REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

# IOT SECURITY BY DETECTION OF THE NETWORK INTRUSION TRAFFIC USING DL

**Nadia Qutaiba Mohamedsaeed ALSABBAGH**

MASTER OF SCIENCE THESIS

Department of Computer Engineering

Program of Computer Engineering

Supervisor

Prof. Dr. Hasan Hüseyin BALIK

July,2024

**REPUBLIC OF TÜRKİYE**

**YILDIZ TECHNICAL UNIVERSITY**

**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# IOT SECURITY BY DETECTION OF THE NETWORK INTRUSION TRAFFIC USING DL

A thesis submitted by Nadia Qutaiba Mohamedsaeed ALSABBAGH in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE is approved by the committee on 03.07.2024 in the Department of Computer Engineering, Program of Computer Engineering.

Prof. Dr. Hasan Hüseyin BALIK
Yildiz Technical University
Supervisor

**Approved By the Examining Committee**

Prof. Dr. Hasan Hüseyin BALIK, Supervisor
Yildiz Technical University

Dr.Öğr.Üye. Mustafa Utku KALAY, Member
Yildiz Technical University

Doç.Dr. Can EYÜPOĞLU, Member
Hava Harp Okulu

I hereby declare that I have obtained the required legal permissions during the data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven’t falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of IOT SECURITY BY DETECTION OF THE NETWORK INTRUSION TRAFFIC USING DL supervised by my supervisor, Prof. Dr. Hasan Hüseyin Balık In the case of a discovery of a false statement, I am to acknowledge any legal consequence.

Nadia Qutaiba Mohamedsaeed ALSABBAGH

Signature

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) Grant No: 2210.

*Dedicated to my family*

*and my best friend*

# ACKNOWLEDGEMENTS

First and foremost, praises and thanks to God, the Almighty, for His showers of blessings.I would like to express my deepest and sincere gratitude to my research supervisor, Prof. Dr. Hasan Hüseyin Balık, for his invaluable patience and feedback. This study could not have been possible without his great support and guidance.

I would like to thank the Republic of Turkey, the Ministry of Higher Education, and Yildiz Technical University in Istanbul.

Also, I would like to thank the Republic of Iraq, the Ministry of Higher Education, and the University of Mosul for supporting me in continuing my academic studies.

Thanks, I should also go to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future.

Lastly, I would be remiss in not mentioning my family, especially my husband and my children for their belief in me, keeping me in high spirits, and they give me more love, understanding, and prayers for continuing to complete this research work.

Nadia Qutaiba Mohamedsaeed ALSABBAGH

# TABLE OF CONTENTS

# LIST OF SYMBOLS

| | |
|---|---|
| Tanh | Hyperbolic tangent of activation function |
| $e^x$ | Based on the natural logarithms |
| Σ | The sum of Multiple Terms |
| ≥ | greater than or equal to |
| < | less than |
| σ | sigmoid activation |

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ADAM | Adaptive Moment Estimation |
| AIDS | Anomaly-Based Intrusion Detection System |
| ANN | Artificial neural networks |
| API | Application Programming Interface |
| AUC | Area Under the Curve |
| CNN | Convolutional neural network |
| CPU | Central Processing Unit |
| DBN | Deep Belief Network |
| DDoS | Distributed Denial of Service |
| DoS | Denial of Service |
| DNN | Deep neural network |
| FN | False Negative |
| FP | False Positive |
| GPU | Graphics Processing Unit |
| GRU | Gated recurrent unit |
| HIDS | Host-based Intrusion Detection System |
| HTTP | Hypertext Transfer Protocol |
| IDS | Intrusion Detection System xi |
| IOT | Internet of Things |
| IP | Internet Protocol |
| LSTM | Long short-term memory |
| MLP | MultiLayer Perceptron |
| ML | Machine Learning |

| | |
|---|---|
| NIDS | Network intrusion detection system |
| R2L | Remote To Local |
| RBM | Restricted Boltzmann Machine |
| ReLU | Rectified linear unit |
| RF | Random forests |
| RNN | Recurrent neural networks |
| SDN | Software-Defined Networking |
| SIDS | Signature-Based Intrusion Detection Systems |
| SIEM | Security Information and Event Management |
| SMOTE | Synthetic Minority Over-sampling Technique |
| SOM | Self-organized maps |
| STL | Self-taught learning |
| TCP | Transmission Control Protocol |
| TN | True Negative |
| TP | True Positive |
| U2R | User To Root |
| UDP | User Datagram Protocol |

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

## IOT SECURITY BY DETECTION OF THE NETWORK INTRUSION TRAFFIC USING DL

Nadia Qutaiba Mohamedsaeed ALSABBAGH

Department of Computer Engineering
Master of Science Thesis

Supervisor: Prof. Dr. Hasan Hüseyin BALIK

This thesis "IoT Security by Detection of the Network Intrusion Traffic Using DL," describes a technique for detecting and preventing network intrusion assaults on IoT devices using deep learning (DL) techniques. The thesis indicates a method for reading network visitor records and identifying patterns of hobby that point to a likely intrusion using a multilayer perceptron (MLP) and a long-short-term memory (LSTM) neural network. The author highlights the vulnerability of IoT gadgets to community intrusion because of their limited processing and reminiscence potential, in addition to the absence of security mechanisms in numerous IoT devices. The proposed method operates a deep learning technique to analyse network visitors' records and recognise suspicious patterns of hobbies to deal with this trouble. Furthermore, via executing examinations on a dataset of network site visitor records obtained from IoT devices, the thesis elucidates the efficacy of the suggested approach in figuring out network intrusion dangers. The results display how nicely the LSTM-based totally technique has acknowledged several sorts of network intrusion attempts, consisting of denial of service (DoS) attacks and malware contamination assaults. Overall, the Thesis highlights the significance of imposing powerful security measures for IoT devices and demonstrates the capability of DL techniques for detecting and preventing network intrusion assaults on those devices. The proposed framework can be beneficial for improving the safety of IoT devices and networks, and for mitigating the risk of cyber-attacks and breaches.

We will evaluate the performances in terms of training, prediction time, accuracy, and different simulation outcomes based totally on precision, taking into account, the F1 score, and log loss with the use of the dataset KDDCUP-99.

**Keywords**: Deep Learning, Intrusion Detection Systems, IoT Attacks, LSTM, MLP, RNN, Security of the Internet of Things.

# ÖZET

## DL KULLANILARAK AĞ İZİNSİZ TRAFİĞİNİN TESPİTİ İLE IOT GÜVENLİĞİ

Nadia Qutaiba Mohamedsaeed ALSABBAGH

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Prof. Dr. Hasan Hüseyin BALIK

Bu Tez " DL Kullanılarak Ağ İzinsiz Trafiğinin Tespiti ile IOT Güvenliği," derin öğrenme (DL) tekniklerini kullanarak IoT aygıtlarına yönelik ağa izinsiz giriş saldırılarını tespit etmeye ve önlemeye yönelik bir tekniği açıklar. Tez, ağ ziyaretçilerinin kayıtlarını okumak ve çok katmanlı bir algılayıcgı (MLP) ve Uzun Kısa Süreli Bellek (LSTM) sinir ağı kullanarak olası bir izinsiz girişe işaret eden hobi kalıplarını belirlemek için bir yöntem göstermektedir. Yazar, çok sayıda IoT cihazında güvenlik mekanizmalarının bulunmamasına ek olarak, sınırlı işleme ve hatırlama potansiyelleri nedeniyle IoT cihazlarının ağa izinsiz giriş denemelerine karşı savunmasızlığını vurguluyor. Önerilen yöntem, bu sorunla başa çıkmak için ağ ziyaretçi kayıtlarını analiz etmek ve şüpheli hobi kalıplarını tanımak için derin öğrenmeye dayalı teknik kullanıyor. Ayrıca, IoT cihazlarından elde edilen ağ sitesi ziyaretçi kayıtlarından oluşan bir veri seti üzerinde incelemeler yapılarak tez, önerilen yaklaşımın ağa izinsiz giriş tehlikelerini belirlemedeki etkinliğini açıklamaktadır. Sonuçlar, LSTM tabanlı tamamen tekniğin, hizmet reddi (DoS) saldırıları ve kötü amaçlı yazılım bulaşma saldırıları da dahil olmak üzere çeşitli türde ağa izinsiz giriş denemelerini ne kadar başarılı bir şekilde kabul ettiğini gösteriyor. Genel olarak Tez, Nesnelerin İnterneti aygıtları için güçlü güvenlik önlemlerinin uygulanmasının önemini vurgulamakta ve bu aygıtlara yönelik ağa izinsiz giriş saldırılarını tespit etmek ve önlemek için DL tekniklerinin yeteneğini göstermektedir. Önerilen çerçeve, IoT cihazlarının ve ağlarının güvenliğini artırmak ve siber saldırı ve ihlal riskini azaltmak için faydalı olabilir.

Performansları eğitim, tahmin süresi, doğruluk ve farklı simülasyon sonuçları açısından tamamen hassasiyete dayalı olarak değerlendireceğiz, F1 puanını dikkate alacağız ve KDDCUP-99 veri kümesinin kullanımındaki günlük kaybını değerlendireceğiz.

**Anahtar Kelimeler:** Derin Öğrenme, Saldırı Tespit Sistemleri, IoT Saldırıları, LSTM, MLP, RNN, Nesnelerin İnternetinin Güvenliği.

# 1 INTRODUCTION

In the recent years of information technology (IT), the complexity of numerous information devices has increased, connected concurrently, and persists to produce and save significant digital data .

Accordingly, this introduces challenges to individuals and corporations. Given these points, the approaches to attack detection should also be more competent and practical than before to combat attack hackers, which are also developing persistently. For detecting abnormal behaviours (attacks), the current technologies commonly employ detection methods based on event examination with preset rules. So, insufficient data regarding real attacks or inaccurate identification of attacks in the security domain; result in financial losses and the limited utilisation of such systems. Consequently, their primary function is the automated gathering and analysis of varied security event data for risk assessment. To ensure essential computing capacity, automatic intrusion detection systems must leverage cloud computing resources and acquire insights into attack methodologies for adaptive responses in the security industry. This thesis aimed to determine and prevent malicious activities targeting Internet of Things devices. DL algorithms analyse network traffic patterns to detect anomalies and classify them as intrusions, allowing timely responses to protect sensitive data and support system integrity.

## 1.1 Literature Review

Networks and their record-protective capacities have stepped forward in a manifold manner to make cutting-edge strides in communication technologies. Still, this improvement has brought forth a group of the latest and most insidious threats. A lot of studies on network protection can correctly discover attacks amidst this sea of rising dangers that target vulnerabilities specific to IoT devices. The range of related gadgets keeps skyrocketing, fueling online traffic further compounded by the use of our insatiable desire for all matters internet-driven. Because of this, state-of-the-art IoT packages can now actualize their information flow from giving up to surrender without relying on archaic techniques [1]. The ever-developing quantity makes it nearly impossible to hit upon assaults and avoid them; therefore, there is a need for DL techniques in particular tailored within the course of IDS through deep learning knowledge of architectures. Here, we talk approximately using DL fashions as a revolutionary technique toward hazard detection within the Internet of Things (IoT), and figuring out malicious facts in its early stages is a specially tough challenge in which these state-of-the-art models could function as surrogate brains capable of encapsulating real life functionalities, all completed through the synthetic technique.

The development of the Internet of Things (IoT) is considered one of the most crucial advances in technology honestly, which notably impacts all spheres of human life, depending on one-of-a-kind use times. IoT tool interconnection that might be leveraged for information series.

In addition to quick developments in the actions with the world spherical us, the IoT is not on my own in this difference: there have been several different top-notch technical advances in today's years that have had huge influences on each society as well as economies.

The proliferation of smartphones and other cellular gadgets has extensively revolutionized the way humans speak, access information, and behave in their business enterprise dealings. The emergence of generation using renewable power resources adjusts how we produce and use power, this technique is a new wave of innovation in the power; business enterprise. There are some sectors like production, healthcare, and transportation that can introduce new packages through device learning and synthetic intelligence; others consist of functions in diverse industries because they are capable of discovering extra competencies from those technologies.

Nonetheless, the impact of the Internet of Things on exceptional aspects of our everyday life is considerable, and as such, the development in relationships between human beings and vehicles or houses and the ways we care for our health is quickly significantly altered by technology, a phenomenon in which IoT takes the middle ground. The connection of normal gadgets to the net that permits them to interact and talk with everyone has created a new field of revolutionary opportunities; a generation in which creativity reigns. But those possibilities additionally pose risks such as information protection, privacy, and safety trouble when extra devices are connected to the internet, and greater instances of information breaches and cyber assaults are mentioned.

This underscores the need for IoT development with strict adherence to fine practices in conjunction with robust security mechanisms.

While the competition over whether the Internet of Things stands as history's singular best technical soar remains an open debate, what can't be contested is that this behemoth quarter, one that continues to grow at a breakneck pace with innovation holds potential that might redefine how we live our lives.

The majority of groups which include manufacturing, healthcare, and retail at the moment are using IoT sensors for monitoring purchases, remotely tracking sufferers, or even running completely self-reliant warehouses. But many greater packages have come about these days than ever earlier; therefore, there were roughly 9.8 billion IoT gadgets operational in use in 2020, a Figure that had risen to around 13.1 billion in 2022, with a forecast indicating that globally there might be 29.4 billion devices in use through 2030 [2].

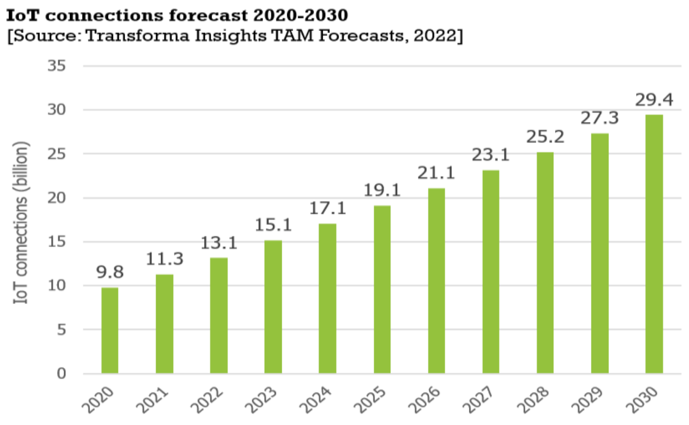


**Figure 1.1** IOT Connections Forecast 2020-2030[2]

The Internet of Things (IoT) is generally understood to be a way that entails proper strategies that link through servers, sensors, and diverse software. The town structure proven in Figure 1.2 has three basic layers, which might be labeled as the fog layer, the cloud layer, and the terminal layer respectively. The records obtained from the Internet of Things gadgets are saved in an environment surroundings called cloud computing (CC), which functions with stronger CPUs and sufficient reminiscence. The rapid increase in the cloud layer may be attributed to the current developments in the Internet of Things technology. A conceivable clarification of such difficulties became integrated into the development of the fog-to-matters system.

In the fog layer, there may be a possibility that gadgets will come across some better statistics values on the way to being successfully moved to the cloud layer. This results in a reduction in the quantity of electricity wasted, the quantity of community site visitors, and the use of bandwidth, together with the demanding situations related to information storage and transmission. Furthermore, it strives to expedite the estimated technique as it processes its very last stages. Enabling a prompt response to urban use with the Internet of Things. Implementing threat detection within the fog-to-things layer has advantages. If those network assaults are detected in the fog layer, either the net service provider or the network administrator can also put into effect preventive movements that would probably reduce the quantity of harm caused. A further advantage of this approach is that it does not keep human beings from going approximately as far in their daily lives as they usually might.

The model observes the quantity of internet traffic produced through every fog-to-subjects node. Identifying network assaults at the fog-to-matters connections is more efficient than doing so on the cloud layer due to the similarity between fog-to-things connections and IoT devices. This may be completed with the aid of comparing the site visitors among the two layers. Threats may be quickly communicated to the network controllers of IoT devices, allowing the controllers to assess and improve their security measures as vital.

It is important to conduct behaviour research on methods that can pick out this type of network visitor before attempting to develop ”intrusion detection and prevention structures.” Through the software of gadget learning (ML) techniques, intrusion detection and prevention structures are able to discover malicious site visitors. “Using algorithms to research statistics and make predictions primarily based on that data is the goal of machine learning (ML), which is a subset of synthetic intelligence (AI).” [3]. “AI algorithms may be applied within the retail, healthcare, and economic sectors, respectively, to foresee customer shopping for traits, affected person fitness issues, and financial institution fraud” [4]. ML has a huge variety of applications, a number of which include those three industries. As a result of the big annual increase in the wide variety of cyberattacks that can be seen on an annual basis, ML processes are being employed to help address the expanding risks of cyberattacks.

Within the area of cybersecurity," DL is positioned to be used for a variety of reasons, one of which is network chance evaluation" [5], which refers back to the procedure of figuring out potential threats to a community. Considering that DL is capable of monitoring incoming and outgoing traffic to perceive traffic that may be suspicious, it may be beneficial for this cause [6]. The research discipline of intrusion detection is widely known and wherein work is being finished. Intrusion detection systems (IDS) can achieve the blessings of deep learning (DL) by becoming more self-sufficient and by appropriately sounding the alert in the event that a probable intrusion is being investigated [7].

To achieve this aim, we're tasked with developing the most superior "deep learning (DL) algorithms" viable to detect assaults on Internet of Things (IoT) networks. These algorithms will employ a contemporary dataset in addition to binary and multi-class category testing.

Machine learning (ML), a subset of the synthetic intelligence era, will end the evaluation and deliver video pictures to people so that they will reply swiftly to concerns and ensure the citizens' protection. ML is an acronym for "machine studying."

Attack detection may be classified as either largely signature-based or predominantly anomaly-based, depending on which technique is used more regularly. In spite of the reality that the latter analyses the behaviour of everyday site visitors, the former makes use of a typically primarily based approach that will check the incoming site visitors for the many kinds of assaults and crimes that are stored inside the database.

Overall, this highlights the growing importance of security for Internet of Things (IoT) gadgets that are increasingly being used in vital packages including healthcare, transportation, and business control structures. The authors should be aware that using IoT devices introduces new safety-demanding situations, as those gadgets are regularly aid-restrained and lack the safety features observed in conventional computing devices.

So one of the most important safety risks for IoT devices is community intrusion attacks, which may additionally undermine the integrity, confidentiality, and availability of IoT networks and data. The thesis indicates using a DL-based method to perceive and prevent network intrusion attacks on IoT devices. The recommended methodology employs a Long Short-Term Memory (LSTM) and Multi-Layer Perceptron (MLP) neural community to scrutinize network traffic records and come across discernible styles of conduct that represent a capability intrusion.

The authors say that DL techniques have shown superb promise in the extensive variety of packages, along with picture recognition, language processing, and speech recognition [8]. However, the application of DL to the sector of network security remains particularly new, and there may be a need for further research and development in this area.

The Thesis ambitions to "address this gap with the aid of offering a DL-primarily based technique for detecting community intrusion assaults on IoT gadgets" [9]. The authors describe experiments performed on a dataset of network visitor statistics from IoT gadgets and display the effectiveness of the proposed approach in detecting numerous sorts of community intrusion attacks.

Overall, they highlight the need for effective security measures for IoT devices and the ability of DL strategies to improve the security of these devices. The thesis contributes to the developing body of research on DL-primarily-based strategies for network safety and offers insights into the demanding situations and possibilities of applying DL to this field.

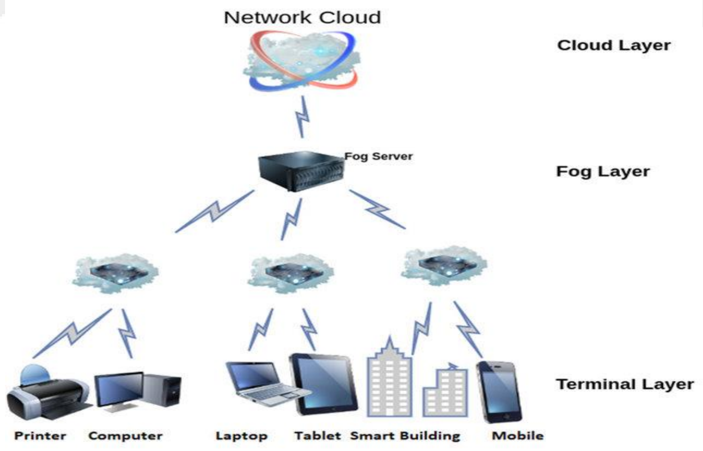


**Figure 1.2** Framework of a smart IoT-based city [10]

A famous area of research is deep learning, a subfield of systems that gain knowledge with fantastic performance. IDSs with deep studying can attain good detection stages when deep learning fashions can generalize nicely enough and have enough training information to locate novel and variational attacks [11]. They are also easy to develop and build because they hardly ever want area expertise.

“Deep learning methods outperform ordinary machine learning approaches in their ability to successfully handle enormous volumes of data. Deep learning algorithms are particularly pragmatic as they function from start to finish and contain the capacity to autonomously collect feature representations from raw data” [12]. The "Data" is the key prerequisite for using deep learning to tackle time series issues. This data may be time-series data with one or more variables, and it can also be time-series data with just one variable. So the taxonomy of IDS Systems [13] is like in Figure1.3:

- Network-based IDS (NIDS)
  - Perimeter NIDS: used to monitor inbound and outgoing traffic at the network's edge.
  - Internal NIDS: deployed within the internal network to monitor traffic between different segments or subnets.
  - Virtual NIDS: deployed in virtualized environments to monitor virtual machines and containers.
- Host-based IDS (HIDS)
  - Server HIDS: installed on servers to screen activities at the host, together with file gadget modifications, login tries, and method behaviours.
  - Endpoint HIDS: installed on endpoints, which include desktops or laptops, to monitor activities on the host and discover capacity attacks or malicious activities.
  - Cloud HIDS: deployed in cloud environments to monitor cloud resources, digital machines, or boxes for security threats.
- Hybrid IDS
  - Network + Host IDS: a mixed approach for thorough chance detection that makes use of both host- and network-based total techniques.
  - Cloud + Network IDS: an incorporated method for keeping a watch on community traffic and cloud sources that mixes both cloud- and community-primarily based processes.
- Signature-based IDS
  - Network Signature-based totally IDS: detects and alerts users to doubtlessly risky hobbies in community facts through the use of preset signatures or styles of recognized assaults.

- Host Signature-based IDS: this system detects and notifies users of dangerous hobbies going on on hosts primarily based on pre-established signatures or known attack styles.

