



الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة قاصدي مرباح ورقلة

كلية التكنولوجيات الحديثة للمعلومات والاتصال

قسم الاعلام الآلي وتكنولوجيا المعلومات

Republic of Algeria Democratic and People's

Ministry of Higher Education and Scientific Research

University of KASDI Merbah – Ouargla

Faculty of New Technologies of Information and Communication

Science Computer of Department and Information Technologies

## Professional Master Thesis

domain: Mathematics and Computer Science

Sector: Computer Science

specialty: Network Administration and Security

### Theme

# An Intrusion Detection System Based on Federated Deep Learning

**Presented by: Tedjini Abdeldjalil and Chenine Mustapha**

**Publicly discussed: 23/06/2024**

**Jury member:**

Dr. BENBEZIANE Mohammed

President

UKM Ouargla

Dr. BENKADDOUR Mohammed Kamel

Supervisor

UKM Ouargla

Dr. KAHLESSENANE Fares

Examiner

UKM Ouargla

**Academic year: 2023/2024**

# Acknowledgment

“

« الحمد لله الذي هدانا لهذا وما كنا لنهتدي لولا أن هدانا الله » سورة الاعراف اية 43

*The first and last thing is to thank Allah, who provided us with sufficient capacity to finish this work. Secondly, we wish to express our heartfelt gratitude to our supervisor, **Dr. BENKADDOUR Mohammed Kamel**, for his continuous support, guidance, and encouragement in accomplishing this work. He consistently contributed to solving the problems we encountered in our work. His guidance throughout the research and writing of our thesis has been invaluable. We also extend our sincere thanks to our co-supervisor, **Dr. ABID Malika**, who tirelessly assisted us in completing this research. We are profoundly grateful to the members of the review committee for accepting to review our work. We are honored by your presence and thank you in advance for your comments.*

*Thirdly, words cannot express our gratitude and appreciation to our beloved parents for their support in our daily lives, their constant guidance, and their unwavering love and encouragement. A special thanks to our families and friends for always standing by our side. Their love has been the main source of spiritual support in our lives. Finally, we would like to thank everyone who has been a source of support and encouragement, helping us to achieve our goal and successfully complete our thesis.*

”

## ملخص

لقد أتاح تطور الشبكات وتقنياتها إمكانية تعرض أجهزتنا وبياناتنا للاختراق بشكل متزايد، خاصةً أثناء مشاركة البيانات وتوزيعها. يمكن أن تكون أنظمة كشف التسلل (IDS) القائمة على نهج التعلم الآلي (ML) والتعلم العميق (DL) حلاً لهذه المشكلة للتعامل مع هذه الهجمات والتهديدات. ومع ذلك، مع وجود العديد من الأجهزة على شبكتنا، قد نحتاج إلى مشاركة البيانات مع الخادم لجمعها وتدريبها لبناء نموذجنا، وهو أمر لا يزال محفوفاً بالمخاطر على الخصوصية والسرية. يعد التعلم الاتحادي (FL) حلاً مناسباً لهذه المشكلة، والذي يضمن عدم مشاركة البيانات مع الخادم للتدريب. بدلاً من ذلك، يسمح لنا بالتدريب محلياً وبناء نموذجنا. ثم يتم إرسال هذا النموذج إلى الخادم ليتم تجميعه وتحديثه لبناء نموذج جديد، وهكذا. في هذه الدراسة، نقترح في هذه الدراسة نظاماً موحدًا قائمًا على التعلم للكشف عن التسلل باستخدام خوارزمية الشبكة العصبية التركيبية (CNN). أظهرت هذه الخوارزمية نتائج جيدة مع مجموعة بيانات UNSW-NB15 في كل من النهجين المركزي واللامركزي، كانت النتائج قريبة من الأفضل بالنسبة للأعمال الحالية، مع ضمان خصوصية البيانات وأمن النموذج في النهج اللامركزي.

---

### كلمات مفتاحية :

التعلم الموحد، نظام كشف التسلل (IDS)، FedAvg CNN UNSW-NB15.

---

# Abstract

The development of networks and their technology has made it possible for our devices and data to be increasingly compromised, especially during data sharing and distribution. Intrusion detection systems (IDS) based on machine learning (ML) and deep learning (DL) approach can be a solution to this problem to deal with these attacks and threats. However, with many devices on our network, we may need to share data with the server for collection and training to build our model, which is still very risky for privacy and confidentiality. Federated learning (FL) is a suitable solution to this problem, which ensures that data is not shared with the server for training. Instead, it allows us to train locally and build our model. This model is then sent to the server to be aggregated and updated to build a new model, and so on. In this study, we propose a unified learning-based intrusion detection system using a neural network algorithm (CNN). This algorithm showed good results with the UNSW-NB15 dataset. In both the central and decentralized approaches, the results were close to better for current works, while ensuring data privacy and model security in the decentralized approach.

---

**Keywords :** Federated learning, IDS, UNSW-NB15, CNN ,FedAvg.

---

# Résumé

Le développement des réseaux et de leur technologie a permis à nos appareils et à nos données d'être de plus en plus compromis, notamment lors du partage et de la distribution des données. Les systèmes de détection d'intrusion (IDS) basés sur l'apprentissage automatique (ML) et l'apprentissage profond (DL) peuvent constituer une solution à ce problème pour faire face à ces attaques et menaces. Cependant, avec de nombreux appareils sur notre réseau, nous pouvons avoir besoin de partager des données avec le serveur pour la collecte et l'entraînement afin de construire notre modèle, ce qui est toujours très risqué pour la vie privée et la confidentialité. L'apprentissage fédéré (FL) est une solution appropriée à ce problème, qui garantit que les données ne sont pas partagées avec le serveur pour la formation. Au lieu de cela, il nous permet de nous entraîner localement et de construire notre modèle. Ce modèle est ensuite envoyé au serveur pour être agrégé et mis à jour afin de construire un nouveau modèle, et ainsi de suite. Dans cette étude, nous proposons un système de détection des intrusions basé sur l'apprentissage fédéré et utilisant un algorithme de (CNN). Cet algorithme a donné de bons résultats avec l'ensemble de données UNSW-NB15. Dans les approches centralisée et décentralisée, les résultats sont proches ou meilleurs que ceux des travaux actuels, tout en garantissant la confidentialité des données et la sécurité du modèle dans l'approche décentralisée.

---

**Mots clés :** Apprentissage fédéré, IDS, UNSW-NB15, CNN FedAvg

---

# Contents

<b>Acknowledgment</b> . . . . .	<b>I</b>
<b>II</b> . . . . .	<b>ملخص</b>
<b>Abstract</b> . . . . .	<b>III</b>
<b>Résumé</b> . . . . .	<b>IV</b>
<b>General Introduction</b> . . . . .	<b>1</b>
<b>1 Information security and intrusion detection systems</b> . . . . .	<b>4</b>
1.1 Introduction . . . . .	5
1.2 Information Security . . . . .	5
1.2.1 Definition . . . . .	5
1.2.2 The Pillars of Information Security . . . . .	5
1.2.3 The Goals of Security . . . . .	7
1.2.4 Definition of attack . . . . .	7
1.2.5 Categories of attacks . . . . .	7
1.2.5.1 Attacks based on effect . . . . .	7
1.2.5.2 Attacks based on source . . . . .	8
1.2.6 Examples of Common Attacks . . . . .	8
1.2.7 Defense mechanisms against Cyber attacks . . . . .	9
1.3 Intrusion Detection System . . . . .	10
1.3.1 A brief history of intrusion detection system . . . . .	10
1.3.2 Definition of Intrusion Detection System . . . . .	11
1.3.3 Intrusion detection system architecture . . . . .	11
1.3.4 Mechanism of Intrusion Detection Systems (IDS) . . . . .	12
1.3.5 Placement of IDS . . . . .	12
1.3.6 Alarm types for intrusion detection system (IDS) . . . . .	13
1.3.7 Taxonomy of Intrusion Detection Systems . . . . .	14
1.3.7.1 Classification of Intrusion detection systems . . . . .	14
1.3.7.2 Intrusion detection methods . . . . .	15
1.3.7.3 Response of intrusion detection . . . . .	16
1.3.8 Firewall versus intrusion detection system (IDS) . . . . .	16
1.3.9 Advantages of using IDS . . . . .	17
1.3.10 Limits of intrusion detection system . . . . .	17
1.4 Conclusion . . . . .	18

<b>2</b>	<b>State of the Art</b>	<b>19</b>
2.1	Introduction	20
2.2	Overview artificial intelligence (AI)	20
2.2.1	Machine learning	21
2.2.1.1	The Methods of Machine Learning	21
2.2.2	Deep learning	22
2.2.3	Deep learning approaches	23
2.2.3.1	Deep neural network(DNN)	23
2.2.3.2	Recurrent neural network(RNN)	24
2.2.3.3	Long Short-Term Memory(LSTM)	25
2.3	Intrusion detection system datasets	26
2.4	Evaluation metrics	28
2.5	Related work	29
2.6	Conclusion	32
<b>3</b>	<b>Federated Deep Learning for intrusion detection system</b>	<b>33</b>
3.1	Introduction	34
3.2	Centralized learning	34
3.3	Decentralized learning	35
3.4	The approach used in this work	35
3.4.1	Convolutional neural network(CNN)	35
3.4.1.1	Layers of convoultion neural networks	36
3.4.1.2	Activation functions	38
3.5	Decentralized Federated Learning (FL)	40
3.5.1	Federated learning process	41
3.5.2	Categorization of federated learning	43
3.5.3	Model Aggregation algorithms	45
3.5.4	Recent developments in federated Learning	46
3.6	Conclusion	47
<b>4</b>	<b>Experiment, Results and Discussion</b>	<b>48</b>
4.1	Introduction	49
4.2	Experimental Environment	49
4.2.1	Material Tools	49
4.2.2	programming language used	49
4.2.2.1	Libraries used	49
4.3	dataset description	51
4.4	Preprocessing dataset	51
4.5	Dataset distribution	53
4.5.1	Independently and identically distributed (IID)	53
4.5.2	Non Independently and identically distributed (Non IID)	54
4.6	Centralized Model based Deep Learning	55
4.6.1	Hyperparameter Settings of CNN model	56
4.6.2	Architecture of local Model Training	58
4.6.3	Cross-validation in model CNN	59
4.7	Architectures of Decentralized Model FedCNN-IDS	60

4.7.1 Federated learning Parameters . . . . .	61
4.8 A proposed aggregation strategy . . . . .	61
4.9 Results in IID and Non-IID Distributions . . . . .	62
4.10 Discussion . . . . .	65
4.11 Conclusion . . . . .	66
<b>General Conclusion . . . . .</b>	<b>67</b>
<b>Bibliography . . . . .</b>	<b>68</b>
<b>Appendices . . . . .</b>	<b>75</b>
<b>A Acceptance certificates in this work . . . . .</b>	<b>76</b>
<b>B Dataset . . . . .</b>	<b>79</b>



# List of Figures

1.1	The Pillars of Information Security CIA . . . . .	6
1.2	Intrusion detection system architecture . . . . .	12
1.3	Placement of IDS on the network . . . . .	13
1.4	Alarm types . . . . .	13
1.5	Taxonomy of IDS . . . . .	14
2.1	Artificial intelligence, machine learning and deep learning . . . . .	21
2.2	Neural network architecture . . . . .	23
2.3	A schematic of Deep neural network(DNN) . . . . .	24
2.4	A schematic of Recurrent neural network(RNN) . . . . .	24
2.5	A schematic of Long Short-Term Memory(LSTM) . . . . .	25
3.1	Centralized learning architecture . . . . .	34
3.2	Decentralized learning architecture . . . . .	35
3.3	Process of convolution layer . . . . .	36
3.4	Process of max pooling layer . . . . .	37
3.5	Process of Flattening layer . . . . .	37
3.6	Activation Function ReLU . . . . .	38
3.7	Activation Function sigmoid . . . . .	39
3.8	Activation Function softmax . . . . .	39
3.9	Federated Learning architecture . . . . .	41
3.10	Rounds between server and client . . . . .	43
3.11	Vertical Federated Learning[80] . . . . .	44
3.12	Horizontal Federated Learning[80] . . . . .	44
3.13	Federated Transfer Learning[80] . . . . .	45
4.1	The step of preprocessing dataset . . . . .	53
4.2	Independently and identically distributed (IID) . . . . .	54
4.3	Non Independently and identically distributed (IID) . . . . .	55
4.4	A comparison of the accuracy outcomes achieved by various model. . . . .	56
4.5	Summary of CNN model . . . . .	57
4.6	Architecture of local CNN Model Training . . . . .	58
4.7	Architecture of cross validation . . . . .	59
4.8	Accuracy by applying 5-fold cross validation . . . . .	59
4.9	Architectures of proposed Model FedCNN-IDS in federated learning . . . . .	60
4.10	The results of testing different combinations of aggregation strategies . . . . .	62
4.11	The accuracy of the model between centralized and federated learning in the IID distribution. . . . .	63

4.12 The accuracy of the model between centralized and federated learning in  
the Non-IID distribution. . . . . 64

# List of Tables

2.1	Summary of dataset . . . . .	27
2.2	Summary of dataset . . . . .	32
4.1	Version of libraries used in work environment . . . . .	50
4.2	Percentage distribution of attack types in the dataset . . . . .	51
4.3	A comparison between the accuracy results of the different approaches . . . . .	56
4.4	Architectures and parameter of CNN model . . . . .	57
4.5	Hyper parameter settings of federated learning architecture . . . . .	61
4.6	Performance comparison of our proposed techniques with state of arts works. . . . .	66

# List of Algorithms

1	Federated Learning: Client-Side Training at Federated Round $T$ . . . . .	42
2	Federated Learning: server-side aggregation $T$ . . . . .	42

# List of abbreviations

<b>AI</b>	<i>Artificial Intelligence</i>
<b>CNN</b>	<i>Convolutional Neural Network</i>
<b>DL</b>	<i>Deep learning</i>
<b>DARPA</b>	<i>Defense Advanced Research Projects Agency</i>
<b>DNN</b>	<i>Deep Neural Network</i>
<b>DoS</b>	<i>Denial Of Service</i>
<b>DMZ</b>	<i>DeMilitarized Zone</i>
<b>DDoS</b>	<i>Distributed Denial of Service</i>
<b>FL</b>	<i>Federated Learning</i>
<b>FedAvg</b>	<i>Federated Averaging</i>
<b>FedProx</b>	<i>Federated Proximal</i>
<b>FedAdagrad</b>	<i>Federated Adaptive Gradient</i>
<b>FedAdam</b>	<i>Federated Adaptive Moment</i>
<b>FP</b>	<i>False Positive</i>
<b>FN</b>	<i>False Positive</i>
<b>HIDS</b>	<i>Host based Intrusion Detection System</i>
<b>IDS</b>	<i>Intrusion Detection System</i>
<b>IID</b>	<i>Independently and identically distributed</i>

<b>IP</b>	<i>Internet Protocol</i>
<b>IoT</b>	<i>Internet of Things</i>
<b>KDD 99</b>	<i>Knowledge Discovery in Databases 99</i>
<b>LSTM</b>	<i>Long Short Term Memory</i>
<b>ML</b>	<i>Machine learning</i>
<b>NIDS</b>	<i>Network based Intrusion Detection System</i>
<b>NPA</b>	<i>Network behavior Analysis</i>
<b>R2L</b>	<i>Root To Local</i>
<b>ReLU</b>	<i>Rectified Linear Units</i>
<b>RNN</b>	<i>Recurrent Neural Network</i>
<b>SVM</b>	<i>Support Vector Machine</i>
<b>TP</b>	<i>True Positive</i>
<b>TN</b>	<i>True Negative</i>
<b>U2R</b>	<i>User To Root</i>
<b>WIDS</b>	<i>Wireless intrusion detection systems</i>

# General Introduction

The exponential growth in computer science has led to the generation of big data, necessitating increased security measures to protect this data (corporate data, research data, patient data in hospitals) and defend against hacking attacks (viruses, worms, Trojan horses, DoS, etc.) that are becoming increasingly common for almost everyone working with an Internet-connected device. These attacks are used to illegally access unauthorized information, damage or change the content of information, or reduce the availability of information to authorized users.

One of the most important and widely used security systems for securing networks and detecting cyber-attacks is known as Network Intrusion Detection Systems (NIDS) [1]. This type of technology allows network users to identify incoming Internet threats by monitoring and analyzing network traffic. It collects packets and scans them for malicious content and behavior. Once a suspicious event is detected, the security process takes action to warn the administrator by writing to log files, which the administrator can read to detect potential intrusions. The advent of artificial intelligence (AI), particularly machine learning (ML) algorithms, has revolutionized the field of cybersecurity. As cyber threats continue to evolve in complexity and frequency, traditional security measures such as firewalls and signature-based detection systems have proven insufficient in identifying and mitigating advanced persistent threats (APTs) and zero-day attacks. Intrusion detection systems (IDS) have long been a critical component of network security, designed to monitor network traffic for suspicious activity and potential intrusions. However, machine learning algorithms have revolutionized how IDSs work, enhancing their effectiveness and efficiency.

Machine learning algorithms excel at pattern recognition and anomaly detection, making them exceptionally well-suited for intrusion detection. Unlike traditional intrusion detection systems that rely on predefined rules and signatures to identify threats, ML-based intrusion detection systems can analyze massive amounts of network data in real-time, learning from historical and current data to identify abnormal behavior that may indicate a security breach. This capability allows previously unknown threats to be detected, significantly improving the system's ability to protect against new attacks. Moreover, the adaptability of ML algorithms enables IDS to continuously improve their detection accuracy. As they process more data, these algorithms refine their models, reducing false positives and ensuring that genuine threats are identified with greater precision.

Security issues represent the biggest challenge in addressing network traffic attacks and anomaly detection. Despite the revolution of machine learning algorithms for anomaly detection, traditional anomaly detection systems often depend on a centralized server for

monitoring and control. All distributed nodes must share their data with a central server for processing and analysis. This approach has several drawbacks, including poor privacy and security, the need for large storage and computing resources, and the risk of a single point of failure.

Federated learning offers a promising solution to these challenges by eliminating the need to upload private or sensitive information to centralized servers and using distributed training to mitigate resource requirements. The goal of this Thesis is to develop a federated learning based intrusion detection system (FL-IDS) capable of detecting attacks while minimizing false alarms.

## Research Motivation

A lot of research has been done on machine learning-based intrusion detection systems (IDS), looking at things like data preprocessing, feature engineering, learning methods, and performance metrics to show how well these systems work. However, with constantly evolving network communication environments and complex network attacks, there is an urgent need to design more effective and efficient intrusion detection and classification systems. Therefore, it is necessary to address various factors, such as improved learning capabilities, data sample distribution, and optimized feature representation, to achieve optimal intrusion detection and classification results.

This necessity emphasizes the importance of developing robust intrusion detection and classification systems to ensure the security of networks, configured devices, and data. Thus, this motivation drives our efforts to design an effective intrusion detection and classification system while maintaining data confidentiality and privacy.

## Research Objective

Improve the system's precision and dependability in identifying genuine threats and irregularities in network traffic, guaranteeing the effective identification and resolution of genuine security issues.

Reduce the number of false alarms triggered by the system, helping to minimize unnecessary interruptions and focus on real security threats, thus improving overall system efficiency and reliability.

Ensure the confidentiality and privacy of sensitive data, prevent unauthorized access; and maintain the privacy and integrity of the data being processed and analyzed.

Detect and identify any malicious actions, suspicious behaviors, or unusual patterns that may indicate a security breach or cyberattack, enabling timely and effective responses to protect the system.



## Research Contributions

Most research in the field of intrusion detection systems (IDS) focuses on enhancing accuracy and reducing the occurrence of false positives in models. Our contribution is the development of IDS models that prioritize privacy and security while enhancing accuracy and minimizing false positives through the use of federated learning, with a particular focus on handling non-IID data distribution.

## Roadmap of the Thesis

The roadmap of the thesis is listed as follows.

- **Chapter 1** presents the different aspects of computer security, mentioning the most common types of attacks, highlighting the most important systems and means of protection against attacks, and providing a detailed description of intrusion detection systems, their different types, and their principles of operation.
- **Chapter 2** discusses the literature review for IDSs, as well as the research results. This chapter also reveals an overview of the intrusion detection datasets used for experimentation as well as the evaluation metrics used to show the performance of the designed intrusion detection systems. Based on the insights gained, different solutions are proposed and discussed in successive chapters.
- **Chapter 3** of the thesis focuses on federated deep learning for intrusion detection systems (IDS). It discusses the limitations of centralized learning, introduces the processes of decentralized and federated learning, and details the proposed approach of CNN. It also covers recent developments in federated learning, highlighting its benefits to intrusion detection systems in terms of enhancing privacy, reducing resource requirements, and eliminating single points of failure.
- **Chapter 4** describes the tools and environment used in this work. It also presents the experimental results of implementing federated learning in an intrusion detection system and compares the results between centralized and decentralized learning.

# Chapter 1

## Information security and intrusion detection systems

## 1.1 Introduction

As technology has evolved and new communication methods have emerged, people worldwide can now share various types of data across networks. However, this increased connectivity creates a major problem: unauthorized individuals might access and manipulate sensitive information.

Strong information security practices, especially those that focus on identification and access control, have become essential to mitigate these risks and keep systems secure.

