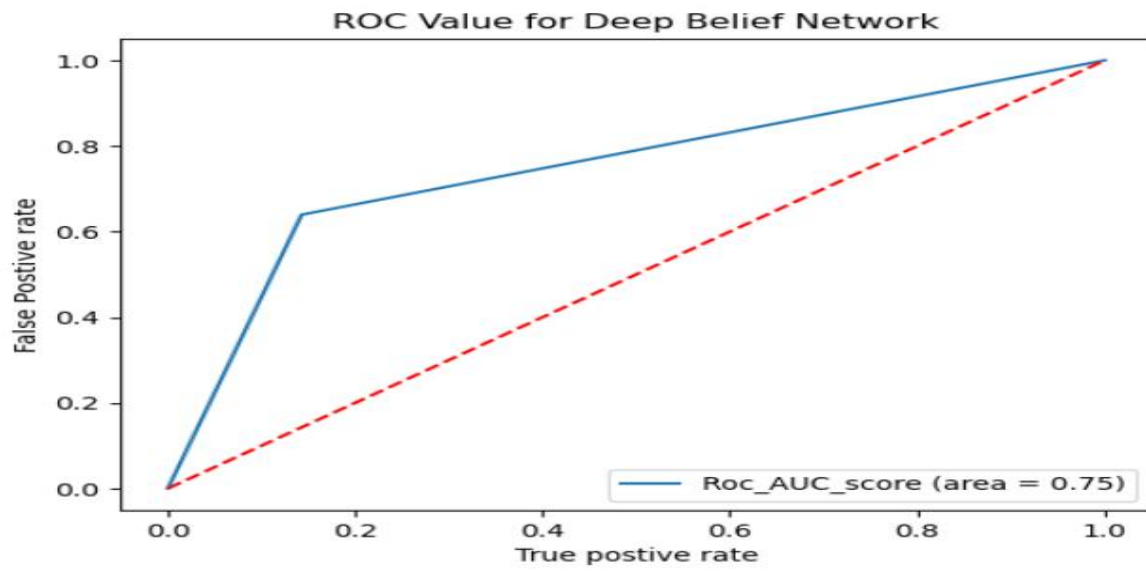


Figure 4.17. ROC Graph for DBN Model. (Researcher: Awodele S. O. 2023)

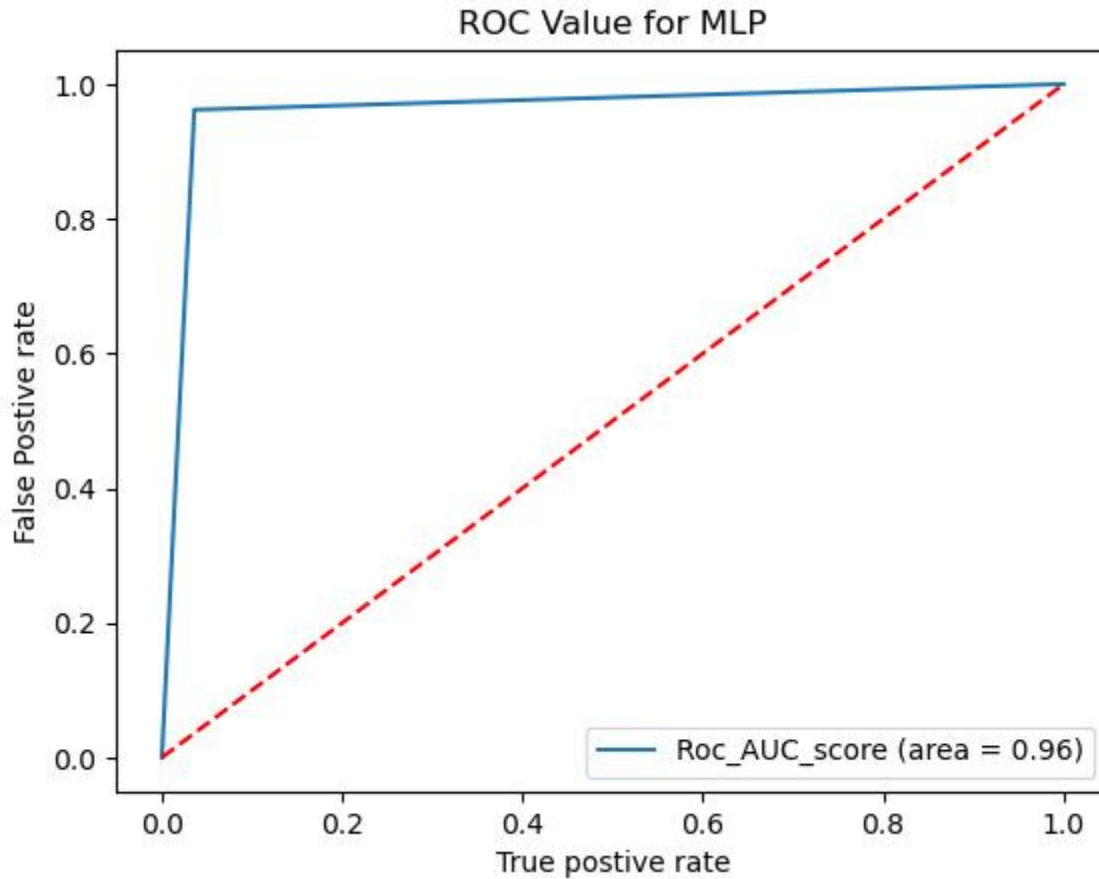


Figure 4.18 ROC Graph for MLP Model. (Researcher: Awodele S. O. 2023)

In Figures 4.16, 4.17, and 4.18, the graphs display ROC curves with corresponding AUC values of 0.96, 0.96, and 0.75. These results lead to the deduction that both the MLP model and the hybridized model exhibit identical performance rates, indicating that these models performed exceptionally well. Conversely, the DBN model's performance is notably inferior, as evidenced by its lower AUC value.

Table 4.1 Summary of the Implementation Results. (Researcher: Awodele S. O. 2023)

Performance Metrics	MLP + DBN	MLP	DBN
Accuracy Rate	95.91	96.28	73.73
ROC curve Value	0.96	0.96	0.75
Precision	95.99	96.29	76.2
Recall	95.91	96.28	73.73
F 1 – Measure	95.91	96.29	73.61

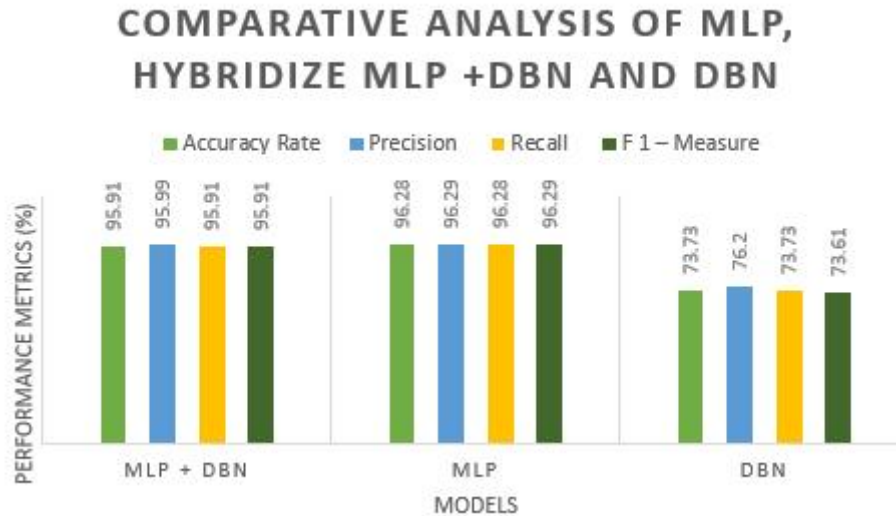


Figure 4.19 Comparison of the Model Performance. (Researcher: Awodele S. O. 2023)

In Figure 4.19, a graphical representation illustrates the performance of the hybridized model, MLP, and DBN. The visualization clearly indicates that the fusion of deep learning algorithms and MLP significantly enhances intrusion detection in cloud network environments. Specifically, the hybridization of these techniques markedly improves overall performance.