- Anomaly-based IDS
  - Network Anomaly-based IDS: monitors network site visitors for deviations from mounted baselines or regular styles to locate anomalies that can indicate potential attacks.
  - Host Anomaly-based IDS: monitors host activities for abnormal behaviours that deviate from established baselines or normal patterns to detect potential attacks.
- Behavior-based IDS
  - Network Connection Behavior-based IDS: the system monitors network connection traffic to detect patterns of activity that might suggest malicious actions, such as several unsuccessful login attempts, abnormal data transfers, or suspicious network behaviours.
  - Host Behavior-based IDS: monitors host activities for styles of conduct that could indicate malicious moves, together with privilege escalation, unauthorized access attempts, or strange system behaviours.
- Cloud-based IDS
  - Infrastructure-level Cloud IDS: monitors the infrastructure and resources in cloud environments, consisting of virtual networks, storage, and databases, for security threats.
  - Application-level Cloud IDS: monitors cloud-primarily based programs, which include internet applications or APIs, for potential attacks or vulnerabilities.
- Distributed IDS
  - Geographical Distributed IDS: deployed across multiple geographical locations or sites to monitor and detect security threats in distributed or remote networks.
  - Segment-based Distributed IDS: deployed in different network segments or subnets to monitor traffic within each segment and detect potential attacks or suspicious activities.
- Host Intrusion Prevention System (HIPS)

- Server HIPS: provides proactive protection on servers employing actively stopping and blocking malicious activities, along with network connections, system executions, or record device modifications.
- Endpoint HIPS: Provides proactive protection on endpoints via actively preventing and blocking off malicious activities, which include unauthorized get-right-of-entry attacks, malware executions, or device adjustments.
- Cloud HIPS: provides proactive defence in cloud environments by actively stopping and blocking malicious activities in cloud assets, virtual machines, or boxes.

The shape of the thesis is as indicated below. In the subsequent part (II), we can articulate the related works. In Section III, we move into detail, describing the methodology of the algorithms. In this component, the evaluation dataset, in addition to the various sorts of assaults and their traits, is supplied. In Section V, the results of the study and critiques of their overall performance are provided. In the concluding element, "Section VI," we present a summary of the thesis and a top-level view of our objectives for future research.

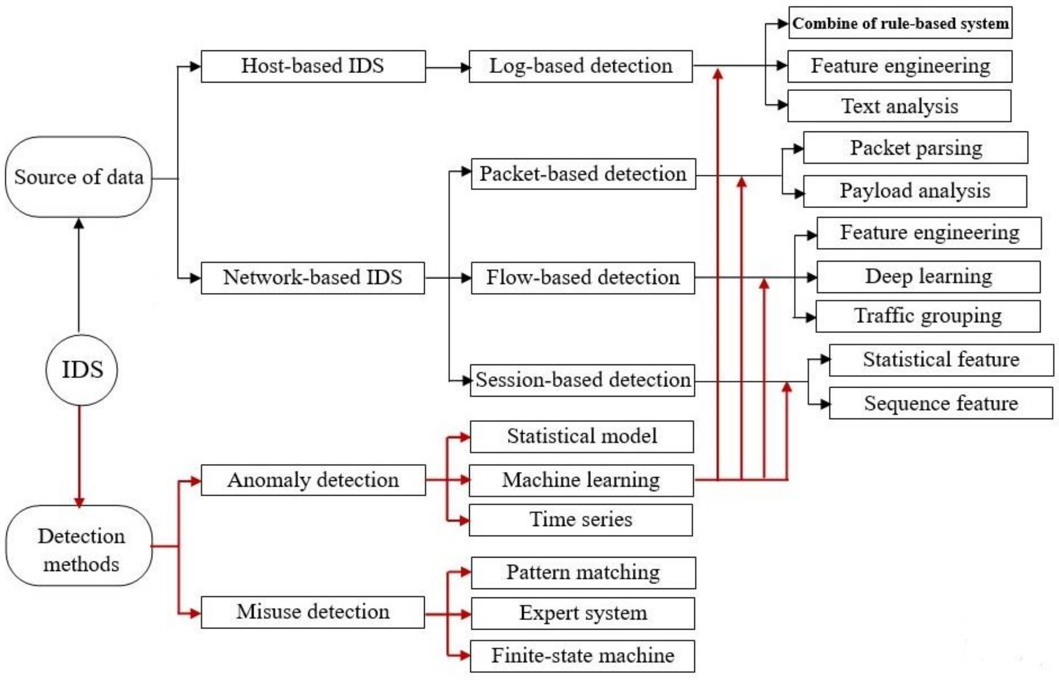


**Figure 1.3** Taxonomy of IDS systems

## 1.2 Objectives of Study

In order to understand future cyberattacks on IoT device networks and protect against them, we should check out IoT security via the use of DL-based strategies in community intrusion traffic detection. However, it needs to be prioritized for steady IoT gadgets because the quantity of IoT devices is growing and they're broadly used in lots of programs that make such protection susceptible: to unauthorized access and record breaches.

Improving the effectiveness and precision of intrusion detection in IoT networks can be better carried out through the employment of deep learning algorithms. The fundamental cause of the project is to create deep knowledge of models that take a look at network traffic statistics in detail, identifying styles and anomalies that might indicate future attacks, through which these fashions are to be taught with large datasets on network traffic to make certain excessive ranges of accuracy and reliability.

The work aims to supply pragmatic and talented strategies for the use of deep learning primarily based on network intrusion website traffic detection for IoT networks. This will result in improved safety of IoT networks and gadgets from cyber threats.

A development in the precision and effectiveness of intrusion detection in Internet of Things (IoT) networks may be accomplished by means of deep learning algorithms. The important purpose of the study is to create deep reading frameworks that surveil network vacationers' statistics on an ongoing basis in an effort to identify styles and anomalies indicating capability destiny assaults. To ensure high accuracy and reliability, those fashions might be professional on massive datasets comprising network traffic.

This takes a look at objectives offering pragmatic techniques for deep learning based totally on community intrusion visitor detection that would only result from carried-out paintings of interior IoT networks; such efforts may want to make substantial contributions to strengthening cybersecurity on IoT networks and gadgets.

## 1.3 Benefit of Study

Studying IoT protection and the usage of deep learning (DL)-primarily based on network intrusion visitor detection has several advantages, consisting of:

- More security for IoT devices and networks: When numerous IoT devices are increasingly being applied for diverse functions, it is crucial to establish their security to block unauthorized access and information breaches. The improvement of the safety of Internet of Things networks and gadgets may be facilitated by employing the development of short algorithms for deep learning primarily based network intrusion traffic detection, which would defend these systems from intruders.

- Network and IoT tool safety needs to be enhanced. As the variety of IoT devices is used for an increasing number of distinct purposes, it's vital to guarantee their protection to prevent unauthorized access and data leakage. Creating brief algorithms for DL-based community intrusion traffic detection may be one way through which researchers can also toughen the safety of Internet of Things networks and devices and, as a consequence, shield these structures from attackers.
- Improved threat intelligence: Deep learning research on IoT security may provide a good understanding of the kinds of attacks that are most often used to exploit weaknesses in IoT networks and devices. This may be used by security experts to create threat information that is more accurate and to design preventative measures against possible intrusions.
- Scalability: Scalability: When the quantity of IoT devices and networks goes up, the regular intrusion detection systems will struggle to handle the large volume of network data. It is feasible for researchers to detect network intrusions in IoT networks in more pragmatic and scalable ways by utilizing DL algorithms.

## 1.4 Proposed Study

The DL-IDS, as defined within the thesis "IOT Security by using Detection Of The Network Intrusion Traffic Using DL," employs separate deep learning methods, mainly lengthy quick-time period memory (LSTM) and multi-layer perceptions (MLPs), to precisely discover and categorize cyber-assaults geared toward IoT networks. The models are evaluated using of the KDDCUP dataset to check their ability to identify diverse forms of attacks and their average effectiveness. The findings indicate that the proposed disbursed system may be very powerful in detecting different types of cyberattacks, with an accuracy rate of up to 99.99% across a variety of configurations. The cautioned architecture for detecting assaults using the allotted deep learning has four fundamental steps: information treatment and pre-processing, education of the deep studying version, testing of the deep learning of the model, and detection of attacks. The technique incorporates many stages: statistics preprocessing and treatment, education and trying out deep learning to know the model, deploying a distributed framework, and the section on detecting and classifying assaults. Given the confined efficacy of conventional centralized intrusion detection systems (IDSs) in thwarting superior and previously unknown (zero-day) assaults, it's highly encouraged to install an allotted solution at the network perimeter to hit upon and mitigate cyber-attacks.

The suggested technique tackles the problem by using deep learning models specifically designed for supervised learning. These models excel at handling highly interconnected characteristics and can effectively solve the given issue.

## 1.5 Hypothesis

Using deep learning models to identify network intrusion traffic in IoT systems will increase security by efficiently detecting and mitigating prospective threats, enabling a proactive and flexible defensive mechanism.

Clarification:

The Internet of Things (IoT) networks' intrinsic complexity IoT networks, which consist of a whole lot of related devices with exceptional functionality, are by way of their very nature diverse and complicated. The continuously evolving characteristics of Internet of Things (IoT) eventualities may additionally offer problems for traditional intrusion detection structures to conform to them.

The growing complexity of infiltration procedures presents trouble for traditional rule-based intrusion detection systems, which can also struggle to keep up with rising attack styles. Deep learning fashions, appreciably neural networks, have the potential to autonomously acquire complex patterns and attributes from network visitor records.

The feature mastering capability of deep learning is characterised by its computerized extraction of hierarchical representations and functions from raw information. Through the analysis of ancient community traffic data, deep learning algorithms may probably discover intrusions effectively by finding minute patterns that point to both felony and criminal conduct. This approach provides an extra comprehensive and complex means of intrusion detection.

Deep learning systems can adapt to evolving hazard situations without having humans enter on an everyday foundation. Adaptability is essential in the Internet of Things environments where gadgets and conversation styles are converging rapidly.

Deep learning fashions have the capability to offer actual-time detection of anomalous patterns in community records while being appropriately deployed and tuned. As a result, the machine can react to attacks extra rapidly, decreasing its vulnerability and boosting its average capability to recover from disruptions.

Deep learning for intrusion detection may also result in fewer false positives than conventional techniques. More correct alarms result from deep learning fashions' capability to differentiate between acceptable-sized safety concerns and standard network pastime versions.

# 2
# RELATED WORKS

## 2.1 Intrusion Detection Research

Multiple studies have recorded the usage of machine learning (ML) in many domain names, along with object identification, model recognition, word processing, and photo processing. Deep Learning (DL) techniques are frequently utilized in protection-targeted studies [14]. In their work, the authors analyze the rise in information volume and the emergence of the "Smart City Internet of Things" as described in reference [15]. The authors of [16] elucidate the process of constructing CC and underscore the importance of big records within the formation of intelligent urban regions. He devised a way targeted at "large information" for the smart town. This approach tackles the difficulties of actual-time selection-making in clever towns [17]. It outlines numerous traits and components of smart towns so one can boom the first-class quality of life for people's lives. The authors of [18] provided a platform structure geared toward safeguarding smart towns from cyberattacks. To distinguish attackers based on the information they have acquired, the structure deploys a unique deep learning algorithm.

To enhance information safety, the authors of [19] used MLP and RF technology to discover the vulnerable factors of two widely used RPL-OF protocols. They additionally identified aggregate assaults on these protocols, together with rank and version, rank and blackhole, and decreased path. Concurrent attack. Consequently, they created a clean dataset for the Internet of Things (IoT) that relied on strength and network traits. Subsequently, these facts changed into implemented as an issue in the RPL Intrusion Detection System/Intrusion Prevention System (IDS/IPS) answer. Based on the idea of DL, the authors in [20] " They set up a " network intrusion detection machine" on the Fog node to detect attacks and identify them without previous expertise, The authors in [21] evolved “a novel method that mixes isolation forests, One Class Support Vector Machine (OCSVM), and an active learning mechanism”. A quick preprocessing or filtering method became paired with a variational auto-encoder that employed reconstruction possibilities via the authors of [22].

The authors of [23] used the ping of death technique to provoke a Distributed Denial of Service (DDoS) attack. This assault was then detected through the use of the RF set of rules and the WEKA tool, achieving a classification accuracy of 99.76%.

Several algorithms of machine learning [24,25] like Naive Bayes (NB), [26] ANN, and [27] fuzzy classification, SVM had been applied.

[28,29] Despite the fact that conventional machine learning methods have been frequently implemented for the detection of network attacks, these methods nevertheless require a great deal of preprocessing. These kinds of algorithms call for the use of attribute engineering and presume the supply of hand-crafted attributes [30].

However, the short development of technology has resulted in a rise in the amount of daily statistics data available to companies. As a consequence, the application of deep learning in trendy device studies involves acceptable human engagement in the compilation of data, this will not be ideal for handling actual-world situations [30,31] Additionally, those strategies are not a hit, but their drawback is a low degree of detection accuracy [31]. Recently, deep learning has gained popularity and appears to provide an answer to practical challenges within the real world. Because of its unheard-of capacity to automatically detect characteristics and correlations in large datasets [31]. Studies of deep learning algorithms for anomalous behaviour-based intrusion detection were undertaken by way of Aldweesh et al. [32]. This entails the classification of awesome IDS and the route of future study. Also, discover IDS from the perspective of deep learning strategies.

In the study undertaken by Vinayakumar and his colleagues, they advised the use of a completely practical hybrid DNN platform. They dubbed it Scale-Hybrid-IDS-AlertNet.5 " This version is hired to effectively surveil community traffic in real-time and offer early caution of risks [33]. It was built on a distributed deep learning model, which the authors verified on many different data sets, such as KDD'99 and NSL-KDD. On NSL-KDD, the optimal F-measure for multiclass classification was 76.5%, while the optimal F-measure for binary classification was 80.7% [33].

Tang et al. “came up with the idea of using a DNN model to spot irregularities in software-defined networking contexts. In practice, a straightforward DNN consisting of an input layer, three hidden layers, and an output layer was utilized”[34]. The NSL-KDD data set was utilized in the training that was carried out. “The model performed an accuracy value of 75.75% when applied to the binary classification problem”[34].

Yin and colleagues presented a technique that uses recurrent neural networks (RNNs) to identify intrusions [35]. When dealing with data that depends on time, RNNs perform very well. The model included both forward and backward propagation phases. Forward propagation is used to calculate the output values, while backpropagation, which passes the collected residuals, is used to maintain the weight current. They used SoftMax for the classification function and Sigmoid for the activation function. The results of the studies carried out using the NSL-KDD data set indicated that the accuracy values for binary and multiclass classification were 83.28% and 81.29%, respectively.

Deep learning and extra conventional device gaining knowledge of algorithms have been blended in the take a look at by using Shone et al.[36]. ”Unsupervised feature learning was done with the assistance of a nonsymmetric deep autoencoder (NDAE). Following that, an RF was applied for the project of classification. Two NDAEs have been stacked on top of one another, and each NDAE contained three discrete layers that were hidden from view”. The number of neurons and the number of capabilities are proportionally disbursed throughout every hidden layer. In order to evaluate the model, the KDD99 record set was utilized. The findings demonstrated a level of accuracy for the 5-class classification of 97.85 %.

Soman et al. [37] conducted studies to assess the performance of a number of one-of-a-kind deep learning algorithms inside the assignment of classifying the KDD'99 information set.

Dong and Wang carried out a “literary and experimental comparison between the utility of conventional NIDS techniques and deep learning techniques” [38]. They observed that deep studying became more effective than conventional strategies. The evaluation they made targeted the software of various techniques. The authors reached the conclusion that “the strategies that had been primarily based on deep learning presented improved detection accuracy throughout a spectrum of pattern sizes and various forms of traffic anomalies” [38]. This end was reached after the authors performed research.

The authors also “demonstrated that troubles associated with imbalanced datasets may be triumphed over by the use of oversampling, for which they used the Synthetic Minority

Oversampling Technique (SMOTE).” In addition, the authors demonstrated that oversampling may be used to triumph over issues related to imbalanced datasets. In addition, the authors confirmed that imbalanced datasets can be conquered by way of using oversampling as a technique of data correction.

In their presentation, Zhao et al. [39] furnished a survey of “the current condition of the work concerning packages of deep learning in gadget vigour monitoring.” They conducted a test wherein they as compared four commonplace deep learning methods to conventional system learning methods. These techniques were “auto-encoders, Restricted Boltzmann Machines (RBM), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN)”. The authors came to the conclusion, primarily based on their research, that deep learning techniques offer better accuracy than traditional strategies.

Deep4MalDroid is the name of the economic Android malware detection framework, this is defined within the paintings with the aid of Hou and colleagues [40]. Their method makes use of stacked auto-encoders, with the realization that three layers offer the best level of accuracy. It has been proven via the use of 10-fold pass validation that, in contrast to deep learning, their method gives advanced detection performance.

You and others [41] advise growing a device that could perform an automated security audit on brief messages (SMS). The RNN model serves as the foundation for his or her method. The authors asserted that their assessments produced an accuracy rate of 92.7%, which represented an improvement over formerly applied classification techniques (inclusive of SVM and Naive Bayes).

Throughout the route of our investigation into the pertinent literature, we came across a number of distinct deep learning knowledge strategies that were mainly proposed for NIDS.

Alrawashdeh and Purdy [42] advocated using an RBM that had one hidden layer on the way to “perform unsupervised characteristic reduction”. This is carried out with the intention of enhancing model accuracy.

The weights are then transferred to a specific RBM, so one can generate a DBN. The weights that have been pre-educated are then fed right into a first-class-tuning layer, which is made from a Logistic Regression classifier that has been skilled with 10 epochs and a multi-magnificence smooth-max. Using the dataset from the KDD Cup '99, the proposed answer was analysed and evaluated. The authors said that their detection rate turned to 97.90%, at the same time as the rate of fake negatives turned to only 2.47%. This is an improvement over the consequences that have been claimed by authors of other papers that were very similar to this one.

The goal of the studies finished by Kim and co-workers [43] changed to zero in on advanced continual threats. They suggest a Deep Neural Network (DNN) with 100 hidden layers combined with the Rectified Linear Unit activation function and the ADAM optimizer. This community might be used to make predictions. Their approach was evaluated by way of making use of the KDD information set, and it was applied on a GPU by means of making use of TensorFlow. The authors asserted that their fashions had an accuracy price of as much as 99% in common, and they concluded that RNN and Long Short-Term Memory (LSTM) models can be essential for the development of future defences.

Javaid et al. Came up with “the idea for a self-taught gaining knowledge of (STL) deep learning model for the purpose of detecting network intrusions” [44]. “The first part of the version turned into unsupervised feature learning, which concerned using a sparse autoencoder to obtain a function representation from a massive unlabeled record set. This part of the model got here first inside the version. The 2nd factor became a synthetic neural network classifier that made use of the SoftMax regression type” [44]. When the version was carried out to the NSL-KDD records set, it achieved accuracy values of 88.39% for the two-class category and 79.10% for the five-class classification, respectively.

Potluri and Diedrich [45] recommend a method that uses 41 functions, and their DNN consists of three hidden layers, broken up among two auto-encoders and one soft-max. The findings that were acquired have been contradictory; however, those who concentrated on a smaller, wider variety of classes performed with a higher degree of accuracy. The authors explained this result by mentioning that positive classes lacked satisfactory training data.

A technique to research models of normal network flows in an unsupervised way was proposed by Cordero et al.[46]. They utilise the RNN, vehicle-encoder, and dropout concepts, which can be related to deep learning. It isn't completely disclosed whether or now not their proposed approach has been correctly evaluated, but the accuracy was entirely high.

Also, Tran Nguyen et al. [47] followed the NDAE model, assessed its efficacy with the use of the KDD Cup '99 and the NSL-KDD datasets, and it obtained a high effect of achievement in terms of accuracy, precision, and recall with minimal training time. The contrast of stacked NDAE models with traditional DBN techniques. The version has been proven to boost accuracy by as much as 5% and limit learning time by as much as 98.81%.

It became proposed with the aid of Kang and Kang [48] to apply an unsupervised DBN to teach parameters to initialize the DNN, which resulted in progressed type outcomes (the precise details of the technique are uncertain). Their evaluation exhibits an improvement in overall performance in terms of the range of incorrect classifications. A whole taxonomy and survey on extraordinary NIDS processes that make use of deep and shallow learning were produced with the aid of Hodo et al. [49]. In addition to this, they've compiled a number of the most pertinent findings from the aforementioned works.

Additionally, there are extra works that are pertinent, such as the DDoS detection gadget that was proposed by Niyaz et al.[50]. They advise an allotted denial of service (DDoS) detection system for a software-defined network community (SDN). This is based totally on deep learning. The evaluation is done on the usage of traffic paths that have been in particular generated. The authors assert that they had been a hit in attaining an accuracy of a binary class of 99.82% and an accuracy of 8-class classification of 95.65%.

Wang et al. [51] present a technique for identifying malicious JavaScript in a laptop program. Their approach uses a 3-layer SdA coupled with linear regression. When compared to other classifier strategies, its evaluation discovered that it had the maximum increased true positive rate at the same time as the second satisfactory false positive rate.

Lee et al. [52] propose the use of deep learning knowledge as a technique for fault monitoring in the production of semiconductors. They provide an unsupervised learning answer by employing a Stacked de-noising auto-encoder (SdA) approach. A contrast with traditional techniques has confirmed that the method increased accuracy by means of “up to 14% throughout a lot of unique use cases.” In numerous one-of-a-kind use instances. They also came to the conclusion that the SdAs with four layers produced the most exceptional outcomes among those that had been analysed (one to four layers).

A Convolutional Neural Network (CNN) is used at the side of Long Short Term Memory (LSTM) to extract precise characteristic representations from statistics, after which they are classified. The suggested approach employs a deep learning model to pick out and classify threats in the IoT. The experimental evaluation used a dataset received from twenty Raspberry Pi-infected IoT devices. The empirical analysis achieved a 96% accuracy rate for assault detection. The proposed technique surpasses other formerly advised deep learning-primarily based assault detection strategies [53].

Alom et al. [54] “Create the primary Intrusion Detection System (IDS) through the use of Dynamic Bayesian Networks (DBN). The solution is evaluated by trying it out on the NSL-KDD dataset, which is normalized by the use of a numerical encoding method. During the classification section, the gadget achieves an accuracy of 97.5%, which is appreciably higher than the 40% accuracy of the training facts within the dataset.”