One such cornerstone technology within the information security domain is the intrusion detection system (IDS). By continuously monitoring for malicious activity, IDS systems provide real-time detection capabilities, enabling the identification and prevention of security breaches.

This first chapter establishes a foundational understanding of information security principles and intrusion detection systems. We commence by defining core concepts in information security, followed by an exploration of various cyber attack classifications with illustrative examples. Subsequently, the second part delves into the intricacies of intrusion detection systems, encompassing their historical development, core definitions, operational principles, and established classification criteria.

## 1.2 Information Security

### 1.2.1 Definition

There have been a number of studies that have attempted to define the concept of security. However, as previous authors have noted, security is inherently multi-dimensional and diverse in practice. This diversity makes it difficult to provide a single, comprehensive definition that encompasses the many different application areas of security. In this paper, we develop a relatively comprehensive definition that is consistent with our work:

Information security is a multidisciplinary approach aimed at preserving the confidentiality, integrity, and availability of digital information assets. It involves the implementation of policies, procedures, controls, and technologies to mitigate risks and threats to information systems, networks, and data. Information security encompasses various domains, including but not limited to, access control, encryption, authentication, data loss prevention, incident response, and security awareness training.

The ultimate goal of information security is to ensure that sensitive information remains protected from unauthorized access, manipulation, or destruction, thereby safeguarding the interests of individuals, organizations, and society as a whole.

### 1.2.2 The Pillars of Information Security

The pillars of information security are confidentiality, integrity, and availability. These pillars form the foundation of all security systems and are essential for ensuring the

protection of information.

Here, we will define each pillar and discuss its importance in ensuring information security.

- **Confidentiality:** Confidentiality refers to safeguarding the data the information from unauthorized clients. There may be some users or clients who might want unnecessary access to the data of other users and may use it for some illegal work. There is possibility that the person may change details of others without authority [2].
- **Integrity:** Integrity refers to the guarantee of authenticity of the data. The data shouldn't be modified by a third-party who hasn't been given the authority to carry out such activities. Consistency, accuracy and trustworthiness of the data must be maintained throughout its lifecycle. The most common way of ensuring the integrity of the data is to make sure back-ups and redundancies are available in case of any data loss or damage [2].
- **Availability:** Availability in computer science, information technology, and applications is a complex concept that has been studied extensively in the context of functions and performance, such as computer networks, information processing systems, databases, and data storage. Here are some definitions of availability: "Enable access to authorized information or resources to those who need them" [3]. "Timely, reliable access to data and information services for authorized users"[4]. "An authorized party should not be prevented from accessing objects to which he, she, or it has legitimate access" [5].

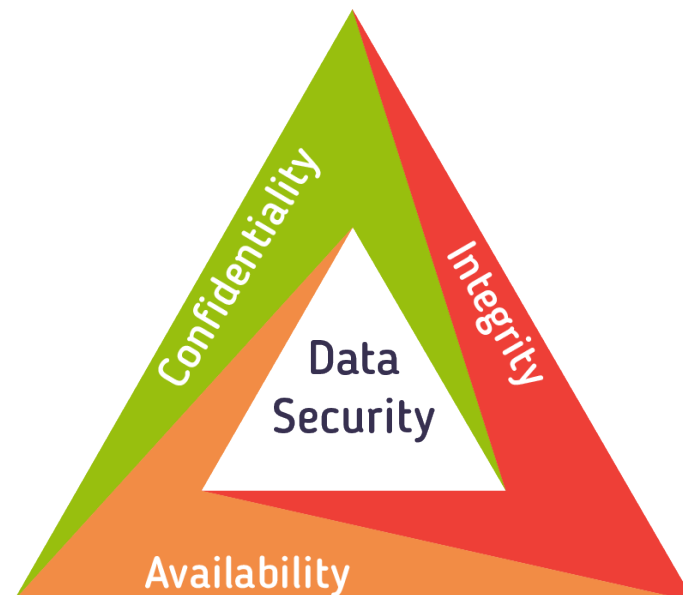


Figure 1.1: The Pillars of Information Security CIA

### 1.2.3 The Goals of Security

Given a security policy is specification of secure and non-secure actions, these security mechanisms can prevent the attack, detect the attack, or recover from the attack. The strategies may be used together or separately[6].

**Prevention:** means that an attack will fail, prevention involves implementation of mechanisms that users cannot override and that are trusted to be implemented in a correct, unalterable way, so that the attacker cannot defeat the mechanism by changing it.

**Detection:** is most useful when an attack cannot be prevented, but it can also indicate the effectiveness of preventative measures. Detection mechanisms accept that an attack will occur; the goal is to determine that an attack is under way, or has occurred, and report it [6].

**Recovery:** has two forms. The first is to stop an attack and to assess and repair any damage caused by that attack, In a second form of recovery, the system continues to function correctly while an attack is under way [6].

### 1.2.4 Definition of attack

An attack is any type of offensive manoeuvre that targets information systems, infrastructure, computer networks, or personal computers and is characterized by their penetration using various means of malicious activities. These attacks aim to disrupt, destroy, or gain unauthorized access to targeted systems, affecting availability, confidentiality, and integrity[7].

### 1.2.5 Categories of attacks

From an information security standpoint, a threat refers to a possible violation of security, this threat can manifest as either inadvertent, purposeful (in the form of an attack), or as either active or passive.

The literature presents many categories of attacks based on different characteristics, including as:

#### 1.2.5.1 Attacks based on effect

Depending on the effects resulting from the attack we can classify the attacks into two main groups: passive attacks and active attacks.

- **Passive attacks:** consist of accessing, using or observing the target system without modifying data or malfunctioned resources of the latter, they are generally undetectable (e.g. content capture, traffic analysis).
- **Active attacks:** consist of making unauthorized changes to the data of systems, to break into network equipment or to disrupt their operations, attacks of this type are obviously more dangerous (e.g.: masquerade and denial of service [DOS]).

### 1.2.5.2 Attacks based on source

- **Internal attacks:** also known as insider threats, are security breaches perpetrated by individuals with authorized access to a system or network. Internal attacks pose a significant threat due to the insider's familiarity with the system's vulnerabilities.
- **External attacks:** are security breaches initiated by individuals or groups outside an organization's network or system. These attacks pose a significant threat as attackers constantly develop new methods to exploit vulnerabilities and gain unauthorized access to sensitive information.

### 1.2.6 Examples of Common Attacks

There are a huge number of attacks that threaten computer systems and networks, however, most of them are just variations of others. Here are examples of the most well-known attacks targeting computer networks today.

1. **Denial of Service (DOS) Attacks:** is a computer attack aimed at purpose of making a service unavailable, of preventing legitimate users of a service from using it. He could be: Flooding of a network in order to prevent its operation The disruption of connections between two machines, preventing access to a particular service Obstruction of access to a service to a particular person Also sending billions of bytes to an internet box. The denial of service attack can thus block a file server, make access to a web server impossible or prevent the distribution of email in a company, the main attacks that can be found are Apache2, Back, Land, Mail bomb, SYN Flood, Ping of death, Process table, Smurf, Syslogd, Teardrop, UDP storm.
2. **Probing:** the attacker of this class begins with a survey of the future victim, what we call scan, this survey will scan each IP port in order to know the services offered by the system (OS, network topology, protections used, etc.) once completed, the intruder's machine (the one carrying out the intrusion) then attempts to identify the operating system used by this victim and to exploit the information, it has collected. This class of attack is the most extensive and requires minimal technical expertise. Examples of this type of attack are: Ipsweep, Mscan, Nmap, Saint, Satan.
3. **User to Root attacks(U2R):** the objective of this class of attacks is to obtain control of the system administrator (Root) from a simple user account by exploiting vulnerabilities, the most common exploits The best known are regular Buffer overflows due to programming errors. The main attacks of this type are: Eject, Ffbconfig, Fdformat, Load module, Perl, Ps, Xterm.
4. **Remote to User attack (R2L):** in this class of attack, the attacker tries to exploit the vulnerabilities of a remote machine in order to have illegal access to the latter. To succeed in this attack, the attacker exploits bugs in applications installed on the target machine, poor configurations of these and the system that hosts them, etc.

5. **IP Spoofing:** the operating principle of this attack is to send IP packets using a source IP address which has not been allocated to the computer which sends these packets for the purpose of hiding the identity of the attacker during an attack on a server or any target in the network, or to impersonate other network equipment to benefit from the services to which he has access.
6. **Network analyzers (Sniffer):** is a device allowing “listening” to the traffic of a network, that is to say, capturing the information circulating there, given that the data in a network not switched are sent to all machines on the network and in normal use the machines ignore packets that are not intended for them. The sniffer can also serve this property to a malicious person having physical access to the network to collect information (e.g. passwords), But a sniffer can also be used as a positive tool for the purpose of studying and capturing the network traffic by network administrators and intrusion detectors (IDS)[8].
7. **Backdoor:** in backdoor attack, attacker can bypass the normal authentication and can obtain unauthorized remote access to a system. Attacker tries to locate the data by doing fraudulent activities to bypass the system security of the system. Hacker uses backdoor programs to install the malicious files, modifying the code or gain access to the system or data [9]

### 1.2.7 Defense mechanisms against Cyber attacks

It is the set of procedures or devices that are designed to detect, prevent or recover from attacks that threaten information security, there are several tools to prevent counter-attacks, We have listed below some mechanisms [10]:

1. **Encryption:** algorithms generally based on keys and transforming data, Its effectiveness depends on the security level of the keys.
2. **Digital signature:** data added to verify data integrity or origin.
3. **Traffic jamming:** data added to ensure confidentiality, particularly in terms of traffic volume.
4. **Notarization:** use of a trusted third party to provide certain security services.
5. **Access control:** verifies an actor’s access rights to data. Does not prevent exploitation of a vulnerability.
6. **Antivirus:** software designed to protect the computer against harmful software (or potentially executable files). Does not protect against an intruder using
7. **Firewall:** an element (software or hardware) of the computer network that controls communications passing through it. Its function is to enforce the network’s security policy, which defines which communications are authorized or prohibited.

The firewall does not prevent an attacker from using an authorized connection to attack the system, nor does it protect against an attack coming from the internal network.

8. **Intrusion Detection Systems (IDS):** are security systems designed to monitor network traffic or system activity for malicious activities or suspicious patterns that might indicate an attempt to gain unauthorized access, compromise systems, or steal data.
9. **Security audit:** identification of system vulnerabilities. Does not detect attacks that have already taken place, or when they will take place.  
However, relying solely on one security mechanism is not enough to ensure adequate information security. Therefore, it is common practice to employ multiple mechanisms simultaneously to achieve an acceptable level of security.

## 1.3 Intrusion Detection System

In the process of addressing harmful viruses or unwanted traffic movements to the intranet we usually use a firewall device or intrusion detection system which is our focus in this search and we will provide an overview of it and its most important points.

### 1.3.1 A brief history of intrusion detection system

Intrusion detection technology first appeared in 1980 thanks to James Anderson's studies into monitoring and monitoring computer security threats. In his article submitted to a government agency, he states the theory of creating and developing an intrusion detection system in the future. The government project initiated by SRI (Stanford Research Institute)International and Dr. Dorothy Denning in 1983 marked the beginning of a new initiative to create an intrusion detection system.[11]

In 1984, Dr. Denning contributed to the creation of the Intrusion Detection Expert System (IDES), the initial model for intrusion detection that laid the groundwork for the final development of IDS technology. Utilizing the research and development efforts undertaken by SRI International, Dr. Denning published foundational work, the Intrusion Detection Model, which revealed the necessary information for developing a commercial intrusion detection system. Her research paper formed the basis for much of the subsequent work in IDS. The subsequent iteration of this tool was named the Distributed Intrusion Detection System (DIDS). DIDS improved upon the existing solution by monitoring client machines in addition to the servers they originally monitored.[11]

The early 1990s saw the beginning of commercial development of intrusion detection solutions. The first company to offer IDS tools commercially was Haystack Labs, with its host-based Stalker product line. In addition, SAIC was working on the Computer Misuse Detection System (CMDS), a host-based intrusion detection system. At the same time, the Automated Security Measurement (ASIM) system was created by the Air Force Cryptographic Logic Support Division to oversee network traffic within the USAF network. [11]

Around 1997, the intrusion detection industry began to take off and become profitable. The industry's leading security company, ISS, created the Real Secure network intrusion



detection system that year. Realizing the value of network intrusion detection, a year later Cisco acquired Wheel Group to produce a security solution that it could sell to customers. Likewise, the merger of the development team from Haystack Labs and the departure of the CMDS team from SAIC created the first visual intrusion detection company, Centrex Corporation.[11]

Subsequently, developments and research began in this field, and competition became great to reach a more effective and more accurate system for protecting networks.

### 1.3.2 Definition of Intrusion Detection System

The intrusion detection system is one of the types of software or systems dedicated to network surveillance to detect abnormal behavior or suspicious threats that may harm the network or enterprise, thereby alerting the network officer or competent authority to take the necessary action and procedures to deal with them, distinguishing between attempts to infiltrate and preparing for infiltration from the normal use of the system.

An intrusion is defined as “Intrusion detection is the process of identifying and responding to malicious activities targeting computing and network resources” [12]”, An intrusion attempt, also known as an attack, refers to a series of actions through which an intruder attempts to take control of a system [13], and any set of actions that attempt to compromise the integrity, confidentiality, or availability of a computer resource” [14].

### 1.3.3 Intrusion detection system architecture

The following section focuses on the basic system architecture of a network-based intrusion detection system.

When implementing an Intrusion detection system, it is not possible to completely resort to a common standard due to varying requirements when utilizing the gathering of the data and its analysis. Nevertheless, almost all of them have some basic components/modules that they all share[15]. These components are:

- **Data gathering:** used for monitoring the source environment. The data gathering is performed using different sensors that observe a specific application or protocol.
- **Detector (Detection engine):** is a module that performs the comparison between the gathered data and the defined rules set and raises alarms in case a deviation is found.
- **Database(Knowledgebase):** is a storage module that contains the rule-sets or the IDs which the detector uses when comparing the received data.
- **Output (Response):** when an alarm is raised a proper action is taken. This could be an active response where the IDS performs a predefined action such as drop the packet, or an inactive response such as logging for later inspection by a human factor to determine the appropriate response.

In Figure 1.2 illustrates the architecture of intrusion detection systems.

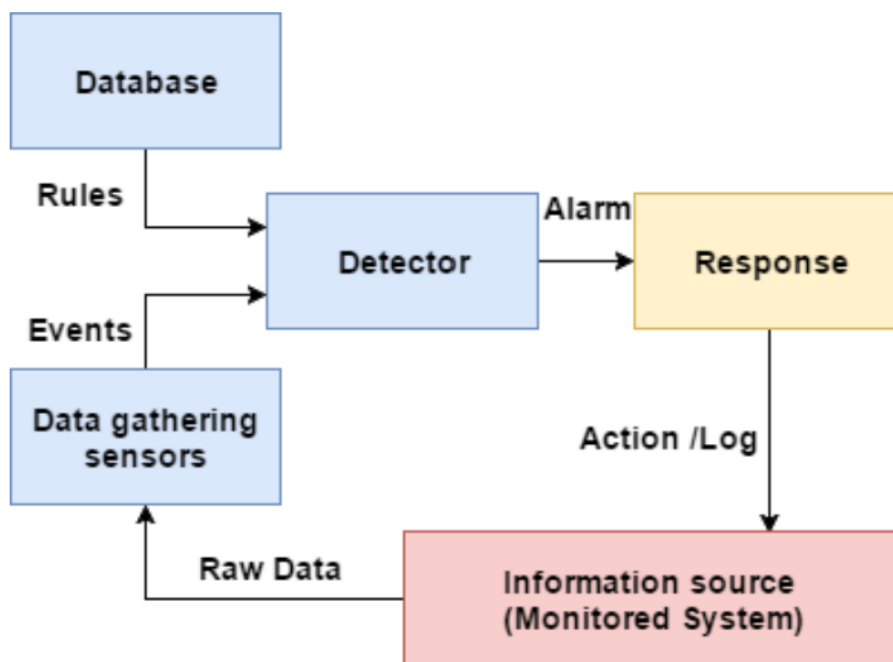


Figure 1.2: Intrusion detection system architecture

### 1.3.4 Mechanism of Intrusion Detection Systems (IDS)

When any data packet enters or there is traffic on the network, it monitors computer network traffic to detect any suspicious activity. It analyzes the data flowing over the network to discover unusual behavior patterns among the data it analyzes or compares, such as source address, destination address, target port, protocol, connection type, etc. An intrusion detection system compares network data against a pre-defined set of rules and patterns to identify any activity that might indicate an attack or intrusion.

If the intrusion detection system detects compliance with one of these rules or patterns, an alert is sent to the system administrator. The system administrator can then verify the alert and take necessary actions to prevent any potential damage or future intrusions.

### 1.3.5 Placement of IDS

Intrusion detection systems are strategically positioned within the network to monitor traffic to and from all devices on it. IDS devices can be configured as servers and placed at the network perimeter before or after the firewall to monitor all traffic, or smaller devices can be installed within network components such as switches, gateways, routers, or even in the demilitarized zone (DMZ) to protect global servers. IDS can also be installed as software on servers or computers.

In Figure 1.3 illustrates the placement of intrusion detection systems in machines and networks.

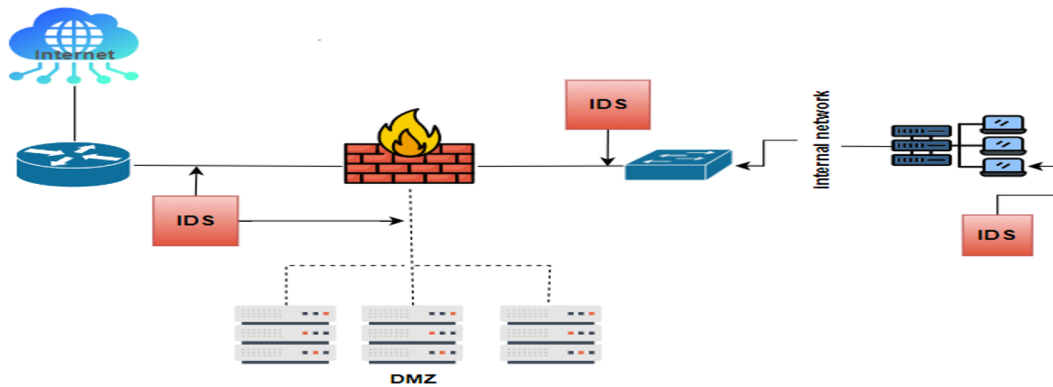


Figure 1.3: Placement of IDS on the network

### 1.3.6 Alarm types for intrusion detection system (IDS)

The IDS takes inputs and categorizes them as normal or malicious. These inputs can be in the form of network traffic, behavioral data packets, and much more. The IDS then applies its detection algorithm and classifies the input as normal behavior or attack, triggering alerts accordingly. Below are points about the various possible alert types along with their meanings:

- **True Positive (TP):** Intrusion traffic is detected as intrusive (Successful identification of attack).
- **False Positive (FP):** Normal traffic is detected as intrusive.
- **True Negative (TN):** Normal traffic is detected as normal (Successful identification of normal traffic).
- **False Negative (FN):** Malicious traffic is detected as normal.