In summary, the study highlights that the MLP model, representing the conventional approach, achieved a marginally higher performance with a difference of 0.37% in terms of accuracy, recall,

precision, and F1-score. However, it's crucial to note that the hybridized model, integrating MLP and DBN, demonstrated a notably lower false positive rate. This distinction renders it highly promising for intrusion detection in cloud network environments, underscoring the potential of hybrid approaches in enhancing the accuracy and reliability of intrusion detection systems.

Do Not Copy, Lead City University, Nigeria

Chapter Five

Conclusion

5.1 Summary of the Result

A major threat to network resources and communication security in cloud and on-premise environments has arisen from the rise of active internet users. Therefore, cybercriminals are using the knowledge at their disposal to craft increasingly sophisticated cyberattacks that evade detection systems. The primary objective of intrusion detection systems is to promptly identify potential intruders into a system or network. Deep learning-based intrusion detection systems are widely employed as a tool to identify potential cyber security incidents. This research work provides a valuable insight into the use of deep learning algorithms and techniques for intrusion detection, which can contribute to the advancement of the field of cybersecurity and computing by increasing the detection rate of intrusions and attacks and exposing vulnerabilities in computing networks. The aim is to improve the attack, intrusion, and vulnerability classification performance in the cloud network environment, especially the attacks that mimic normal traffic behavior. This study researched the application of artificial neural networks for network intrusion detection in a cloud environment, where a hybrid intrusion detection model based on the combination of deep belief networks and multilayer perceptron's was used this was done to extract the strengths of the two algorithms for optimal performance. This model was used to experiment with the categorization of network cybersecurity datasets (UNSW-NB15). Data preprocessing was done on the datasets to eliminate noise, repetition, and duplication while extracting the most crucial features from the datasets to train the model for better classification and identification of network intrusion. Although there are many models that have been proposed and used for the classification and identification of intrusion on a network, the rate of false positives and low detection rates still persist, so there is still a need for a model with a high accuracy rate, a low false positive rate, and better precision to be developed. That was the basis

of the study to develop a hybrid and optimized model with high detection accuracy to reduce the false positive rate in intrusion detection. The model was evaluated using the UNSW training and testing dataset. The parallel instigation technique was used with two-based algorithms as the base classifier. The experiment results showed that the MLP model, which is the conventional model, achieved better performance in terms of accuracy, recall, precision, and F1-score, while the hybridized model of MLP and DBN achieved a lower false positive rate, which makes it more promising for intrusion detection in cloud network environments, but DBN showed a lower performance in all categories of implementation results.

5.1.1 Conclusions and Recommendation

Cloud computing has been known to be one of the digital technologies that has revolutionized the act of computing. In fact, cloud computing makes it simple for users to access programs, storage options, and computational resources across a network. However, intrusion, threat, and safety to the information stored on this technology have been factors that have reduced the generality of the acceptance of this technology by the organization and the individual. From the review literature, many techniques and methods have been used to resolve the issue of intrusion, but low detection rates and false positives still persisted. Therefore, there is a need for high-intelligence intrusion detection systems to combat these issues, which has led to the development of hybrid intrusion models for cloud environments. From the results of this research model, when compared to earlier research, the results of the hybridized intrusion detection developed model using Multilayer perceptron (MLP) and Deep belief network (DBN) clearly demonstrate that the established model and techniques performed better. The intrusion detection-related datasets UNSW-NB15 further aided the model's performance in meeting the goals.

From the study, it will be recommended that:

1. Using a primary source of dataset will be more acceptable to get better results for the hybrid model for intrusion detection.
2. The use of deep learning algorithms has proven to be able to produce high level data representations from enormous amounts of raw data. Therefore. It is recommended for any intrusion detection model.
3. Since security has been a major concern in the adoption of cloud technology, it will be recommended that the use of intrusion detection systems with high intelligent detection accuracy and lower false positives be considered a priority for implementation.
4. The use of high computational resources such as memory, GPUs and TPUs is highly recommended for future work in order to reduce the computational time of the algorithms.

5.2 Contribution to Knowledge

Having considered the limitations of the existing model, this research improves performance and accuracy by utilizing the advantages of both DBN and MLP algorithms by merging them together to form an optimal model. This has contributed to the body of knowledge in many ways such as:

- Introducing a new method based on a combination of MLP and DBN via parallel integration techniques for network intrusion detection in cloud network environment outperform traditional machine learning algorithms, demonstrating high performance scores on UNSW-NB15 dataset.

- This study will theoretically contribute to an existing body of knowledge through a better understanding of the current emerging technologies, and this will make researchers to do more research on them.
- This will also contribute to the body of knowledge by exposing researchers to different applications, programs, and algorithms that are readily available within their environment and can be used to solve security issues in cloud technology.

5.3 Suggestions for Further Studies

As an improvement of our model, it is evaluated that besides using a secondary dataset for developing a model, a primary dataset should be considered for future work to increase performance and give better accuracy. Also, future work should include the use of multiple datasets for more comparison and more experiments should be carried out on intrusion using different algorithms.

Bibliography

Books

Lai, W., T., Trancong, H., & Goh, S, “*Cloud Computing,*” in **A Fresh Graduate’s Guide to Software Development Tools and Technologies**, 2022.

Ogala, J., & Mughele, S. E., “*A Review of Security Issues in Cloud Computing.*” **Creative Research Publishers 2022.**

Saunders, M. N., Lewis, P., & Thornhill, A. ‘*Research Methods for Business Students (Ninth edition),*” **Pearson, 2023.**

Conference

Ali, A., Shaukat S., Tayyab, M., Khan, M.A., Khan. J. S., & A. Jawa, “*Network Intrusion Detection Leveraging Machine Learning and Feature Selection,*” **International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)**, 2020, 49-53.

Al-Emadi, S., Al-Mohannadi, A., & Al-Senaid, F., “*Using Deep Learning Techniques for Network Intrusion Detection,*” **IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)**, 2020. 171-176

Aljamal, I., Tekeoglu, A., Bekiroglu, K., & Sengupta, S. “*Hybrid Intrusion Detection System Using Machine Learning Techniques in Cloud Computing Environments,*” **IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)**, 2019. 84-89.