Omar M. et al. [55] carried out "deep learning techniques to increase deep neural networks (DNN) for intrusion detection systems. They proceeded with the version on the NSLKDD dataset. Of the 41 on-hand features within the NSL-KDD dataset, they best implemented 37 critical characteristics. This selection was primarily based on the method's potential to supply high levels of accuracy, recall, and precision at the same time as simultaneously lowering the training time necessary. The version gave correct consequences to each of the validation and check units. The values for F1 in both examples are 98.209%, and 98.426% respectively. 95% accuracy for all kinds of attacks.

Singhania and his colleagues [56] compared three advanced deep learning frameworks, namely Theano, TensorFlow, and Fast.Ai, in terms of their capacity to identify and categorize unique intrusions. They used a recently released dataset from 2018. Based on the findings, it was continuously noticed that speedy. Fast.AI carried out better than the other frameworks in all of the experiments, accomplishing accuracy results of up to 98.45% with extremely low charges of fake positives and false negatives, each much less than 1%. Moreover. Has attained an F1 rating of 0.99.

Wang et al. [57] chose “the MLP model for assessment. They use the DDoS attack detection preprocessed via select optimal features that perform the finest performance”.

Jullian et al. [58] used the DL-IDS system which employs DL models, especially the feed-ahead neural network (FFNN) and lengthy short-term memory (LSTM), to discover and categorize cyber-attacks in IoT networks. “The models are assessed using two distinct datasets, especially BoT-IoT and, NSL-KDD, to measure their implementation and capability to locate diverse varieties of attacks” [58]. The outcomes imply that the counseled disbursed system could be very efficient in identifying various styles of cyber-attacks, with an accuracy price of as much as 99.95% in unique configurations [58].

Otoum et al.[59] counseled techniques for “detecting attacks through the use of allotted deep learning knowledge, which includes four levels: data preprocessing and treatment, training and testing of deep learning models, deployment of the distributed framework, and detection and categorization of assaults” [59]. The architecture necessitates a decentralized approach at “the periphery for figuring out cyber-attacks, as conventional centralized Intrusion Detection Systems (IDSs) have shown to be insufficient in thwarting new and unknown (zero-day) attacks. The proposed system tackles this trouble by using deep learning methods that are specifically designed for supervised learning knowledge of and exhibit great overall performance when carried out to which highly correlated information” [59].

Another paper titled "Intrusion Detection Usage of Network Traffic Profiling and Machine Learning for IoT" proposes that “a dynamic and proactive intrusion detection system is carried out to profile and display all networked devices for the motive of detecting any efforts to tamper with IoT gadgets, as well as figuring out uncommon network transactions” [60]. The proposed solution utilizes network profiling and machine learning to secure IoT against Cyberattacks. Raw traffic is passed directly to the device learning classifier for examination and identification of possible attacks. The proposed “anomaly detection device grants promising effects with a basic accuracy of 98.35% and 0.98% of fake-fantastic alarms” [60].

This method frames the mission of identifying assaults as a piece of anomaly detection. The cause for that is that assaults typically appear very infrequently; however, after they do, their signature or, in particular, the distribution of the records is extensively one-of-a-kind from that in a state of affairs in which operations are normal. We can decide whether or not the incoming records have an enormously unique signature through the use of this approach.

As seen in Table 2.1, the consequences of our assessment of the pertinent literature have revealed that, despite the fact that high detection accuracies are currently being performed, there's nevertheless room for refinement. The dependence on human operators, the lengthy training times, the volatile average accuracy ranges, and the significant modification of datasets (as an example, balancing or profiling) are all examples of such weaknesses. The subject continues to be in its infancy, and most researchers are nevertheless carrying out experiments on how to combine exclusive algorithms (including education, optimization, activation, and classification) and layering procedures to generate the most correct and efficient solution possible for a given dataset.

**Table 2.1** Summary of existing surveys and papers related work to deep learning

| Method | Accuracy |
|---|---|
| Proposed MLP- Binary classification<br>Proposed MLP- Multiclass classification | 99.99352097511292%<br>99.99271035194397% |
| Proposed LSTM- Binary classification<br>Proposed LSTM- Multiclass classification | 99.83644485473633%<br>99.99756813049316% |
| Khamparia et al. [23] RF and WEKA tool | 99.76% |
| Vinayakumar et al. [33] DNN- Multiclass classification<br>Vinayakumar et al. [33] DNN- Binary classification | 76.5%<br>80.7% |
| Tang et al. [34] DNN- Binary classification | 75.75% |
| Yin et al. [35] RNN- Binary classification<br>Yin et al. [35] RNN- Multiclass classification | 83.28%<br>81.29% |
| Shone et al. [36] NDAE-5 Class classification | 97.85 % |
| L. You et al. [41] RNN | 92.7% |
| Alrawashdeh et al. [42] DBN | 97.90% |
| Kim et al. [43] LSTM | 99% |
| Javaid et al. [44] STL- Binary classification<br>Javaid et al. [44] STL- 5 Class classification | 88.39%<br>79.10% |
| [47] stacked NDAE | 98.81% |
| Niyaz et al. [50] SDN- Binary class<br>Niyaz et al. [50] SDN -8 Class classification | 99.82%<br>95.65%. |
| Lee et al. [52] SdA | up to 14% more than a lot of unique use cases |
| Sahu et al. [53] LSTM | 96% |
| Alom et al. [54] DBN | 97.5% |
| Almejarb et al. [55] DNN | 95% |
| Singhania et al. [56] Fast.Ai | 98.45% |
| Jullian et al. [58] LSTM | 99.95% |
| Rose et al. [60] network profiling | 98.35% |

# 3
# METHODOLOGY

## 3.1 Approaches to Solve Attack Detection

Their age alone over 70 years would qualify connectionist architectures as relics, but their recent rise to prominence in the artificial intelligence realm through innovative designs and GPU utilization places them at the cutting edge of AI research. Deep learning is not a singular methodology but rather a confluence of different topologies and techniques that make it possible to address a wide range of problems.

Far from being new, deep learning has received traction rapidly because of excessive layers coupled with GPUs. Large layered neural networks are rather efficient at computation considering they involve many straightforward multiplications. This efficiency has also led to the uptake of massive records, as deep studying is predicated on instance statistics for education in those networks with the aid of profitable them based totally on how nicely they educate; as it is constructed on the method of training neural networks through the use of example data, after which rewarding them depending on the effectiveness of their education.

Many exclusive structures and methodologies are carried out in deep learning. This phase offers six awesome deep learning knowledge of configurations that have been created over the last long time, as indicated in Figure 3.1. Deep learning frameworks are divided into two collections: supervised and unsupervised. Commonly deployed techniques consist of convolutional neural networks (CNN) and Long Short-Term Memory (LSTM). Those are the two oldest methods on this list.

Additionally, the self-organizing map (SOM), gated recurrent unit (GRU), autoencoder (AE), deep perception network (DBN), and restricted Boltzmann gadget (RBM) are also discussed)".

### 3.1.1 Supervised Deep Learning

Supervised learning is the term used in system learning to indicate a specific trouble location wherein the goal variable to be predicted is truly marked inside the training data.

#### 3.1.1.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a selected form of multilayer neural network or deep learning structure that takes cues from organic visual systems. Many computer-imaginative and natural language processing applications stand to benefit from using CNN.

Convolutional networks, which are generally called CNNs or ConvNets, are a particular type of deep neural network that may be deployed to identify images. It's a synthetic neural network that is both difficult and long-lasting. It's vital to understand that feedforward neural networks, once in a while termed multilayer belief (MLP), are the most popular model of deep learning. These kinds of models are referred to as "feedforward" models for the reason that information may additionally pass freely via the model. There are no feedback connections in the model; therefore, the output of the version isn't always returned to the model itself.

Yann LeCun is recognized as the originator of the first concept that led to the development of CNN. The recognition of handwritten characters played a central role in those days' architectural landscape, much like postal codes. An artificial neural network's deeper layers take information recombined at each stage and locate edges (simple features), then use these edges to reconstruct more complex properties located at higher levels of the network, while the first levels in turn find features within images. The presence of many layers is one characteristic of a deep network [61].

#### 3.1.1.2 Recurrent Neural Network (RNN)

Most deep learning blueprints are set up on a basic network framework called the recurrent neural network (RNN). Recurrent networks differ from ordinary multilayer networks due to the fact that, apart from having connections that transmit information and gradient data to higher layers, they also have connections going backwards or sideways. These forms of lateral and backward connections assist RNNs in remembering what they have seen in advance inside the sequence, a capacity useful for coding sequential statistics such as time series or natural language sentences [62].

**LSTM**

The Long Short-Term Memory (LSTM) came up with the concept of a reminiscence cell, which breaks away from the traditional neural network topologies built on neurons. However, it is feasible for the reminiscence cellular to maintain its value in reminiscence for an arbitrary length of time based on when it gets the inputs, therefore, allowing it to not forget what is essential as opposed to simply what was most recently computed [63].

**GRU Networks**

The Gated Recurrent Unit (GRU) was unveiled in 2014 as a more simplistic model of the LSTM. In contrast to the LSTM model, which consists of three gates, with an output gate being the third gate, this version has only two gates. The first gate is a reset gate, while the second is an update gate. The update gate makes a decision about what part of the previous cell state might be introduced to the current cell candidate, creating an up-to-date cell state. The way those are combined involves passing through a pointwise multiplication operation, which permits the gradient to drift freely. The update gate conveys information on the proportion of the preceding cell's content material that is meant to be stored. The reset gate governs how the new input is blended into the previous cell's content. A common Recurrent Neural Network (RNN) can be honestly represented by the use of a Gated Recurrent Unit (GRU), which has a value of 1 for the reset gate and a value of zero for the update gate [64].

#### 3.1.1.3 Deep Stacking Networks

DSN, which stands for the Deep Convex Network, might be the ultimate running platform. DSN differs from traditional deep learning frameworks since it contains several impartial networks that each have their own personal hidden layer in place of an unmarried deep network. The layout of the structure is supposed to remedy one of the challenges connected with deep learning: the complexity of the training method. Because the complexity of education rises with the addition of layers in a deep learning version, DSN alternatively approaches training as a series of troubles that are treated simultaneously [67].

### 3.1.2 Unsupervised Deep Learning

When the training facts do not consist of any goal labels, then the system gaining knowledge of the method is called unsupervised learning. This kind of difficult region is called a" problem area".

#### 3.1.2.1 Self-Organized Maps

The Kohonen map, also called the self-organized map (SOM), was introduced in 1982 with the aid of Dr. Teuvo Kohonen. It is an unmonitored neural network that helps in data classification by lowering the dimensionality of the input data set. Although there are some massive commonalities between a self-organizing map (SOM) and a traditional artificial neural network, they vary substantially [65].

#### 3.1.2.2 Autoencoders

In 1987, regardless of its misty dawn and early packages, LeCun unveiled that autoencoders had been used for the first time. The centre layer, hidden layer, and output layer (ANN) are the three different layers that constitute this specific kind of artificial neural community. Before being passed to the hidden layer from the input layer, it is first encoded. The number of nodes inside the hidden layer is much less than inside the input layer. It is inside this hidden layer where the compressed shape of the original data is held. The task of the output layer is to try replicating the input layer by using decoder operations [66].

#### Restricted Boltzmann Machines

The first RBMs were created by Paul Smolensky in 1986, and they were referred to as harmoniums at that point. Their creations no longer enticed an awful lot of attention. An RBM is a two-layer neural network with the input layer and hidden layer arranged one after the other; every node in the visible layer is attached to each node in the hidden layer, but there aren't any connections between nodes within a single layer. This differentiation was brought about because all nodes of both layers had been interconnected within the original Boltzmann machine, which ended up being too computationally expensive, as verified with the aid of restricted Boltzmann machines [67].

### 3.1.2.3 Deep Belief Networks

The DBN presents a common network architecture, but the manner of its training is completely uncommon. An RBM is created for each pair of layers that are connected in a DBN because the DBN is a system with many hidden levels and is, therefore, complex. A DBN can be seen as just a collection of RBMs.

It is more than this kind of representation. The sensory inputs are what the input layer of a DBN presents, while the buried layers develop abstract representations of data and the output layer has to categorize the network; thus all, these different roles imply that they operate differently from each other. Unsupervised pretraining comes first, and fine-tuning comes next, during which an instructor plays a directing role [67].

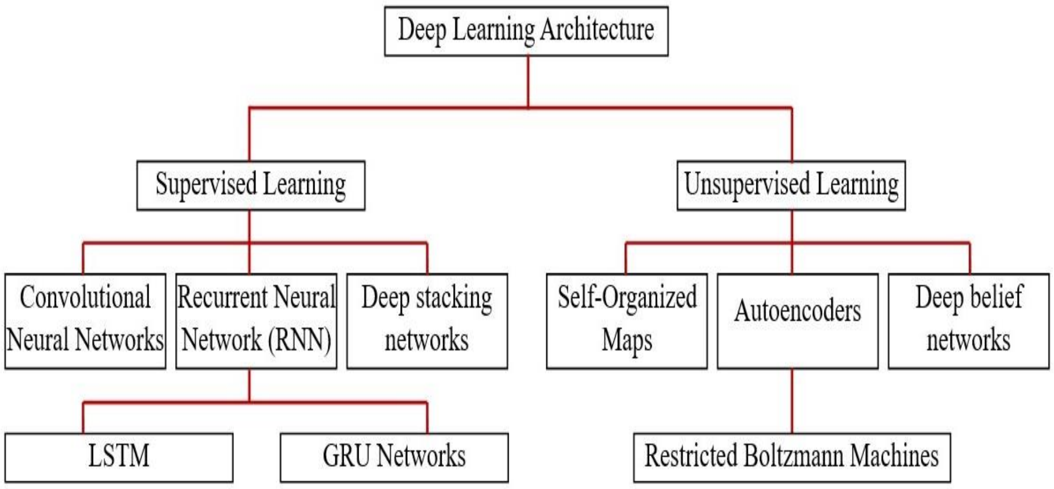


**Figure 3.1** Deep Learning Architecture

## 3.2 Detect Attacks using DL

As seen in Figure 3.2, in this part, the challenges related to attack detection are explored via the statistical categorization of measures and the use of DL. Classification is often achieved via the use of the supervised learning labelled data approach. In the course of our inquiry, we will be leveraging the RNN to acquire the best findings.

This approach recasts the challenging challenge of identifying assaults as the process of identifying anomalies. This is because assaults generally don't happen very frequently, but when they do, their signature or, more precisely, the distribution of datasets them apart from what one would see in an environment where operations are carried out regularly. This evolution of a signature is reflected in the data gathered over time. To be more precise, we use a model called a single hot encoder to discover the distribution of a given data set under typical circumstances. With this method, we may ascertain if the incoming data has an appreciably different signature.

Detecting assaults using Deep Learning (DL) includes utilizing the capabilities of neural networks to learn patterns and behaviours indicative of attacks from large datasets.

a) **Data Collectio** Gathering a labeled dataset of machine log or network traffic data that carries examples of both legitimate and malicious actions. The DL version makes use of this dataset as training records.  Broadly applicable datasets, KDD_99.

b) **Preprocessing Data:** To ensure data quality and remove noise, the collected data must be cleaned and preprocessed. To account for class imbalances, this may include standardizing, balancing, and removing redundant attributes from the dataset.

c) **Architecture Model:** Successful identification of attacks relies on choosing a strong DL architecture. Common designs say that LSTM networks are good because they are a type of RNN-based deep learning architecture and are effective for analyzing temporal patterns in data since they are presented one after another [68]. and MLP (MultiLayer Perceptron), which looks at all the parts of data simultaneously, the work by Abdullah et al.  [69]. Developed such a CNN-based model for DDoS attack detection in the IoT environment.

d) **Training of Model:** The deep Learning model may be trained via backpropagation of modifications to the version parameters by first sending preprocessed data via the chosen model architecture. This method allows the model to identify functions and styles that differentiate attacks from everyday behavior.

e) **Evaluation Model:** The skilled model's overall performance wishes to be assessed with a validation set. The area under the Receiver Operating Characteristic (ROC) curve, F1-score, Recall, accuracy, and precision are ordinary assessment metrics for DL-primarily based intrusion detection structures, other than F1-rating, which is an assessment metric that measures a check's accuracy in modeling the true values. Alazab et al.'s [70] work supplied performance metrics for the evaluation of a CNN-based model for detecting internet-based total attacks.

f) **Deployment of Model**: After demonstrating adequate performance, the DL model may be used in a production setting to keep an eye out for potential attacks in incoming data. Real-time monitoring and alerting would require integrating the DL model into an IDS or SIEM system.

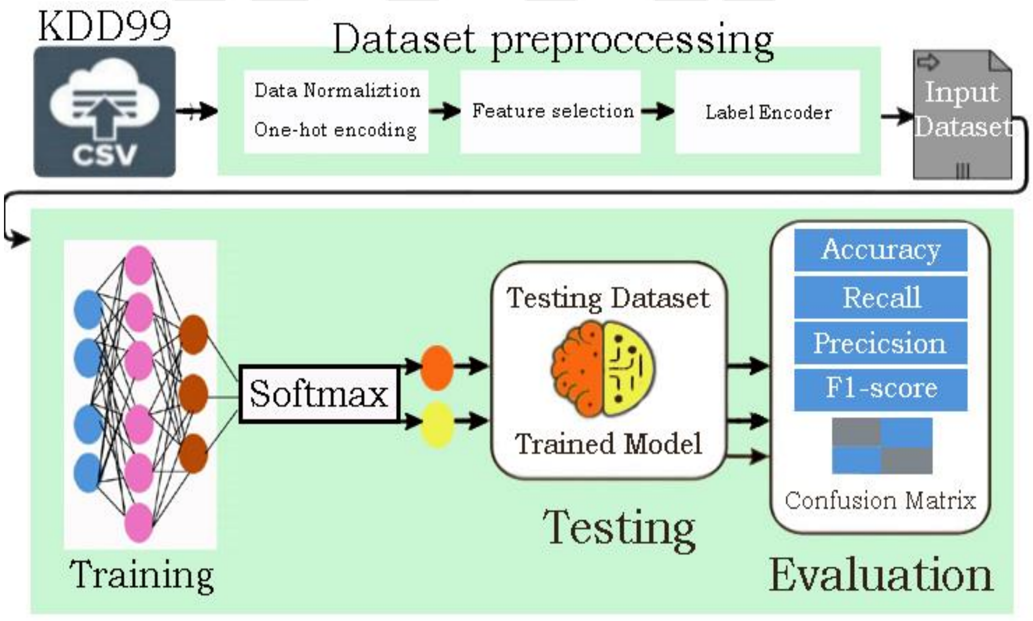


**Figure 3.2** Deep learning architecture of the proposed models

### 3.2.1 Multi-Layer Perceptron

MLP is a normal kind of artificial neural network; it is used for classification and regression-demanding situations and is called a feedforward neural network algorithm (MLP). It's made of numerous layers of neurons or network components, with each layer providing an output for the next layer's input.

The fundamental composition of MLP is composed of three main types of layers:

**Input Layer:** The input layer accepts traits that are entered, which are frequently termed feature vectors or characteristic representations, and they are transmitted to succeeding layers. The attributes of the input information are represented by every node in the input layer.

**Hidden Layers**: The layer that occurs between the input layer and the output layer is called a hidden layer. By activating a node, they are in charge of picking up intricate representations of the incoming data. To make the network capable of identifying nonlinearity in the information, every node in the hidden layers applies a nonlinear activation function to the weighted sum of its inputs. Additionally, the various hidden layer nodes ought to be correct, because a number of the hidden layer nodes contribute to the model now not functioning well with complex information.

**Output Layer:** he output layer includes the conclusion of the network. This is probably a numerical result for regression tasks or a projected class label for classification tasks. The output layer utilizes various activation functions compared to the hidden layer. For example, the output layer generally employs a Sigmoid function for binary category responsibilities and a SoftMax characteristic for multiclass class responsibilities.

Employ unique optimization techniques, like backpropagation. Backpropagation determines the slope of the loss function primarily based on the model's parameters and adjusts the parameters to minimize the disparity among the predicted and real outputs, weights, and biases of the neurons in the MLP via training. It acquired information using the procedure.

MLP is a versatile algorithm that can be used in numerous applications, including Speech recognition, natural language processing, photo classification, etc. With the right optimization of hyperparameters and cautious data management, packages may be simple and feature extremely good accuracy. However, without enough optimization and tweaking, overfitting and gradual convergence may additionally result.

A type of feedforward artificial neural network that might be entirely linked is called a multilayer perceptron, or MLP. The period "multilayer perceptron" (MLP) is now and then used inconsistently. Sometimes feedforward artificial neural networks (ANN), Sometimes are used more particularly to refer to networks that are comprised of multiple layers of perceptrons that have a threshold of activation [71].

An MLP can handiest expand with at least 3 node layers. Every node inside the network, besides the input nodes, acts as a neuron and makes use of a non-linear activation characteristic. Backpropagation, which is another term for the supervised learning technique used in MLP schooling, is based on the chain regulations of computerized differentiation reverse strategy [72][73][74][75][76][77] The reason MLP differs from a linear perceptron is because it has multiple layers and non-linear activation; this enables it to differentiate records that aren't linearly separable [78].

Each neuron in a multilayer perceptron undergoes the transformation of its input weights into its output value through the linear activation function. This property allows for the simplification of any quantity of layers right down to a tow-layer input-output model using techniques from linear algebra since the output value of every neuron is directly proportional to the inputs via a linear equation. The use of nonlinear activation features in MLPs is influenced by way of biological plausibility; actual neurons no longer surely summate their inputs, but, instead fire action possibilities based on them. Consequently, those features are chosen to resemble the firing frequency of real neurons. Sigmoidal activations include two well-known forms that have been used substantially:

$$y(v_i) = \tanh(v_i) \ \ and \ \ y(v_i) = (1 + e^{-v_i})^{-1} \tag{3.1}$$

The first is a -1 to 1 hyperbolic tangent, while the second is a logistic function, despite having almost the same structure. The logistic function's range is 0 to 1 instead of -1 to 1. In this case, $v_i$ represents the weighted total of the input connections to the node (neuron), while $y_i$ represents the output of the node. Reverser and Softplus are two more activation functions that have been suggested as workable alternatives. One kind of more complex activation function is the radial basis function. Radial basis networks are one family of supervised neural network models that make use of these functions.

In the field of deep learning rectified linear unit (ReLU) has garnered interest lately as a potential fix for numerical problems with the sigmoid.

Since MLP is completely integrated, each layer's node is coupled to the next layer's node with a certain weight $w_{ij}$.