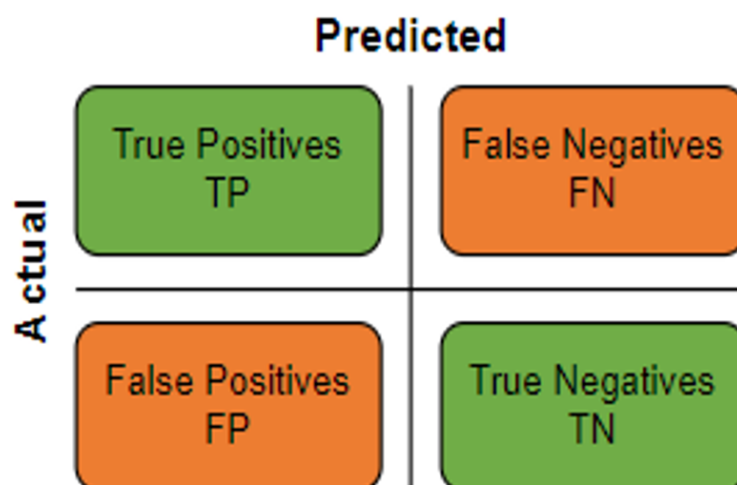


Figure 1.4: Alarm types

### 1.3.7 Taxonomy of Intrusion Detection Systems

The techniques and classifications of intrusion detection system can be illustrated in the following Figure 1.5:

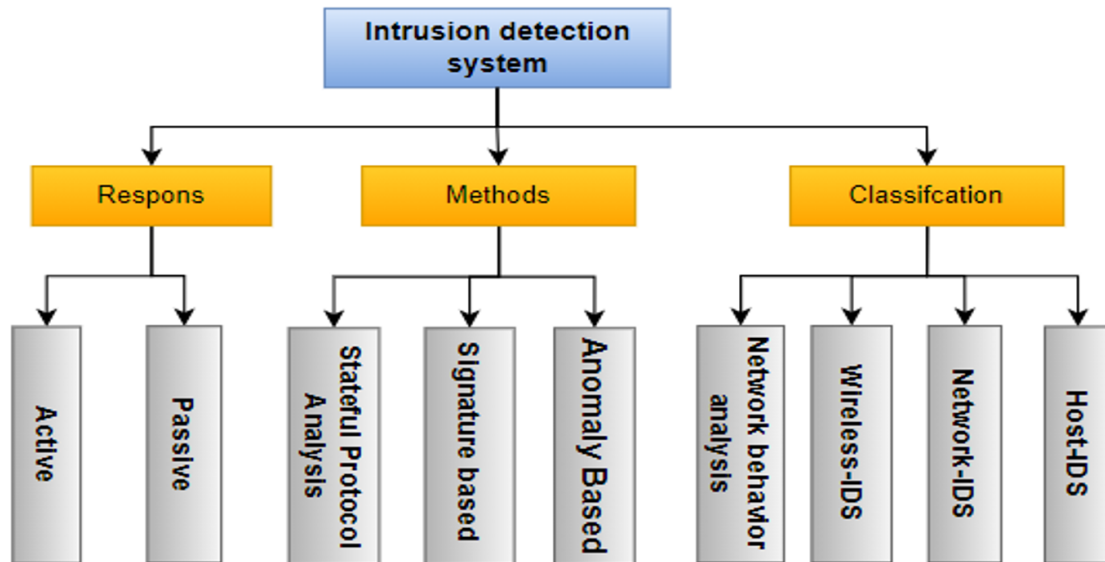


Figure 1.5: Taxonomy of IDS

#### 1.3.7.1 Classification of Intrusion detection systems

can be classified into four different types which are

1. **Host intrusion detection systems (HIDS):** run on individual hosts or devices on a network. HIDS only monitors incoming and outgoing packets from the device and will alert the user or administrator if suspicious activity is detected. HIDS analyzes network traffic and system-specific settings such as software calls, system logs, local security policy, system configuration, local log audits, and more. HIDS must be installed on each device and requires configuration specific to that operating system [16].
2. **Network intrusion detection systems (NIDS):** it analyzes and monitors outgoing and incoming network traffic for suspicious activity using an intrusion detection method. It is placed between different networks, such as locations near routers or firewalls, to monitor data flow between internal and external networks [16].
3. **Wireless intrusion detection systems (WIDS):** it monitors the wireless network for suspicious traffic by analyzing wireless networks and scanning protocol activities. The power of wireless intrusion detection systems lies in the fact that they are the only type of intrusion detection system capable of monitoring wireless protocol activities [16].

4. **Network behavior Analysis (NBA):** it examines network traffic to identify threats that may lead to the generation of unusual traffic flows, such as Distributed Denial of Service (DDoS) attacks, and analyzes network activities at a high level to identify any individual trends or patterns. This type of intrusion detection system is commonly deployed on internal networks but is also sometimes used to monitor flow between internal and external networks. It is most effective in detecting activities that result in significant changes in network behavior. This includes scanning, where the intruder scans the system to map out the targeted network before launching the attack, and Denial of Service (DoS) attacks [16].

### 1.3.7.2 Intrusion detection methods

The IDSs utilized by most organizations rely on one of three basic methods to detect attacks on a network.

This includes anomaly-based detection, stateful protocol analysis and signature based detection. Each method has its respective strength and weaknesses.

1. **Anomaly Based Detection:** anomaly-based detection involves monitoring enterprise network traffic by IDS, which then compares it to predetermined profiles. These profiles are created by enabling the intrusion detection system (IDS) in the "learning mode", enabling it to collect usage information from the system and to use statistical techniques to determine expected user actions. For example, if the number of times a user tries to log into the system during a period that far exceeds the expected number of logins, this will be considered an anomaly. It is then marked as a potential threat and reported to the authorities. User profiles evolve over time, with the IDS continuously collecting information about the network.

Generally speaking, the work of an anomaly detection strategy can be likened to the methodology used by many banks to identify cases of credit card abuse. Bank employees are alerted when the system detects behaviors that deviate from an individual's usual spending patterns. Anomaly-based systems use machine learning techniques to build a baseline for natural behavior. This detection strategy can be critical in addressing emerging risks.

2. **Stateful Protocol Analysis:** stateful protocol analysis is a common and effective detection methodology that operates by comparing predetermined profiles of acceptable protocol behaviors for protocol states against observed activities to detect deviations and misbehaviors. Stateful protocol analysis is a typical detection methodology in IT security [17]. However, a recent survey has shown that approximately 80% of cyber-attacks originate in the application layer [18].

So it can be said that stateful protocol analysis creates a list in which it identifies any abnormal packets that violate pre-defined protocol state behaviors.

3. **Signature Based Detection:** signature-based intrusion detection systems monitor network packets and compare them against a database of known signatures or patterns of malicious threats, making signature-based detection the simplest method of detection. For example, if an employee receives an email with an attachment

known to contain a virus, it would be detected by the intrusion detection system and classified as inappropriate activity. The primary advantage of the signature-based approach is its highly effective detection of previously known threats, as long as they have been accurately entered into the system. However, configuring a signature-based intrusion detection system can be costly and time-consuming, especially for an organization with complex traffic [19].

Additionally, the signature-based approach is not very flexible; if a new threat is discovered in the network environment, it may not be detectable because the signature has not been entered into the intrusion detection system. To mitigate these risks, the database must be updated at specified intervals.

### 1.3.7.3 Response of intrusion detection

Intrusion detection system (IDS) response mechanisms can be categorized into two main types

- **Passive:** it is a system that is limited to monitoring and detecting harmful behavior and alerting the user only
- **Active:** it is a system usually called an intrusion prevention system (IPS) that monitors, detects, prevents and responds to harmful behaviors, but it may also have drawbacks.

### 1.3.8 Firewall versus intrusion detection system (IDS)

- **Intrusion Detection System (IDS):**
  - It monitors unusual or suspicious activities within the network.
  - Identifies and records potential threats and sends alerts to analysts to investigate and respond.
  - Helps detect hacks, intrusions, and unwanted behaviors.
- **Firewall:**
  - It acts as a barrier between the internal network and the external network (the Internet, for example).
  - It analyzes incoming and outgoing traffic and decides whether it should be allowed or blocked according to pre-defined rules.
  - Prevents unauthorized network access and protects internal systems from external attacks.

Basically, an intrusion detection system focuses on detecting and monitoring threats and informing administrators to take appropriate action, while a firewall regulates traffic and prevents unauthorized access to the network.

### 1.3.9 Advantages of using IDS

1. **Foiling expected network attacks:** IDS protect systems against network attacks such as backdoor detection, IP address spoofing, DoS, worms, Trojans, viruses, Botnets, rootkits, Spyware, and other threats that could harm the network. Active IDS take automatic measures against security threats and risks encountered.
2. **Alerting Network Administrators of Potential Security Events:** the fundamental function of intrusion detection systems is to generate alerts where external, internal threats, or violations of network security policy exist, and also to provide administrators with detailed information on data movement within the network.
3. **Time-saving:** using IDS provides significant time and effort savings in understanding what is happening in the network and can also run continuously without human supervision.
4. **Control of Programs Used by Employees to Monitor the Internet:** IDS can help discover programs dealing with the internet, enabling better control and protection of the network.
5. **Gaining Customer Trust:** IDS assist organizations in protecting their customers' data from theft and security breaches. This fosters trust among customers and partners and maintains a good reputation for the organization.
6. **Cost Savings:** through IDS, organizations can identify suspicious movements in the network and report the responsible parties to take proactive measures to protect the network and save money that would be spent if a security breach occurred in the network or if personal information theft took place.

### 1.3.10 Limits of intrusion detection system

The current limitations of the intrusion detection system include:

1. **False Positives:** IDS may generate alerts for events that are not actual threats, leading to wasted time and resources in investigating false alarms.
2. **Scalability:** IDS may struggle to handle large volumes of network traffic efficiently, resulting in performance degradation or missed detections.
3. **Signature-based Detection:** traditional IDS rely on known attack signatures, making them ineffective against new or unknown threats.
4. **Encrypted Traffic:** IDS may face challenges in inspecting encrypted network traffic, limiting their ability to detect malicious activities hidden within encrypted communications.
5. **False Sense of Security:** Over-reliance on IDS without considering other security measures can create a false sense of security, leaving the system vulnerable to attacks that bypass detection.

## 1.4 Conclusion

In this chapter, we discussed various concepts of information security, elaborated on attacks and their classifications, and elucidated the mechanisms used to prevent these attacks. Among these mechanisms, we detailed Intrusion Detection Systems (IDS) as our focus in this paper. Furthermore, we mentioned the main methods of using IDS and the possible types of alarms. IDS methods were presented and based on this it was decided to build our IDS system based on anomaly detection technology.

This is due to the fact that IDS anomaly detection technology is more automated and can detect an unknown attack. In next chapters, we will outline how we developed IDS using federated learning to enable it to recognize the largest possible number of unregistered attacks and intrusions while maintaining the security of the client dataset.



## Chapter 2

### State of the Art

## 2.1 Introduction

Artificial intelligence (AI) has experienced significant advancements in recent years, encompassing a range of computer science disciplines focused on developing systems capable of emulating human-like intelligence, including learning, reasoning, vision, and decision-making. Machine learning (ML) and deep learning (DL) are integral components of AI, involving the study of algorithms and statistical models that enable computer systems to execute tasks without explicit programming. ML and DL algorithms are utilized for diverse applications such as data extraction, image processing, predictive analytics, and decision-making, offering the advantage of automating tasks once the algorithm learns from the data.

Based on these technologies, as mentioned earlier in the previous chapter, Intrusion Detection System (IDS) is one of the tools that examines network traffic to ensure its confidentiality, integrity, and availability. However, it still faces challenges in improving detection accuracy while reducing false alarm rates and detecting new intrusions. The use of IDS based on both ML and DL has begun as potential solutions to increase the accuracy of detecting network intrusions effectively.

However, since we want to apply this system to the network, there is a higher likelihood of data sharing and thus the possibility of data exposure to theft and fraud. A new technology has emerged in recent years, known as federated learning, which is an approach to AI aimed at protecting data and models. It does not share data for learning purposes but shares device models to the server and then builds a final model for intrusion detection based on federated learning.

This chapter provides a comprehensive introduction to AI and its various branches. It briefly discusses the datasets used in IDS and also mentions the existing research on federated learning in IDS and other fields.

## 2.2 Overview artificial intelligence (AI)

Artificial Intelligence (AI) stands at the forefront of technological innovation, reshaping industries, revolutionizing processes, and redefining the boundaries of what machines can accomplish. According to certain researcher's definitions: "Artificial intelligence (AI) is intelligence exhibited by machines" [37]. At its core, AI aims to replicate human-like intelligence in machines, enabling them to perceive, reason, learn, and make decisions autonomously. With its profound impact on virtually every aspect of our lives, AI represents a paradigm shift in how we interact with technology and perceive the capabilities of intelligent systems. Since its inception in the mid-20th century, AI has experienced significant development due to advancements in processing power, algorithms, and data accessibility. Machine Learning (ML) is a very influential field of AI that enables machines to acquire knowledge from data, recognize patterns, and make predictions or judgements without the need for explicit programming. Deep Learning (DL) has revolutionized the field of Machine Learning by enabling advanced processing and comprehension of intricate data. Deep Learning algorithms, also called neural networks, have made significant advancements in computer vision, natural language processing, and speech recognition by

drawing inspiration from the structure and function of the human brain.

Figure 2.1 illustrates that artificial intelligence is a broad discipline encompassing both machine learning and deep learning.

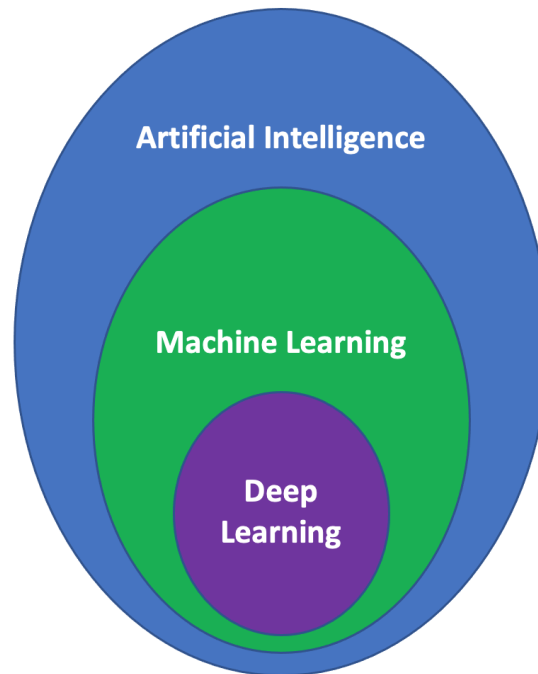


Figure 2.1: Artificial intelligence, machine learning and deep learning

## 2.2.1 Machine learning

Machine learning (ML) is a field within computer science and a subset of artificial intelligence that enables computers to think and learn without explicit programming. Machine learning is applied in diverse computational tasks, aiming to train the machine using provided data. the data can be labelled in case of supervised learning and unlabeled in case of unsupervised learning in order to produce better outcomes for the specified problem. The main focus is to make computers learn from past experience [38].

### 2.2.1.1 The Methods of Machine Learning

Two of the most widely adopted machine learning methods are supervised learning and unsupervised learning. Most machine learning about 70 percent is supervised learning. Unsupervised learning accounts for 10 to 20 percent. Semi supervised and reinforcement learning are two other technologies that are sometimes used.

1. **Supervised learning:** is algorithms need labelled data, and the data is split into two parts, one is testing data set, and the other is training data set [39].  
The trained data set has some output that needs to be predicted. The task is to make the machine learn from some similar kind of patterns obtained from the

training data set and apply the same on the data set to be tested to predict the real-valued output [40].

2. **Unsupervised learning:** is a form of machine learning that requires no labelled data to the machine. An algorithm is made based upon the input data, and then, the algorithm is analyzed on a set of data. The training data set is used in creating and training of the model, whereas the testing data set helps in predicting the correct values [41]. The machine predicts the outcome based on past experiences and learns from the previously introduced features to predict the real-valued outcome
3. **Semi-supervised learning:** is used for the same applications as supervised learning. These type of algorithms deal with the both labeled and unlabeled data sets [42].

It means using a small amount of labeled data with a large amount of unlabeled data (because unlabeled data is less expensive and takes less effort to obtain). This type of learning can be used with methods such as classification, regression, and prediction.

Semi-supervised learning is useful when the cost associated with labeling is too high to allow for a fully labeled training process.

4. **Reinforcement learning:** is the training of machine learning models so that they can make a sequence of decisions. The agent figures out how to accomplish a goal in an unsure, possibly complex environment. In reinforcement learning, the agent faces a game-like circumstance. The agent utilizes experimentation to return up with a response to the issue. To make the machine attempt to do what the programmer needs, the agent gets either rewards or penalties for the activities it performs. Its goal is likely to expand the full reward.

## 2.2.2 Deep learning

Deep learning, also known as hierarchical learning, is a part of ML algorithms and architectures. It includes all machine learning methods which are based on learning data representations [43]. The learning can be categorized as supervised, semi-supervised or unsupervised. In supervised learning, classification is done, and in unsupervised learning, similar features or characteristics are grouped [44]. Deep learning algorithms extract features implicitly, and the significance of the word deep means the number of layers throughout from which the data is to be transformed. Algorithms of deep learning are applied to both supervised and unsupervised learning. In unsupervised learning, there is more amount of unlabeled data as compared to supervised learning; hence, this is more beneficial. Deep learning extracts the best features, and the solution is an end-to-end method [45][46].

The deep learning paradigm is currently implemented with Deep Neural Networks (DNNs), that are Artificial Neural Networks (ANNs) based on several hidden layers between input and output (Figure 2.2).

Each layer learns higher-level features that are later processed by the following layer [47]. When a suitable high-level representation is reached, a classifier can perform the final decision. Modern DNNs provide a very powerful framework for supervised learning: when adding more layers, in fact, a deep network can represent functions of increasing complexity and can potentially reach higher levels of semantic representations.

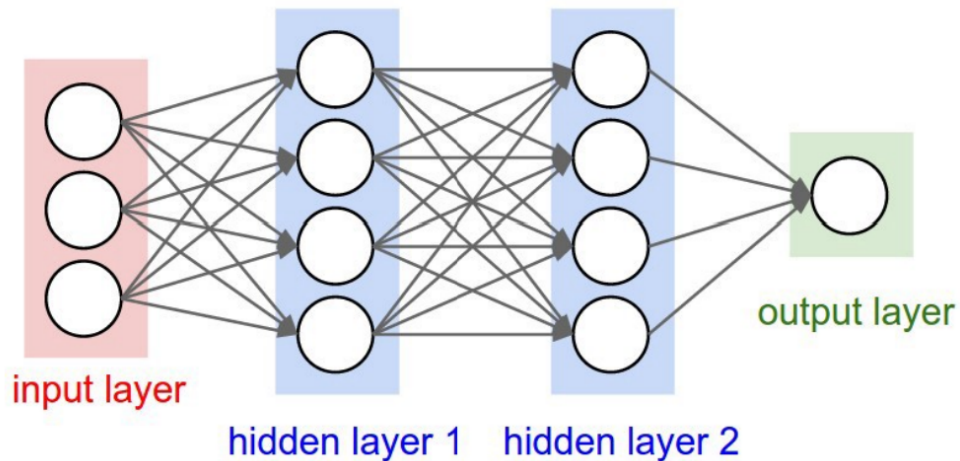


Figure 2.2: Neural network architecture

## 2.2.3 Deep learning approaches

### 2.2.3.1 Deep neural network(DNN)

A deep neural network(DNN) consists of artificial neural network (ANN) nodes in different hierarchical layers (Figure 2.3). These layers consist of input layer, hidden layers, and the output layer. The number of input and output layer is fixed similar to the sequential models, and the number of nodes in these 2-layers are also fixed. The input layers contain as many nodes as the number of features in the input, and when used as a classifier, the output layer contains as many nodes as the number of classes. There could be one or more hidden layers in the network, and the number of nodes in each hidden layer may differ [48].

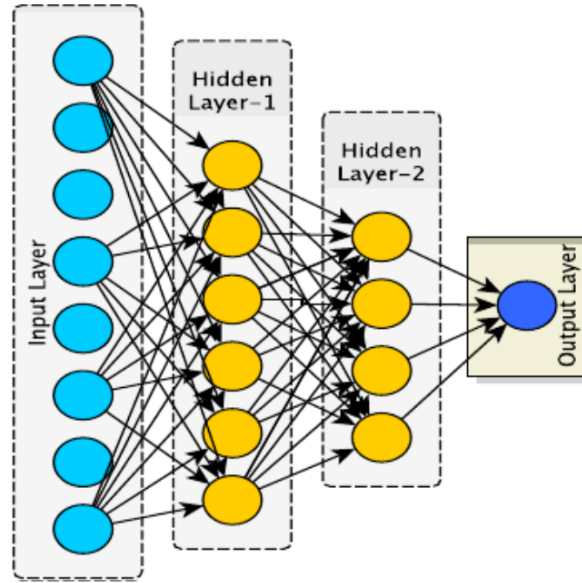


Figure 2.3: A schematic of Deep neural network(DNN)

### 2.2.3.2 Recurrent neural network(RNN)

A recurrent neural network (RNN) is a type of deep learning model designed to receive sequential data input and generate a corresponding sequential data output. For example, in situations where the occurrence of two inputs is interdependent and these inputs influence subsequent inputs, the appropriate architecture to use is the recurrent neural network (RNN). A recurrent neural network may be conceptualised as a set of identical network instances, each transmitting information to the subsequent node. The primary concept underlying the use of recurrent neural networks (RNNs) is to leverage sequential data for the purpose of deep learning[49].

The architecture of an Recurrent neural network(RNN) differs from that of a canonical neural network, as shown in Figure 2.4, in that it includes recurrent connections that allow information to persist over time. This repeated connection forms a loop, allowing the node to retain the memory of previous inputs while processing the current input.

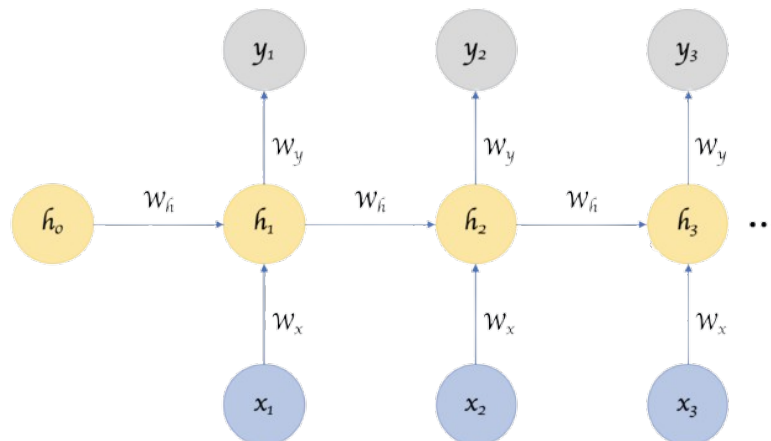


Figure 2.4: A schematic of Recurrent neural network(RNN)

It has significant use in forecasting the next word in a phrase, discerning the meaningful sequence of words, recognizing handwriting, detecting voice, translating text using machines, and other applications that involve sequential data[49].