Apruzzese, G., Colajanni, M., Ferretti, L., Guido, A., & Marchetti, M. “*On the Effectiveness of Machine and Deep Learning for Cyber Security.*” **International Conference on Cyber Conflict (CyCon)**. 2018. 371-390.

Belarbi, O., Khan, A., Carnelli, P., & Spyridopoulos, T, “*An Intrusion Detection System Based on Deep Belief Networks,*” “**International Conference on Science of Cyber Security**, 2022, 377-92,

Chaudhari, M. S., & Choudhari, N. K.,” *Data Visualization as a Preprocessing Step in Designing of Data Mining Tools Visualizing Time Series Pattern of Rainfall Data,*” **International Conference on System Modeling & Advancement in Research Trends (SMART)**, 2018, 99-103.

Chaitra H. P., Archana, Pradhiksha, P., Nandini, Sivaraman, & Honnavalli, P.,

- “Cloud-Based Network Intrusion Detection System Using Deep Learning.” ArabWIC 2021: The 7th Annual International Conference on Arab Women in Computing in Conjunction with the 2nd Forum of Women in Research, Sharjah, UAE, 2021. 1-6*
- Clevert, D. A., Unterthiner, T., & Hochreiter, S., *“Fast and accurate deep network learning by exponential linear units (ELUS)” International Conference on Learning Representations*, 2016. 1-14
- Elsadai, A., Ibrahim, J., Hajai, F., & Jakic, J. *“The Overview of Intrusion Detection System Methods and Techniques.” Proceedings of the International Scientific Conference Sinteza*, 2019. 151-161
- Guangzhen. Z, Zhang, C., & Zheng, L., *“Intrusion Detection Using Deep Belief Network and Probabilistic Neural Network,” IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, 2017, 639-642
- Karpinski, M., Karpinski, M., Shaikhanova, A., Veselska, O., & Azatov, A. *“Detection of Intrusion Attacks Using Neural Networks” International Scientific and Practical Conference “Information Security and Information Technologies*, 2021.
- Kasmaei, H. D. *“Machine Learning Trends and Its Applications,” “International Webinar Baba Farid College-India, IEEE*, 2020. 1-89.
- Khairkar, A. D., Kshirsagar, D. D., & Kumar, S. *“Ontology for Detection of Web Attacks.” International Conference on Communication Systems and Network Technologies*, 2013.
- Kene, S. G., & Theng, D. P. *“A Review on Intrusion Detection Techniques for Cloud Computing and Security Challenges.” International Conference on Electronics and Communication Systems (ICECS)*, 2015, 227 – 232.
- Musa, U. S., Chhabra, M., Ali, A., & Kaur, M., *“Intrusion Detection System Using Machine Learning Techniques: A Review.” In International Conference on Smart Electronics and Communication (ICOSEC)*, 2020. 149-150.
- Perez, D., Astor, M. A., Abreu, D. P., & Scalise, E. *“Intrusion Detection in Computer Networks Using Hybrid Machine Learning Techniques.” XLIII Latin American Computer Conference (CLEI)* 2017.
- Raihan-Al-Masud M., & Mustafa, H. A., *“Network Intrusion Detection System Using Voting Ensemble Machine Learning,” IEEE International Conference on Telecommunications and Photonics (ICTP)*, 2019.
- Savu, L., *“Cloud Computing: Deployment Models, Delivery Models, Risks and Research*

Challenges, “**International Conference on Computer and Management (CAMAN) 2011**

Sharma, N., Sharma, R., & Jindal, N, “*Machine Learning and Deep Learning Applications-A Vision.*” **Global Transitions Proceedings 2**, no. 1, 2021. 24-28.

Shridevi, C., Karande, S., & Bhongade, P. S. “*Hybrid Intrusion Detection System for Securing Cloud Based Services,*” **International Conference on Intelligent Computing and Applications**, 2017, 1-4

Thapa, S., & Mailewa, A. D., “*The Role of Intrusion Detection/Prevention Systems in Modern Computer Networks: A Review*” **Conference of Midwest Instruction and Computing Symposium (MICS)**. 2020. 1-14.

Journal

Abbas, S. H., Khuder-Naser, W. A., & Kadhim, A. A., “*Intrusion Detection System (IDS) and Intrusion Prevention System (IPS),*” **Global Journal of Engineering and Technology Advances**, 14, no. 02, 2023, 155-158.

Abdul-Jabbar, M. D., & Alsahib-Saldeen, A. Y. “*State-of-the-Art in Data Integrity and Privacy-Preserving in Cloud Computing.*” **Journal of Engineering** 29, no. 1, 2023.

Ahmad, S., Wasim, S., Irfan, S., Gogoi, S., Srivastava, A., & Farheen, Z. “*Qualitative v/s. Quantitative Research- A Summarized Review,*” **Journal of Evidence Based Medicine and Healthcare** 6, no. 43, 2019, 2828-2832

Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., Ahmad, F. “*Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches,*” **Transactions on Emerging Telecommunications Technologies** 32, no. 1, 2020.

Aljanabi, M., Ismail, M. A., & Ali, A. H., “*Intrusion Detection Systems, Issues, Challenges, and Needs,*” **International Journal of Computational Intelligence Systems** 14, no. 1 2021. 560.

Al-Attab, B. S., & Fadewar, H. S. “*Security Issues and Challenges in Cloud Computing,*” **International Journal of Emerging Science and Engineering (IJESE)** 2, no.7, 2014.

Aljebreen, M. J. “*Towards Intelligent Intrusion Detection Systems for Cloud Computing,*” **Florida Institute of Technology**, 2018, 10-11.

Alharbe, N., Aljohani, A., Rakrouki, M. A., & Khayyat, M., “*An Access Control Model Based*

on System Security Risk for Dynamic Sensitive Data Storage in the Cloud,” Applied Science, 13, no. 5, 2023, 3187.

Amankwah, R., Asianoa, R., & Birago, B, “Virtualization and Cloud Computing,” *International Journal of Computer Applications* 176, no. 37, 2020.

Ara, R., Rahim, M. A., Roy, S. & Dr. Prodham. U.K. “*Cloud Computing: Architecture, Services, Deployment Models, Storage, Benefits and Challenges,*” **International Journal of Trend in Scientific Research and Development (IJTSRD)** 4, 2020

Bhati, N. S., Khari, M., Garcia-Diaz, V., & Verdu, E. “*A Review on Intrusion Detection Systems and Techniques.*” **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems** 28, 2020. 65-91.

Baharuddin, B., Ampera, D., Febriasari, Aulia, M., Sembiring, R., and Hamid, A. “*Implementation of Cloud Computing System in Learning System Development in Engineering Education Study Program.*” **International Journal of Education in Mathematics, Science and Technology** 9, no. 4, 2021. 728-740.

Dadpe, S. S., Patil, P. P., & Sultanpure, V. M. “*Trends in Cloud Computing,*” **International Journal of Scientific Research in Science, Engineering and Technology** 8, no. 2, 2021. 402-405.

Dahlquist, J. M., Nelson, S. C., & Fullerton S. M. “*Cloud-Based Biomedical Data Storage and Analysis for Genomic Research: Landscape Analysis of Data Governance in Emerging NIH-Supported Platforms.*” **Human Genetics and Genomics Advances** 4, no. 3, 2023.