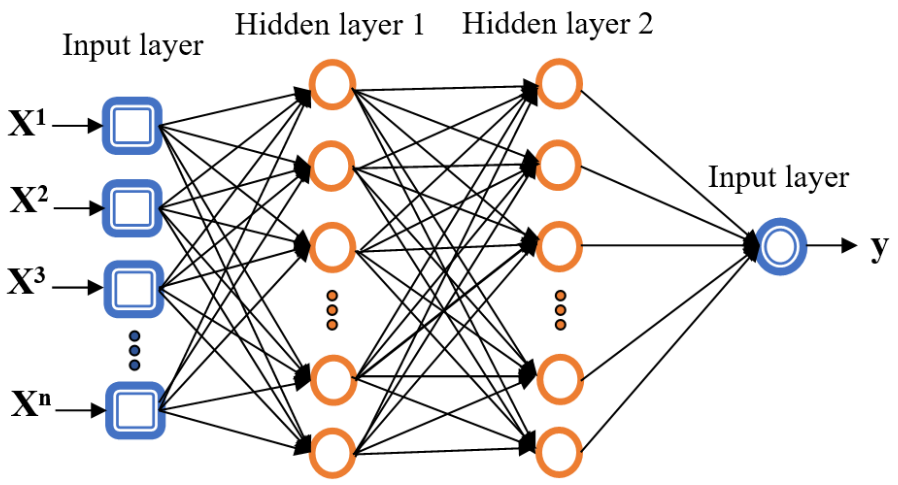


**Figure 3.3** MLP Architecture

In the deep layers of neural networks, there is a kind of activation function that is called the Rectified Linear Unit (ReLU) because it remains persistently in the layers. The expression of ReLU mathematically takes the form $\mathrm{f(x)} = \max{(0, \mathrm{x})}$, with x being the input to the neuron. In essence, what ReLU does is leave positive values unchanged and change negative input values into zeros [79].

Using ReLU as an activation function in an MLP helps the network learn complex representations and speeds up the training process, as it allows for the efficient backpropagation of gradients during the training phase. Expressing it mathematically:

$$y_{out} = \{ {}^{x, x \geq 0}_{0, x < 0} \qquad (3.2)$$

The dropout layer is a component employed in the development of neural networks to mitigate overfitting. During training, specific nodes are selectively omitted in different runs with a certain probability, simulating their absence as if they were not integral to the network structure [80].

The risk of overfitting, represented by connected layers that lead to decision-making, which is characterized by good performance on training data but failing in fresh data, creates the problem of overfitting. A possible solution is to include a dropout layer during the model architecture (see Figure 3.4). In this layer, both incoming and outgoing edges linked to deactivated components are removed; this is achieved by randomly deactivating some of the neurons along with their connections. Consequently, the network will have a more compact topology than it would otherwise have had; it should then be trained using input data. The weights of all nodes that were turned off at this stage should be brought back to their initial values after the session has been completed. By using dropout, overfitting is greatly reduced, improving the model's capacity to generalize and perform well on data that was not previously recognized. This method guarantees a more dependable and strong model [81].

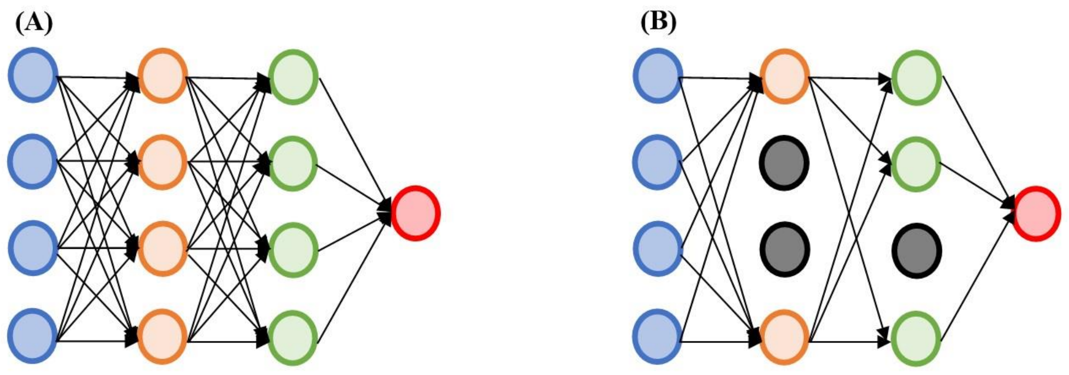


**Figure 3.4** (A) Simple Neural Network, (B) After Dropout, the Neural Network

### 3.2.1.1 Sigmoid Activation Function

The Sigmoid Activation Function is a mathematical function represented by its distinct "S" shaped curve. It is useful for use cases of logistic regression or building simple neural networks. The Sigmoid Function is the proper choice for executing a classifier when dealing with situations that have several correct answers. Use this function element-wise on raw output. Typically, the output of the Sigmoid Function falls within the range of 0 to 1 or -1 to 1 [82]. This is what the Sigmoid Function looks like:

$$f(x) = sigmoid(x) = \frac{1}{1+e^{-x}} \tag{3.3}$$

### 3.2.1.2 Softmax Activation Function

The intriguing Softmax Activation Function, also known as SoftArgMax or Normalized Exponential Function, operates on real number vectors as input and transforms them into a probability distribution where each component is computed as the exponential of the corresponding input element. It allows negative and non-integer inputs that might not sum up to one before transformation; all elements are scaled between 0 and 1 after applying Softmax, with the total sum equating exactly to 1, hence forming a valid probability mass function [82]. A more insightful description can be obtained when more numbers are included in the input set. The formula for the Softmax function is as follows:

$$softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} for\ j = 1, \dots, k \tag{3.4}$$

## 3.2.2 Recurrent Neural Network

A specific type of neural network is known as a recurrent neural network, abbreviated as RNN. These networks were created with the intention of dealing with sequential data. Unlike processing input in a straightforward manner as done by normal neural networks, RNNs have feedback loops that allow information to persist within the network; hence, RNNs turn out to be ideal for handling sequential data, including but not limited to time series, audio, and natural language, because of their inherent memory property that conventional neural networks lack.

A recurrent neural network (RNN) can store a hidden state, which allows it to remember information over previous time steps. This makes each time step not only produce an output but also have that output be determined by the previous hidden state and the input at that time step specifically. The inability of standard feedforward networks to recognize such connections in sequential data would make them unsuitable for this task; however, the network can do so now, thanks to RNNs.

RNNs are available in various structures. The most common are the vanilla RNN, LSTM network (long short-term memory), and GRU network (gated recurrent unit). LSTM and GRU networks are more advanced types of RNNs. They were designed to address the issue of vanishing gradients, which can occur during training deep neural networks.

Recurrent Neural Networks (RNNs) find their way into language translation, emotion analysis, voice recognition, and picture captioning, among other uses. In speech recognition and picture captioning ( two applications), they are used to translate audio signals into text or provide natural language descriptions of images, respectively. These applications involve digital data processing. In sentiment analysis, RNNs help determine whether a given text is being positively evaluated or not. In addition to being used for language translation between different languages based on situations where RNNs have been deployed, such details show how RNNs can be universally applied as an effective tool for processing sequential data.

One type of artificial neural network is recurrent neural networks (RNNs), where the node connections may form loops. This allows the output of some nodes to be fed back as input to the same nodes in future time steps within the network, resulting in dynamic temporal behaviour. RNNs differ from other network types in that they can process sequences of inputs of variable length since these sequences are entirely dependent on the state information that is kept within the network during processing, which flows through cycles [83] [84] [85]. As a consequence, applications such as connected cursive handwriting recognition or speech synthesis can be implemented with RNNs [86]. [87] [88] Being Turing complete entities, recurrent neural networks have no limitations regarding the amount and duration of computational tasks that could be executed since any program might be presented as a set sequence produced by some function on its inputs and storing internal results or sketching outputs for future use [89].

Convolutional neural networks are said to have a fixed impulse response, while recurrent neural networks have an infinite impulse response. Both types of networks exhibit temporal behaviour [90]. Recurrent networks can be thought of as including the feedback from their output in addition to the next input, with other terms more descriptive of that direct cycle use: additional stored states are used if more general dynamic information is included and kept for some time (even though part of it would decay), or more generally. A feedforward network cannot include such connections because this would create cycles, and they can be found in both long-term short-term memory networks (LSTMs) and gated recurrent units. A Feedback Neural Network (FNN) is another name for this type of network.

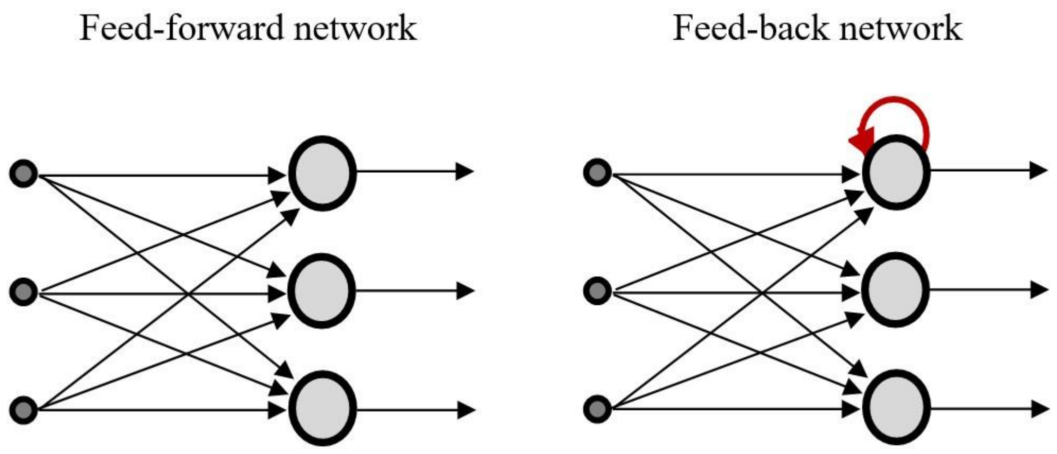


**Figure 3.5** Recurrent Neural Network.

### 3.2.2.1 Long Short-Term Memory

A type of structure found in recurrent neural networks, Long Short-Term Memory (LSTM), is widely used in deep learning applications such as time series prediction, language modeling, and speech recognition [91].LSTMs were introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997 as a more effective alternative to standard RNNs aiming to improve long-term memory capabilities and combat the vanishing gradient problem. Ever since their inception, LSTMs have outperformed themselves, with newer accuracy benchmarks set consistently across various domains [92].

At the core of LSTMs lies the concept of memory cells: entities capable of storing data for long durations while also deciding which information to retain or forget. An LSTM cell typically consists of three parts that help facilitate this functionality:

- **Input Gate:** A gate that determines the maximum data intake and processing for updating the memory cell. To figure out the blend between input data and past hidden states, LSTM utilizes a sigmoid activation function, which helps in deciding when is the best time for memory cell updates.
- **Forget Gate:** This gate decides on the removal of any unimportant information from the memory cell by controlling incoming input and hidden state through a sigmoidal activation function.
- **Output Gate:** This is what decides which part of the memory cell to reveal as its hidden state in the view of typical LSTM workings; it sums up the incoming input and

previous hidden state through sigmoid activation, sometimes also deciding when to send out stored data.

The LSTM cell does have a hidden state (which represents the output of the cell after passing through this gate) and a memory cell that stores updated data from input and forget gates. At each time step, these two entities are modified by LSTM so that long-term dependencies can be captured from the input sequence.

The problem of vanishing gradients in traditional RNNs is tackled by the design of LSTM, as it allows gradients to flow across the gates smoothly. LSTMs can deal with long sequences of data because they selectively update and forget information from the memory cell, which helps preserve important information over some time, thus making them efficient for modelling temporal interactions.

In general, Long Short-Term Memory (LSTM) networks are a very efficient kind of recurrent neural network often used for analyzing sequence data. Due to their remarkable capacity to capture long-term correlations, they are particularly well-suited for tasks that require modelling complex temporal patterns.

At the same time, LSTM models began to outperform traditional models in many applications related to speech, which include voice recognition, natural language translation, and video analysis. This led to a drastic change in the field of voice recognition.

[93]. The first RNN that won a pattern recognition competition was an LSTM network trained with Connectionist Temporal Classification (CTC); after winning several competitions, it became the first RNN to do connected handwriting recognition in 2009 [94]. In 2014, just using CTC-trained RNNs, Baidu was able to surpass the 2S09 Switchboard Hub5'00 benchmark from the voice recognition dataset without utilizing traditional techniques of speech processing [95].

The Google Android case involved using LSTM to better the synthesis of text-to-speech in addition to large-vocabulary voice recognition [87][88] and text-to-speech synthesis [96][94][97]Google's voice recognition, coming up in 2022, reportedly showed outstanding performance with an accuracy of 99.36% achieved through CTC-trained LSTM[98].

LSTM contributed significantly to language modeling and multilingual language processing and also led to improvements in machine translation down the line. The development of automated picture captioning saw major strides as a result of using Long Short-Term Memory (LSTM) and convolutional neural networks (CNNs) [99].

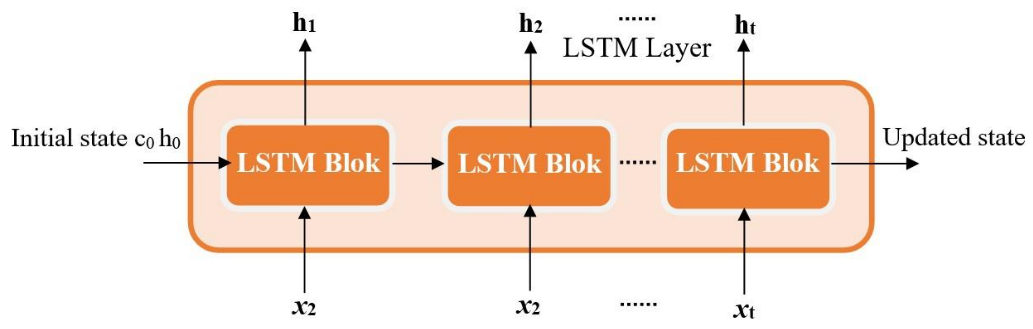


**Figure 3.6** LSTM Architecture

**LSTM Equations**

The inputs and outputs of an LSTM are depicted in Figure 3.7 below for a single timestep. This represents the input, output, and equations for a single timestep of a time-unrolled representation. The CNN output or the input sequence itself can both be used as the LSTM's input $x_t$ . The inputs from the prior timestep LSTM are $h_{t-1}$ and $c_{t-1}$. The LSTM's output for this timestep is $o_t$. Moreover, the LSTM generates the $c_t$ and$h_t$ for use by the subsequent time step LSTM [91].

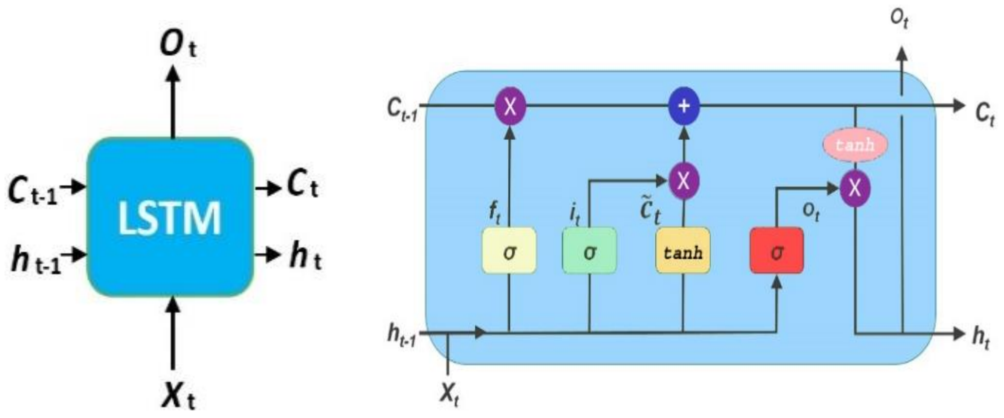


**Figure 3.7** LSTM input outputs and the corresponding equations for a single timestep

The LSTM cell has several equations that govern how it processes input and updates its internal state. The equations are as follows:

First, the input gate ($i_t$) equation controls how much new information is added to the memory cell. It takes the input at time step t ($\boldsymbol{x_t}$ )and the previous hidden state $h_{t-1}$ as inputs, and produces a value between 0 and 1 that determines how much of the input should be added to the memory cell $i_t$:

$$i_t = \sigma(W\_\{x_i\}\, x_t + W\_\{h_i\}\, h_{t-1} + b_i) \tag{3.5}$$

where $w_{xi}$, $w_{hi}$, and $b_i$ are learnable parameters, and σ is the sigmoid activation function.

Next, the forget gate equation ($f_t$) sets the proportion of the prior memory cell state $c_{t-1}$ that should be preserved. The output gate receives the same inputs as the input gate and generates a value between 0 and 1. This value defines the extent to which the prior cell state $f_t$ should be retained:

$$f_t = \sigma(W\_\{x_f\}\, x_t + W\_\{h_f\}\, h_{t-1} + b_f) \tag{3.6}$$

where
$W\_\{x_f\}$, $W\_\{h_f\}$, and $b_f$ are learnable parameters, and σ is the sigmoid activation function.

The next step is to update the cell state with the information from the input gate and forget gate. This is done by first creating a candidate value $C_t'$ that could be added to the cell state and then combining it with the previous cell state $c_{t-1}$ based on the forget gate and input gate:

$$C_t' = tanh(W\_\{x_g\} x_t + W\_\{h_g\}\, h_{t-1} + b_g) \tag{3.7}$$

$$c_t = f_t * c_{t-1} + i_t * C_t' \tag{3.8}$$

where $W\_\{x_g\}$, W_{ $h_g$}, and $b_g$ are learnable parameters, and tanh is the hyperbolic tangent activation function.

Finally, we have the output gate ($o_t$) equation, which determines how much of the updated cell state should be output at the current time step ($h_t$). It takes the input and previous hidden state as before, along with the updated cell state ($c_t$), and produces a value between 0 and 1 that determines how much of the updated cell state should be output ($o_t$):

$$o_t = \sigma(W\_\{x_o\} x_t + W\_\{h_o\}\, h_{t-1} + W\_\{c_o\}\, c_t + b_o) \tag{3.9}$$

where $W\_\{x_o\}$, $W\_\{h_o\}$ , $W\_\{c_o\}$, and $b_o$ are learnable parameters, and σ is the sigmoid activation function.

The final hidden state at time step t is then computed as:

$$h_t = o_t * tanh(c_t) \tag{3.10}$$

where $tanh$ is the hyperbolic tangent activation function.

# 4
# APPROACH OF DL ALGORITHMS

## 4.1 APPROACH OF ALGORITHMS

Applying machine learning to network intrusion detection involves constructing a model using historical network information to gather knowledge about ordinary network conduct. Subsequently, this skilled model is used to hit upon deviations from regular community patterns in actual-time community site visitors. This may additionally be a useful resource in the detection of viable protection vulnerabilities, consisting of malware or undesirable intrusion attempts.

Machine learning can be utilized in a whole lot of approaches to network intrusion detection. It is useful to employ supervised studying, wherein classified samples of each regular and aberrant community interest are used to educate the version. The model may also then be used to categorize new network site visitors as traditional or unusual based totally on how similar they are to the samples in the training dataset.

The objective of the use of gadget learning in network intrusion detection is to enhance the efficacy and precision of figuring out possible protection vulnerabilities and to facilitate safety groups indirectly and effectively addressing them.

One way to try may be to use the scikit-study bundle for Python, which has quite a few algorithms for supervised and unsupervised mastering. These methods can be used to educate a model about the usage of historical community facts. The trained version may additionally be used to classify clean network visitors as both extraordinary and regular.

The following are the basic steps involved in the use of machine learning to gain knowledge of community intrusion detection:

1. Gather and prepare community records: This involves accumulating statistics from numerous resources, along with community site visitor facts, and then organizing and exceptional-tuning them for you to put together for schooling gadget mastering models. This would possibly consist of recognizing and classifying examples of regular and bizarre conduct.

2. Train a machine learning model: This includes the use of supervised learning to teach a model on the accumulated and preprocessed community data. Typically, this method requires splitting the data into training and test sets. The developments and patterns of traditional community hobbies are then determined by the usage of the training information.

3. Assess the running prototype: Test records ought to be applied after education for you to affirm the accuracy and capability of the model. This will contribute to ensuring that the version can correctly perceive both regular and aberrant community events.

4. To discover network breaches in real-time, use the following version: Once trained and assessed, the version may be used in real-time to classify new community traffic and detect any safety flaws. To immediately notify protection specialists of any threats, this could include configuring indicators or notifications or easily incorporating the model into protection gear and systems that are already in use.

The number one goal of using system learning in community intrusion detection is to enhance the accuracy and efficacy of figuring out viable safety dangers and to empower security groups to reply directly and successfully.

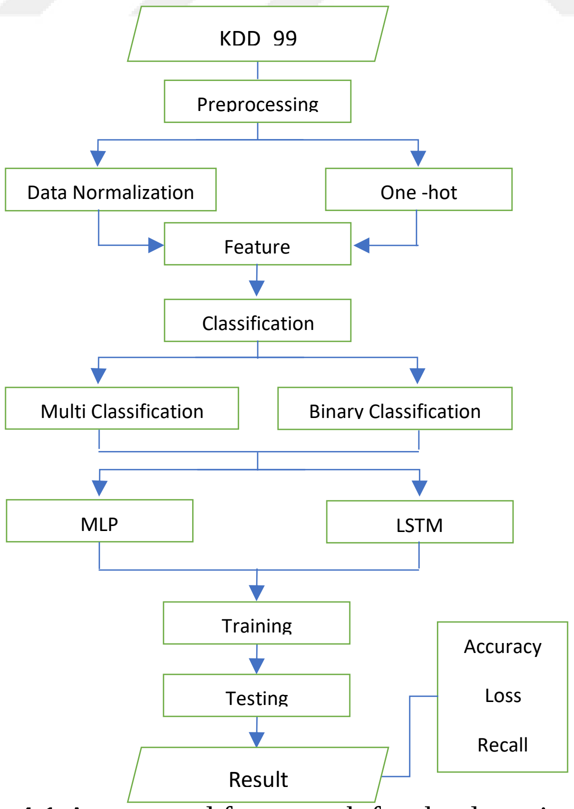


**Figure 4.1** A proposed framework for the detection of attacks

## 4.2 Data Normalization

The mixed integer and floating point values, which varied from 0 to 1, made learning challenging, as in Tables 4.1 and 4.2. We used the min-max normalization strategy to fix the issue by converting the numerical attribute values $K_{ij}$ into the range of 0 to 1. This method aims to improve training consistency and model performance by aligning features in the following ways:

$$\widetilde{K}_{fj} = \frac{K_{fj} - \min(K_f)}{\max(K_f) - \min(K_f)} \tag{4.1}$$

The max and min values of the numeric attribute $K_f$ denote a $\max(K_f)$ and $\mathrm{m}\,(K_f)$ respectively, while $\widetilde{K}_{fj}$ represents the normalized feature value within the range of 0 to 1.