### 2.2.3.3 Long Short-Term Memory(LSTM)

LSTMs were introduced to handle the vanishing gradient problem of the traditional RNNs. An LSTMs architecture is similar to an RNN, but the recurrent cells are replaced with LSTM cells [1] which are constructed to retain information for extended periods. Similar to the cells in standard RNNs, the LSTM cell recurs the previous output, but also, it keeps track of an internal cell state, which is a vector serving as long-term memory. Hence, the LSTMs have access to both short-term memory, i.e. the hidden state, and long-term memory, i.e. the cell state  $c_t$ , resulting in the name: long short-term memory cells [57].

The cell state can metaphorically be viewed as a conveyor belt that passes information down the line, to be used in later calculations. The content of this cell state vector is carefully altered through operations by gates inside the LSTM cell, which lets essential information and gradients to flow unchanged.

The main idea of the LSTM cell is to regulate the updates of the long-term memory (cell state), such that information and gradients (for training) can flow unchanged between iterations. To carefully regulate the cell-state, the LSTM uses the three yellow gates in Figure 2.5.

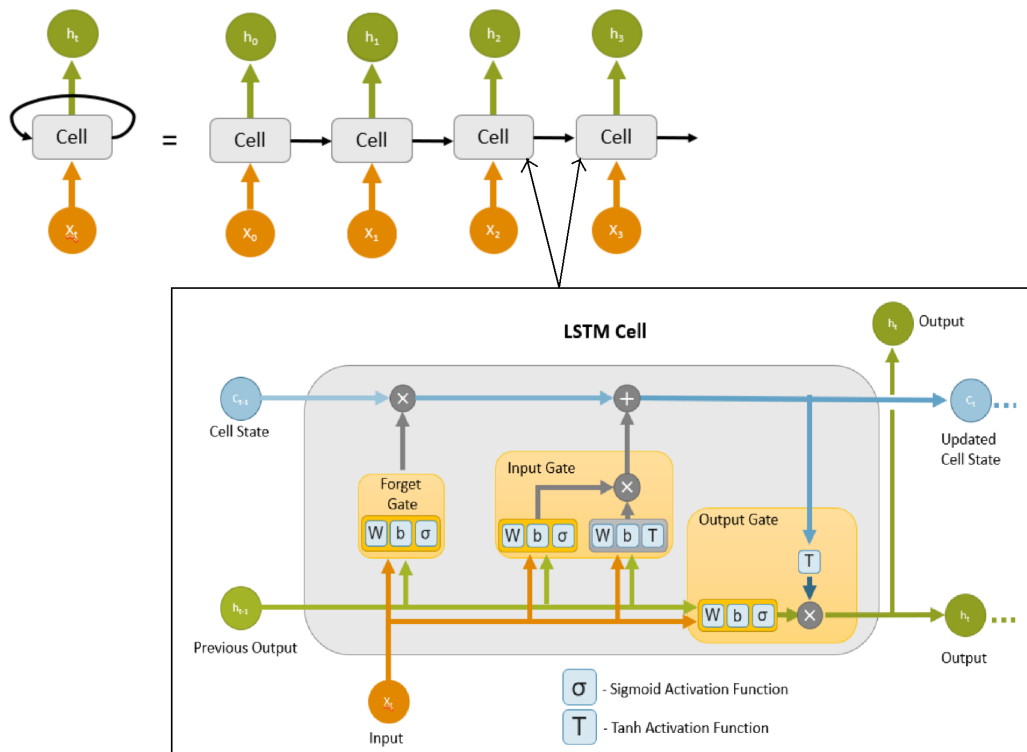


Figure 2.5: A schematic of Long Short-Term Memory(LSTM)

## 2.3 Intrusion detection system datasets

Datasets for intrusion detection and classification are developed by gathering data from different devices and applications being used in the underlying network environment. Intrusion detection datasets consist of data samples that represent traffic flowing through web applications, system configuration, execution system processes, and audit log files[32].

One of the earliest datasets used for the performance evaluation of IDS is DARPA dataset.

- **DARPA Dataset:**

The basic dataset to assess the performance of IDS was made in 1998 by Defence Advanced Research Project Agency(DARPA) [33].

DARPA 1998 was created at the Massachusetts Institute of Technology Lincoln Laboratory by collecting host log files and network packets from the emulated network environment for 9 weeks. Then, the range of accessible attack scenarios was extended to construct DARPA 1999 for the purpose of providing a realistic evaluation environment.

The dataset contains both raw packets and tags. There are five types of labels: normal, denial of service (DOS), probe, user-to-root (U2R), and remote-to-local (R2L).

DARPA 1998/99 datasets have contributed immensely to the improvement research on IDS. However, they are criticised for the huge amount of record redundancy and considered an outdated.

- **KDD 99 Dataset:**

Knowledge Discovery in Databases 99 (KDD)[33] is the improved version of DARPA 1998 and one of the most widespread datasets to evaluate IDSs.

It contains around 4 million records classified as normal connection and attack traffic. Likewise, this dataset incorporates in excess of 20 distinct kinds of attacks and 41 features. Nevertheless, KDD 99 has serious limitations like the missing statistics about the number of dropped packets while creating the dataset and the unbalanced distribution of attacks resulting in biased majority-class classification results. Also, around 78% of training records and 75% of the testing dataset are duplicated[34].

However, besides the fact that this dataset does not present examples of new attacks, it remains a useful benchmark dataset to evaluate and compare diverse intrusion detection techniques.

- **NSL-KDD Dataset:**

NSL-KDD was developed in 2009 to overcome the problems of the KDD 99 dataset and eliminate duplicate records[34].

The records in the NSL-KDD were carefully selected on the basis of KDD99. Records from different classes are balanced in the NSL-KDD, which avoids the problem of classification bias. The NSL-KDD also removes duplicate and redundant records, so it contains only a moderate number of records. As a



result, experiments can be implemented on the dataset, and results from different papers are consistent and comparable. NSL-KDD mitigates the problems of bias and data redundancy to some extent. However, NSL-KDD does not include new data, so minority class samples are always lacking, and its samples are always obsolete.

- **UNSW-NB15 Dataset:**

UNSW-NB15 is a network intrusion detection dataset collected by a team of researchers at the University of New South Wales in 2015. The dataset comprises 2,540,044 network traffic records. Furthermore, the UNSW-NB15 dataset encompasses a broader range of attack types compared to the KDD99 dataset, and its features are more extensive[35]. The UNSW-NB15 dataset includes 49 features and categorizes attacks into 9 types: Normal, Analysis, DoS, Exploits,, Fuzzers, Reconnaissance, Backdoors, Shellcode, Worms, and Generic.

- **CICIDS 2017 Dataset:**

CICIDS2017 dataset contains the most up-to-date common attacks along with 80 network flow features from the captured network traffic. Records were collected over five days from an emulated environment by using CIC FlowMeter to analyze the network traffic[36].

Additionally, it includes the normal behavior of users depending on the HTTPS, HTTP, SSH, FTP, and email protocols. Furthermore, it presents naturalistic, benign traffic and contains a variety of attack scenarios. Typically, this dataset can be suitable for general evaluation settings.

Dataset Name	Developed By	Attack types	Number of features
DARPA	MIT Lincoln Laboratory	Dos, R2L, U2R, Probe	41
KDD	University of California	Dos, R2L, U2R, Probe	41
NSL-KDD	University of Californiain in 2009	Dos, R2L, U2R, Probe	41
UNSW-NB15	team of researchers at the University of New South Wales in 2015	Analysis, DoS, Exploits, Fuzzers, Reconnaissance, Backdoors, Shellcode, Worms, and Generic	49
CI-CIDS2017	Canadian Institue of Cyber Security	Brute force, Portscan, Botnet,Dos, DDoS, Web,Infiltration	80

Table 2.1: Summary of dataset

## 2.4 Evaluation metrics

Performance metrics are essential in validating the performance of any classification technique for a given application problem. Performance metrics illustrate how a classification technique has captured the problem and how it is interpreting the data. For instance, for intrusion detection and classification, the accuracy performance measure signifies that out of all the data samples, how many data samples were correctly classified. Whereas precision shows that out of all the data samples labelled as attacks, how many are actually attack. Hence, every performance metric has its own significance in representing the effectiveness and efficiency of a classifier [36].

The results of the experiments performed for the proposed approaches are represented using accuracy, precision, recall and f-score. These performance measures are derived using True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These terms are explained as follows.

- **True Positive (TP):** intrusion traffic is detected as intrusive (Successful identification of attack).
- **False Positive (FP):** normal traffic is detected as intrusive.
- **True Negative (TN):** normal traffic is detected as normal (Successful identification of normal traffic).
- **False Negative (FN):** malicious traffic is detected as normal.

**Accuracy** :it is a performance metric that provides an indication of the accuracy of the planned model. Accuracy is the quotient of the number of accurately predicted samples divided by the total number of samples.

The formula for calculating accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (2.1)$$

**Precision:** is a statistical measure that quantifies the accuracy of positive predictions by calculating the ratio of properly anticipated positive samples to the total number of positive samples.

The formula for calculating accuracy is as follows.

$$Precision = \frac{TP}{TP + FP} \quad (2.2)$$

**Recall:** is a statistical measure that quantifies the proportion of accurately predicted positive samples out of the total number of samples in the actual class.

The equation for calculating recall is as follows:

$$Recall = \frac{TP}{TP + FN} \quad (2.3)$$

**F1-score:** is a calculated measure that represents the weighted average of accuracy and recall.

The formula for calculating the f-score is as follows.

$$F1 - score = 2 \frac{Recall Precision}{Recall + Precision} \quad (2.4)$$

## 2.5 Related work

Currently, there are many studies and research in federated learning that have developed significantly in recent years, in order to further preserve data protection and confidentiality, in this section we will present some studies related to FL and IDS technology in data protection .

Suwannalai and Polprasert [20]. They propose the use of deep reinforcement learning as a new approach to solve the problem of network intrusion detection for network monitoring and analysis. They propose the AE-DQN (deep Q-Learning) algorithm as an alternative to improve the attack detection performance of anomaly-based NIDS, the performance of their proposal was examined on the NSL-KDD dataset. They focused on the 5-label classification problem.

Chen et al [21]. developed a network intrusion detection system based on a CNN-based data collection and monitoring control system to safeguard the Industrial Internet of things from DDoS attacks and other cyberattacks, cyber assaults against SCADA systems in general and cyberattacks on SCADA systems in particular.

Zhao et al[22]. To protect the training data, they used a federated learning architecture to train the detection model so that:

- They proposed a multi-task deep neural network in federated learning (MT-DNN-FL) for network anomaly detection and network traffic analysis. In MT-DNN-FL, participants do not share their training data with a third party, which may prevent attackers from exploiting the training data.
- MT-DNN-FL can perform multiple tasks at the same time, network anomaly detection task, VPN (Tor) traffic recognition task, and traffic classification task. Compared with using multiple single-task learning methods, MT-DNN-FL can save training time.
- Experiments conducted on three representative datasets, CICIDS2017, ISCXVPN2016, and ISCXTor2016, showed that the detection and classification performance achieved by the proposed method is better than the basic methods (deep neural network, logistic regression, nearest neighbors, and random forest) in central regions.

In their proposed multi-task learning framework in federated learning, a network anomaly detection task, a VPN or Tor traffic recognition task, and a traffic classification task are collaboratively trained by  $n$  participants. So each user uses their own data to train the local model and then upload parameter updates to the server. The server averages these parameter updates to replenish the global model and then sends the new global model to each participant.

Man et al [23]. Proposed an intelligent intrusion detection mechanism FedACNN, which assists the deep learning model CNN to complete intrusion detection through the FL mechanism, they used the NSL-KDD dataset to evaluate the model, the server used in they experiment is the Windows10 operating system and they used Python's deep learning library Pytorch to program FedACNN. they performed a preprocessing operation on the original dataset. The preprocessing procedure includes numerical, normalization, and visualization, accuracy was first used to evaluate the performance of the centralized learning (CL) model and the federated learning model on NSL-KDD. For the centralized model, they used a CNN model as well as a centralized FedAVG algorithm that uploads data to the server for centralized training, the results in the case of 40 rounds of iteration gave 99.65% for CL-CNN, because the CL-CNN model contains more complete data sets. Compared to FedAVG, FedACNN gave an accuracy of 99.12%. Through their experiments, they showed that the Federal Education Cooperative Training could achieve optimal accuracy while protecting data privacy. In other words, FedACNN sacrifices some accuracy to protect data privacy.

Liu et al [24]. they proposed a new communication-efficient on-device federated learning (FL)-based deep anomaly detection framework for sensing time-series data in IIoT. Specifically, they introduced an FL framework to enable decentralized edge devices to collaboratively train an anomaly detection model, which can improve its generalization ability, they proposed an attention mechanism-based convolutional neural network long short-term memory (AMCNN-LSTM) model to accurately detect anomalies. The AMCNN-LSTM model uses attention mechanism-based convolutional neural network units to capture important fine-grained features, thereby preventing memory loss and gradient dispersion problems, they proposed framework is applied to four real-world data sets, i.e., power demand, 1 space shuttle, 2 ECG, 3 and engine 4 for performance demonstration. These data sets are time-series data sets collected by different types of sensors from different fields.

Bo Cao et al [25], They proposed a network intrusion detection model that integrates a convolutional neural network and a gated recurrent unit to address the problems associated with the low accuracy of existing intrusion detection models for multi-classification of intrusions, the proposed network intrusion detection model that combines convolutional neural network and GRU, referred to as CNN-GRU model, consists of three main stages: first, the preprocessing stage, in which the original data is transformed into digital features and normalized, and then the dataset is normalized by ADRDB algorithm, and then the features are extracted by RFP algorithm and finally converted into gray scale map; Second, the training phase, where the pre-processed data are assigned different weights to the features by the Convolutional Mass Attention Module (CBAM) based on the residuals first, then the spatial features are extracted by the CNN module, and the spatial information is extracted and further aggregated by combining Averagepooling and Maxpooling. Then, the temporal features are extracted by multiple GRUs. Finally, clas-

sification is performed by the Softmax function; Third: The testing phase, in which the test set is passed to the training model for classification, the proposed model is evaluated on UNSW\_NB15, NSL-KDD and CIC-IDS2017 datasets.

Sultana et al [27]. Software Defined Networking (SDN) technology provides an opportunity to effectively detect and monitor network security issues that are due to the advent of programmable features. Recently, machine learning (ML) methods have been applied in SDN-based network intrusion detection systems (NIDS) to protect computer networks and overcome network security issues. They evaluated deep learning techniques in developing SDN-based NIDS. In this survey, they cover tools that can be used to develop NIDS models in an SDN environment. and discuss ongoing challenges in implementing NIDS using ML/DL and future work.

Nguyen et al [28]. As many IoT devices become widespread, they are vulnerable due to insecure design, implementation, and configuration. However, current intrusion detection techniques are not effective in detecting compromised IoT devices due to the sheer scale of the problem in terms of the number of different types of devices and manufacturers involved. As a result, in their paper they present D<sup>2</sup>IoT, an autonomous self-learning distributed system for detecting IoT devices. vulnerable. D<sup>2</sup>IoT effectively relies on device type-specific communication profiles without human intervention or labeled data that is subsequently used to detect anomalous anomalies in devices' communication behavior, potentially caused by malicious adversaries, D<sup>2</sup>IoT uses a federated learning approach to efficiently collect behavior profiles. They reported that it is the first system to use a federated learning approach to intrusion detection based on anomaly detection. Therefore, D<sup>2</sup>IoT can deal with new and unknown attacks, they conducted a systematic and extensive evaluation of more than 30 long-term IoT-ready devices. To evaluate D<sup>2</sup>IoT, they applied it to a real-life IoT malware detection use case. They chose Mirai for this purpose, since its source code is publicly available, they collected extensive datasets about the communication behavior of IoT devices in laboratory and real-world deployment settings, and For evaluating the effectiveness of D<sup>2</sup>IoT at detecting attacks, they collected a dataset comprising malicious traffic of IoT devices infected with Mirai malware .

Fan et al [29]. proposed an FL-based intrusion detection framework for 5G IoT. This framework built a network intrusion detection model based on convolutional neural network (CNN) and aggregated different local intrusion detection models through federated learning to train a powerful intrusion detection model.

Paper Name	Dataset(s)	Approach
Network intrusion detection systems using adversarial reinforcement learning with deep Q-network[20].	NSL-KDD.	DQL(deep q-learning)
A novel network intrusion detection system based on CNN[21].	Industrial IoT, SCADA systems.	CNN-based data collection, monitoring control system
Multi-task network anomaly detection using federated learning[22].	CICIDS2017, IS-CXVPN2016, ISCX-Tor2016.	Federated learning, multi-task deep neural network (MT-DNN-FL),
Intelligent intrusion detection based on federated learning for edge-assisted internet of things[23].	NSL-KDD.	FedAVG, FedACNN
Liu et al Communication efficient federated learning for anomaly detection in industrial internet of things[24].	Industrial IoT (time series).	CNN-LSTM
Network Intrusion Detection Model Based on CNN and GRU[25].	UNSW_NB15, NSL-KDD, CIC-IDS2017.	CNN-GRU
DIoT: A federated self-learning anomaly detection system for IoT[28].	IoT devices data.	GRU(Gated Recurrent Units)
A federated transfer learning intrusion detection framework for 5 g iot[29].	5G IoT.	CNN

Table 2.2: Summary of dataset

## 2.6 Conclusion

This comprehensive overview highlights the pivotal role of artificial intelligence (AI), machine learning (ML), and deep learning (DL) in intrusion detection systems (IDS). Transitioning to the context of intrusion detection system datasets, this study underscores the importance of robust datasets for training and evaluating such systems. We also highlight the evaluation metrics that accurately measure the performance of intrusion detection models, thereby facilitating comparisons and improvements. We also reviewed related work in this field to contextualize advancements and ongoing research efforts.

## Chapter 3

# Federated Deep Learning for intrusion detection system

### 3.1 Introduction

In the world of AI and machine learning, data availability is very important for generating high-performance and accurate models. However, the manner in which such data are stored, processed and used varies considerably based on the approved infrastructure. So there are two basic models in this case: centralized and decentralized learning.

### 3.2 Centralized learning

The centralized learning or what is known as the centralized architecture is where local data samples are collected from various sources and transmitted to a central server [58]. The central entity holds all data samples, ideally reflecting an overall statistical representation of the organizational network structure.

The learning and testing stages are carried out on the central server, where the learning models experience and extract useful patterns from heterogeneous network traffic. Therefore, IDS can effectively detect network intrusions in non-independently and identically distributed (non-IID) data samples [59]. However, centralized learning requires direct sharing of data samples between participants and a central entity [60]. This architecture presents deficiency in privacy and security due to the nature of the transmitted data as shown in (Figure 3.1). Network data often contain sensitive information related to users' browsing sessions, applications, and services utilized.

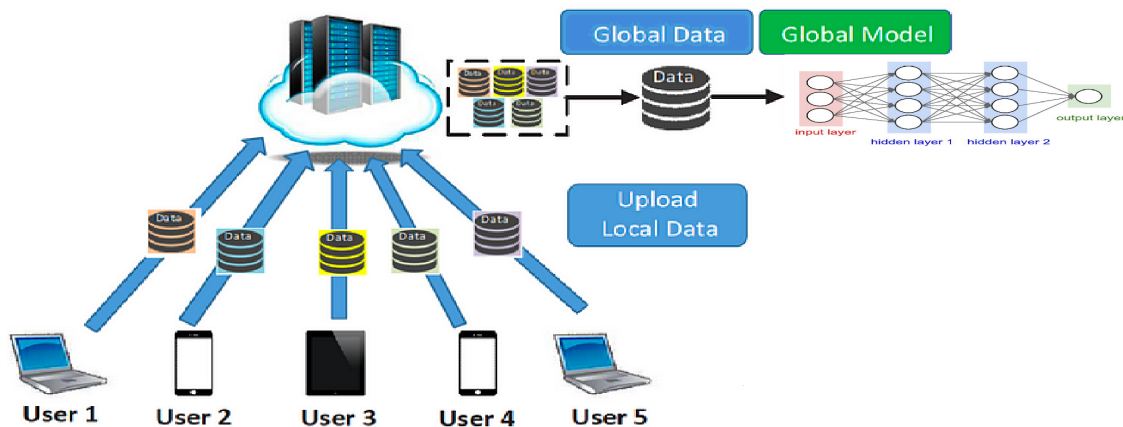


Figure 3.1: Centralized learning architecture



### 3.3 Decentralized learning

Decentralized learning is a architecture of machine or deep learning in which the training data is stored on several devices or nodes, such as cellphones, sensors, and computers, that are spread out over a network. Decentralised learning algorithms enable collaborative training of models on individual devices, eliminating the need to submit all data to a central server for processing. The models or their updates are then swapped for aggregation, enhancing the global model without necessitating the consolidation of all the data in a single location[61].

This architecture addresses privacy and security concerns regarding the client's data by allowing the client to send only the model to the server, without sharing the dataset, as illustrated in Figure 3.2.