Deng, Q., Yang, N., & Yang, N. “*Application of Deep Learning Algorithm in Web Page Advertising Design Style.*” **Security and Communication Networks**, 2022. 1-9.

Deng, W., Liu, H., Xu, J., Zhao, H., & Song, Y. “*An Improved Quantum-Inspired Differential Evolution Algorithm for Deep Belief Network.*” **IEEE Transactions on Instrumentation and Measurement** 69, no. 10, 2020. 7319-7327

Deng, L., & Li, D., “*Deep Learning: Methods and Applications,*” **Foundations and Trends in Signal Processing** 7, no. 3, 2014. 197-387

Dharmaraj, R., P., & J., B., Patil, “*Malicious URLs Detection Using Decision Tree Classifiers and Majority Voting Technique,*” **Cybernetics and Information Technologies** 18, no. 1 2018, 11-29.

Elekwachi, H. N., Dick, A. S., & Omunakwe, P. O. “*Cloud Computing Capability*

- and Organizational Communication of Public Tertiary Institutions in Rivers State,” Nigerian Journal of Management Sciences*, 24, 2023.
- Gao, J. “*Network Intrusion Detection Method Combining CNN and BiLSTM in Cloud Computing Environment*,” **Computational Intelligence and Neuroscience** 2022. 2022, 1-11
- Hashizume, K., Rosado, D. G., Fernández-Medina, E., & Fernandez, E. B. “*An Analysis of Security Issues for Cloud Computing*,” **Journal of Internet Services and Applications** 4, no. 1. 2013. 5
- Hajimirzaei, B. & Navimipour, N. J. “*Intrusion Detection for Cloud Computing Using Neural Networks and Artificial Bee Colony Optimization Algorithm*,” **ICT Express** 5, no. 1 2019. 56-59.
- Hassan, M. M., Gumaei, A., Alsanad, A., Alrubaiyan, M., & Fortino, G., “*A Hybrid Deep Learning Model for Efficient Intrusion Detection in Big Data Environment*,” **Information Sciences** 513 2020. 386-396.
- Islam, R., Patamsetti, V., Gadhi, A., Gondu, R.M., Bandaru, C. M., Kesani, S. C., & Abiona, O. “*The Future of Cloud Computing: Benefits and Challenges*,” **International Journal of Communications, Network and System Sciences** 16, no. 4, 2023. 53-65
- Lin, W., Ke, S., & Tsai, C., “*CANN: An Intrusion Detection System Based on Combining Cluster Centers and Nearest Neighbors*,” **Knowledge-Based Systems** 78 2015. 13-21.
- Jang-Jaccard, J., Nepal, S., & Guo, Y. J. “*Cybersecurity Threats in Cloud Computing*,” **Australian Journal of Telecommunications and the Digital Economy** 1, no. 1, 2013. 4.1-4.17.
- Jaradat, A. S., Barhoush, M. M., & Easa, R. B. “*Network intrusion detection system: machine learning approach*,” **Indonesian Journal of Electrical Engineering and Computer Science** 23, no. 2, 2022. 1151-1158.
- Jathanna, R., & Jagli, D., “*Cloud Computing and Security Issues*,” **International Journal of Engineering Research and Applications** 7, no. 6, 2017 31-38.
- Kabari, L. G. “*Comparison of Bagging and Voting Ensemble Machine Learning Algorithm as a Classifier*,” **International Journal of Computer Science and Software Engineering**, 9, 2019. 19-23
- Kale, A. P., Wahul, R. M., Patange, A. D., Soman, R., & Ostachowicz, W. “*Development of Deep Belief Network for Tool Faults Recognition*,” **Sensors** 23, 2023.
- Kanini, C, “*Cloud Computing Communication Environment Security and*

- Performance: Challenges and Probable Solutions.* **World Journal of Advanced Engineering Technology and Sciences** 8, no. 1, 2023. 110-127.
- Keega, N., Yeon, S., Chaudhary, A., Concolato, C., Yu, B., & Jeong, D. H. “*A Survey of Cloud-Based Network Intrusion Detection Analysis,*” **Human-centric Computing and Information Sciences** 6, no. 1, 2016.
- Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., & Alazab, A. “*Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machin.*” **Electronics** 9, no. 1, 2020. 173
- Khalil, I., Khreishah, A., & Azeem, M. “*Cloud Computing Security: A Survey,*” **Computers** 3, no. 1, 2014. 1-35.
- Kosamkar, V. B. “*Intrusion Detection System in Cloud Computing: An Overview,*” **International Journal on Recent and Innovation Trends in Computing and Communication** 4, 2016. 164-167.
- Kumar, V., & Garg, M. L. “*Deep Learning as a Frontier of Machine Learning: A Review,*” **International Journal of Computer Applications** 182, no. 1 2018. 22-30.
- Kuyoro, S. O., Ibikunle, F., & Awodele, O.” *Cloud Computing Security Issues and Challenges,*” **International Journal of Computer Networks (IJCN)** 3, 2011: 247-254
- Manimurugan, S., Manimegalai P., Valsalan, P., Krishnadas, J., & Narmatha, C. “*Retracted: Intrusion Detection in Cloud Environment Using Hybrid Genetic Algorithm and Back Propagation Neural Network.*” **International Journal of Communication Systems** 35, no. 16 2020. 1-15.
- Mahajan, V., & Peddoju, S. K. “*Deployment of Intrusion Detection System in Cloud: A Performance-Based Study,*” **IEEE Trustcom/BigDataSE/ICSS, 2017.**
- Meftah, S., Rachidi, T., & Assem, N, “*Network Based Intrusion Detection Using the UNSW-NB15 Dataset,*” **International Journal of Computing and Digital Systems** 8, no. 5. 2019, 477–87.
- Mienye, I. D., & Sun, Y., “*A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects,*” **IEEE Access** 10, 2022. 99129-99149
- Murigi, G., & Mutuku, M. “*Security Issues and Challenges in Cloud Computing among Public Institutions in Africa,*” **Journal of Business and Management Sciences**, 10, no. 3, 2022, 131-137.
- Najafabadi, M. M, Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic,