**Table 4.1** Data before normalization

| | duration | Protocol_type | service | flag | Src_bytes | Dst_bytes | land | Wrong_fragment | urgent | hot | … | dst_host_sry_count | dst_host_same_sry_rate | dst_host_same_sry_rate | dst_host_same_sry_rate | Label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | tcp | http | SF | 181 | 5450 | 0 | 0 | 0 | 0 | … | 9 | 1.0 | 0.0 | 0.0 | Normal |
| 1 | 0 | tcp | http | SF | 239 | 486 | 0 | 0 | 0 | 0 | … | 19 | 1.0 | 0.0 | 0.0 | |
| 2 | 0 | tcp | http | SF | 235 | 1337 | 0 | 0 | 0 | 0 | … | 29 | 1.0 | 0.0 | 0.0 | |
| 3 | 0 | tcp | http | SF | 219 | 1337 | 0 | 0 | 0 | 0 | … | 39 | 1.0 | 0.0 | 0.0 | |
| 4 | 0 | tcp | http | SF | 217 | 2032 | 0 | 0 | 0 | 0 | … | 49 | 1.0 | 0.0 | 0.0 | |
| 5 rows x 42 columns | | | | | | | | | | | | | | | | |

**Table 4.2** Data after normalisation

| | duration | Protocol_ty | service | flag | Src_bytes | Dst_bytes | land | Wrong_fragment | urgent | hot | dst_host_sry_count | dst_host_same_sry_rate | dst_host_srv_diff _host rate | label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.067792 | tcp | http | SF | -0.002879 | 0.138664 | 0.006673 | -0.04772 | -0.002571 | -0.044136 | -1.694322 | 0.599394 | -0.158629 | Normal |
| 1 | -0.067792 | tcp | http | SF | -0.002820 | 0.138664 | 0.006673 | -0.04772 | -0.002571 | -0.044136 | -1.600018 | 0.599394 | -0.158629 | |
| 2 | -0.067792 | tcp | http | SF | -0.002824 | 0.138664 | 0.006673 | -0.04772 | -0.002571 | -0.044136 | -1.505714 | 0.599394 | -0.158629 | |
| 3 | -0.067792 | tcp | http | SF | -0.002840 | 0.138664 | 0.006673 | -0.04772 | -0.002571 | -0.044136 | -1.411410 | 0.599394 | -0.158629 | |
| 4 | -0.067792 | tcp | http | SF | -0.002842 | 0.138664 | 0.006673 | -0.04772 | -0.002571 | -0.044136 | -1.317106 | 0.599394 | -0.158629 | |
| 5 rows x 42 columns | | | | | | | | | | | | | | |

## 4.3 One-Hot Encoding

One technique for converting categorical data into a numerical representation appropriate for mathematical calculations is one-hot encoding, which is used in data analysis and machine learning [100].

A technique called "one-hot encoding" encodes every category as a binary vector. The number of categories in the variable equals the length of the vector. The vector's members each stand for a distinct category; the first element is "hot" (value = 1), while the remaining elements are "cold" (value = 0). The encoded data item's specific category is shown by the hot element, while the other categories are shown by the cold elements. The benefit of one-hot encoding is that it makes it possible to include categorical data in machine-learning models that were solely intended to process numerical input. We may employ categorical data in mathematical operations like regression analysis and computing the separations between data points by providing it with a numerical representation. The one-hot-encoding technique was used to translate the three category features of protocol type, service, and flag into numerical values. to choose the features of the dataset that depended on it for classification, we use (TCP, UDP, and ICMP) which are the three main protocol types of packet data. As a result, the packet data is divided into normal and attack categories using these three protocols(protocol type) as well as the service categories and their flag state. The protocol uses flags, which is a complex handshake procedure to confirm that a trustworthy connection has been established between the communication users. As seen in the accompanying table.

**Table 4.3** Categorical features transform into numeric values

| Index | Protocol_type | Service | Flag |
|---|---|---|---|
| 0 | Tcp | HTTP | SF |
| 1 | Tcp | HTTP | SF |
| 2 | Tcp | HTTP | SF |
| 3 | Tcp | HTTP | SF |
| 4 | Tcp | HTTP | SF |

**Table 4.4** Applying one-hot encoding to convert categorical attributes to numerical

| | protocol_type_icmp | protocol_type_tcp | protocol_type_udp | service_IRC | service_X11 | service_Z39_50 | service_auth | service_bgp | service_courier | service_csnte_ns | … | flag_REJ | flag_RSTO | flag_RSTOS0 | flag_RSTR | flag_S0 | flag_S1 | flag_S2 | flag_S3 | flag_SF | flag_SH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 5 rows x 80 columns | | | | | | | | | | | | | | | | | | | | | |

## 4.4 Binary Classification

One of the machine learning assignments is binary classification, which involves predicting one of two potential outputs using a given set of input attributes. The two possible results are often known as the "positive" and "negative" categories, or as "class 1" and "class 0" [101].

For example, within the vicinity of clinical diagnostics, it can be useful to expect, based totally on an affected person's signs and check findings, whether or not or no longer the affected person has a positive illness. In this case, the negative elegance would represent people who are unaffected by the illness, at the same time as the nice magnificence would represent people who are.

Typically, step one in fixing a binary class hassle is to bring together a classified dataset that carries occasions with both advantageous and negative results. A gadget studying a set of rules will then gain knowledge of the prior dataset to create a version that could effectively classify new fashions.

Support vector machines, choice trees, random forests, and logistic regression are a few of the techniques used in binary classification. Furthermore, binary type is an important use of deep learning. The selection of the algorithm is contingent upon the characteristics of the data and the particular issue being addressed. As seen in Figure 4.2

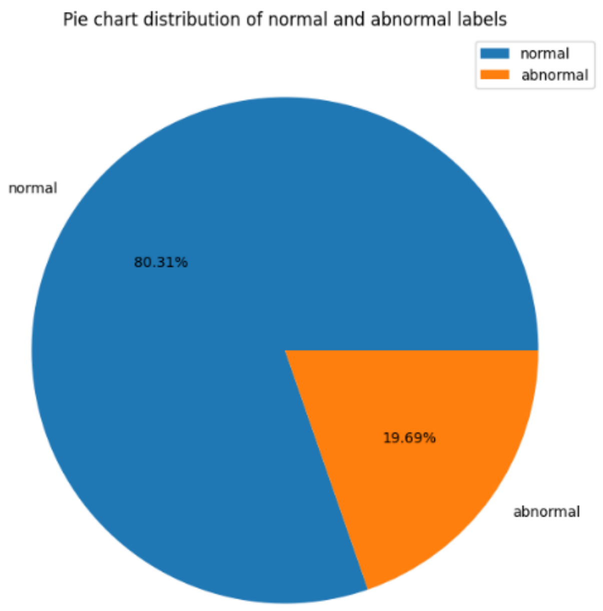


**Figure 4.2** Binary- Classification

### 4.4.1 Binary Classification with MLP Classifier

For binary type, a specific sort of neural network version known as a multi-layer perceptron (MLP) can be used. A Multi-Layer Perceptron (MLP), a type of neural community, includes an input layer, an output layer, and one or more hidden layers of artificial neurons.

To use a multilayer perceptron (MLP) for binary type, one must first get a labeled dataset that includes examples of each tremendous and poor eventuality. We then use this dataset to teach a Multilayer Perceptron (MLP) for you to construct a model that efficiently classifies new cases.

During the training process, the Multilayer Perceptron (MLP) acquires the capacity to alter the weights and biases of the artificial neurons in every layer. This update takes into account the input data and the intended output. The purpose is to limit a loss characteristic that quantifies the discrepancy between the anticipated and real outputs of the MLP.

After the training segment is over for the binary class, we may also utilize a Multilayer Perceptron (MLP) to predict the class of new instances. An estimate of the fantastic elegance's chance is frequently the MLP's output. To flip this probability estimate right into a binary prediction, we can also follow a threshold. We forecast the negative class if the probability estimate is beneath the threshold, and we anticipate the positive class otherwise.

A Multilayer Perceptron's (MLP) performance for the binary classes is often assessed using metrics including accuracy, F1 score, don't forget precision, and recall. The metrics compare the performance of the MLP on the check dataset, which consists of samples that are not employed within the training segment.

An MLP's ability to establish complex, non-linear connections between the class designation and the enter capabilities is a plus on the subject of binary classification. The MLP's success is, however, dependent on the selection of hyperparameters, which include the range of hidden layers, the variety of neurons in every layer, and the learning ratio. These hyperparameters can be modified by using strategies like grid seek or random search to enhance the MLP's overall performance on the test dataset.

### 4.4.2 Binary Classification with LSTM Classifier

To use an LSTM for the binary categories, you normally need initially a categorized dataset that consists of examples of both positive and negative conditions. Next, we educate an LSTM model using the formerly said dataset to get the functionality effectively classifying new examples.

During the training segment, the LSTM set of rules learns to gradually system the incoming entry flow, altering its internal nation at each step. The LSTM creates a hidden state with the resource of the present-day input and the previous inputs in the sequence.

After reading the whole sequence, the final hidden state of the LSTM is acquired with the aid of a feedforward neural network, which transforms it into a binary category result. The result regularly yields an estimate of the probability of the positive category.

Once the training section is complete, it makes predictions on the class of fresh instances using the LSTM version for binary classification. The input series feeds to the LSTM incrementally, updating the internal state at each time step. Using the feedforward neural network's output, we can generate a probability estimate for the positive class after the collection processing is finished. We might also then flip this chance estimate into a binary prediction by adding a threshold.

Measures like accuracy, F1 score, recollect, and precision are regularly used when inspecting the utility of an LSTM version for binary categories. Metrics are used to assess the overall performance of the LSTM model on the looked at the dataset, which contains examples that have not been used within the training segment.

An LSTM can acquire and recognize long-time period dependencies within the input collection and research in-depth details about the nonlinear correlations among the category label and the entered information, which can be useful in binary classification problems. Nonetheless, the hyperparameters that affect the LSTM's effectiveness consist of the wide variety of LSTM layers, the wide variety of neurons in every layer, and the learning rate. To improve the LSTM's performance on the test dataset, those hyperparameters can be improved by using strategies like random or grid seek.

## 4.5 Multi-Class Classification

Multi-class class is a device for gaining knowledge, where the objective is to predict one of several possible results based on input characteristics. Multi-class categorization is the term used when there are more than two potential effects, and every final result is typically represented via a special class label [102].

To distinguish pictures, it is probably beneficial, for example, to categorize a photo into many categories, such as "dog," "chook," or "cat." Here, every class is connected to a unique special label; the goal is to determine which label best appropriately describes the photo.

The manner of resolving multi-class category trouble frequently begins with the compilation of a dataset including classified examples for every capability class. Then, using the formerly given dataset, we train a machine learning algorithm to construct a model that can effectively categorize new instances.

It will be used as a multi-class classification version to appropriately become aware of the class of fresh instances after the training section is over. An estimate of the likelihood for each class label is typically the result of a multi-class classification model. We may identify the class with the highest probability as the expected class for the given input instance.

Multi-class class fashions are regularly assessed using the F1 rating, accuracy, keep in mind, and precision as key performance indicators. These metrics compare the version's overall performance on the test dataset, which includes samples that were not included in the training segment after applying the version. The aim is to create a model that demonstrates exceptional performance on unlabeled information and can accurately classify input instances into their respective class labels.

### 4.5.1 Multi-Class Classification using MLP Classifier

One type of neural community model that may be used to solve problems like multi-class categorization is the multi-layer perceptron (MLP). A Multi-Layer Perceptron is fabricated from a neural community's feedforward production (MLP). It is made up of one or more hidden layers constructed of synthetic neurons, an input layer, and an output layer.

When using Multi-Layer Perceptron (MLP) for multi-class classification, the first step is to get a categorized dataset that consists of samples for every class. Next, we use this dataset to educate a Multilayer Perceptron (MLP) with the intention of creating a model that may, as it should, classify novel instances.

Throughout the training segment, the Multilayer Perceptron (MLP) modifies the weights and biases of its artificial neurons in every layer based on the input information and the favoured output. The goal is to reduce a loss function that quantifies the difference between the predicted output and the actual output of the MLP.

When the MLP for multi-class classification has completed its training segment, we may additionally use a Multi-Layer Perceptron (MLP) to expect the class of fresh instances. The MLP generally offers a chance estimate for every class label as an output. The projected class for the given input instance might also then be decided based on which class has the best possibility.

The performance of a Multi-Layer Perceptron (MLP) in multi-class classification tasks is often evaluated using metrics that include accuracy, F1 score, recall and precision. These metrics examine the MLP's performance when it comes to checking dataset samples that were not used within the training segment.

The capacity of an MLP to set up complicated, non-linear correlations between the entered information and the class designation is one benefit of using multi-class classification.

The hyperparameters that affect the MLP's performance include the range of hidden layers, the variety of neurons in each layer, and the learning rate. Grid seek and random search strategies can be used to adjust those hyperparameters, to enhance the MLP's overall performance on the check dataset. Regularization techniques will also be used to reduce overfitting and improve the generalization capability of the model.When the MLP for multi-class classification has completed its training segment, we may additionally use a Multi-Layer Perceptron (MLP) to expect the class of fresh instances. The MLP generally offers a chance estimate for every class label as an output. The projected class for the given input instance might also then be decided on based on which class has the finest possibility.

The performance of a Multi-Layer Perceptron (MLP) in multi-class classification tasks is often evaluated the usage of metrics which include accuracy, F1 score, take into account, and precision. These metrics examine the MLP's performance while it comes to check dataset samples that were not used within the training segment.

The capacity of an MLP to set up complicated, non-linear correlations between the enter information and the class designation is one gain of using in multi-class classification. The hyperparameters that have an effect on the MLP's performance include the range of hidden layers, the variety of neurons in each layer, and the learning rate. Grid seek and random search strategies can be used to adjust those hyperparameters, in an effort to enhance the MLP's overall performance at the check dataset. Regularization techniques will also be used to reduce overfitting and improve the generalization capability of the model.

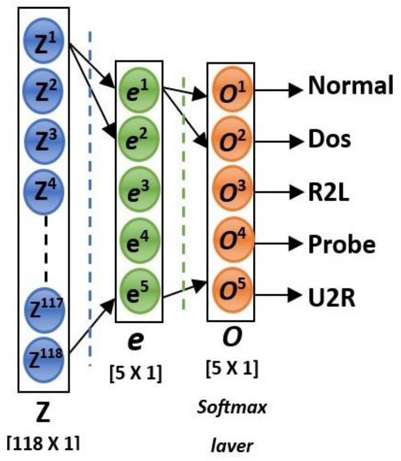


**Figure 4.3** MLP architecture. The architecture has a single hidden layer with 5 units.

### 4.5.2 Multi-Class Classification using LSTM Classifier

Using an LSTM for multi-class classification frequently begins with acquiring a categorized dataset along with samples from numerous classes. Next, we train an LSTM (Long Short-Term Memory) model using the dataset noted earlier. The goal of this method is to come to be properly categorized new instances into the relevant classes.

During the training section, the Long Short-Term Memory (LSTM) version carefully evaluates the incoming sequence of information and modifies its internal nation at every step. Each time a generation takes a region, the LSTM creates a hidden state using records from the inputs that came earlier than and from the current entry.

After the complete collection is analysed, the LSTM's very last hidden state is sent right into a feedforward neural network, which transforms it into a multi-class output. The final results are often a vector of possibilities, in which each likelihood is related to a certain class.

After training, a class of new instances will be expected using an LSTM version for multi-class classification. The LSTM can manner the input series sequentially, updating its inner state at each step. The feedforward neural community generates an output when sequence processing is finished, which can be used to build a vector of chances, each of which is connected to a certain class. It might also, therefore, be decided that the category with the highest opportunity is the one that is anticipated.

An LSTM is frequently assessed for multi-class categories with the use of measures consisting of accuracy, F1 rating, and, precision and recall. Metrics are used to evaluate the overall performance of the LSTM model on the check dataset, which includes examples that were excluded from the training set.

LSTMs are great for multi-elegance classification because they can detect lengthy-time period dependencies inside the input collection and set up complex, nonlinear connections among the entered records and the class labels. However, the quantity of LSTM layers, the range of neurons in each layer, and the learning ratio are a few of the hyperparameters that define the LSTM's overall performance. To enhance the LSTM's overall performance by looking at a dataset, these hyperparameters can be changed using grid search and random search strategies.

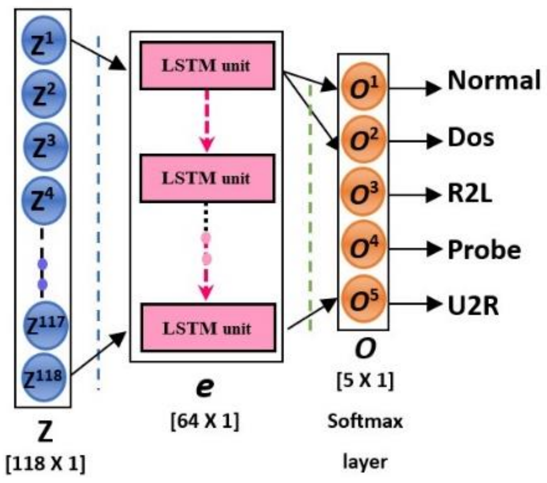


**Figure 4.4** LSTM Classifier under consideration. The model has a single hidden layer with 64 units.

## 4.6 Compiled Model

After processing the dataset and categorizing it into binary and multiclass classifications, we will train the model that we will use in the classification of the two models:

### 4.6.1 MLP Compiled Model

As Figures 4.3 and 4.5 illustrate, it is comprised of one input layer, five hidden layers, and the activation "ReLU". Also, the "Dense" has 118 inputs when it first appears.

The output was sent to a dense, fully linked layer that included two or five neurons in binary or multiclassification, using "sigmoid" or "SoftMax" activation functions, respectively. The dropout layer has a dropout rate of 0.2 to minimize overfitting.

"Binary crossentropy" and "categorical crossentropy" from binary class and multiclassification, respectively, as well as the metric chosen the "accuracy," obtained by the "ADAM" (adaptive moment estimation) optimizer, were used to train the MLP model.

In the case of two categories, the definite must make the loss "binary crossentropy," while in the case of more than two categories, the definite must make the loss "categorical crossentropy."

We applied the model two times. The input dataset was first divided into training and testing subsets of 80% and 20%, respectively. After that, it is divided into training and testing portions of 75% and 25%, respectively, using test sizes of (0.2) and (0.25) and random state (42). We used the model twice in this way. Training with 500 batch sizes, 20 epochs, and 0.2 validation rates is the next step. These requirements were developed after thorough testing conducted in real-world scenarios.

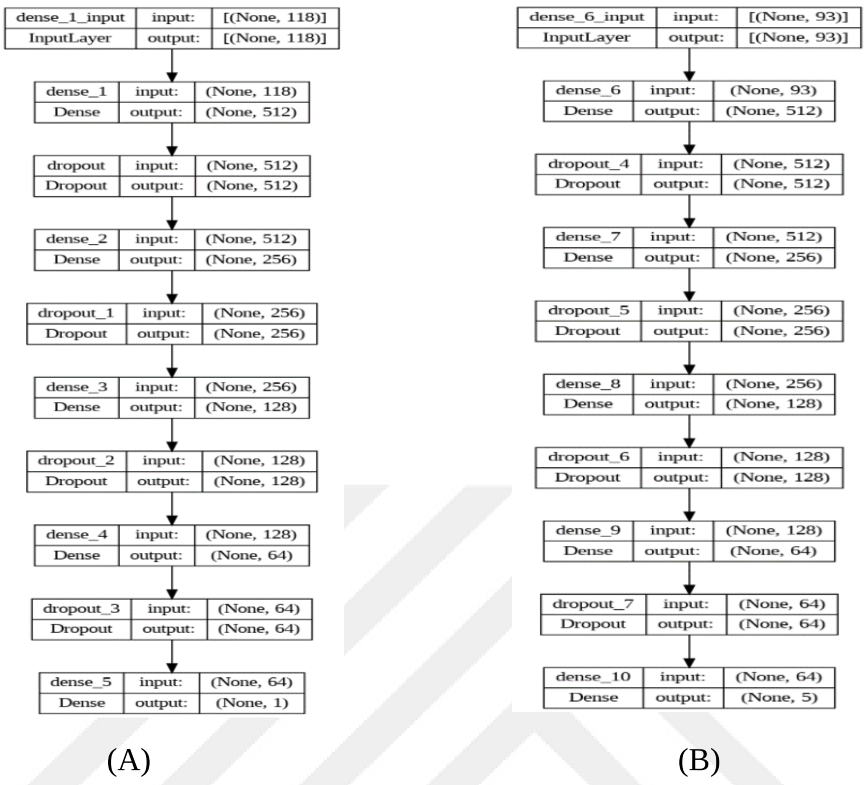

dense_1_input | input: | [(None, 118)]
InputLayer | output: | [(None, 118)]
dense_1 | input: | (None, 118)
Dense | output: | (None, 512)
dropout | input: | (None, 512)
Dropout | output: | (None, 512)
dense_2 | input: | (None, 512)
Dense | output: | (None, 256)
dropout_1 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_3 | input: | (None, 256)
Dense | output: | (None, 128)
dropout_2 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_4 | input: | (None, 128)
Dense | output: | (None, 64)
dropout_3 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_5 | input: | (None, 64)
Dense | output: | (None, 1)
(A)
dense_6_input | input: | [(None, 93)]
InputLayer | output: | [(None, 93)]
dense_6 | input: | (None, 93)
Dense | output: | (None, 512)
dropout_4 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_7 | input: | (None, 512)
Dense | output: | (None, 256)
dropout_5 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_8 | input: | (None, 256)
Dense | output: | (None, 128)
dropout_6 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_9 | input: | (None, 128)
Dense | output: | (None, 64)
dropout_7 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_10 | input: | (None, 64)
Dense | output: | (None, 5)
(B)

**Figure 4.5** Architecture MLP Classifier (A) Binary Classification, (B) Multi-class Classification

### 4.6.2 LSTM Compiled Model

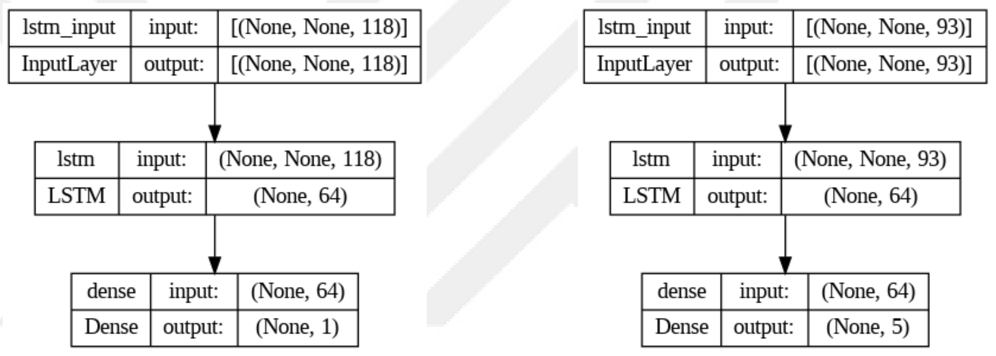
Figure 4.4 and Figure 4.6 depict a neural network architecture with an input layer, a single LSTM layer, and an output layer. To guarantee fairness in comparison, the LSTM layer used 64 neurons for storing the input information, which corresponds to the number of memory units or cells in the LSTM layer. Additionally, the input dimensions were set to 118 for binary classification and 93 for multiclassification.