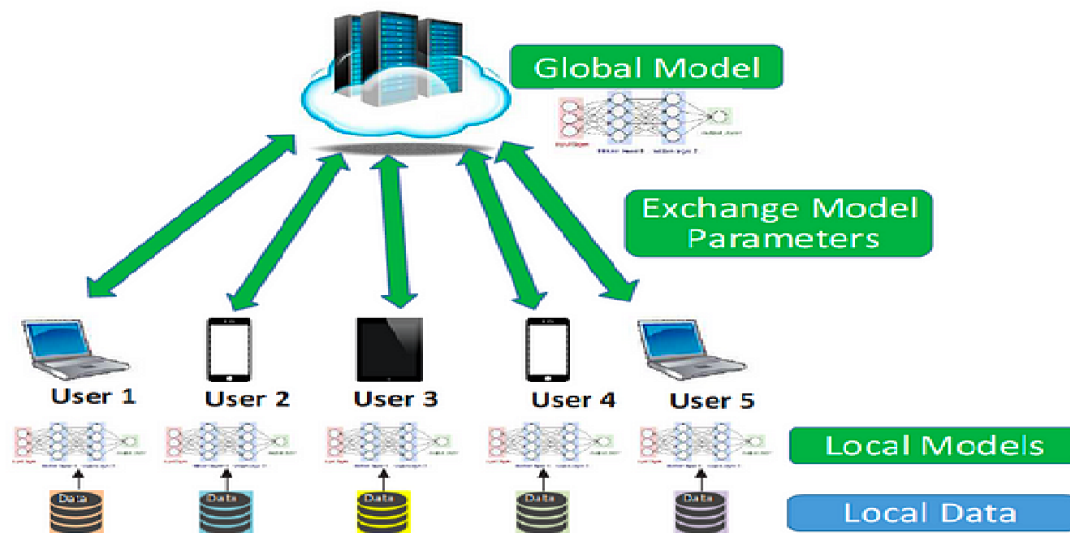


Figure 3.2: Decentralized learning architecture

## 3.4 The approach used in this work

### 3.4.1 Convolutional neural network(CNN)

Convolutional Neural Network(CNN) are one of the best machine learning algorithms .and is one of the most popular deep neural networks. It take this name from mathematical linear operation between matrixes called convolution. CNN have multiple layers; including convolutional layer, non-linearity layer, pooling layer and fully-connected layer. The convolutional and fully-connected layers have parameters but pooling and non-linearity layers don't have parameters. The CNN has an excellent performance in machine learning problems. Specially the applications that deal with image data, such as largest image classification data set (Image Net), computer vision, and in natural language processing (NLP) and the results achieved were very amazing[50].

### 3.4.1.1 Layers of convolution neural networks

In Neural Networks There is an input layer, one or many hidden layers and an output layer. All the layers have nodes and each node has a weight which is considered while processing information from one layer to the next layer[51].

1. **Input layer:** in this layer, we supply our model with inputs. It is training information.
2. **Hidden layer:** the hidden layer receives the input from the input layer after that. Several hidden layers may exist, contingent on the amount of the data and our model. . Each layer's output is determined by matrix multiplying the output of the layer before it with its learnable weights, adding its learnable biases, and then applying an activation function to make the network non-linear.
3. **Output layer:** a logistic function, such as sigmoid or softmax, receives the output from the hidden layer and uses it to translate the output of each class into a probability score for each class.
4. **Convolution layer:** convolution is a mathematical operation. It can be used for edge detection, and extract features. We can achieve this by performing a convolution operation between the kernel and the image or matrix (a kernel is a constant grid constance matrix)(Figure 3.3). We achieve this by adding each multiplication of the kernel index and the image grid index and placing the result in the middle of the grid

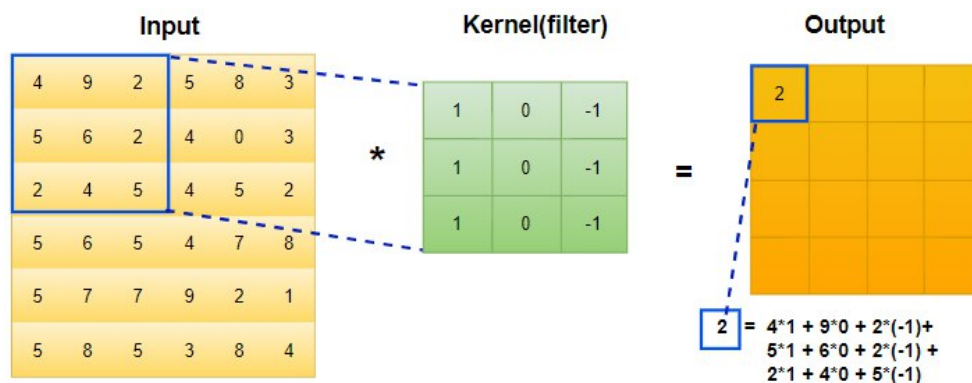


Figure 3.3: Process of convolution layer

5. **Pooling layer:** feature motifs, which result as an output of convolution operation, can occur at different locations in the image or matrix. Once features are extracted, its exact location becomes less important as long as its approximate position relative to others is preserved. Pooling or down-sampling is an interesting local operation. It sums up similar information in the neighborhood of the receptive field and outputs the dominant response within this local region[52].

Figure 3.4 illustrates the process of max pooling.

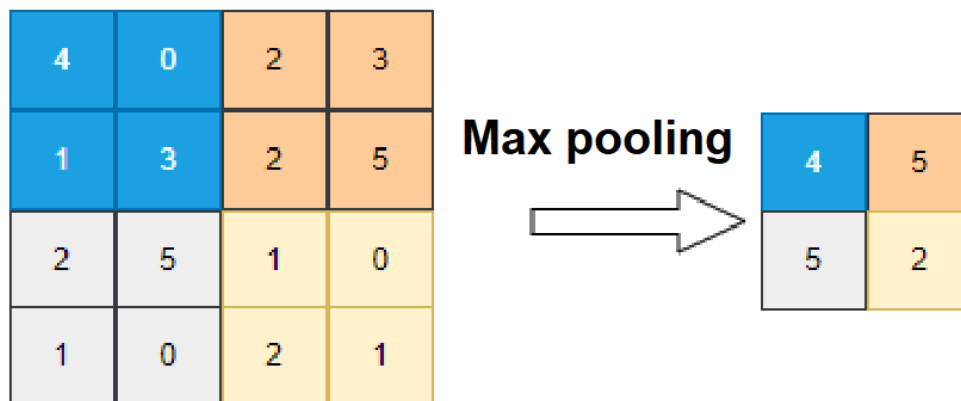


Figure 3.4: Process of max pooling layer

6. **Dropout Layer:** dropout regularization technique address the issue of over-fitting in feed-forward neural networks. The basic idea of applying standard dropout during training phase of neural network is to randomly drop/deactivate neurons [53]
7. **Flattening layer:** flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector.

Figure 3.5 illustrates the process of flattening.

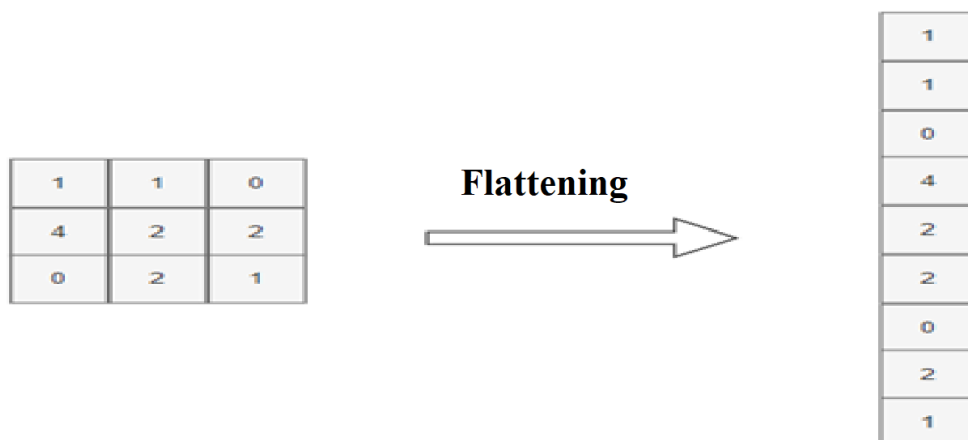


Figure 3.5: Process of Flattening layer

8. **Fully connected Layer:** fully connected layer is mostly used at the end of the network for classification. Unlike pooling and convolution, it is a global operation. It takes input from feature extraction stages and globally analyses the output of all the preceding layers [54].

Consequently, it makes a non-linear combination of selected features, which are used for the classification of data [55].

### 3.4.1.2 Activation functions

Activation function serves as a decision function and helps in learning of intricate patterns. The selection of an appropriate activation function can accelerate the learning process[56].

Among the activation functions we mention the following:

**Function Rectified Linear Unit (ReLU) :** ReLU stands for rectified liner unit and is a non-linear activation function which is widely used in neural network. The upper hand of using ReLU function is that all the neurons are not activated at the same time. This implies that a neuron will be deactivated only when the output of linear transformation is zero[51]. It can be defuned mathematically as :

$$G(E) = \max(0, E) = \begin{cases} E & \text{if } E \geq 0 \\ 0 & \text{else} \end{cases} \quad (3.1)$$

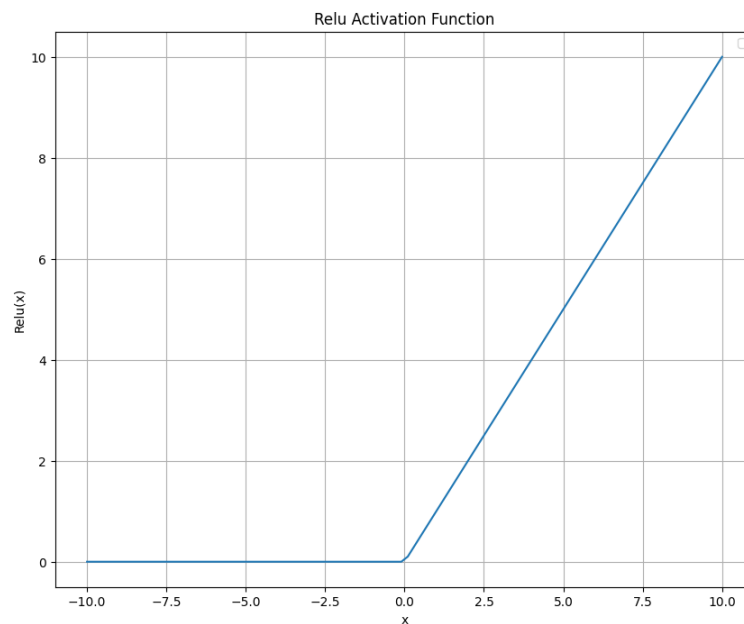


Figure 3.6: Activation Function ReLU

**Function sigmoid :** it is the most widely used activation function as it is a non-linear function. Sigmoid function transforms the values in the range 0 to 1[51].

$$\alpha(x) = \frac{1}{1 + e^{-x}} \quad \forall x \in \mathbb{R} \quad (3.2)$$

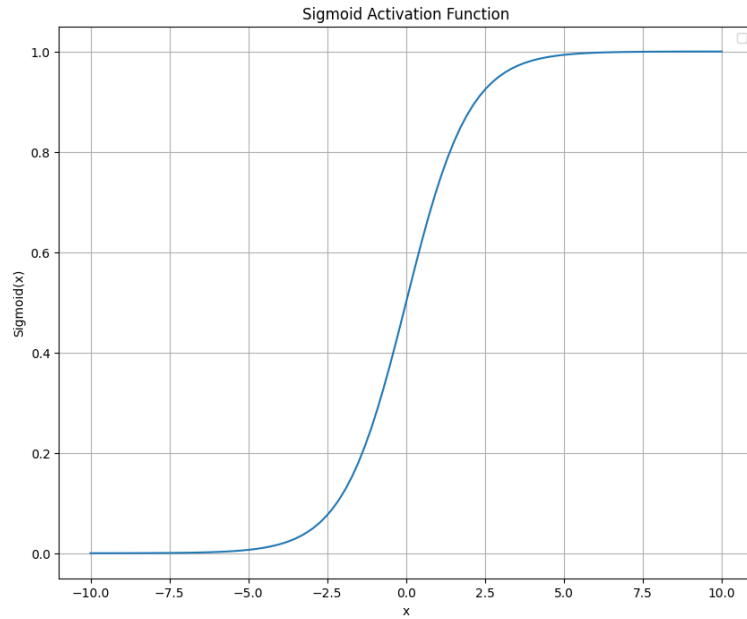


Figure 3.7: Activation Function sigmoid

**Function softmax :** the Softmax function is a combination of multiple sigmoid functions. As we know that the sigmoid function returns values in the range 0 to 1, they can be treated as probabilities of data points of a given class.

The Softmax function can be used in multiclass classification problems, unlike sigmoid functions which are used for binary classification. It can be expressed as follows:

$$G(e_j) = \frac{e^{e_j}}{\sum_i e^{e_i}} \quad (3.3)$$

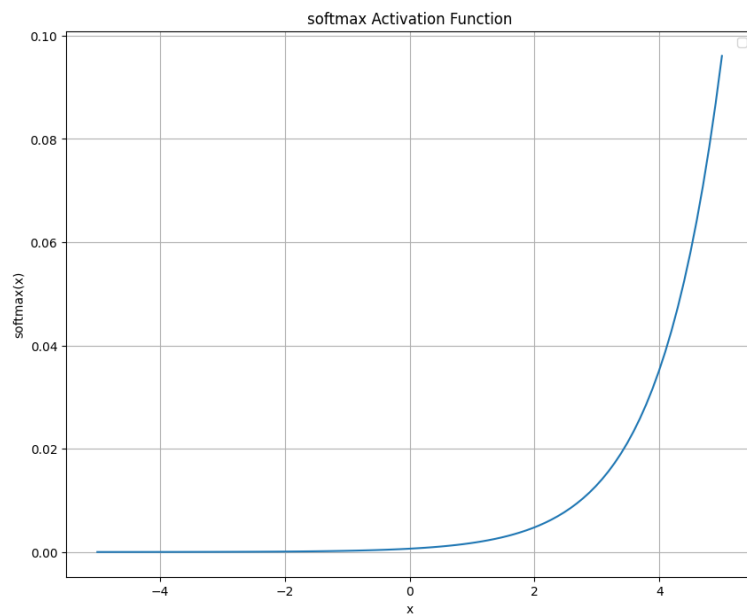


Figure 3.8: Activation Function softmax

### 3.5 Decentralized Federated Learning (FL)

The availability of top-notch training data is essential for ML applications. However, there are times when privacy concerns make it impossible to transfer training data to a central data repository where it can be curated and managed for the ML process. The method known as federated learning (FL) was first put forth in [62] to train ML models using training data from various sources without the need for centralized data collection.

The lack of adoption of a central data repository has been largely attributed to various jurisdictions' differing laws governing consumer privacy. Examples of legal frameworks for the gathering and use of consumer data include the General Data Protection Regulation (GDPR), the Health Insurance Portability and Accountability Act (HIPAA), and the California Consumer Privacy Act (CCPA) [63].

Moreover, the proliferation of news articles on data breaches has heightened individuals' consciousness of the potential hazards associated with retaining confidential customer data. Federated Learning enables the utilization of data without the need to store it in a central repository, thereby mitigating this potential hazard. Data movement across jurisdictions, such as different nations, is also subject to regulatory restrictions.

This decision was made due to the potential inadequacy of data protection in other countries or its connection to national security, which requires the storage of important data within the country. Global enterprises with operations in several markets have the challenge of complying with national and regional rules when attempting to train a model utilizing their whole dataset. In addition to meeting legal obligations, using data from several sources might also be advantageous. Obtaining centralized data collection may prove unattainable due to the unreliability of communication links, the immense amount of data gathered by sensors or telecommunication devices, or a combination of both factors.

Additionally, FL enables various businesses to collaborate and develop models for their mutual benefit without disclosing their trade secrets. In the FL approach, a number of different parties who each have control over their own training set work together to develop a machine learning model. They carry out this action without disclosing their training information to any other parties or outside organizations. In the related work, parties to the collaboration are also referred to as clients or devices. Parties include consumer electronics like smartphones or automobiles, but they can also be cloud services from various providers, data centers processing enterprise data in various nations, application silos within a business, or embedded systems like manufacturing robots in an automotive plant.

Although the FL collaboration can be carried out in various ways, its most typical form is shown in Figure 3.9 In this method, the collaboration is facilitated by an aggregator, also known as a server or coordinator. On the basis of their personal training data, parties conduct local training processes. When local training is complete, they update the aggregator with their model parameters. Depending on the type of machine learning model being trained, the model updates may take the form of network weights, for example, in the case of a neural network.

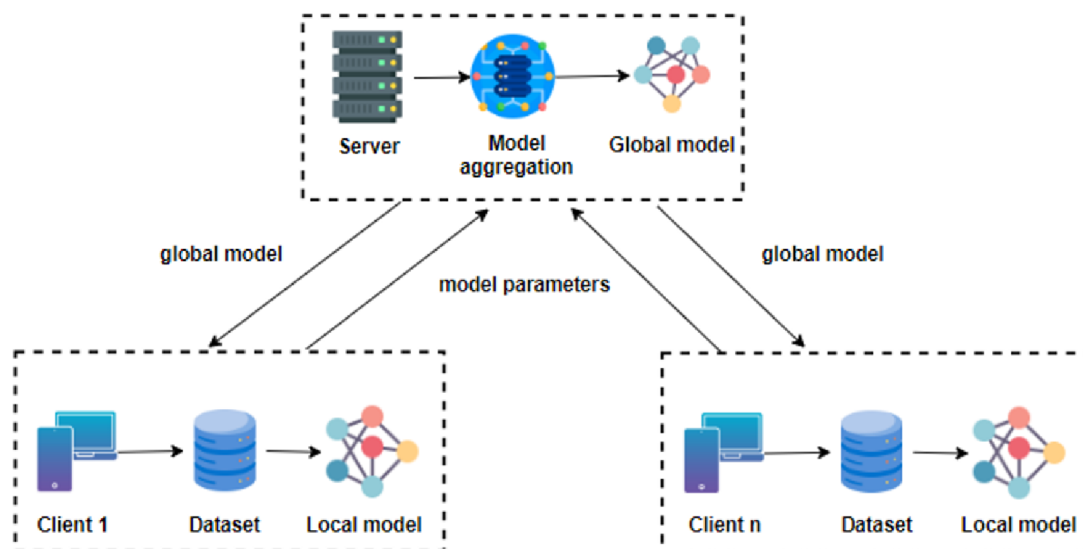


Figure 3.9: Federated Learning architecture

The model updates from the parties can then be combined into a single model through a process we call model fusion after the aggregator has received them. This can be accomplished in the neural network example by simply averaging the weights, as suggested by the FedAvg algorithm [64].

### 3.5.1 Federated learning process

The federated learning Process will be thoroughly explained, providing a comprehensive breakdown of its six steps.

#### Step 1: Initialize the global model

The process of FL begins with the initialization of a global model, which serves as the starting point for the training process and is usually pre-trained on a large, representative dataset, the global model is initialized with a set of weights and biases that define its initial state, the goal of FL is to refine this global model using the data from the different nodes in the network. The training process happens in a sequence of rounds, and in each round, the global model is sent to the nodes, at each node, the global model is used to make predictions on the local data, and the local model's weights and biases are updated using a local optimization algorithm, such as stochastic gradient descent. The updated weights and biases are then sent back to the server, where they are aggregated to update the global model. The initialization of the global model is an important step in federated learning because it provides a starting point for the training process, the quality of the global model can have a significant impact on the speed and accuracy of the Federated Learning process. A good initialization can lead to faster convergence and better results [65].

**Step 2: Send the global model to a number of connected organizations/devices (client nodes)**

In this step, the central server sends the global model to multiple participating organizations or devices, referred to as client nodes, these latter can be various healthcare institutions, IoT devices, or even personal mobile devices [65].

As illustrated in ( Algorithm 1) it operates at the client level, where the model is trained locally.

---

**Algorithm 1** Federated Learning: Client-Side Training at Federated Round  $T$

---

**Require:** Local learning rate  $\eta_t$  and loss function  $\ell$

**Require:** Number of local epochs  $E$  and local training data  $D$

```

1: procedure CLIENT_UPDATE( $w^t$ )
2:    $w \leftarrow w_{t-1}$  ▷ Initialize local model
3:    $B \leftarrow$  Split  $p_k$  into batches of size B
4:   for each local epoch  $i$  from 1 to  $E$  do ▷ With SGD optimizer
5:     for each batch  $b$  in  $B$  do
6:       Compute gradient  $g_i^b \leftarrow \nabla \ell(w; b)$ 
7:       Update local model  $w \leftarrow w - \eta g_i^b$ 
8:     end for
9:   end for
10:  Return:  $w$  ▷ Upload to server
11: end procedure

```

---

**Step 3: involves sending form updates to the server** After the local training concludes, every client node transmits its model updates to the central server. These updates may encompass gradients, weights, or other model parameters adjusted during the local training process (see Algorithm 2). To uphold privacy, the updates can undergo encryption or masking before being transmitted to the server [65].