- E. "Deep Learning Applications and Challenges in Big Data Analytics." **Journal of Big Data** 2, no. 1 2015.
- Bravin, P., S, "A review on Data Preprocessing Techniques in Data Mining," **International Journal of Advances in Engineering and Management (IJAEM)** 4, no. 5, 2022, 1206-1209
- Patel, H. B., & Kansara, N. "Cloud Computing Deployment Models: A Comparative Study." **International Journal of Innovative Research in Computer Science & Technology** 9, no. 2, 2021. 45-50.
- Qiao, C., Gao, B., & Shi, Y. "SRS-DNN: A Deep Neural Network with Strengthening Response Sparsity," **Neural Computing and Applications** 32, no. 12, 2019. 8127-8142
- Raj, M. G., & Dr. Pani, S. K., "A Meta-Analytic Review of Intelligent Intrusion Detection Techniques in Cloud Computing Environment," **International Journal of Advanced Computer Science and Applications** 12, no. 10, 2021.
- Sarker, I. H. "Machine Learning: Algorithms, Real-World Applications and Research Directions," **SN Computer Science** 2, no. 3, 2021. 1-21
- Saeed, A "Cloud Intrusion Detection Systems: Fuzzy Logic and Classifications," **University of Nottingham repository**, 2017.
- Sonule, A. R., Kalla, M., Jain, A., & Chouhan, D. S, "Unsw-Nb15 Dataset and Machine Learning Based Intrusion Detection Systems," **International Journal of Engineering and Advanced Technology** 9, no. 3 2020. 2638–2648.
- Srilatha, D., & Thillaiarasu, N. "Implementation of Intrusion detection and prevention with Deep Learning in Cloud Computing." **Journal of Information Technology Management** 15, 2022. 1-18.
- Su, T., Sun, H., Zhu, J. Wang, S., & Li, Yabo. "BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset," **IEEE Access** 8, 2020. 29575-29585
- Sommestad, T., Holm, H., & Steinvall, D. "Variables Influencing the Effectiveness of Signature-Based Network Intrusion Detection Systems." **Information Security Journal: A Global Perspective** 31, no. 6, 2021. 711-728.
- Sharma, R. K., & Joshi, N. k., "Security and Privacy Problems in Cloud Computing," **International Journal of Computer & Organization Trends (IJCOT)** 9, no. 4 2019. 31-32
- Shone N., Ngoc, T. N., Phai, V/. D., & Shi, Q. "A Deep Learning Approach to Network

Intrusion Detection,” IEEE Transactions on Emerging Topics in Computational Intelligence 2, no. 1, 2018. 41-50

Taye, M., M, “*Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions,*” **Computers** 12, no. 5, 2023.

Tadapaneni, N. R., “*Cloud Computing Security Challenges,*” **International Journal of Innovations in Engineering Research and Technology [IJIERT]** 7, 2020

Talwar, M., “*Security Issues in Internet of Things,*” **International Journal on Emerging Technologies** 6, no. 2, 2015. 364-367

Taj, M. S.,” *Enhancing Anomaly Based Intrusion Detection Techniques for Virtualization in Cloud Computing Using Machine Learning,*” **International Journal of Computer Science and Information Security (IJCSIS)** 18, no. 5, 2020. 60-78

Thakur, K., Tao, L., Wang, T., & Ali, M. L. “*Cloud Computing and Its Security Issues.*” **Application and Theory of Computer Technology** 2, no. 1, 2017. 1-10.

Thilagam, T., & Aruna, R., “*Intrusion Detection for Network Based Cloud Computing by Custom RC-NN and Optimization,*” **ICT Express** 7 no. 4, 2021. 512-520

Tiwari, M., Kuma, R., Bharti, A., & Kishan, J. “*Intrusion Detection System,*” **International Journal of Technical Research and Applications** 5, 2017. 38-44.

Tissir, N., Kafhali, S. E., & Aboutabit, N. “*Cybersecurity Management in Cloud Computing: Semantic Literature Review and Conceptual Framework Proposal,*” **Journal of Reliable Intelligent Environments** 7, no. 2 2020, 69-84.

Tripathy, S. S., & Behera, B., “*Performance Evaluation of Machine Learning Algorithms for Intrusion Detection System.*” **Journal of Biomechanical Science and Engineering** 2023.

Wang, Z., & Chen, X., “*Intrusion Detection-Data Security Protection Scheme Based on Particle Swarm-BP Network Algorithm in Cloud Computing Environment.*” **Journal of Electrical and Computer Engineering** 2023, 2022. 1-10.

Wang, W., Du, X., Shan, D., Qin, R., & Wang, N. “*Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine,*” **IEEE Transactions on Cloud Computing** 10, no. 3, 2020. 1634-1646

Wang, G., Qiao, J., Liu, C., & Shen. Z. “*How Deep Is Deep Enough for Deep Belief Network for Approximating Model Predictive Control Law.*” **IEEE Transactions on Automation Science and Engineering** 2021.1-12

Internet

- BBC News, "Amazon 'thwarts largest ever DDoS cyber-attack,'" **BBC News**, 2020,
1. <https://www.bbc.com/news/technology-53093611>
- Bento, C. "Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis," **Towards Data Science**, 2021, <https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141>
- Craig, S, "The ultimate guide to cybersecurity planning for businesses," **TechTarget Security**, 2023. <https://www.techtarget.com/searchsecurity/The-ultimate-guide-to-cybersecurity-planning-for-businesses>
- GeeksforGeeks, "ML Overview of Data Cleaning," 2023, <https://www.geeksforgeeks.org/data-cleansing-introduction/>.
- Geisle, T, "What Is an Intrusion Detection System," **Datto A Kaseya company**, 2019,
3. <https://www.datto.com/blog/what-is-an-intrusion-detection-system>
- Inamdar, A., "Ensemble Learning Techniques in Machine Learning," **Fire blaze AI School**, 2021, <https://www.fireblazeaischool.in/blogs/ensemble-learning-techniques-in-machine-learning/>
- Kaspersky, "DDoS attacks hit a record high in Q4," **Kaspersky**, 2021, https://www.kaspersky.com/about/press-releases/2022_ddos-attacks-hit-a-record-high-in-q4-2021
- Kiner, F., Konduru, S, "How Google Cloud blocked the largest Layer 7 DDoS attack at 46 million rps," **Google Cloud**, 2022. <https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps>
- Kumar, A. "Feature Scaling in Machine Learning: Python Examples - Data," 2022, <https://vitalflux.com/python-improve-model-performance-using-feature-scaling/>.
- Kundu, R., "The Complete Guide to Ensemble Learning," **V7**, 2022, <https://www.v7labs.com/blog/ensemble-learning>
- Morgan, S., "Humans on The Internet Will Triple From 2015 To 2022 And Hit 6 billion," **Cybercrime Magazine**, 2019, <https://cybersecurityventures.com/how-many-internet-users-will-the-world-have-in-2022-and-in-2030/>
- Mohanty, A. "Multi-layer Perceptron (MLP) Models on Real World Banking Data." **Becoming Human: Artificial Intelligence Magazine**,

2019, <https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f>

Otten, N. V., “*Deep Belief Network Explanation, Application & How to Get Started in TensorFlow.*” **Machine Learning, Natural Language Processing**, 2023. <https://spotintelligence.com/2023/02/10/deep-belief-network/>

Picotte, A, “*Best Intrusion Detection Techniques in Cloud Computing,*” **Uptycs**, 2020. <https://www.uptycs.com/blog/intrusion-detection-in-cloud-computing>

Rishabh, U, “*Review of Accent Classification Using Deep Belief Network,*” **Conceicao Rodrigues College of Engineering, Bandra, Mumbai**. 2017.