The result was fed into a densely linked layer with either 2 or 5 neurons, using "sigmoid" and "SoftMax" activation functions for binary and multiclass classification, respectively. The dropout layer is implemented using a dropout rate of 0.2 in order to mitigate the problem of overfitting.

The LSTM model was trained using the "ADAM" optimizer, which is an adaptive moment estimation optimizer. The loss function used was "binary crossentropy" for binary classification and "categorical crossentropy" for multiclassification. The measure used to evaluate the model's performance was "accuracy".

When dealing with two categorical variables, it is necessary to use the "binary crossentropy" loss function. Simultaneously, the loss function known as "categorical crossentropy" is used when dealing with more than two categories.

Additionally, we used the model twice. Initially, the input dataset was divided into 80% for training and 20% for testing. Then, it was further divided into 75% for training and 25% for testing, with test sizes of 0.2 and 0.25 correspondingly, and a random state of 42. Next, the training will be conducted using 20 epochs, a batch size of 500, and a validation rate of 0.2. The selection of these values was based on an empirical investigation carried out via a series of diverse tests.


lstm_input | input: | [(None, None, 118)]
InputLayer | output: | [(None, None, 118)]
lstm | input: | (None, None, 118)
LSTM | output: | (None, 64)
dense | input: | (None, 64)
Dense | output: | (None, 1)
lstm_input | input: | [(None, None, 93)]
InputLayer | output: | [(None, None, 93)]
lstm | input: | (None, None, 93)
LSTM | output: | (None, 64)
dense | input: | (None, 64)
Dense | output: | (None, 5)

(A) (B)

**Figure 4.6** Architecture LSTM Classifier (A) Binary Classification, (B) Multi-class Classification

The Compile model with Adam optimizer, binary cross-entropy loss, and accuracy metric. If we have two categorical must make the loss=binary_crossentropy :

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

However, when having more than categorical must be created the loss=categorical_crossentropy :

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

- **Optimizer: Adam**, the optimizer The Adam optimizer is a well-liked optimization method that's often used in deep learning model training. The algorithm modifies the learning rate during training by considering the prior gradients, resulting in an extraordinarily successful and efficient optimization technique.

- **Loss Function**: To assess the difference between two probability distributions, a mathematical function known as binary cross-entropy is used, especially when dealing with binary classification problems. The binary cross-entropy loss function is often used in binary classification tasks, where events must be categorized into one of two categories, such as incursion or normal. Greater model performance is indicated by lower values. The statistic calculates the discrepancy between the true binary labels and the estimated probability.
- **Metrics:** Precision Accuracy is a common evaluation criterion for classification tasks. The percentage of accurately recognized instances among all occurrences in the dataset is computed. In this instance, the accuracy measure will be used to assess the model's performance throughout training and testing. Better accuracy metric values correspond to better performance.

You may use *Keras*' compile function to specify the optimizer, loss function, and evaluation metrics for the model before it is trained. You are building the model to be trained with these particular parameters, which are often used for network intrusion detection applications. During model compilation, you specifically define the Adam optimizer, binary cross-entropy loss, and accuracy measure.

## 4.7 Description of the Network Attack

A network attack is a deliberate attempt to unlawfully infiltrate an organization's network, steal data, or carry out other malicious activities. There are two main classifications of network attacks:

### 4.7.1 Passive

Attackers enter a network and can clandestinely monitor or pilfer personal data while abstaining from altering the information, and keeping it in its original state.

### 4.7.2 Active

Attackers not only receive illegal access but also exploit data by deleting, encrypting, or driving other forms of damage to it.

Network assaults are distinct from several other types of attacks.

Endpoint assaults include the unauthorized infiltration of user devices, servers, or other endpoints. These assaults often compromise the devices by infecting them with malicious software.

Unauthorized program introduction into IT resources is known as a malware attack. This kind of assault gives attackers access to systems, allows them to steal data, and causes damage. Ransomware is another one of these assaults.

The firm is prone to vulnerabilities, exploits, and attacks that specifically target flaws in software. These attacks aim to gain unauthorized access, compromise, or cause harm to the organization's systems.

Advanced persistent threats (APTs) are highly complex and varied forms of attacks that span several types of assaults, such as network attacks and others.

During a network attack, attackers focus on penetrating the perimeter of the company network and gaining access to inside systems. Often, after an intrusion occurs, attackers may include other types of attacks, such as compromising a device, spreading malicious software, or exploiting a weakness in a system inside the network.

## 4.8 Intrusion Detection System( IDS)

Intrusion detection structures, or IDSs for short, are a type of security technology aimed at detecting and preventing unauthorized access, abuse, and intrusions into laptop networks and systems. An intrusion detection gadget's (IDS) goal is to pick out unusual or malicious behaviour taking place on a network or machine and notify system directors or security professionals so they may take the perfect measures to lower the chance [103].

There are two main types of IDS: network-based and host-based:

**Network-based Intrusion Detection Systems (NIDS)** Analyze community information for signs of doubtful behaviour and unlawful access. Port scanning, denial of carrier (DoS) assaults, and efforts to take advantage of weaknesses in community protocols or programs are just a few of the threats that Network Intrusion Detection Systems (NIDS) may additionally pick out.

**Host-based Intrusion Detection Systems (HIDS)** are mounted on non-public computer systems and reveal activities and system logs for indicators of malicious activity. Host-based intrusion detection structures (HIDS) may also become aware of numerous threats, consisting of attempts to get unauthorized access, adjustments to device configuration documents, and malware infestations.

IDS may use a variety of detection strategies, which include heuristic, signature-primarily based, and anomaly detection. Anomaly detection searches for styles of behaviour that deviate from regular hobbies, while signature-based detection compares device or network interest to a database of recognized threat signatures. Heuristic detection is the system of identifying, in all likelihood, suspicious pastimes by way of the use of attempted and actual assault strategies.

When something appears suspicious, an intrusion detection gadget (IDS) might also alert the gadget administrator or safety specialists. Alerts may be dispatched via e-mail, textual content messages, safety dashboard notifications, or different channels. Administrators or security personnel might also pick up any vulnerabilities within the machine or software, disconnect the affected device from the network, or block the attack's supply after reading the warning.

IDS often provides a large contribution to an all-encompassing security strategy that safeguards terminal networks and structures. They offer an extra line of protection against cyber threats and reduce the danger of protection breaches and information loss.

An intrusion detection system (IDS) is a monitoring tool that is used to identify and analyze all potentially dangerous network actions. **Anomaly-based total detection** and **signature-based detection** are the two principal strategies used to build an intrusion detection machine (IDS). An intrusion detection gadget (IDS) compares incoming visitors to a database of previously produced signatures, making use of attack signatures. Therefore, an assault is only detectable with the aid of the gadget if its signature is observed in the database. An anomaly-pushed intrusion detection device (IDS) keeps an eye fixed on network data to identify any uncommon conduct that doesn't shape the anticipated styles of typical network interest. The signature-based detection method has inherent faults, such as being easy to attack by zero-day attacks or attacks that have been altered to avoid detection by the signature database. Machine Learning techniques and knowledge of methods are very compatible with anomaly-primarily-based intrusion detection structures (IDS) because they allow the IDS to be successfully taught to discriminate between benign regular site visitors and malicious attack site visitors.

The Three Approaches to Intrusion Detection Systems

### 4.8.1 Intrusion Detection Based on a User's Signature

The purpose of Signature-Based Intrusion Detection Systems, commonly known as SIDS, is to find patterns and correlate them with known signs of intrusion.

A “SIDS” takes advantage of a database that includes knowledge about prior assaults. The detection system will send an alert to the administrator if any behavior on your network has a "signature" that is equivalent to that of an attack or security breach in the database [104].

Ensuring continuous changes to the database is vital for a SIDS solution since it is the basic backbone of the system. A SIDS system can only identify risks that it has already encountered. Hence, even with continuous changes to the database, your company's security may still be exposed to innovative infiltration tactics.

### 4.8.2 Intrusion Detection Based on a Unusual Behavior

On the other hand, An Anomaly-Based Intrusion Detection System, or AIDS, may be able to identify these unique zero-day assaults.

AIDS uses machine learning (ML) and statistical data to create a model of "normal" behavior. If a transmission departs in any manner from its typical pattern, the system will flag it as possibly malicious [104].

When comparing AIDS with SIDS, the biggest worry is the potential for an erroneous diagnosis. Ultimately, not all alterations are the consequence of illegal behaviour; rather, some are just an expression of shifts in the organization's general ethics. However, since SIDS do not have access to a database of known assaults, they are more likely to flag any and all abnormalities as intrusions.

### 4.8.3 Intrusion Detection using Hybrid Methods

With a hybrid approach, the benefits of both kinds of systems might be combined. A hybrid intrusion detection system can detect both novel and pre-existing intrusion scenarios since it examines both isolated occurrences and recurring trends [104]. The greater number of issues that are found is the only disadvantage of a hybrid approach. However, because the goal of an intrusion detection system (IDS) is to identify potential intrusions, it is difficult to see this rise of flags as a negative development.

## 4.9 Confusion Matrix

A confusion matrix is a popular technique for examining the performance of a machine-learning system. It is a matrix containing actual values against expected values, where each column shows the percentage of samples that the model predicted correctly or inaccurately.

The confusion matrix is often employed in scenarios involving binary categorization, where there are two potential outcomes (e.g., positive or negative). This 2 x 2 matrix may produce the following four products:

•True Positive (TP): The sample in the positive class was accurately predicted by the model.

•False Positive (FP): An incorrect prediction was made by the model indicating that the sample was in a positive class.

•True Negative (TN): The sample's membership in the negative class was correctly predicted by the model.

•False Negative (FN): The sample was incorrectly predicted by the model to belong to the negative class.

A variety of criteria are used to assess the efficacy of a classifier. Table 4.5 provides a brief explanation of each of the four potential outcomes associated with a detection strategy.

Using a tabular form called a confusion matrix, an actual attack is compared to an expected attack. by dividing the projected intrusion value (cell value in the confusion matrix row) by the total number of real invasions to get a normalized result. The recall of the model is the number of predicted values that agree with the actual values. It may be located by analyzing the values along the principal diagonal members of the confusion matrix. In an ideal model, a value of "0" would be shown everywhere else, and a value of "1" just on the main diagonal. The accuracy is determined by dividing the total number of occurrences (the sum of the whole matrix) by the number of cases that were correctly predicted (the sum of the diagonal terms), as shown in the following Figures.

**Table 4.5** Confusion Matrix

| | | Actual | |
|---|---|---|---|
| | | Normal | Attacks |
| Predicted | Normal | True Positive | False Positive |
| | Attacks | False Negative | True Negative |

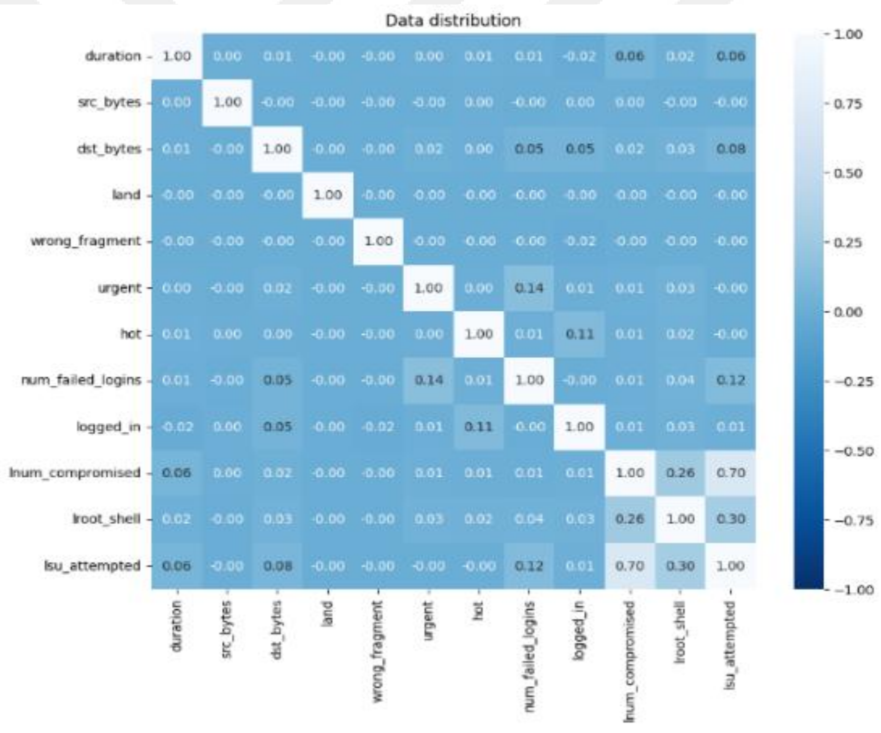


**Figure 4.7** Matrix of Confusion the Attacks Distribution

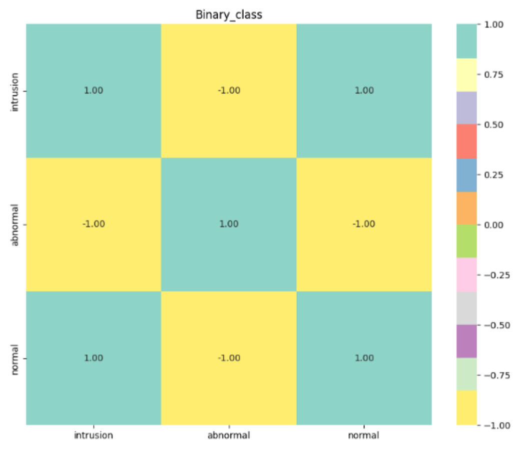


**Figure 4.8** Matrix of Confusion for Binary Attacks Distribution

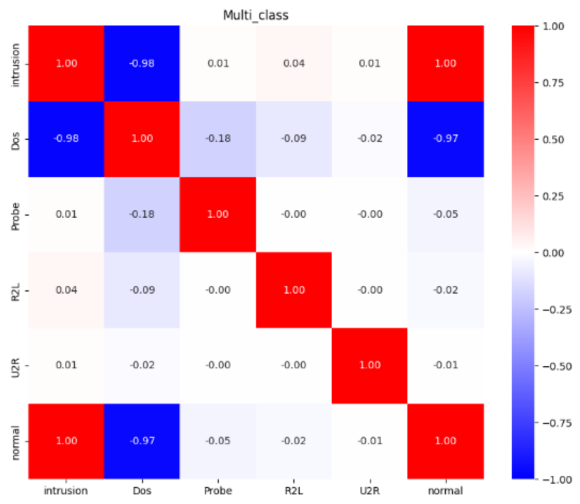


**Figure 4.9** Confusion Matrix of Multiclass Attacks Distribution

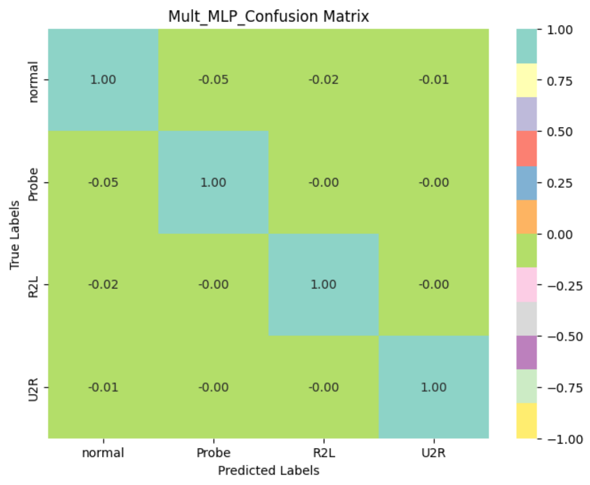


**Figure 4.10** Confusion Matrix of Multi-Class Classification using MLP Classifier

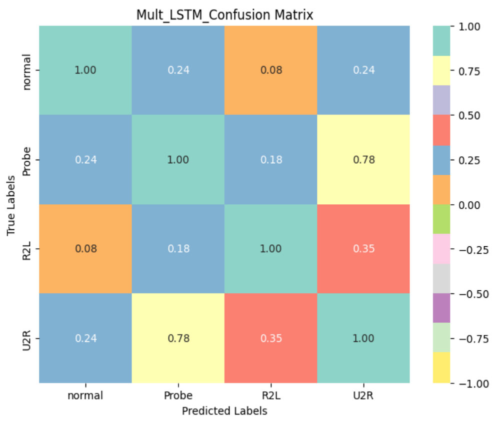


**Figure 4.11** Confusion Matrix of Multi-Class Classification using LSTM Classifier

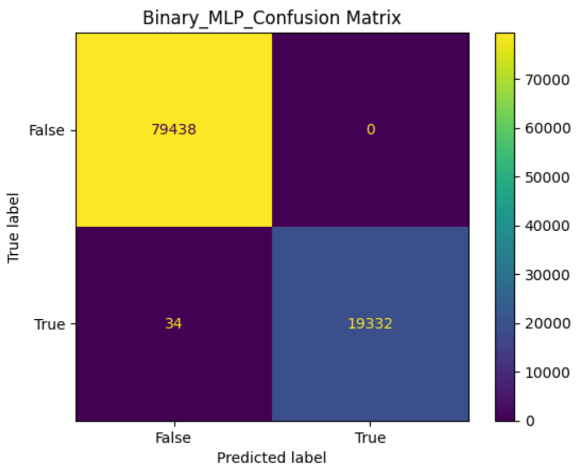


**Figure 4.12** Confusion Matrix of Binary Classification with MLP Classifier

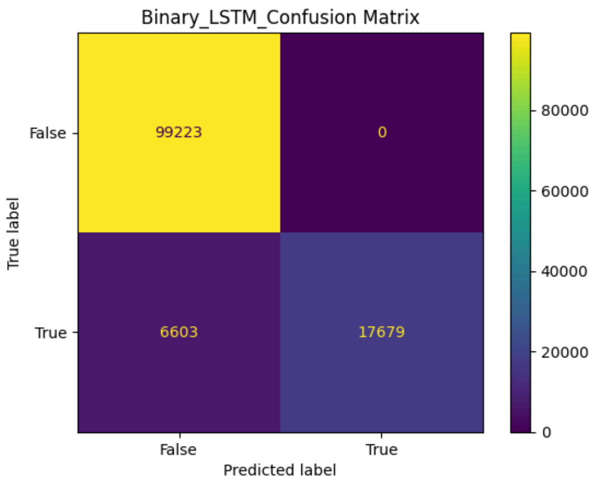


**Figure 4.13** Confusion Matrix of Binary Classification with LSTM Classifier

# 5
# RESULTS AND DISCUSSION

The whole set of our study's findings is presented in this section. It begins by outlining the assessment criteria, the KDD99 datasets that are used, and the programming environment. It then provides comprehensive results for every strategy. It concludes by describing the results obtained.

## 5.1 Programming Environments

The deep neural network necessitates numerous computations primarily involving matrix multiplications and additions. Therefore, employing a GPU is crucial to minimizing the time spent on model training.

- **Google Colab**: The Google Collaboratory (CoLab) is a cloud-based interactive development environment that is free to use for Jupyter notebooks that are hosted on Google’s servers.

The following libraries are used in this work: There are also many built-in deep learning libraries, but Tensorflow and Keras are mostly used ones.

• **Numpy:** includes mathematical functions.

**• Pandas:** for data analysis and manipulation.

**• ScikitLearn:** is a machine-learning library used for the predictive analysis of data

(SVM, PCA, train-test dataset splitting, k-fold, performance metrics, etc. . . .).

• **Tensorflow:** deep learning library developed by Google (probability, 2D, and

3D convolution, etc. . . .)

**• Keras:** is a deep learning library (CNN, Transfer learning models,

EffcientNet performance measurement functions, etc. . . .).

## 5.2 Datasets

We define our intrusion detection learning problem using the KDD99 dataset. KDD99 is the most widely used benchmark dataset for intrusion detection systems (IDS) it contains 494020 rows and 41 columns. To make raw TCP packets more useful for machine learning models, compilers used the DARPA1998 dataset and extracted 41-dimensional features, which comprised both labels and raw TCP packets. As a result, the KDD99 dataset was curated to generate a new dataset.

The labels in the DARPA 1998 and KDD 1999 datasets are identical. The attributes of KDD99 may be categorized into four groups: main characteristics, host-based statistical characteristics, content-based statistical characteristics, and time-based statistical characteristics [105].

After training the model on the test set (also referred to as a validation set), we calculate the effectiveness metrics and calibrate the decision parameters, such as the classifier threshold. We next use the production set, which is an unlabeled dataset, to find the abnormalities. When the model is put into production, it will make predictions on real-time data without labels to compare; therefore, the last step is crucial. Consequently, to ensure that the model does not behave weirdly on this unknown batch of unlabeled data, it is usually a good idea to wait for a production set (rather than a validation set or test set).