---

**Algorithm 2** Federated Learning: server-side aggregation  $T$

---

**Require:**  $T$ : num federated rounds

```

1: procedure AGGREGATING( $C, K$ )
2:   Initialize global model  $W^0$ 
3:   for  $t = 1, 2, \dots, T$  do
4:      $m \leftarrow \max(CK, t)$ 
5:      $S_t \leftarrow$  random set of  $m$  clients ▷ Selected Clients for round t
6:     for each client  $k \in S_t$  do ▷ Run in parallel
7:       send  $w^{t-1}$  to client  $k$ 
8:        $w_k^t \leftarrow$  CLIENT_UPDATE( $w^{t-1}$ )
9:     end for
10:     $w^t \leftarrow \sum_{k=1}^K \frac{n_k}{N} w_k^t$  ▷ Aggregating clients update
11:  end for
12:  Return  $w$ 
13: end procedure

```

---



#### Step 4: Aggregating the model updates into a new global model

The central server gathers the model updates from all involved client nodes and combines them to form an updated global model. This aggregation can employ diverse strategies, including averaging the model updates or utilizing advanced techniques like secure multi-party computation (SMPC) or differential privacy. The objective of this step is to generate a new global model that benefits from the collective knowledge of all participating nodes while preserving privacy [65].

#### Step 5: Rounds between server and clients

The federated learning process continues iteratively in rounds between client and server until the global model converges or reaches a fixed performance threshold (Figure 3.13), convergence can be monitored using various metrics like loss, accuracy, or other domain-specific measures, the number of iterations may depend on factors like the complexity of the problem, the size and diversity of the local datasets, and the desired level of model performance [65].

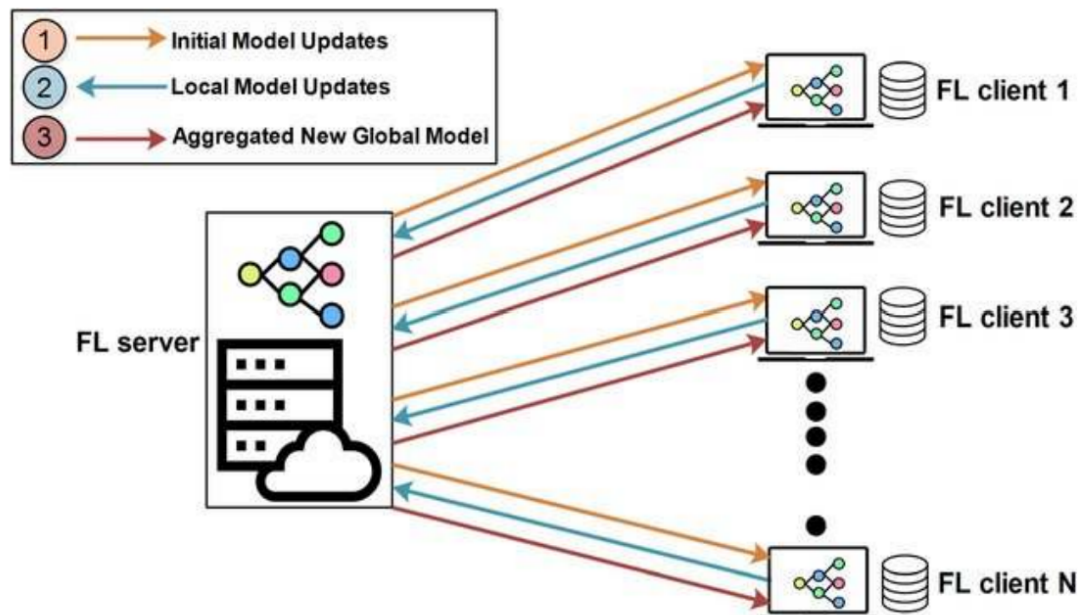


Figure 3.10: Rounds between server and client

### 3.5.2 Categorization of federated learning

In this section, we introduce different types of Federated Learning frameworks:

1. **Vertical Federated Learning** is used for cases in which each device contains dataset with different features but from sample instances. For instance, two organizations have data about the same group of people with different feature set can use Vertical FL to build a shared ML model [66].

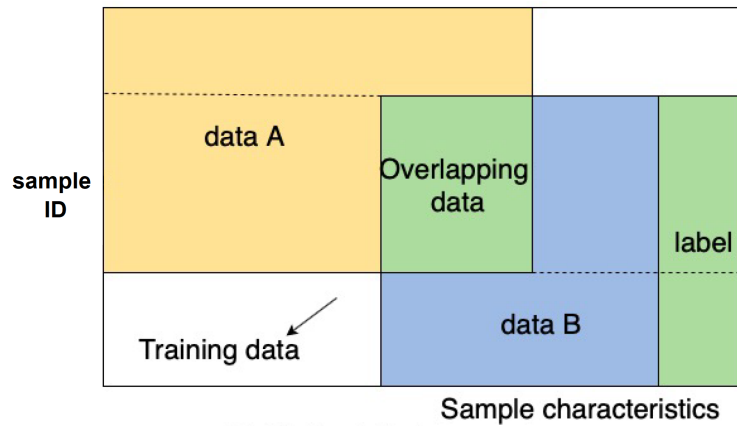


Figure 3.11: Vertical Federated Learning[80]

2. **Horizontal Federated Learning:** is used for cases in which each device contains dataset with the same feature space but with different sample instances. The first use case of FL- Google keyboard uses this type of learning in which the participating mobile phones have different training data with same features [66].

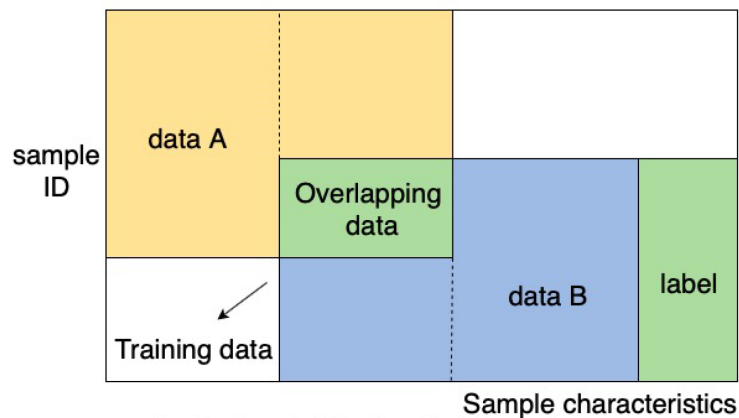


Figure 3.12: Horizontal Federated Learning[80]

3. **Federated Transfer Learning:** is similar to the traditional Machine Learning, where we want to add a new feature on a pre-trained model. The best example would be for giving an extension to the vertical federated learning. If we want to extend the ML to more number of sample instances which are not present in all of the collaborating organizations [66].

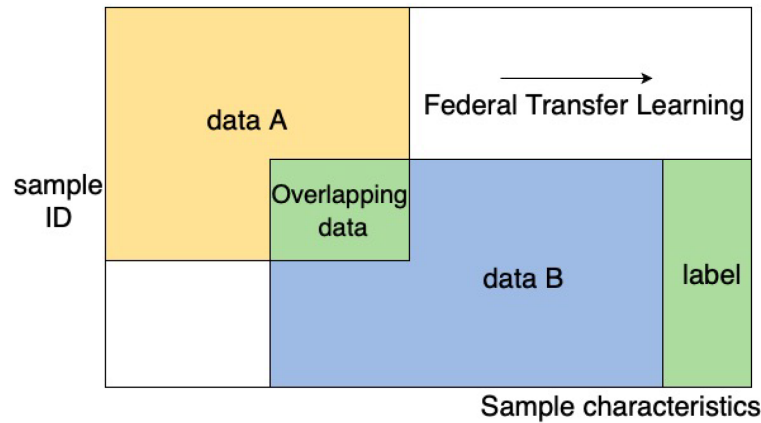


Figure 3.13: Federated Transfer Learning[80]

### 3.5.3 Model Aggregation algorithms

In federated learning, various model aggregation strategies are employed to merge models trained by different clients. Frameworks like Flower have integrated several of these strategies. In the domain of Intrusion Detection Systems (IDS). The selection of an aggregation strategy is contingent upon the specific application's needs and limitations. Here are some prevalent models of aggregation strategies utilized in federated learning.

1. **FedAvg Strategy** : Federated Averaging stands out as a widely used model aggregation strategy in federated learning. In this approach, clients train their local models on their individual datasets and subsequently transmit the updated model parameters to a central server. The server aggregates these updates by computing the average of the parameters and returns the updated global model to the clients. This iterative process continues until convergence is achieved [67].
2. **FedProx Strategy** : Federated Proximal builds upon the principles of FedAvg by incorporating a proximal term into the optimization objective. This addition serves to promote similarity between local models and the global model by penalizing significant parameter updates. By doing so, Federated Proximal addresses the challenge of data heterogeneity among clients and fosters stability in the learning process within federated settings [68].
3. **FedAdagrad Strategy** : Federated Adaptive Gradient presents an evolution of the Adagrad optimizer tailored for federated learning contexts. It integrates adaptive learning rates for individual parameters within the global model, factoring in both the frequency and magnitude of updates received from clients. By assigning distinct learning rates to each parameter, FedAdagrad adeptly manages non-IID data distributions across clients in federated learning scenarios [69].
4. **FedAdam Strategy** : Federated Adaptive Moment Estimation expands upon the widely-used Adam optimizer for federated learning scenarios. It merges the advantages of adaptive learning rates from Adam with the averaging mechanism inherent in federated learning. FedAdam dynamically adjusts the learning rates for each pa-

parameter by considering their historical gradients, enabling clients to learn at varying speeds [70].

### 3.5.4 Recent developments in federated Learning

#### 1. One-shot federated Learning:

In most of the federated learning frameworks, there will be multiple rounds of communication between devices and the central server, which increases the communication overheads. Recently, there is a growing interest in one-shot federated learning which is first introduced by [71], where the global model is learned in a single round of communication.

In order to overcome communication overheads of sending bulky gradients, [72] proposes a distilled one-shot federated learning, where each device distills their data and send the fabricated data to the central server. The server then learns the global model by training over the combined data from all the devices.

#### 2. Incentive Mechanisms:

Current FL approaches work under the assumption that devices will cooperate in the learning process whenever required without considering the rewards. Whereas in actual practice, devices or clients must be economically compensated for their participation. To encourage/improve device participation in FL, works such as [73][74] propose a reputation based incentive mechanism i.e., devices get rewards based on their model accuracy, data reliability and contribution to the global model. However, these works did not talk about how to model convergence and additional communication overheads induced into the framework.

#### 3. Federated Learning as a Service:

The machine Learning as a Service, primarily offering centralized services. However, with the increasing demand for privacy-preserving techniques like Federated Learning (FL), there's a need for FL to be incorporated into cloud services. This integration requires collaboration among third-party applications. A recent study [75] aimed to address this gap by developing a Federated Learning framework as a service. This framework enables third-party applications to contribute and collaborate on machine learning models, thereby facilitating the adoption of FL in cloud-based offerings.

#### 4. Asynchronous Federated Learning:

The majority of existing Federated Learning (FL) aggregation methods are tailored for devices operating synchronously. However, because of the heterogeneity in systems and data, training and model transfer occur asynchronously. Consequently, scaling federated optimization in a synchronous manner may not be feasible [76]. Studies like in [77][78] explore conducting federated learning in an asynchronous setting. Unlike FedAvg, which operates synchronously, asynchronous Federated Averaging techniques can accommodate a larger number of devices and accept updates at various times.

### 5. Blockchain in FL:

An aggregator is essential for updating the global model while handling the asynchronous arrival of parameters from devices. However, this requirement may hinder the widespread adoption of Federated Learning (FL) models. Alternatively, blockchain, being a decentralized network, enables devices to learn collaboratively without a central aggregator [79].

## 3.6 Conclusion

Artificial intelligence is a highly active research field, continuously making progress to enhance performance results.

One of the recent emerging areas in artificial intelligence is federated learning, which represents a promising approach to address privacy concerns while benefiting from collective learning of distributed devices. Through examining related works, providing an overview of artificial intelligence, and exploring federated learning in detail, including its processes, categorization, model aggregation algorithms, and recent advancements, a comprehensive understanding of this evolving field has been achieved.

The comparison between centralized and decentralized learning architectures underscores the importance of federated learning in contemporary artificial intelligence research.

As federated learning continues to evolve and witness advancements, it holds immense potential to revolutionize collaborative model training across decentralized networks.

## Chapter 4

# Experiment, Results and Discussion

## 4.1 Introduction

In this chapter, we will introduce the environment and tools used in our work, provide a description and preprocessing of the dataset, and present all the experiment results using various types of graphs, bar charts, tables, and some text to illustrate the different presentations. We will demonstrate the model's performance in binary classification and multi classification in centralized and decentralized architectures, test several aggregation strategies, and finally compare the results in both architectures. This chapter aims to enhance the model's performance while safeguarding the confidentiality of the clients dataset in decentralized architectures.

## 4.2 Experimental Environment

In this section, we will present the hardware and software, and libraries then the data used in our work.

### 4.2.1 Material Tools

The experiment were conducted on CPU Intel(i7) with 2 cores and 8 GB RAM to implement the proposed approach and construct our model. We utilized the jupyter editor IDE.

### 4.2.2 programming language used

In our work we use python[81], a high-level general purpose programming language widely used in data science and to produce deep learning algorithms. In this brief tutorial we offer libraries and frameworks used. Our model needs to perform better than this. we are using the following libraries: Pandas, Numpy, Keras, Matplotlib, TensorFlow and flower.

#### 4.2.2.1 Libraries used

**TensorFlow[82]:** TensorFlow is an end-to-end open source platform for machine learning. It offers a complete and flexible ecosystem of tools, libraries and community resources allowing researchers to advance in the field of machine learning, and developers to easily create and deploy applications that exploit this technology.

**Pandas[83]:** Pandas is an open-source Python library that uses strong data structures to provide high-performance data manipulation and analysis. Pandas is derived from the term Panel Data, which is an Econometrics from Multidimensional data. Wes McKinney, a developer, began developing pandas in 2008 in response to a demand for a high-performance, versatile tool for data analysis. Python was mostly used for data munging and preparation prior to Pandas. It made very little contribution to data analysis. Pandas solved this issue. We can use Pandas to do five common phases in data processing and analysis, independent of data origin: load,

prepare, manipulate, model, and analyze. Python with Pandas is utilized in a variety of academic and commercial disciplines such as finance, economics, statistics, analytics, and so on.

**Scikit-Learn**[84]: Scikit-Learn is a versatile machine learning library in Python, with a wide range of tools for dataset preprocessing. This essential step involves converting raw data into a format suitable for modeling. Scikit-Learn offers several preprocessing techniques, such as scaling, normalization, encoding categorical variables, and handling missing values.

**Numpy**[85]: This library, whose name means numerical Python, constitutes the core of many other Python libraries that have originated from it. Indeed, NumPy is the foundation library for scientific computing in Python since it provides data structures and high-performing functions that the basic package of the Python cannot provide. In fact, NumPy defines a specific data structure that is an N dimensional array defined as ndarray.

**Keras**[86]: Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

**Matplotlib**[87]: This package is the Python library that is currently most popular for producing plots and other data visualizations in 2D. Since data analysis requires visualization tools, this is the library that best suits this purpose.

**Flower**[88]: Flower is a recent FL framework that provides higher-level abstractions enabling researchers to extend and implement FL ideas on a reliable stack. It is one of the very few frameworks that can support heterogeneous clients running on different ML frameworks and using different programming languages. Flower also has a large suite of built-in Strategies representing state-of-the-art FL algorithms for users to freely extend, modify, and use for their experiments.

Table 4.1 shows the versions of the libraries used in this work.

Libraries	Version
Tensorflow	2.15
keras	2.12
Scikit-Learn	1.3.2
pandas	2.1.3
framework Flower	2.0.1
Numpy	1.26.2
Matplotib	3.8.1

Table 4.1: Version of libraries used in work environment



### 4.3 dataset description

Data sets are used to accurately assess the model’s ability to detect attacks. The quality of the dataset ultimately affects the results of any network intrusion detection system (NIDS). Here, in our research we use the UNSW-NB15 dataset, which plays a crucial role in understanding, combating and detecting cyberattacks. The reasons for his choice are described below:

- This collection has been characterized by the availability of up-to-date and comprehensive data on cyberattacks, making it easier for us to study attackers’ behavioural patterns and extract security patterns and warnings.
- This collection is a rich source of high volume data to contain about 2,540,044 network traffic records, which can be used to develop and test our intrusion detection model and improve its accuracy.
- It contains a large number of cyber attacks compared with other data such as NSL-KDD and this makes our model comprehensive identification and identification of attacks.

Both the following table and the Relative Service aim to give an overview of the types of attacks and their distribution within the unsw-nb15 dataset:

Type of Attack	Number of traffic records	Percentage
Normal	60000	36.1%
Analysis	2677	1.04%
Backdoor	2329	0.90%
DoS	16353	6.35%
Exploits	44525	17.3%
Fuzzers	24246	9.41%
Generic	58871	22.8%
Reconnaissance	13987	5.43%
Shellcode	1511	0.58%
Worms	174	0.067%

Table 4.2: Percentage distribution of attack types in the dataset

### 4.4 Preprocessing dataset

It is a set of operations and manipulations that we apply to our data set to make it fit and adapt to our model, and then train it. The steps are:

**Data collection:** We have imported the dataset into the working environment, which consists of the training file and the test file, fetched from the official website

**Data cleaning:** The goal of data cleaning is to verify accuracy, ensure consistency and remove errors within the dataset. Incorrect or inconsistent data are likely to negatively affect the performance of machine learning models that are:

- **Identify and deal with missing values:** It is necessary to identify and deal with missing values to prevent inaccurate conclusions. However, in our dataset, no missing values were identified.
- **Delete duplicate field data :** This step is to search for duplicate records and remove them from the dataset which can simplify them, reduce jamming and improve accuracy quality.

**Encoding Categorical Data:** Given that machine learning models require numerical inputs, encoding categorical data is essential. The UNSW-NB15 dataset contains several categorical features that need to be transformed. These features include 'proto', 'service', 'state', and the attack type feature 'attack\_cat'. We use two methods were used to determine which provides better results:

- **One-Hot Encoding:** Converts each category into a column with values of 1 or 0.
- **Numerical Encoding:** Replaces each category with a numerical value.

After testing both methods, it was found that Numerical Encoding yielded better results than One-Hot Encoding.

**Splitting the Dataset:** We split the dataset for the purpose of training and testing the model. Following the data pre-processing step, we split the dataset into three subsets: the training, validation, and testing sets. We train the proposed model on the training set, which contains 60% of all the data. The validation set has 20% of the data, tunes the model's hyperparameters, and evaluates its performance during training. The last 20% of the data is the testing set, which verifies how well the model works after training and validation.

This splitting strategy helps to avoid overfitting and test the model's generalization ability. Overfitting occurs when the model undergoes extensive training on the training data, causing it to memorize the data rather than understanding the underlying patterns. This can make it challenging to do well with new data. The validation set is used to tune the model's hyperparameters, which are settings that control the learning process.

**Feature Scaling:** Standardize independent features to a fixed range during data pre-processing. This is done to handle variations in sizes, values, or units. Two methods include:

- **Normalization:** Rescale values to a range of [0-1].
- **Standardization:** Rescale values to have a mean of 0 and a standard deviation of 1.

After testing both methods, it was found that Standardization yielded better results than Normalization.

The goal of testing both feature scaling and encoding categorical data steps was to use them in subsequent experiments to achieve better results.

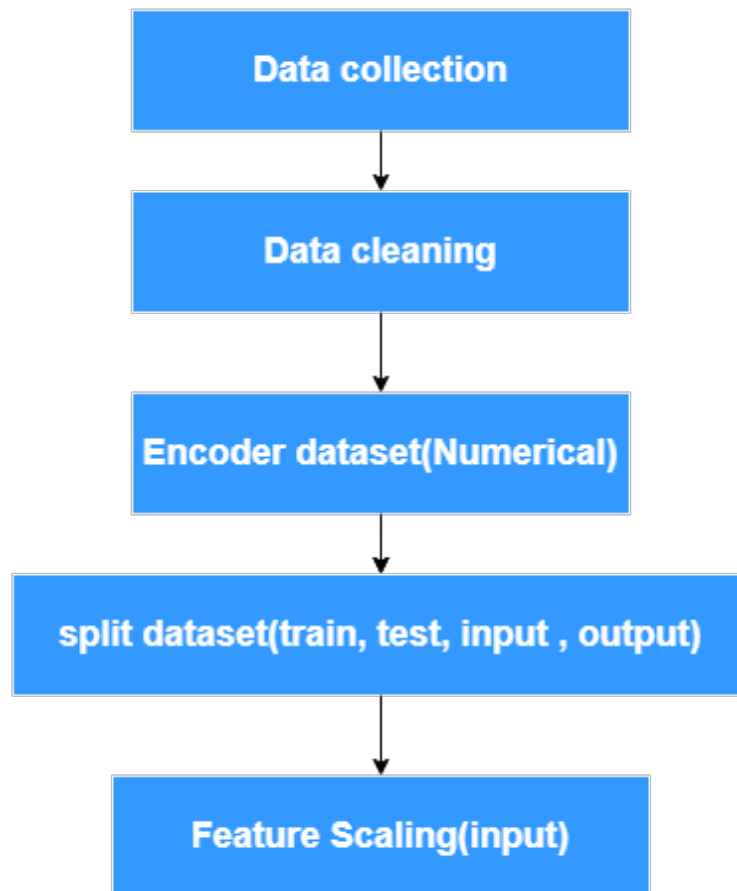


Figure 4.1: The step of preprocessing dataset

## 4.5 Dataset distribution

In the federated learning approach, datasets are distributed to clients in two forms: IID (Independent and Identically Distributed) and non-IID (Non-Independent and Identically Distributed)

### 4.5.1 Independently and identically distributed (IID)

In the context of federated learning, Independently and Identically Distributed (IID) refers to a scenario where datasets distributed to multiple clients exhibit similar statistical properties and are drawn from the same distribution.