RF Wireless World, “*traditional storage vs cloud storage | difference between traditional storage and cloud storage,*” **RF Wireless World.com**, 2012, <https://www.rfwireless-world.com/Tutorials/traditional-storage-vs-cloud-storage.html>

Secure security,” *Cisco Meraki provides Umbrella SD-WAN connector to enhance cloud security across the SD-WAN network*”, **Secure security**, 2022. <https://www.sourcesecurity.com/news/cisco-meraki-umbrella-sd-wan-connector-co-15001-ga-npr.1643261707.html>

Wolfewicz, A. “*Deep Learning vs. Machine Learning -What’s The Difference,*” **Levity**, 2023. <https://levity.ai/blog/difference-machine-learning-deep-learning>

Appendix

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

```
import os
```

```
os.chdir('C:/Users/user/Documents/Project/intrusion detection')
```

```
# In[2]:
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
# In[3]:
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
import plotly.graph_objects as go
```

```
# In[4]:
```

```
df=pd.read_csv('UNSW.csv')
```

```
# In[5]:
```

```
#dataset shape
```

```
df.shape
```

```
# In[6]:
```

```
df.head(5)
```

```
# In[7]:
```

```
#Check If there are any missing values  
df.isnull().sum().sum()
```

```
# In[8]:
```

```
list_drop = ['id','attack_cat']
```

```
# In[9]:
```

```
df.drop(list_drop,axis=1,inplace=True)
```

```
# In[10]:
```

```
df.head(4)
```

```
# In[11]:
```

```
# Clamp extreme Values
```

```
df_numeric = df.select_dtypes(include=[np.number])
```

```
df_numeric.describe(include='all')
```

```
# In[12]:
```

```
DEBUG =0
```

```
for feature in df_numeric.columns:
```

```
    if DEBUG == 1:
```

```
        print(feature)
```

```
        print('max = '+str(df_numeric[feature].max()))
```

```
        print('75th = '+str(df_numeric[feature].quantile(0.95)))
```

```
        print('median = '+str(df_numeric[feature].median()))
```

```
        print(df_numeric[feature].max()>10*df_numeric[feature].median())
```

```
        print('-----')
```

```
    if df_numeric[feature].max()>10*df_numeric[feature].median() and
```

```
df_numeric[feature].max()>10 :
```

```
        df[feature] = np.where(df[feature]<df[feature].quantile(0.95), df[feature],
```

```
df[feature].quantile(0.95))
```

```
# In[13]:
```

```
df_numeric = df.select_dtypes(include=[np.number])
```

```
df_numeric.describe(include='all')
```

```
# In[14]:
```

```
df_numeric = df.select_dtypes(include=[np.number])
```

```
df_before = df_numeric.copy()
```

```
DEBUG = 0
```

```
for feature in df_numeric.columns:
```

```
    if DEBUG == 1:
```

```
        print(feature)
```



```

print('nunique = '+str(df_numeric[feature].nunique()))

print(df_numeric[feature].nunique()>50)

print('-----')

if df_numeric[feature].nunique()>50:

    if df_numeric[feature].min()==0:

        df[feature] = np.log(df[feature]+1)

    else:

        df[feature] = np.log(df[feature])

df_numeric = df.select_dtypes(include=[np.number])

# In[15]:

df_cat = df.select_dtypes(exclude=[np.number])
df_cat.describe(include='all')

# In[16]:

DEBUG = 0

for feature in df_cat.columns:

    if DEBUG == 1:

        print(feature)

        print('nunique = '+str(df_cat[feature].nunique()))

        print(df_cat[feature].nunique()>6)

        print(sum(df[feature].isin(df[feature].value_counts().head().index)))

        print('-----')

```

```

if df_cat[feature].nunique()>6:

    df[feature] = np.where(df[feature].isin(df[feature].value_counts().head().index), df[feature],
    '-')

```

```

# In[17]:

```

```

df_cat = df.select_dtypes(exclude=[np.number])
df_cat.describe(include='all')

```

```

# In[18]:

```

```

df['proto'].value_counts().head().index

```

```

# In[19]:

```

```

df['proto'].value_counts().index

```

```

# In[20]:

```

```

# Feature Selection
from sklearn.feature_selection import SelectKBest, chi2

```

```

best_features = SelectKBest(score_func=chi2,k='all')

```

```

X = df.iloc[:,4:-2]
y = df.iloc[:, -1]
fit = best_features.fit(X,y)

```

```

df_scores=pd.DataFrame(fit.scores_)

```

```

df_col=pd.DataFrame(X.columns)

```

```

feature_score=pd.concat([df_col,df_scores],axis=1)

```

```

feature_score.columns=['feature','score']

```

```
feature_score.sort_values(by=['score'],ascending=True,inplace=True)
```

```
fig = go.Figure(go.Bar(  
    x=feature_score['score'][0:21],  
    y=feature_score['feature'][0:21],  
    orientation='h'))
```

```
fig.update_layout(title="Top 20 Features",  
    height=1200,  
    showlegend=False,  
    )
```

```
fig.show()
```

```
# In[21]:
```

```
X = df.iloc[:, :-1]  
y = df.iloc[:, -1]
```

```
# In[22]:
```

```
X.head()  
feature_names = list(X.columns)  
np.shape(X)
```

```
# In[23]:
```

```
from sklearn.compose import ColumnTransformer
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1,2,3])],  
remainder='passthrough')  
X = np.array(ct.fit_transform(X))
```

```
# In[24]:
```

```
np.shape(X)
```

```
# In[25]:
```

```
df_cat.describe(include='all')
```

```
# In[26]:
```

```
len(feature_names)
```

```
# In[27]:
```

```
for label in list(df_cat['state'].value_counts().index[::-1][1:]):
```

```
    feature_names.insert(0,label)
```

```
for label in list(df_cat['service'].value_counts().index[::-1][1:]):
```

```
    feature_names.insert(0,label)
```

```
for label in list(df_cat['proto'].value_counts().index[::-1][1:]):
```

```
feature_names.insert(0,label)
```

```
# In[28]:
```

```
len(feature_names)
```

```
# In[29]:
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
                                                    test_size = 0.2,
```

```
                                                    random_state = 0,
```

```
                                                    stratify=y)
```

```
# In[30]:
```

```
df_cat.describe(include='all')
```

```
# In[31]:
```

```
# 6 + 5 + 6 unique = 17, therefore the first 17 rows will be the categories that have been encoded,  
start scaling from row 18 only.
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train[:, 18:] = sc.fit_transform(X_train[:, 18:])
```

```
X_test[:, 18:] = sc.transform(X_test[:, 18:])
```

```
# In[32]:
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
def print_score(clf, X_train, y_train, X_test, y_test, train=True):
```

```
    if train:
```

```
        pred = clf.predict(X_train)
```

```

clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))

print("Train Result:\n=====")

print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")

print("_____")

print(f"CLASSIFICATION REPORT:\n{clf_report}")

print("_____")

print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

elif train==False:

    pred = clf.predict(X_test)

    clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))

    print("Test Result:\n=====")

    print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")

    print("_____")

    print(f"CLASSIFICATION REPORT:\n{clf_report}")

    print("_____")

    print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")

# In[33]:

from sklearn.neural_network import BernoulliRBM

from sklearn.pipeline import Pipeline

from sklearn.base import clone

from sklearn.model_selection import GridSearchCV

from sklearn.neural_network import MLPClassifier

from sklearn.ensemble import VotingClassifier