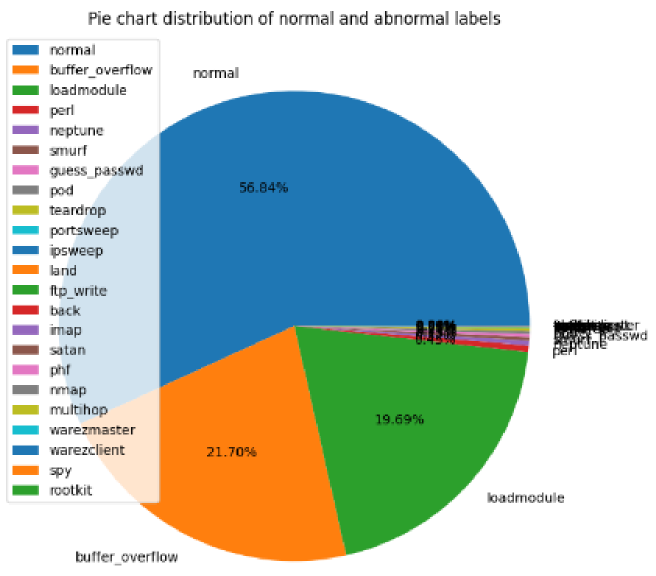


**Figure 5.1** Distribution of normal and abnormal Attack

**Table 5.1:** Examples of KDD CUP 99 Features

| Attribute name | Type | Description |
|---|---|---|
| Protocol type | Discrete | type of the protocol, e.g. tcp, udp, etc. |
| Service | Discrete | network service on the destination, e.g., http, telnet, etc. |
| Flag | Discrete | normal or error status of the connection. |
| Logged in | Discrete | 1 if successfully logged in; 0 otherwise. |
| Land | Discrete | 1 if the connection is from/to the same host/port; 0 otherwise |
| Duration | Continuous | Length (in seconds) of the connection |
| Dst bytes | Continuous | number of data in bytes sent from destination to source. |
| Src bytes | Continuous | number of data in bytes sent from the source to the destination. |
| Urgent | Continuous | number of urgent packets. |
| wrong fragment | Continuous | number of “wrong” fragments. |
| Hot | Continuous | number of “hot” indicators. |
| Num failed logins | Continuous | number of failed login attempts. |

**Table 5.2** Distribution of KDD Cup 99 Classes

|  | Normal | DoS | Probe | R2L | U2R |
|---|---|---|---|---|---|
| Training | 97278 (19.69%) | 391458 (79.24%) | 4107 (0.83%) | 1126 (0.23%) | 52 (0.01%) |
| Test | 60593 (19.48%) | 229855 (73.90%) | 4166 (1.34%) | 16345 (5.26%) | 70 (0.02%) |

**Table 5.3** Number of attack labels

| Name of Attack | Smurf | Neptune | Normal | Back |
|---|---|---|---|---|
| No. of Attack | 280790 | 107201 | 97277 | 2203 |
| Name of Attack | Satan | Ip sweep | Port sweep | Warez client |
| No. of Attack | 1589 | 1247 | 1040 | 1020 |
| Name of Attack | Teardrop | Pod | Nmap | Guess password |
| No. of Attack | 979 | 264 | 231 | 53 |
| Name of Attack | Buffer overflow | land | Ware master | Lmap |
| No. of Attack | 30 | 21 | 20 | 12 |
| Name of Attack | Rootkit | Load module | Ftp_write | Multihop |
| No. of Attack | 10 | 9 | 8 | 7 |
| Name of Attack | Phf | Perl | Spy | |
| No. of Attack | 4 | 3 | 2 | |

datasets are clustered into four different attack classes, As seen in Table 5.4 and Figure 5.2.

### 5.2.1 Denial OF Service (DOS)

stands for "denial of service," an assault that has the potential to force a computer to crash or operate more slowly. These attacks are carried out by providing the server with more traffic data than it is capable of handling at once. DoS attacks have a detrimental effect on legally permitted network traffic or service accessibility. Consequently, denial of service (DoS) attacks are among the most prevalent types of attacks on network resources that stop users from using network services. DoS attacks may take many different forms, and each one has a unique method of depleting network resources to accomplish the hacker's objective, which is to stop users from accessing the network [106].

### 5.2.2 Root to Local (R2L)

A specific kind of computer network assault is called "root to local," in which an attacker sends a sequence of packets to a server or next-side machine across a network that the hacker is not permitted to access as a local user. One may claim that by remotely delivering false packets to the system, they provide illegal local access to a machine. R2L assaults are referred to as "root to local" attacks.

### 5.2.3 User to Root (U2R)

A user-to-root attack (U2R) is a different kind of attack where the hacker repeatedly tries to get access to network resources as a normal user before managing to become a complete-access user. Hackers exploit vulnerabilities in the system in the same way that regular users do [106].

### 5.2.4 Prob

"Probing" is another word for a probe. A third kind of attack is known as probing, in which a hacker looks for open ports or weaknesses in the topological design of network devices, and then subsequently leverages such flaws to get personal information without permission. Numerous tools are available for network probing, such as Nmap, portsweep, and ipsweep. is an attack that collects network data with the goal of getting past security controls.

for this reason, "IDS works against all intruder attacks, making it an essential part of building computer networks to capture these kinds of attacks in their early stages."

**Tabel 5.4** Attack profiles of DoS, R2L, U2R, Probe Classes

| Attack class | Attack profile |
|---|---|
| DoS | back, neptune, land, smurf, pod, teardrop, apache2, mailbomb, processtable, udpstorm,worm. |
| R2L | ftp write, imap, guess passwd, multihop, phf, warezclient, spy, warezmaster,,httptunnel,amed,sendmail, snmpgetattack,snmpgues,aware client,,xloc ,xsnoop. |
| U2R | loadmodule, buffer overflow, rootkit, perl,ps,sqlattack,xterm. |
| Probe | nmap, ipsweep, satan, portsweep,,mscan,saint. |

**Table 5.5** Attack class of DoS, R2L, U2R, Probe by count

| Dos | normal | Probe | R2L | U2R |
|---|---|---|---|---|
| 391458 | 97277 | 4107 | 1126 | 52 |

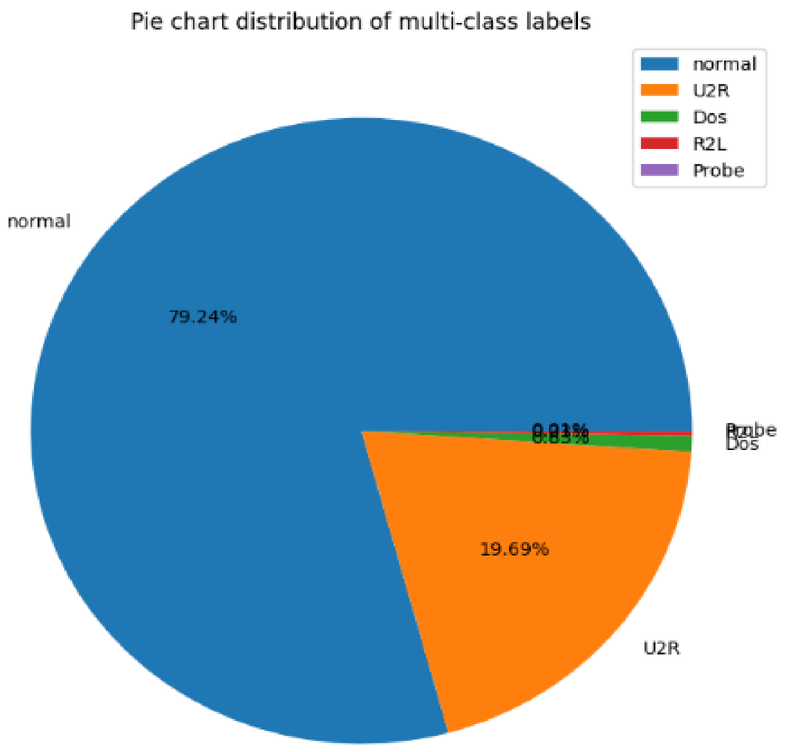


**Figure5.2** Multi-class Classification

## 5.3 Evaluation Metrics

To evaluate the LSTM and MLP Models performance, we use the assessment measures. They provide numerical metrics that assist us in choosing the most appropriate model for the task at hand. Common assessment measures include F1-score, recall, accuracy, and precision.

**Accuracy** is the percentage of records classified correctly, and it is computed using the following formula:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{5.1}$$

**Precision (P)** is the percentage of records correctly classified as anomalies out of the total number of records classified as anomalies. Precision is computed as follows:

$$Precision = \frac{TP}{TP+FP} \tag{5.2}$$

**Detection rate (DR)**, also called True Positive Rate or Recall, is the percentage of records correctly classified as anomalies out of the total number of anomaly records. A method to get the detection ratio is as follows:

$$Recall = \frac{TP}{TP+FN} \tag{5.3}$$

**F-measure (F)**, which is the mathematical average of accuracy and recall, is computed using the harmonic mean. It is used to assess test accuracy and acts as a direct indication of the model's performance. A statistic having a range of 0 to 1, the F1 score seeks to get a number as close to 1. It is calculated as follows:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision+Recall} \tag{5.4}$$

where the number of instances correctly identified as anomalous is expressed as TP (True Positive); the number of instances that are correctly classified as normal is known as TN (True Negative); False positives, or FPs, are the number of regular traffic patterns that are mistakenly identified as anomalous, whereas false negatives, or FNs, are the number of abnormal traffic patterns that are mistakenly identified as normal. The performances of the proposed architectures (LSTM, MLP) were examined in binary (Normal, Abnormal) and multi-classification (Normal, Dos, Probe, R2L) modalities in order to assess the classifier's ability to correctly discriminate between normal and abnormal attacks. Tables 5.6 and 5.7 provide an overview of the accuracy, precision, recall, F1-score, and loss for the suggested models (LSTM and MLP). All cases had a model accuracy of more than 99%, which increased the detection rate.

### 5.3.1 Binary Classification

Table 5.6 reports the outcomes of binary classification experiments; when splitting the dataset 75% for training and 25% for testing, The LSTM and MLP classifiers achieved F1 scores of 0.9958465271209442% and 0.9998352553542009% in detecting normal and abnormal categories, respectively, whereas, LSTM also performed well in terms of accuracy, with a performance rate of up to 99.83644485473633% with a loss of 0.7034842856228352% as compared to the MLP classifier which achieved accuracies of 99.99352097511292% with a loss of 0.04522902600001544%. LSTM in terms of performance time with 152.73029232025146 seconds, compared to the MLP classifier which achieved a performance time of 1008.4796767234802 seconds. However, compared with the result in Tabel 5.7, when splitting the dataset 80% for training and 20% for testing, The LSTM and MLP classifiers achieved F1 scores of 0.999767675985441% and 0.9998708911095618 %, respectively, whereas, LSTM also performed well in terms of accuracy, with a performance rate of up to 99.99089241027832% with a loss of 0.03050541563425213% as compared to the MLP classifier which achieved accuracies of 99.99493956565857% with a loss of 0.03381400019861758%.LSTM in terms of performance time with 170.23747491836548 seconds, compared to the MLP classifier which achieved a performance time of 1111.6987471580505 seconds.

Also, we got from the Tabel 5.6, Precision with the result as 0.9944152431011827 for LSTM and Recell with 0.9972819372374598. Likewise Precision MLP 0.999917620891342, and Recell with 0.9997529033852236.

While, In Tabel 5.7, the Precision for LSTM is 0.9995870541475249, and Recell 0.9999483631106062, Likewise for Precision MLP 1.0, and Recell with 0.9997418155530311. With the different hidden layers, our results are measured in Figures 5.3 and 5.4.

### 5.3.2 Multi Classification

Table 5.6 reports the outcomes of multi-classification experiments; when splitting the dataset 75% for training and 25% for testing, The LSTM and MLP classifiers achieved F1 scores of 0.997145913650768% and 0.9999028355816101% in detecting normal and abnormal categories, respectively, whereas, LSTM also performed well in terms of accuracy, with a performance rate of up to 99.99756813049316% with a loss of 0.014561950229108334% as compared to the MLP classifier which achieved accuracies of 99.99271035194397% with a loss of 0.06861906149424613%. LSTM in terms of performance time with 150.2320737838745 seconds, compared to the MLP classifier which achieved a performance time of 1057.5126361846924 seconds. However, compared with the result in Tabel 5.7, when splitting the dataset 80% for training and 20% for testing, The LSTM and MLP classifiers achieved F1 scores of 0.9952305406562414% and 0.9998532440652406 %, respectively, whereas, LSTM also performed well in terms of accuracy, with a performance rate of up to 99.99696612358093% with a loss of 0.014984085282776505% as compared to the MLP classifier which achieved accuracies of 99.98583197593689% with a loss of 0.46352697536349297%.LSTM in terms of performance time with 184.8968095779419 seconds, compared to the MLP classifier which achieved a performance time of 1111.741890668869 seconds.

Also, we got from the Tabel 5.6, Precision with the result as 0.9943160776104983 for LSTM and Recell with 0.9999919031618153. Likewise for Precision MLP 0.9999271249159912, and Recell with 0.9998785474272297.

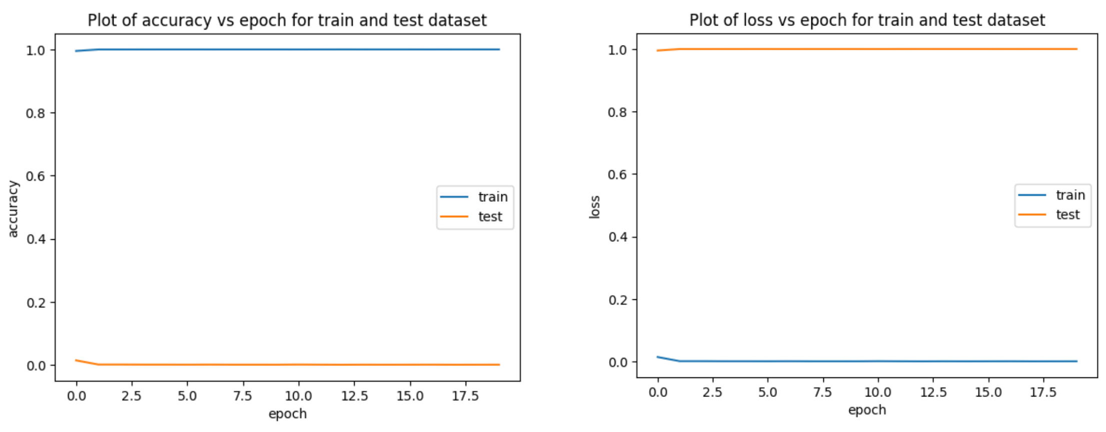
In Tabel 5.7, the Precision for LSTM is 0.9905063608384879, and Recell 1.0, Likewise for the MLP, the Precision is 0.999858303897655, and Recell with 0.9998481842840371. With the different hidden layers, our results are measured in Figures 5.3 and 5.4.

**Table 5.6** Schedule of performance of models (LSTM and MLP), splitting the dataset 75% for training and 25% for testing

| **Model** | **LSTM** | | **MLP** | |
|---|---|---|---|---|
| **Class** | **Binary** | **Multiclass** | **Binary** | **Multiclass** |
| **Accuracy** | 99.836444854473633% | **99.99756813049316%** | 99.99352097511292% | 99.99271035194397% |
| **Precision** | 0.9944152431011827 | 0.9943160776104983 | 0.999917620891342 | **0.9999271249159912** |
| **Recall(DR)** | 0.9972819372374598 | **0.9999919031618153** | 0.9997529033852236 | 0.9998785474272297 |
| **F1- score** | 0.9958465271209442 | 0.997145913650768 | 0.9998352553542009 | **0.9999028355816101** |
| **Loss** | 0.703484285622835% | **0.0145619502291083%** | 0.04522902600001544% | 0.06861906149424613% |
| **Time(s)** | 152.73029232025146 | **150.2320737838745** | 1008.4796767234802 | 1057.5126361846924 |

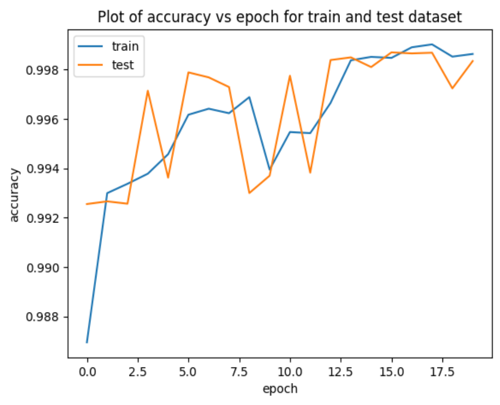
**Table 5.7** Schedule of performance of models (LSTM and MLP), splitting the dataset 80% for training and 20% for testing

| **Model** | **LSTM** | | **MLP** | |
|---|---|---|---|---|
| **Class** | **Binary** | **Multiclass** | **Binary** | **Multiclass** |
| **Accuracy** | 99.99089241027832% | **99.99696612358093%** | 99.99493956565857% | 99.98583197593689% |
| **Precision** | 0.9995870541475249 | 0.9905063608384879 | **1.0** | 0.999858303897655 |
| **Recall(DR)** | 0.9999483631106062 | **1.0** | 0.9997418155530311 | 0.9998481842840371 |
| **F1- score** | 0.999767675985441 | 0.9952305406562414 | **0.9998708911095618** | 0.9998532440652406 |
| **Loss** | 0.030505415634252l3% | **0.0149840852827765%** | 0.03381400019861758% | 0.46352697536349297% |
| **Time(s)** | **170.23747491836548** | 184.8968095779419 | 1111.6987471580505 | 1111.741890668869 |

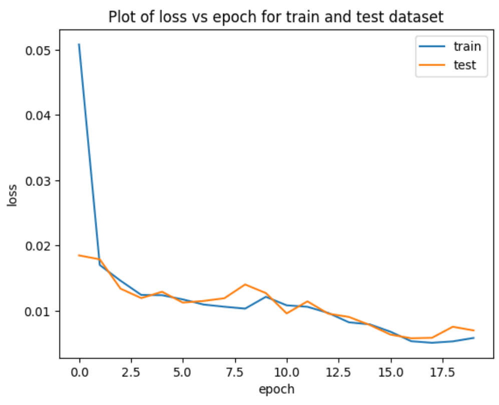


(A) (B)

**Figure 5.3** (A) Accuracy throughout Epoch, (B) loss throughout Epoch for the MLP Model (Binary Classification)

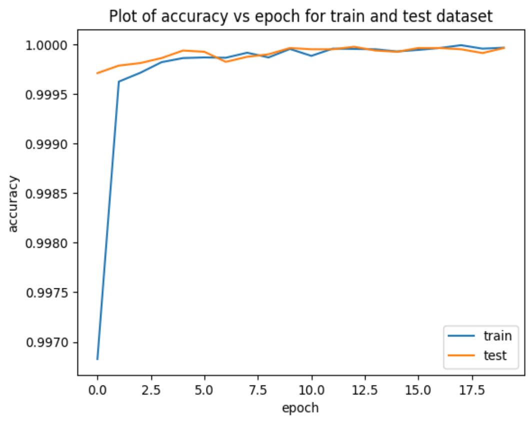


(A)

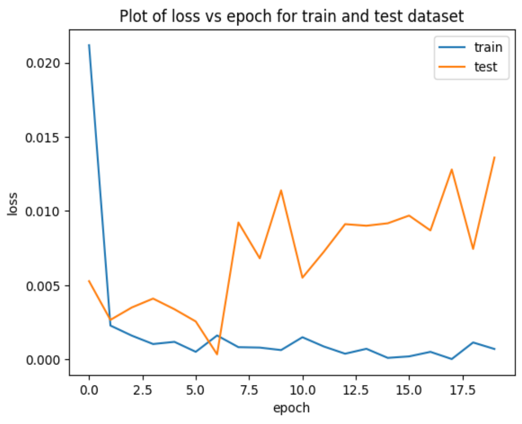


(B)

**Figure 5.4** (A) Accuracy throughout Epoch, (B) loss throughout Epoch for the LSTM Model (Binary Classification)

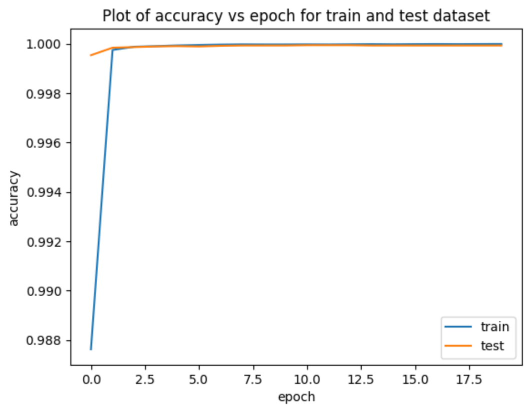


(A)

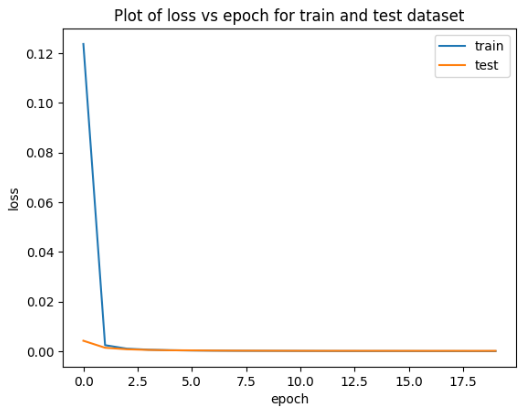


(B)

**Figure 5.5** (A) Accuracy throughout Epoch, (B) loss throughout Epoch for the MLP Model ((Multi-class Classification)



(A)



(B)

**Figure 5.6** (A) Accuracy throughout Epoch, (B) loss throughout Epoch for the LSTM Model ((Multi-class Classification)

# 6 CONCLUSIONS

The use of Deep Learning (DL) for detecting network intrusion traffic in IoT devices is a promising approach to enhancing IoT security. By analyzing the network traffic data using DL algorithms, it is possible to identify suspicious activities and prevent potential cyberattacks.

The results of this study showcase remarkable performance metrics, with accuracy and detection rates consistently averaging an impressive 99.99%. Notably, the loss rate attained a remarkably low figure of 0.7%, underscoring the system's capability to discern normal data from potential attacks with exceptional precision. The study in this domain has scrutinized and categorized individual traffic data.

However, as part of prospective endeavours, the exploration of real-time deep analysis of data emerges as a promising avenue. We underscored our trajectory by the pursuit of criteria facilitating faster decision-making processes, a concerted effort to mitigate computational complexity, and an orientation towards handling the challenges posed through big data in intrusion detection contexts.

Some of the key conclusions from using DL for IoT security are:

DL-based intrusion detection systems (IDS) can improve the accuracy of detecting network attacks in IoT devices.