Research has consistently shown that federated learning algorithms perform optimally under IID conditions. This effectiveness stems from the alignment of objectives between the central server orchestrating the learning process and the individual clients contributing their local data. In IID settings, the assumptions of statistical independence and identi-

cal distribution facilitate efficient model aggregation and learning convergence across all participating clients.

Based on Figure 4.2, we distributed the dataset into 10 clients as they show a similar data distribution, which means that each customer’s dataset shares similar statistical characteristics with the others.

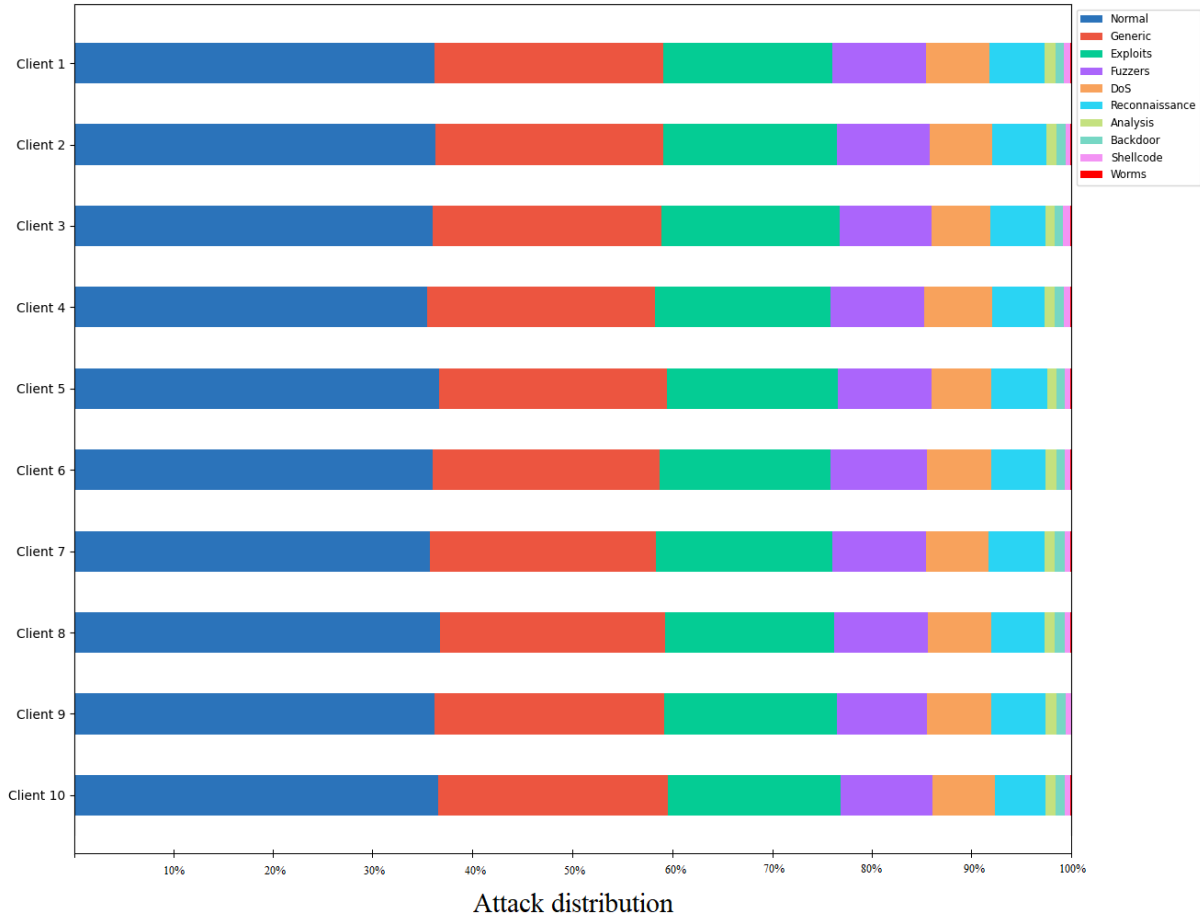


Figure 4.2: Independently and identically distributed (IID)

## 4.5.2 Non Independently and identically distributed (Non IID)

Non-Independently and Identically Distributed (Non-IID) refers to datasets distributed among multiple clients in a federated learning system where the statistical properties of the data vary significantly across clients. Unlike IID distributions where all clients receive comparable data distributions, Non-IID distributions reflect real-world scenarios where data ownership and characteristics differ among clients.

In order to generate non-IID data sets in different cases, we utilized the Dirichlet distribution [90]. The Dirichlet distribution is a crucial multi-dimensional continuous distribution in probability statistics, commonly denoted as  $Dir(\alpha)$ , and is governed by the parameters of the positive real vector  $\alpha$ . The Dirichlet distribution can be defined as follows: let  $\theta = [1, \dots, m]$  be an  $m$ -dimensional vector, where for any  $i \in [1, 2, \dots, m]$ ,  $\theta_i \geq 0$  and  $\sum_{i=1}^m \theta_i = 1$ . Let  $\alpha$  be a  $k$ -dimensional vector,  $\alpha = [\alpha_1, \dots, \alpha_k]$ , where for

any  $i \in [1, 2, \dots, k]$ ,  $\alpha_i > 0$ . If  $Dir(\alpha_1, \dots, \alpha_k)$ , then the Dirichlet distribution probability density function is given by[91]:

$$P(\theta_1, \dots, \theta_m) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^m \theta_k^{\alpha_k - 1} \quad (4.1)$$

$$\Gamma(\alpha_k) = \int_0^{+\infty} x^{\alpha_k - 1} e^{-x} dx (x > 0) \quad (4.2)$$

The parameter  $\alpha$  can control the skew degree of the generated data distribution. The smaller  $\alpha$ , the higher the non-IID level of each client’s data distribution[91]. Figure 4.3, we show the data distribution for 10 labels(type attacks) and 10 clients when  $\alpha = 0.5$ .

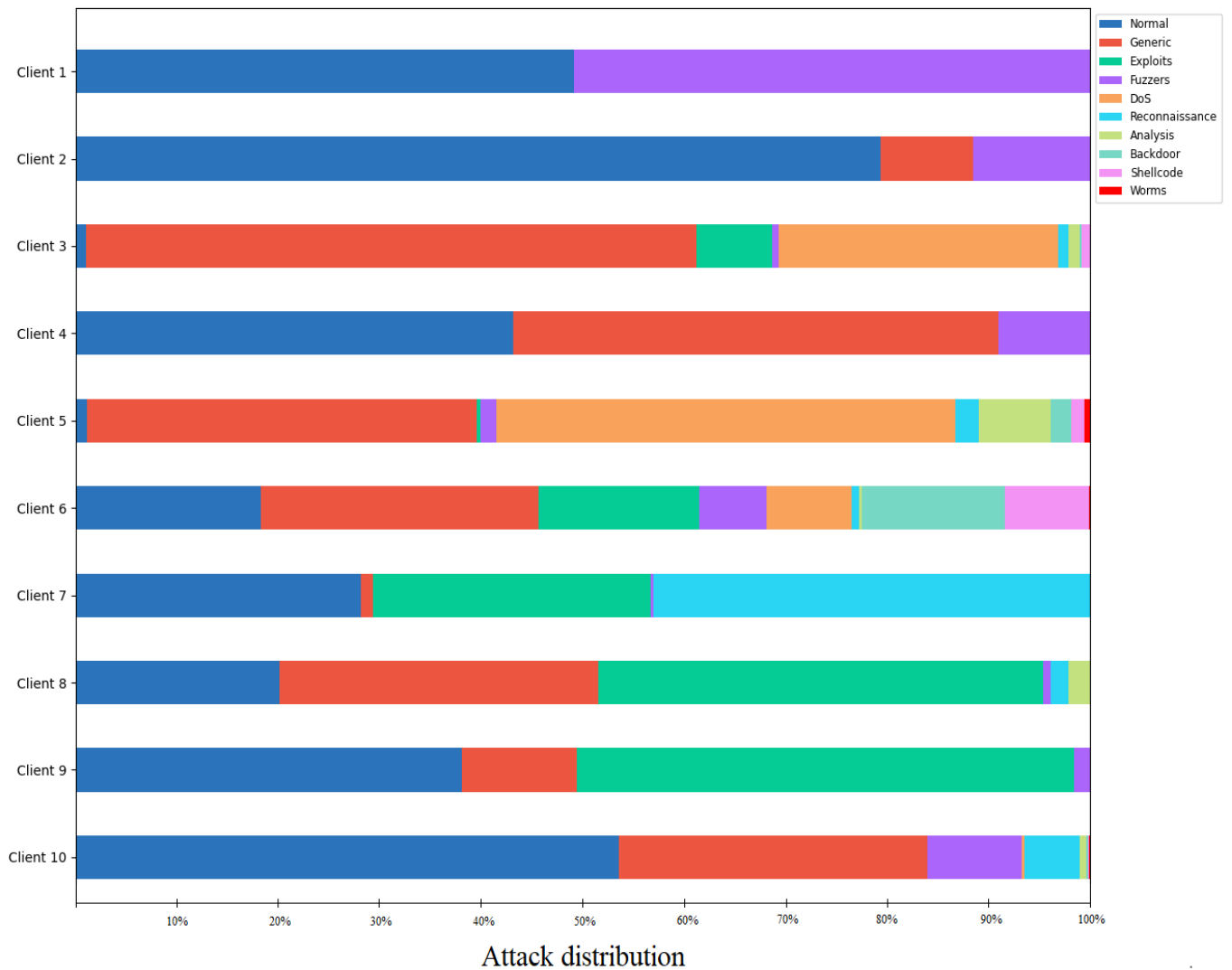


Figure 4.3: Non Independently and identically distributed (IID)

## 4.6 Centralized Model based Deep Learning

In this study, before starting the federated learning process, it was necessary to carefully select the model to achieve better results. Therefore, we have used different deep learning

technique . Several other deep learning algorithms were tested, Convolutional neural network (CNN), Deep neural network (DNN), Recurrent neural network (RNN), and Long Short-Term Memory (LSTM). The results were as shown in the and Table 4.6.

Table 4.3: A comparison between the accuracy results of the different approaches

Model	Accuracy	Precision	Recall	F1-score
CNN	95.92%	95.97%	95.92%	95.93%
DNN	93.81%	93.82%	93.81%	93.81%
RNN	90.47%	91.49%	90.47%	90.61%
LSTM	94.13%	94.14%	94.13%	94.14%

The following Figure 4.4 better illustrates the accuracy and measurements of the proposed models and shows the extent to which the CNN model outweighs the rest of the models.

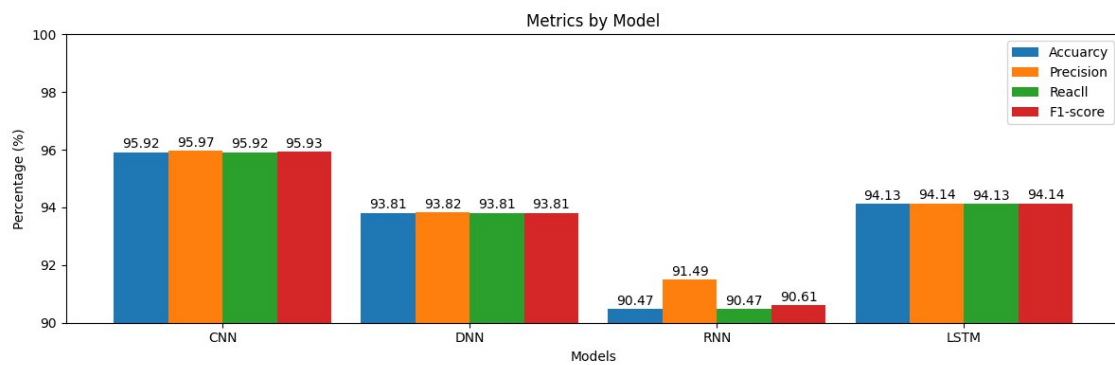


Figure 4.4: A comparison of the accuracy outcomes achieved by various model.

### 4.6.1 Hyperparameter Settings of CNN model

We chose the CNN model to implement it in the decentralized approach (Federated Learning).

Table 4.4 presents the CNN architectures. The model comprises convolutional layers, pooling layers, dropout regularization, and fully connected layers. The input shape is determined by the features and data depth. For binary classification, the model includes a single neuron with a sigmoid activation function, while for multi-class classification, it contains 10 neurons with a softmax activation function. The model optimizes using cross-entropy loss and the Adam optimizer.

the Figure 4.5 illustrates a summary of the CNN model, identifying each layer, output shape, and parameter.

Layer	Parameter
Convolutional Layer	3 Layer is 1D
MaxPooling	3 layer
Activation function in hidden layer	ReLU
Dropout	20%
Optimizer function	Adam
Activation function : binary classification	Sigmoid
Activation function : multi classification	Softmax
Loss function on binary classification	Binary cross-entropy
Loss function on multi classification	Categorical cross-entropy

Table 4.4: Architectures and parameter of CNN model

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 40, 32)	128
conv1d_1 (Conv1D)	(None, 38, 32)	3104
max_pooling1d (MaxPooling1D)	(None, 12, 32)	0
conv1d_2 (Conv1D)	(None, 10, 64)	6208
conv1d_3 (Conv1D)	(None, 8, 64)	12352
global_average_pooling1d (GlobalAveragePooling1D)	(None, 64)	0
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 64)	4160
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 10)	330
=====		
Total params: 28,362		
Trainable params: 28,362		
Non-trainable params: 0		

Figure 4.5: Summary of CNN model

### 4.6.2 Architecture of local Model Training

The figure 4.6 illustrates the structure of the local model training process, starting with the preprocessing of the dataset, followed by training and testing the CNN model on the test dataset.

We categorize the results into two types: binary classification, which determines whether the traffic is normal or an attack, and multi-class classification, which identifies the specific type of attack or normal if there is no attack.

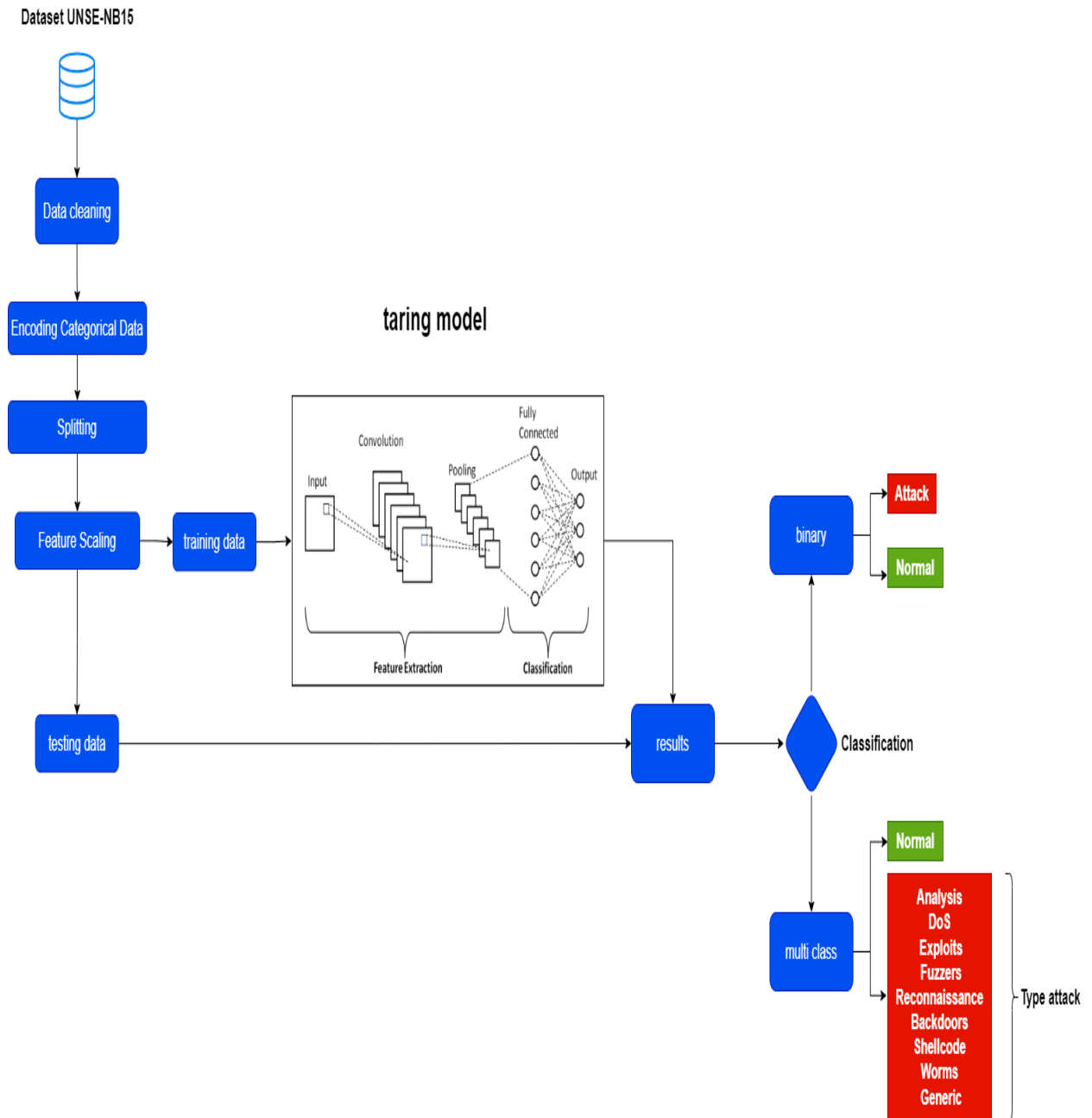


Figure 4.6: Architecture of local CNN Model Training



### 4.6.3 Cross-validation in model CNN

K-fold Cross Validation (KCV) is a commonly used method by professionals to pick models and estimate errors of classifiers[92]. In our work, the dataset was divided into 5 subsets, with subsets used to train the model, while the remaining subsets were used to evaluate its performance as shown in Figure 4.7:

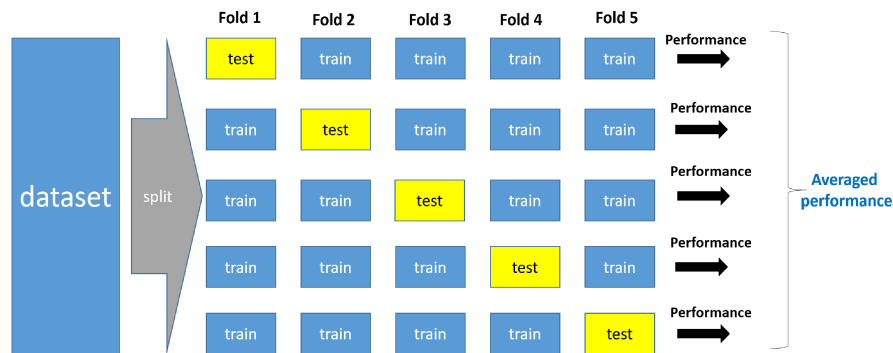


Figure 4.7: Architecture of cross validation

The figure 4.8 present the accuracy of the model with different folds of cross-validation. The accuracy is relatively high and consistent across all folds.

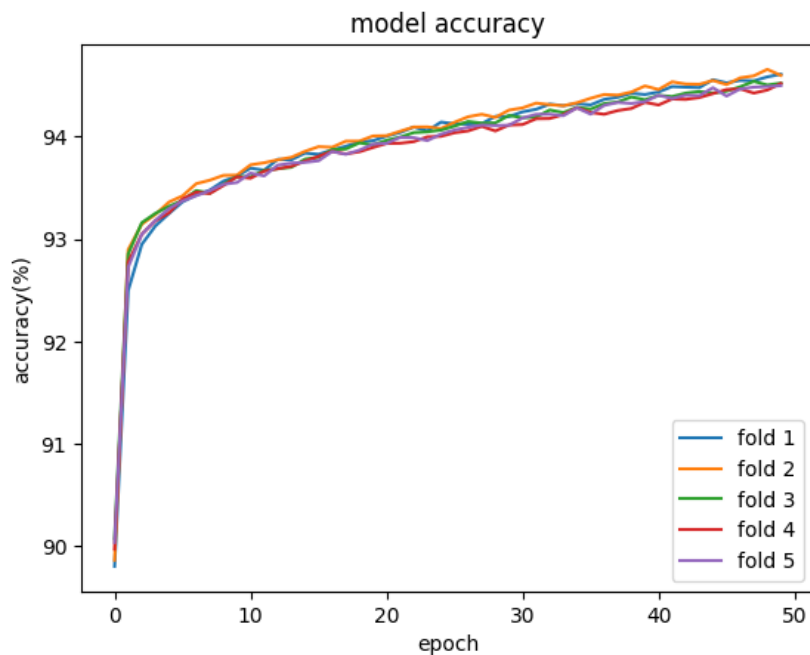


Figure 4.8: Accuracy by applying 5-fold cross validation

## 4.7 Architectures of Decentralized Model FedCNN-IDS

The Figure 4.9 illustrates the structure of the proposed FedCNN-IDS model. In this model, clients train locally using the CNN model. When each client finishes training, it sends the resulting weight parameters to the server. The server then aggregates all the weights sent by the clients using the FEDAVG algorithm. After the aggregation process, the weights are sent back to the clients, and this process continues for a predetermined number of rounds.