```

```
# In[42]:
```

```
mlp_gs = MLPClassifier(max_iter=100)
```

```
parameter_space = {
```

```
    'hidden_layer_sizes': [(10,30,10),(20,)],
```

```
    'activation': ['tanh', 'relu'],
```

```
    'solver': ['sgd', 'adam'],
```

```
    'alpha': [0.0001, 0.05],
```

```
    'learning_rate': ['constant','adaptive'],
```

```
}
```

```
# In[43]:
```

```
gridsearchcv=GridSearchCV(mlp_gs, parameter_space, n_jobs=-1, cv=5)
```

```
# In[44]:
```

```
# Train the MLP model
```

```
gridsearchcv.fit(X_train, y_train)
```

```
print_score(gridsearchcv, X_train, y_train, X_test, y_test, train=True)
```

```
print_score(gridsearchcv, X_train, y_train, X_test, y_test, train=False)
```

```
# In[45]:
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
import matplotlib.pyplot as plt
```

```
classNames = ['Normal', 'Abnormal']
```

```
predictions = gridsearchcv.predict(X_test)
```

```

cm = confusion_matrix(y_test, predictions, labels=mlp.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,

                               display_labels=classNames) # Use the classNames here

```

```

fig, ax = plt.subplots(figsize=(8, 6))

disp.plot(ax=ax) # Specify the ax parameter to set the title later

# Set the title

ax.set_title('Confusion Matrix - Multilayer Perceptron')

plt.show()

```

```
# In[48]:
```

```

from sklearn.metrics import roc_curve, roc_auc_score

def performance_record(clf):

    pred = clf.predict(X_test)

    fpr, tpr, threshold = roc_curve(y_test, gridsearchcv.predict(X_test))

    roc_auc_score = roc_auc_score(y_test, pred)

    plt.figure()

    plt.plot(fpr, tpr, label="Roc_AUC_score (area = %0.2f)" % roc_auc_score_)

    plt.plot([0,1],[0,1], "r--")

    plt.xlabel('True postive rate')

    plt.ylabel('False Postive rate')

    plt.title('ROC Value for MLP')

```



```
plt.legend(loc="lower right")
```

```
# In[49]:
```

```
from sklearn.metrics import roc_curve, roc_auc_score
```

```
def performance_record(clf):
```

```
    pred = clf.predict(X_test)
```

```
    fpr, tpr, threshold = roc_curve(y_test, hybridise_model.predict(X_test))
```

```
    roc_auc_score_ = roc_auc_score(y_test, pred)
```

```
    plt.figure()
```

```
    plt.plot(fpr, tpr, label="Roc_AUC_score (area = %0.2f)" % roc_auc_score_)
```

```
    plt.plot([0,1],[0,1], "r--")
```

```
    plt.xlabel('True postive rate')
```

```
    plt.ylabel('False Postive rate')
```

```
    plt.title('ROC Value for DBN & MLP')
```

```
    plt.legend(loc="lower right")
```

```
# In[50]:
```

```
performance_record(hybridise_mode
```

```
{
```

```
    "cells": [
```

```
        {
```

```
            "cell_type": "code",
```

```
            "execution_count": 37,
```

```
            "id": "7726d85e",
```

```
            "metadata": {},
```

```
            "outputs": [],
```

```
            "source": [
```

```
                "import os \n",
```

```
                "os.chdir('C:/Users/Kamar/Desktop/Project/Intrusion')"
```

```

]
},
{
  "cell_type": "code",
  "execution_count": 38,
  "id": "2c6c38e9",
  "metadata": {},
  "outputs": [],
  "source": [
    "import warnings\n",
    "warnings.filterwarnings('ignore')\n"
  ]
},
{
  "cell_type": "code",
  "execution_count": 39,
  "id": "962b9556",
  "metadata": {},
  "outputs": [],
  "source": [
    "import pandas as pd\n",
    "import numpy as np\n",
    "import matplotlib.pyplot as plt\n",
    "import seaborn as sns\n",
    "%matplotlib inline\n",
    "import plotly.graph_objects as go\n"
  ]
},
{
  "cell_type": "code",
  "execution_count": 40,
  "id": "f8d60dc0",
  "metadata": {},
  "outputs": [],
  "source": [
    "df=pd.read_csv('UNSW.csv')\n"
  ]
},
{
  "cell_type": "code",

```

```

"execution_count": 41,
"id": "1486d79a",
"metadata": {},
"outputs": [
  {
    "data": {
      "text/plain": [
        "(82332, 45)"
      ]
    },
    "execution_count": 41,
    "metadata": {},
    "output_type": "execute_result"
  }
],
"source": [
  "#dataset shape\n",
  "df.shape"
],
{
  "cell_type": "code",
  "execution_count": 42,
  "id": "93bf160e",
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/html": [
          "<div>\n",
          "<style scoped>\n",
          "  .dataframe tbody tr th:only-of-type {\n",
          "    vertical-align: middle;\n",
          "  }\n",
          "\n",
          "  .dataframe tbody tr th {\n",
          "    vertical-align: top;\n",
          "  }\n",
          "\n",
          "  .dataframe thead th {\n",

```

```

"    text-align: right;\n",
"  }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>id</th>\n",
"      <th>dur</th>\n",
"      <th>proto</th>\n",
"      <th>service</th>\n",
"      <th>state</th>\n",
"      <th>spkts</th>\n",
"      <th>dpkts</th>\n",
"      <th>sbytes</th>\n",
"      <th>dbytes</th>\n",
"      <th>rate</th>\n",
"      <th>...</th>\n",
"      <th>ct_dst_sport_ltm</th>\n",
"      <th>ct_dst_src_ltm</th>\n",
"      <th>is_ftp_login</th>\n",
"      <th>ct_ftp_cmd</th>\n",
"      <th>ct_flw_http_mthd</th>\n",
"      <th>ct_src_ltm</th>\n",
"      <th>ct_srv_dst</th>\n",
"      <th>is_sm_ips_ports</th>\n",
"      <th>attack_cat</th>\n",
"      <th>label</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>1</td>\n",
"      <td>0.000011</td>\n",
"      <td>udp</td>\n",
"      <td>-</td>\n",
"      <td>INT</td>\n",
"      <td>2</td>\n",
"      <td>0</td>

```

```

"    <td>496</td>\n",
"    <td>0</td>\n",
"    <td>90909.0902</td>\n",
"    <td>...</td>\n",
"    <td>1</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>1</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>Normal</td>\n",
"    <td>0</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>1</th>\n",
"    <td>2</td>\n",
"    <td>0.000008</td>\n",
"    <td>udp</td>\n",
"    <td>-</td>\n",
"    <td>INT</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>1762</td>\n",
"    <td>0</td>\n",
"    <td>125000.0003</td>\n",
"    <td>...</td>\n",
"    <td>1</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>1</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>Normal</td>\n",
"    <td>0</td>\n",
"  </tr>\n",
"  <tr>\n",

```

```

"    <th>2</th>\n",
"    <td>3</td>\n",
"    <td>0.000005</td>\n",
"    <td>udp</td>\n",
"    <td>-</td>\n",
"    <td>INT</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>1068</td>\n",
"    <td>0</td>\n",
"    <td>200000.0051</td>\n",
"    <td>...</td>\n",
"    <td>1</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>1</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"    <td>Normal</td>\n",
"    <td>0</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>3</th>\n",
"    <td>4</td>\n",
"    <td>0.000006</td>\n",
"    <td>udp</td>\n",
"    <td>-</td>\n",
"    <td>INT</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>900</td>\n",
"    <td>0</td>\n",
"    <td>166666.6608</td>\n",
"    <td>...</td>\n",
"    <td>1</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",