DL-based IDS can analyse large amounts of network traffic data in real time, which is essential on IoT devices that generate vast amounts of data.

The accuracy of DL-based IDS can be further enhanced by using different DL models and techniques, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs).

The implementation of DL-based IDS in IoT devices can also help in reducing false positives and false negatives, which are significant issues in traditional IDS.

DL-based IDS can be trained to detect new and previously unknown types of attacks, which is a significant advantage over rule-based IDS.

In summary, DL-based intrusion detection systems have the potential to provide robust security for IoT devices by accurately detecting and preventing cyberattacks. However, the effectiveness of these systems depends on the quality and quantity of training data, the selection of appropriate DL models, and the continuous improvement of the system to adapt to new types of attacks.

# REFERENCES

[1] G. Jaina and R. R. Prasad, "IoT Enabled Multi-Antenna Communication Channel with Deep Learning Network," Proceedings of the International Conference on Recent Advances in Computational Techniques (IC-RACT), pp. 179-184, 2020.

[2] A. Saint, "Where next for the Internet of things, Information Technology Internet of Things," Engineering and Technology, vol.10, no. 1, pp. 72-75, 2015.

[3] K. Jagdale, C. Shelke and R. Achary, "Artificial Intelligence and its Subsets: Machine Learning and its Algorithms," Deep Learning and their Future Trends. vol. 9, pp. 112-117, 2022.

[4] Diwakar Bhardwaj, "Study of Significant and Competent Intrusion Detection System by FA-SVM Technique," European Journal of Molecular and Clinical Medicine, vol. 7, no.4, 2020.

[5] M. M. N. Aboelwafa, K. G. Seddik, M. H. Eldefrawy, Y. Gadallah and M. Gidlund, "A Machine-Learning-Based Technique for False Data Injection Attacks Detection in Industrial IoT," IEEE Internet of Things Journal, vol. 7, no. 9, pp. 8462-8471, 2020.

[6] L. Liu, J. Yang and W. Meng, "Detecting malicious nodes via gradient descent and support vector machine in the Internet of Things," Computers and Electrical Engineering, vol. 77, pp. 339-353, 2019.

[7] M. Anwer, Khan, S. M., Farooq and Waseemullah, "Attack Detection in IoT using Machine Learning. Engineering," Technology and Applied Science Research, vol. 11, no. 3, pp. 7273-7278, 2021.

[8] R. Shyam and G. Awasthi, "Role of Deep Learning in Image Recognition," Journal of Image Processing and Pattern Recognition Progress, vol. 8, no. 2, pp. 34-39, 2021.

[9] A. Jayaswal and R. Nahar, "Detecting Network Intrusion through a Deep Learning Approach," International Journal of Computer Applications, vol. 180, no. 14, pp. 15-19, 2018.

[10] Usman Inayat, Muhammad Fahad Zia, Sajid Mahmood, Haris M. Khalid and Mohamed Benbouzid, "Learning-Based Methods for Cyber Attacks Detection in IoT Systems: A Survey on Methods, Analysis, and Future Prospects," Technology and Applied Science Research, vol. 11, no. 9, pp. 7273-7278, 2022.

[11] M. Kohler, and A. Krzyzak, "Over-parametrized deep neural networks minimizing the empirical risk do not generalize well," Bernoulli, vol. 27, no. 4, pp. 2564-2597, 2021.

[12] Md. Imtiaz Ahmed and Fatima Shefaq, "A Study on Machine Learning and Supervised and Deep Learning Algorithms to Predict the Risk of Patients," International Journal of Practical Healthcare Innovation and Management Techniques, vol. 9, no.1, pp. 1-12, 2022.

[13] S. Niksefat, P. Kaghazgaran and B. Sadeghiyan, "Privacy issues in intrusion detection systems: A taxonomy, survey and future directions," Computer Science Review, vol. 25, pp. 69-78, 2017.

[14] S.M. Khan, M. U. Farooq and W. Waseemullah, "Attack Detection in IoT using Machine Learning," Engineering, Technology and Applied Science Research, vol. 11, no. 3, pp. 7273-7278, 2021.

[15] Z. Allam, and Zaynah A. Dhunny, "On big data, artificial intelligence and smart cities," Cities, vol. 89, pp. 80-91, 2019.

[16] Krishna Kumar Mohbey, "An Efficient Framework for Smart City Using Big Data Technologies and Internet of Things," Advanced Computing and Intelligent Engineering, pp. 319-328, 2019.

[17] Y. Khlaponin, Lesja M. Kozubtsova, I. Kozubtsov and R. Shtonda, "Functions of The Information Security and Cybersecurity System of Critical Information Infrastructure," Cybersecurity Education, Science Technique, vol. 3, no. 15, pp. 124-134, 2022.

[18] A. Elsaeidy, I. Elgendi, K. S. Munasinghe, D. Sharma and A. Jamalipour, "A smart city cyber security platform for narrowband networks," International Telecommunication Networks and Applications Conference (ITNAC), pp.1-6, 2017.

[19] J. Foley, N. Moradpoor and H. Ochen, "Employing a Machine Learning Approach to Detect Combined Internet of Things Attacks against Two Objective Functions Using a Novel Dataset," Security and Communication Networks vol. 2, pp. 1-17, 2020.

[20] N. Sahar, R. Mishra and S. Kalam, "Deep Learning Approach-Based Network Intrusion Detection System for Fog-Assisted IoT," International Conference on Big Data Machine Learning and Their Applications, pp. 39-50, 2021.

[21] S. Kavitha, U. Maheswari, R.Venkatesh, "Network Anomaly Detection for Nsl-Kdd Dataset Using Deep Learning," Information Technology In Industry, vol. 9, no. 2, pp. 821-827, 2021.

[22] H. Neuschmied, M. Winter and B. Stojanovic, "APT-Attack Detection Based on Multi-Stage Autoencoders, " Recent Advances in Cybersecurity and Computer Networks, vol. 12, no. 13, 2022.

[23] A. Khamparia, S. D. Pande, D. Gupta and A. K. Sangaiah, "Multi-level framework for anomaly detection in social networking," Hi-Tech, vol. 38, no. 2, pp. 350-366, 2020.

[24] Aswani K. Cherukuri, and I. S. Thaseen, "Intrusion detection model using feature extraction and LPBoost technique," International Journal of Internet Technology and Secured Transactions, vol. 8, no. 4, pp. 635-652, 2018.

[25] B. A. Tama, M. Comuzzi and K. H. Rhee, "TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System," IEEE Access, vol. 7, pp. 94497-94507, 2019.

[26] M. Panda, A. Abraham, S. Das, and M. R. Patra, "Network intrusion detection system: A machine learning approach," Intelligent Decision Technologies, vol. 5, no. 4, pp. 347-356, 2011.

[27] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," International Conference on Signal Processing and Communication Engineering Systems (SPACES), pp. 92-96, 2015.

[28] S. Mukkamala, A. H. Sung, and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," Journal of Network and Computer Applications, vol. 28, no. 2, pp. 167-182, 2005.

[29] H. Alqahtani, I. H. Sarker, A. Kalim and S. Hossain, "Cyber Security Intrusion Detection using Machine Learning Techniques," International Conference on Computing Science, Communication and Security (COMS2), vol. 2, 2020.

[30] K. Aggarwal, M. M. Mijwil, S. Garg and S. Abdul Rahman, "Has the Future Started? The Current Growth of Artificial Intelligence, Machine Learning, and Deep Learning," Iraqi Journal for Computer Science and Mathematics, vol. 3, no.1, pp. 115-123, 2022.

[31] Yuefei Zhu, Jinlong Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," IEEE vol. 5, pp. 21954-21961, 2017.

[32] A. Aldweesh, Abdelouahid Derhab, and A. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," Knowledge-Based Systems, vol. 189, pp. 105-124, 2020.

[33] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," IEEE Latin-American Conference on Communications (LATINCOM), vol. 7, pp. 41525-41550, 2019.

[34] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi and M. Ghogho, "Deep Learning Approach for Network Intrusion Detection in Software Defined Networking," International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258-263, 2016.

[35] K. M. Sydney, "A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework," Computer Communications, vol. 199, pp.113-125, 2023.

[36] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, 2018.

[37] R. Vinayakumar, Kp. Soman and P. Poornachandran, "Evaluation of Recurrent Neural Network and Its Variants for Intrusion Detection System (IDS)," International Journal of Information System Modeling and Design, vol. 8, no. 3, pp. 43-63, 2017.

[38] B. Dong and X. Wang, "Comparison Deep Learning Method to Traditional Methods Using for Network Intrusion Detection," IEEE International Conference on Communication Software and Networks (ICCSN), pp. 581-585, 2016.

[39] R. Zhao, R. Yan, Zh. Chen, K. Mao, P. Wang and R. X. Gao, "Deep Learning and Its Applications to Machine Health Monitoring," Mechanical Systems and Signal Processing, vol. 115, pp. 213-237, 2019.

[40] S. Hou, A. Saas, L. Chen and Y. Ye, "Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs," IEEE International Conference on Web Intelligence Workshops (WIW), pp. 104-111, 2016.

[41] L. You, Y. Li, Y. Wang, J. Zhang, and Y. Yang, "A deep learning based RNNs model for an automatic security audit of short messages," International Symposium on Communications and Information Technologies (ISCIT), pp. 225-229, 2016.

[42] K. Alrawashdeh and C. Purdy, "Toward an Online Anomaly Intrusion Detection System Based on Deep Learning," IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 195-200, 2016.

[43] Jin Kim, Nara Shin, S. Y. Jo and S. H. Kim, "Method of intrusion detection using deep neural network," IEEE International Conference on Big Data and Smart Computing (Big Comp), pp. 313-316, 2017.

[44] A. Javaid, Q. Niyaz, W. Sun and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," International Conference on Bio-Inspired Information and Communications Technologies (BIONETICS), pp. 21-26, 2016.

[45] S. Potluri and C. Diedrich, "Deep Feature Extraction for Multi-Class Intrusion Detection in Industrial Control Systems," International Journal of Computer Theory and Engineering, vol. 9, no. 5, pp. 374-379, 2017.

[46] C. G. Cordero, S. Hauke, M. Mühlhäuser and M. Fischer, "Analyzing Flow-Based Anomaly Intrusion Detection Using Replicator Neural Networks," Annual Conference on Privacy, Security and Trust (PST), pp. 317-324, 2016.

[47] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, 2018.

[48] M. J. Kang and J.W. Kang, "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security." PLOS One, vol. 11, no. 6, 2016.

[49] E. Hodo, X. Bellekens, A. Hamilton, Ch. Tachtatzis and R. Atkinson, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey," Computer Science-Cryptography and Security, vol.1, 2017.

[50] Q. Niyaz, W. Sun, A. Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," EAI Endorsed Trans. Security Safety, vol. 4, no. 12, 2017.

[51] Y. Wang, W. Cai, P. Wei, "A Deep Learning Approach for Detecting Malicious JavaScript Code," Security and Communication Networks, vol. 9, no. 11, pp. 1520-1534, 2016.

[52] H. Lee, N. Kim and J. Lee, "Deep Neural Network Self-Training Based on Unsupervised Learning and Dropout," International Journal of Fuzzy Logic and Intelligent Systems, vol. 17, no. 1, pp. 1-9, 2017.

[53] A. K. Sahu, S. Sharma, M. Tanveer and R. Raja, "Internet of Things Attack Detection Using Hybrid Deep Learning Model," Computer Communications, vol. 176, pp. 146-154, 2021.

[54] M. Z. Alom, V. Bontupalli and T. M. Taha, "Intrusion Detection Using Deep Belief Networks," National Aerospace and Electronics Conference (NAECON), pp. 339-344, 2015.

[55] R. M. Almejarb, O. M. Sallabi, F. F. Bushaala, A. Ali Mohamed, A. Fawzi and R. Adel, "A Proposed Method for Detecting Network Intrusion Using Deep Learning Approach," IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), pp. 83-88, 2023.

[56] T. Singhania, V. Agarwal, Sunakshi, P.S. Giridhar and T. Gupta, "A Novel Intrusion Detection System Using Deep Learning," International Conference on Innovative Computing and Communications, Advances in Intelligent Systems and Computing, vol. 1394, pp. 343-352, 2022.

[57] M. Wang, Y. Lu and J. Qin, "A Dynamic MLP-Based DDoS Attack Detection Method Using Feature Selection and Feedback," Computers and Security, vol. 88, 2020.

[58] O. Jullian, B. Otero and E. Rodriguez, "Deep-Learning Based Detection for Cyber-Attacks in IoT Networks: A Distributed Attack Detection Framework," Journal of Network and Systems Management, vol. 31, no. 2, 2023.

[59] Y. Otoum, D. Liu and A. Nayak, "DL-IDS: A Deep Learning–Based Intrusion Detection Framework for Securing IoT," Transactions on Emerging Telecommunications Technologies, vol. 33, no.3, 2022.

[60] J. R. Rose, M. Swann, G. Bendiab, S. Shiaeles and N. Kolokotronis, "Intrusion Detection Using Network Traffic Profiling and Machine Learning for IoT," IEEE 7th International Conference on Network Softwarization (NetSoft), pp. 409-415, 2021.

[61] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti and D. De, "Fundamental Concepts of Convolutional Neural Network," Recent Trends and Advances in Artificial Intelligence and Internet of Things, Intelligent Systems Reference Library, vol. 172, pp. 519-567, 2019.

[62] W. Zaremba, I. Sutskever and O. Vinyals, "Recurrent Neural Network Regularization," Neural and Evolutionary Computing, vol. 5, 2015.

[63] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks," Computer Science - Neural and Evolutionary Computing, vol. 1909, 2019.

[64] H. M. Lynn, S. B. Pan and P. Kim, "A Deep Bidirectional GRU Network Model for Biometric Electrocardiogram Classification Based on Recurrent Neural Networks," IEEE Access, vol. 7, pp. 145395-145405, 2019.

[65] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," Biological Cybernetics, vol. 43, no. 1, pp 59-69, 1982.

[66] M. Sewak, S. K. Sahay and H. Rathore, "An Overview of Deep Learning Architecture of Deep Neural Networks and Autoencoders," Journal of Computational and Theoretical Nanoscience, vol. 17, no. 1, pp 182-188, 2020.

[67] M. P. Hosseini, S. Lu, K. Kamaraj, A. Slowikowski and H.C. Venkatesh, "Deep Learning Architectures," Deep Learning: Concepts and Architectures, vol 866, pp. 1-24, 2019.

[68] G. V. Houdt, C. Mosquera and G. Napoles, "A Review on the Long Short-Term Memory Model," Artificial Intelligence Review. vol. 53, no.1, pp. 5929-5955, 2020.

[69] L. Ma, Y. Chai, L. Cui, D. Ma, Y. Fu and A. Xiao, "A Deep Learning-Based DDoS Detection Framework for Internet of Things," IEEE International Conference on Communications (ICC), pp. 1-6, 2020.

[70] M. Gopinath, S. Chakkaravarthy and Sethuraman, "A comprehensive survey on deep learning-based malware detection techniques," Computer Science Review, vol. 47,2023.

[71] Fatih Taşpınar and Zehra Bozkurt, "Application of Artificial Neural Networks and Regression Models in The Prediction of Daily Maximum Pm10 Concentration in Düzce, Turkey," vol. 23, no. 10, 2014.

[72] M. M. Poulton, "Chapter 3 Multi-layer perceptrons and back-propagation learning," Handbook of Geophysical Exploration: Seismic Exploration, vol. 30, pp. 27-53, 2001.

[73] A.D. Dongare, R.R. Kharde and A. D. Kachare, "Introduction to Artificial Neural Network," International Journal of Engineering and Innovative Technology (IJEIT), vol. 2, no.1, 2012.

[74] M. C. Popescu, V. E. Balas, L. P. Popescu and N. Mastorakis, "Multilayer Perceptron and Neural Networks," Wseas Transactions on Circuits and Systems, vol. 8, no.7, 2009.

[75] K. A. Eşidir, Y. E. Gür, and A. Yoğunlu, "Estimation of Elaziğ Tourism Satisfaction Analysis Results Using with Multilayer Perceptron (MLP) and Radial Basis Function (RBF) Prediction Models," Journal of Social and Humanities Sciences Research, vol. 9, no. 86, pp.1682-1703, 2022.

[76] H. Abdel, and Tamer H. M. Soliman, "A Multi-Layer Perceptron (MLP) Neural Networks for Stellar Classification: A Review of Methods and Results," International Journal of Advances in Applied Computational Intelligence (IJAACI), vol. 3, no. 2, pp. 29-37, 2023.

[77] K. Y. Chan, B. Abu-Salih and R. Qaddoura, "Deep neural networks in the cloud: Review, applications, challenges and research directions," Neurocomputing, vol. 545, 2023.

[78] T. Menzies, E. Kocagüneli and B. Turhan, "Sharing Data and Models in Software Engineering," Sharing Data and Models. Ch. 3, Book 2015.

[79] S. Sharma, S. Sharma and A. Athaiya, "Activation Functions in Neural Networks," International Journal of Engineering Applied Sciences and Technology, Vol. 4, no. 12, pp. 310-316, 2020.

[80] B. Ko, H. G. Kim, K.J. Oh and H. J. Choi, "Controlled dropout: A different approach to using dropout on deep neural network," IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 358-362, 2017.

[81] C. Garbin, X. Zhu and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," Multimedia Tools and Applications, vol. 79, pp.12777-12815,2020.

[82] M. A. Mercioni and S. Holban, "The Most Used Activation Functions: Classic Versus Current," International Conference on Development and Application Systems (DAS), pp. 141-145, 2020.

[83] D. Susmita, A. Tariq, T. Santos, S. S. Kantareddy and I. Banerjee, "Recurrent Neural Networks (RNNs): Architectures, Training Tricks, and Introduction to Influential Research Machine Learning for Brain," Neuromethods, vol. 197, pp. 117-138, 2023.

[84] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. AbdElatif Mohamed, H. Arshad, "State-of-the-art in artificial neural network applications: A survey," Heliyon, vol. 4, no. 11, 2018.

[85] A. Tealab, "Time Series Forecasting using Artificial Neural Networks Methodologies: A Systematic Review," Future Computing and Informatics, vol. 3, no. 2, pp. 334-340, 2018.

[86] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, pp. 855-868, 2009.

[87] H. Sak, A. Senior and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," Proceedings of the Annual Conference International Speech Communication Association, pp.338-342, 2014.

[88] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," IEEE International Conference Speech and Signal Processing (ICASSP), pp. 4520-4524, 2015.

[89] S. Grossberg, "Recurrent Neural Networks," Scholarpedia, vol. 8, no. 2, pp.1888, 2013.

[90] M. Jovo, K. Slobodan, R. Guberinic and K. Zarko, "Definisanje Maksimalnog Koraka Napredovanja Mehanizovane Hidraulicne Podgrade (Mhp) Za Uslove Rudnika Strmosten" Technical Sciences, vol. 7, no.1, pp. 35-42, 2012.

[91] S. Yanagawa, "Basic Unit: As a Common Module of Neural Networks," Electrical Science and Engineering, vol. 3, no.1, 2021.

[92] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp 1735-1780, 1997.

[93] S. Fernandez, A. Graves, J. Schmidhuber, "An Application of Recurrent Neural Networks to Discriminative Keyword Spotting," Artificial Neural Networks (ICANN), vol. 4669, pp. 220- 229, 2007.

[94] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," Neural Networks, vol. 61, pp. 85-117, 2015.

[95] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates and A. Y. Ng, "Deep Speech: Scaling up End-To-End Speech Recognition," Computation and Language, vol. 1, 2014.

[96] B. Fan, L. Wang, F. K. Soong and L. Xie, "Photo-Real Talking Head with Deep Bidirectional LSTM," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4884-4888, 2015.

[97] Z. Heiga, Y. Agiomyrgiannakis, N. Egberts, F. Henderson and P. Szczepaniak, "Fast, Compact, and High-Quality LSTM-RNN Based Statistical Parametric Speech Synthesizers for Mobile Devices," Computer Science and Sound, vol. 1, 2016.

[98] J. Oruh, S. Viriri and A. Adegun, "Long Short-Term Memory Recurrent Neural Network for Automatic Speech Recognition," IEEE Access, vol. 10, pp. 30069-30079, 2022.

[99] O. Vinyals, A. Toshev, S. Bengio, D. and Erhan, " Show and Tell: A Neural Image Caption Generator," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3156-3164, 2015.

[100] K. H. Kim, B. Chang, and H. K. Choi, "Deep Learning Based Short-Term Electric Load Forecasting Models using One-Hot Encoding," Journal of IKEEE, vol. 23, no. 3, pp. 852-857, 2019.

[101] A. Aldelemy, and R. A. Abd-Alhameed, "Binary Classification of Customer’s Online Purchasing Behavior Using Machine Learning," Journal of Techniques, vol. 5, no. 2, pp. 163-186, 2023.

[102] B. M. Alsafy, Z. Mosad and W. K. Mutlag, "Multiclass Classification Methods: A Review," International Journal of Advanced Engineering Technology and Innovative Science (IJAETIS), vol.5, no.3, pp. 1-10, 2020.

[103] H. J. Liao, C. H. Richard Lin, Y. C. Lin and K. Y. Tung, "Intrusion Detection System: A Comprehensive Review," Journal of Network and Computer Applications, vol. 36, no. 1, pp. 16-24, 2013.

[104] A. Khraisat and A. Alazab, "A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Validation Strategy, Attacks, Public Datasets and Challenges," Cybersecurity, vol.4, no.18, 2021.

[105] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani," Detailed Analysis of the KDD CUP 99 data set," IEEE Symposium, Computational Intelligence for Security and Defense Applications (CISDA), vol. 2, 2009.

[106] C. Ieracitano, A. Adeel, F. C. Morabito and A. Hussain, "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach," Neurocomputing, vol. 387, pp. 51-62, 2020.

# PUBLICATIONS FROM THE THESIS

## Conference Papers

1. Nadia ALSABBAGH and Hasan Hüseyin BALIK” Detection of the Network Intrusion Traffic using Deep Learning”, In the Proceedings of the MFBK (3rd INTERNATIONAL CONGRESS ON ENGINEERING AND SCIENCES), at the Virtual Conference of 3rd INTERNATIONAL CONGRESS ON ENGINEERING AND SCIENCES, Istanbul, Turkey.11th - 12th-MAY-2024, pp. 300 Publisher Certificate No: 52866
E-ISBN: 978-625-6471-27-6
https://drive.google.com/file/d/1PdN3851T6_2xkGjCU1Idseb9ipyte688/view?usp=sharing