```

```

"    <td>0</td>\n",
"    <td>2</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"    <td>Normal</td>\n",
"    <td>0</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>4</th>\n",
"    <td>5</td>\n",
"    <td>0.000010</td>\n",
"    <td>udp</td>\n",
"    <td>-</td>\n",
"    <td>INT</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>2126</td>\n",
"    <td>0</td>\n",
"    <td>100000.0025</td>\n",
"    <td>...</td>\n",
"    <td>1</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>2</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"    <td>Normal</td>\n",
"    <td>0</td>\n",
"  </tr>\n",
"</tbody>\n",
"</table>\n",
"<p>5 rows × 45 columns</p>\n",
"</div>"

```

],

"text/plain": [

```

" id    dur proto service state  spkts  dpkts  sbytes  dbytes  \\n",
"0  1  0.000011  udp    -  INT    2    0    496    0  \n",
"1  2  0.000008  udp    -  INT    2    0   1762    0  \n",

```

```

"2 3 0.000005 udp - INT 2 0 1068 0 \n",
"3 4 0.000006 udp - INT 2 0 900 0 \n",
"4 5 0.000010 udp - INT 2 0 2126 0 \n",
"\n",
" rate ... ct_dst_sport_ltm ct_dst_src_ltm is_ftp_login \\n",
"0 90909.0902 ... 1 2 0 \n",
"1 125000.0003 ... 1 2 0 \n",
"2 200000.0051 ... 1 3 0 \n",
"3 166666.6608 ... 1 3 0 \n",
"4 100000.0025 ... 1 3 0 \n",
"\n",
" ct_ftp_cmd ct_flw_http_mthd ct_src_ltm ct_srv_dst is_sm_ips_ports \\n",
"0 0 0 1 2 0 \n",
"1 0 0 1 2 0 \n",
"2 0 0 1 3 0 \n",
"3 0 0 2 3 0 \n",
"4 0 0 2 3 0 \n",
"\n",
" attack_cat label \n",
"0 Normal 0 \n",
"1 Normal 0 \n",
"2 Normal 0 \n",
"3 Normal 0 \n",
"4 Normal 0 \n",
"\n",
"[5 rows x 45 columns]"
]
},
"execution_count": 42,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.head(5)"
]
},
{
"cell_type": "code",
"execution_count": 43,

```



```
"id": "513b87e5",  
"metadata": {},  
"outputs": [  
  {  
    "data": {  
      "text/plain": [  
        "0"  
      ]  
    }  
  }  
]
```

Do Not Copy, Lead City University, Nigeria

Biodata

A. Personal

Data Full name: Awodele Simon Olufikayo
Address: College of Medicine Quarters, DIB
Eni Njoku Road, University of Lagos.
Email Address: talk2ficky@gmail.com
Phone Number: 08035163072
Date of Birth: 09/01/1980
Place of Birth: Osi, Kwara State.
Gender: Male
Name of Next of kin: Awodele Taiwo Abidemi

B. Educational Background

- Lead City University, Ibadan, PhD in Computer Science (in view)
- M. Sc Electrical Engineering (Communication Option)
University of Lagos. (2012)
- B.Eng. Electrical/Computer Engineering
Federal University of Technology Minna, Niger State. (2006)

C. Work Experience with Dates

New Horizons Systems Solutions Ikeja, Lagos.

5, Babatola drive, off Awolowo Way Ikeja,

Lagos

Designation: Technical & Quality Assurance Officer

(October 2009 – August 2015)

Responsibilities:

- Monitoring and maintaining Computer Systems and Networks, Planning and undertaking scheduled maintenance upgrades.
- Talking to staff or clients through a series of actions, either face to face or over the telephone to help set up systems or resolve issues.
- Plan and Implementation required configuration changes to Local area network, manage the installations and monitoring of organization's Local area network system.
- Make recommendations to administrator regarding standards for the computer and computer network.
- Training, supervising and coordinating the Industrial trainee students.
- Provide Technical support
- General Hardware and Software Troubleshooting and Maintenance

Network Manager (September 2015 – July 2018)

Ivetec Global Service

6, Babatola Drive, off Awolowo Way Ikeja, Lagos

Responsibilities:

- Manage and supervise the company projects.

- Network planning, implementation and system networking (LAN, WAN AND WIMAX).
- Configuration, installation, maintenance and troubleshooting of wireless network.
- Installation, configuration, maintenance and troubleshooting of CISCO Routers and Switch
- General hardware and software troubleshooting and maintenance.
- Basic troubleshooting, installation, maintenance and repair on other designated equipment.
- Presenting ideas for systems Improvement and Liaise with the ISP.
- Utilizing the escalation process to resolve customer service delivery issues

WiniGroup limited

6, Babatola Drive, off Awolowo Way Ikeja, Lagos

TECHNICAL (August 2018 – January 2023)

Responsibilities:

- Giving technical support to the company clients in person and remotely
- Working with client to establish and meet business
- Following standard procedures for proper escalation of unresolved issues to the technical team lead

New Horizons Systems Solutions Ikeja, Lagos.

5, Babatola drive, off Awolowo Way Ikeja,

Lagos

Designation: Assistance General Manager (Technical)

(February 2023 – Present)

Responsibilities:

- Helps the general manager to ensure that day-to-day business operations run smoothly
- Providing general supervision, including interviewing and hiring, employee engagement, personal attendance and training
- Ensuring that all office workflows are carried out in a timely and accurate manner
- Contributing to plans for the organization's progress and growth

D. Membership

- Corporate Member Nigeria Society of Engineers (MNSE)
- Registered Engineer; COREN

E. Professional Qualifications:

- Cisco Certified Network Associate (CCNA)
- Cisco Certified Network Professional (CCNP)
- Microsoft Certified PROFESIONAL (MCP)
- Network Security (Certiport)
- Networking (Certiport)
- Data Analysis (Certiport)
- Cybersecurity (Certiport)
- Cloud Computing (Certiport)

F. Publication Journal Article:

- Intrusion Detection System in Cloud Computing: A Review

(Authors: Awodele Simon Olufikayo, & Dr. Wilson Sakpere)

G. Major Conferences/Workshops Attended

- 4th International Faculty of Basic Medical and Applied Sciences
Conference (FASCON 2022).

H. References

- Engr, Ofusori Temidayo
Senior Lecturer
Department of Electrical and Electronics Engineering
Federal University Oye-Ekiti
08036968132
- Mr. Dayo Folowoshele
Gasridders Nig. Ltd, Lagos.
08035070677

Do Not Copy, Lead City University, Nigeria

The University Compliance Form

This is to certify that this thesis by Simon Olufikayo Awodele with Matriculation Number LCU/PG/001505 in the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan is in full compliance with the approval of the University's format and style.

.....

Signature

.....

Date

Do Not Copy, Lead City University, Nigeria

Do Not Copy, Lead City University, Nigeria