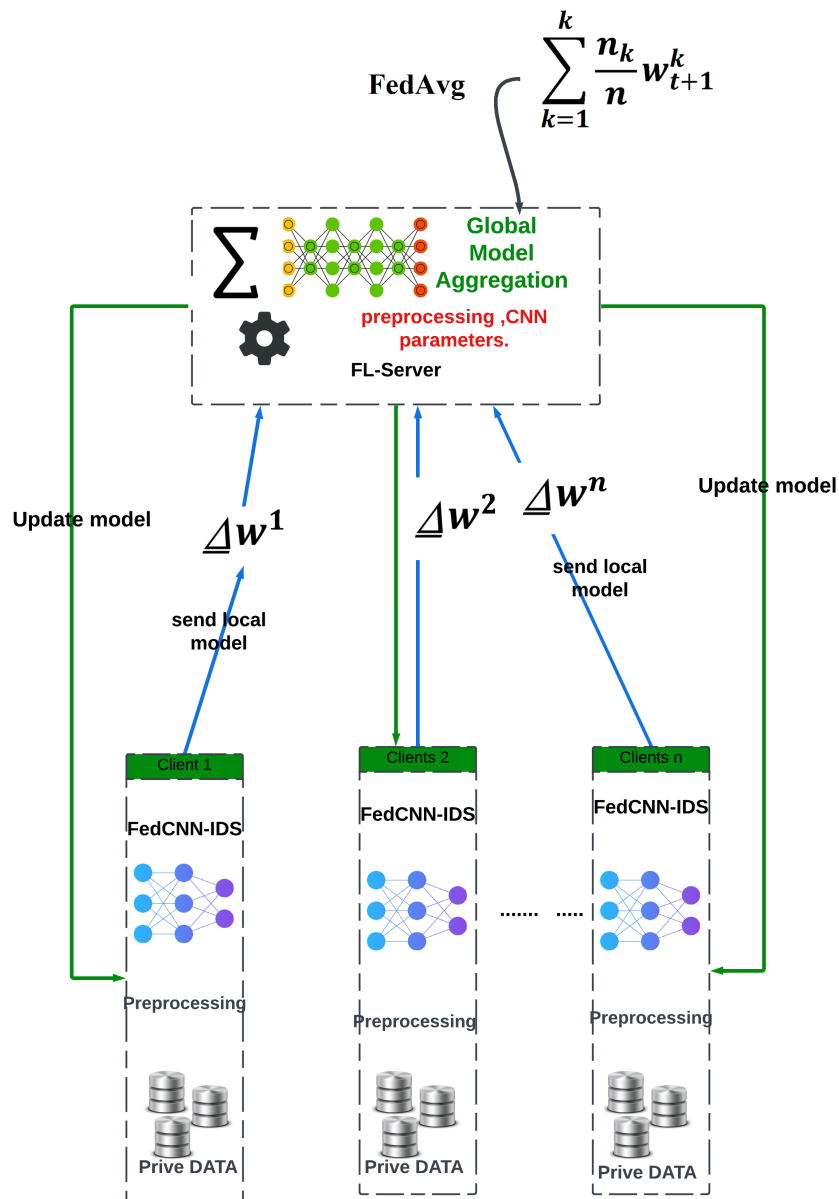


Figure 4.9: Architectures of proposed Model FedCNN-IDS in federated learning

### 4.7.1 Federated learning Parameters

The total of 10 clients participate in implantation of our decentralized model. Each round, we choose 70% of the clients to take part in the training. Each client trains on its local dataset for 50 epochs with a batch size of 128. There are 50 communication rounds between the client and the server.

<b>Hyperparameter Settings</b>	
Proposed Model Approach	CNN
Total of clients	10
Method aggregation	FedAvg
Epochs	10
Batch size	128
Number of communication rounds	50

Table 4.5: Hyper parameter settings of federated learning architecture

## 4.8 A proposed aggregation strategy

Figure 4.10 illustrates the results of testing set of different aggregation strategies. The FedAvg and FedProx strategies showed close performance, in contrast to other strategies like FedAdam and FedAdagrad, which demonstrated poorer convergence and less results in accuracy.

Based on these results, the FedAvg and FedProx strategies demonstrated superior performance. For our work, we selected the FedAvg aggregation strategy as it yielded slightly better results compared to the FedProx strategy.

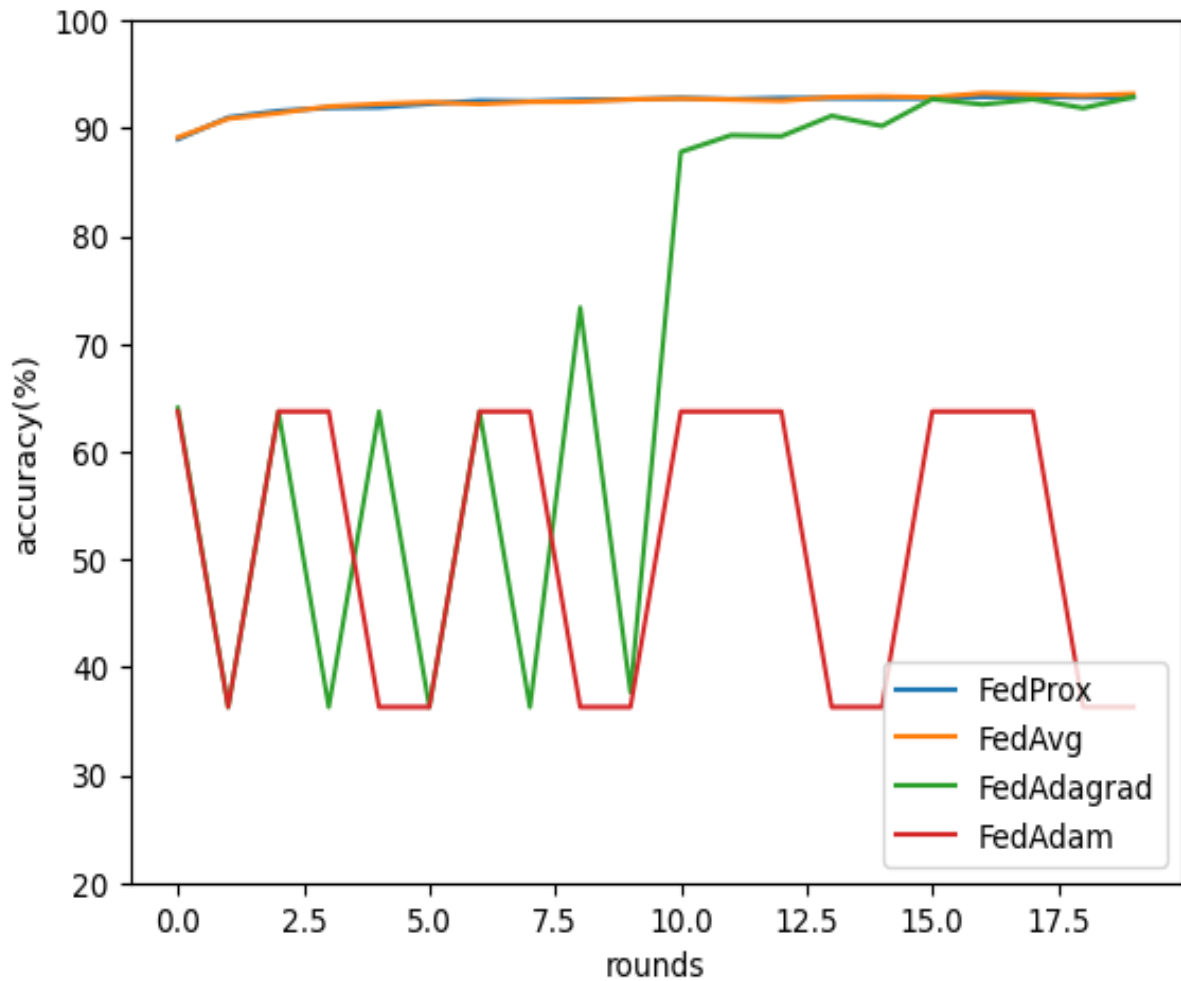
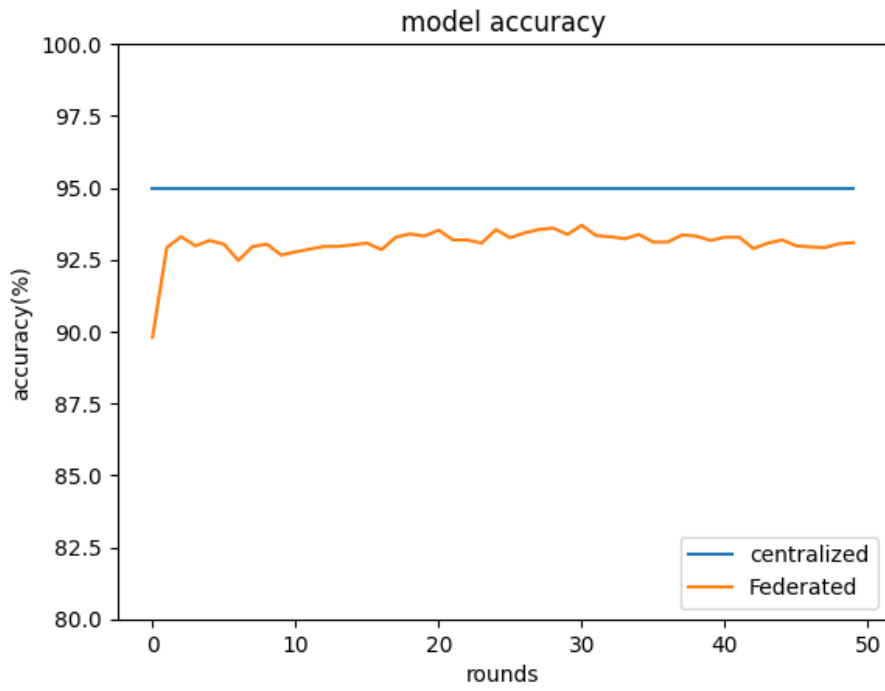


Figure 4.10: The results of testing different combinations of aggregation strategies

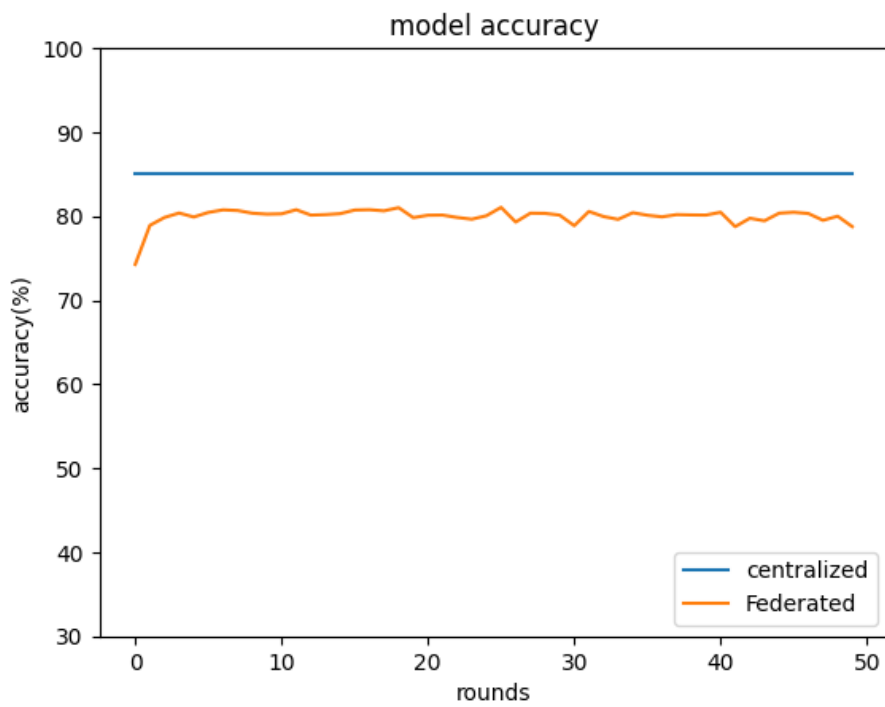
## 4.9 Results in IID and Non-IID Distributions

We trained the model on both IID and non-IID distributions in binary and multi-class classifications and compared the outcomes to the centralized architectures.

The results (Figure 4.11 and Figure 4.12) obtained in IID and Non-IID distributions, in both binary and multi-class classifications, demonstrate an improvement and convergence in the accuracy of the model between centralized learning and federated learning. federated learning achieved higher accuracy on both classifications compared to centralized learning. However, the difference in accuracy was smaller for binary classification compared to multi-class classification, this is due to the distribution of dataset over the clients. The results were as follows:

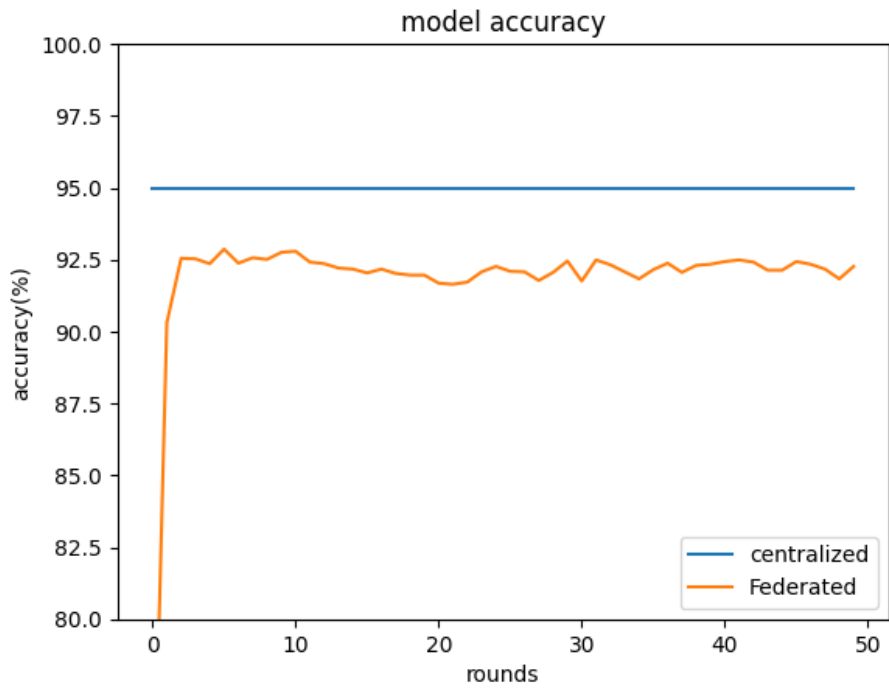


(a) binary classification

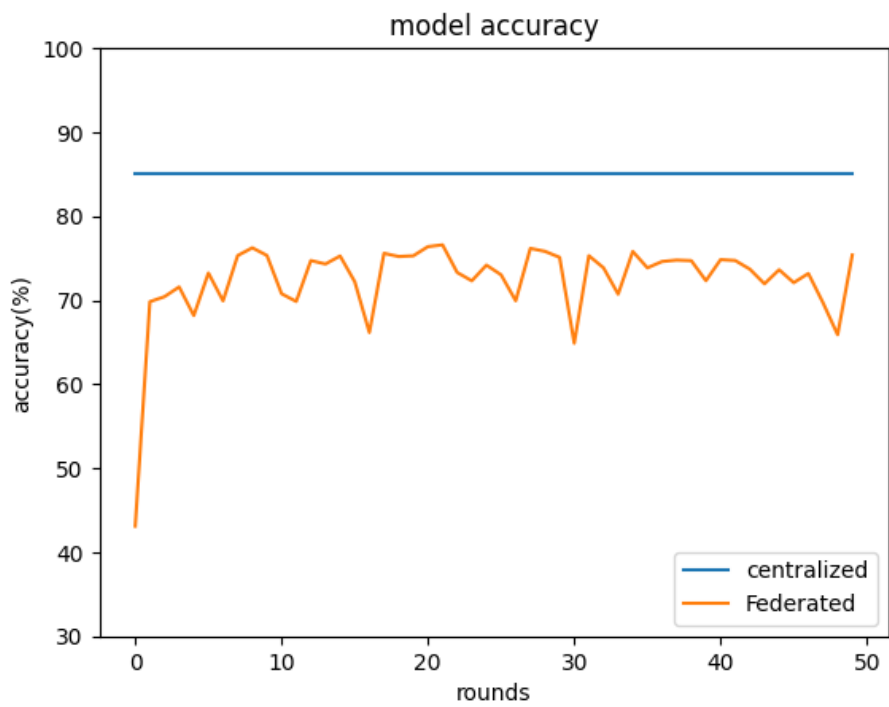


(b) multiclass classification

Figure 4.11: The accuracy of the model between centralized and federated learning in the IID distribution.



(a) binary classification



(b) multiclass classification

Figure 4.12: The accuracy of the model between centralized and federated learning in the Non-IID distribution.

## 4.10 Discussion

Preprocessing a dataset is one of the most important steps to achieve good results. We performed several preprocessing steps after collecting the dataset.

First, we ensured that our data set was free of missing values to prevent problems with the model. We also checked for and removed duplicate samples. Next, we checked for the presence of text values and the encoding applied by replacing each category with a numeric value. After experimenting with the two types of encryption mentioned previously, we found that digital encryption gave better results than single encryption. Next, we split the dataset into training and testing sets, separating the inputs and outputs. For binary classification, the outputs were binary, located in the Label column, and for multi-class classification, the column was `Attack_cat`. In addition, we observed that there are differences in values across different features. To address this issue, we applied normalization. This approach provided better results compared to standardization.

After preparing the data, we conducted experiments on several deep learning models, including LSTM, RNN, DNN, and CNN. After obtaining the results, we found that the CNN model gave better results compared to other models, with an increase in the percentages of evaluation metrics, accuracy, precision, recall, and F1 score.

After choosing the CNN model, we moved on to unified learning, so that our new proposed model, FedCNN-IDS, based on unified deep learning, uses a decentralized training technique with central assembly. This means that, unlike traditional mainframe systems, there is no central server responsible for training and storing the data. Instead, we distribute the training process across multiple clients, each of which stores its own local data, and we coordinate the learning process through a central aggregation mechanism, such as the FedAvg algorithm we use. This decentralized approach ensures that no single authority controls the entire network, and it overcomes the challenges of centralized data storage and processing and privacy concerns in sharing sensitive data. Therefore, we conducted extensive experiments with the FedCNN-IDS model in both binary and multi-class classifications, as well as in different scenarios represented by IID and non-IID distributions, and the results were compared with those of centered learning. The results were close and the performance was high, which indicates the high confidentiality in maintaining the data set with clients.

One of the fundamentals of federated learning is the clustering strategy. Therefore, we tested a range of strategies, and the results showed that both the FedAvg and FedProx strategies provided good and comparable accuracy compared to the other strategies. Based on these results, we suggested using the FedAvg strategy, due to it achieving slightly better results than FedProx.

These results are especially noteworthy when considering non-IID (non-independent and identically distributed) scenarios, where data is unevenly distributed across different clients. The performance of our FL model demonstrates its robustness and adaptability in these complex data environments, highlighting its potential for real-world applications in network intrusion detection.

The experimental results obtained indicate that our federated learning (FL) model achieves good results in both binary and multiclass classification, which is equivalent and

comparable to existing centralized architectures and works. To validate our approach, we performed a comparison with two models using the same UNSW-NB15 dataset.

Table 4.6 shows the accuracy comparison between our proposed model and other works. Notably, we addressed binary and multi-class classification, while other works mostly focused on a single classification type. These results highlight the effectiveness of our approach on various classification tasks and confirm the robustness of the preprocessing and model selection steps.

Architecture	Model	Dataset	Binary (Acc)	Multi-class (Acc)
Centralized	Semi-Supervised Learning[30]	UNSW-NB15	N/A	N/A
	CNN [31]	UNSW-NB15	N/A	83.46%
	<b>Our model</b>	<b>UNSW-NB15</b>	<b>95.92%</b>	<b>85.04%</b>
Federated	Semi-Supervised Learning[30]	UNSW-NB15	84.32%	N/A
	CNN [31]	UNSW-NB15	N/A	81.19%
	<b>Our model</b>	<b>UNSW-NB15</b>	<b>93.61%</b>	<b>78.15%</b>

Table 4.6: Performance comparison of our proposed techniques with state of arts works.

## 4.11 Conclusion

In this chapter, we conducted several experiments on the UNSW-NB15 dataset and compared several methods. We chose a CNN to evaluate its performance in both centralized and decentralized (federated learning) architectures. The accuracy was close in both centralized and decentralized architectures. This shows that federated learning is highly accurate and protects both data privacy and model security in the network intrusion detection system.



# General Conclusion

The primary objective of this thesis was to explore and enhance the application of federated learning within the context of network intrusion detection systems (NIDS). Through a comprehensive study encompassing information security principles, machine learning techniques, and federated learning methodologies, we have provided valuable insights and contributions to the field.

Our investigation started with a thorough review of information security and intrusion detection systems, and we discussed the essential pillars and goals of information security, as well as the various types of attacks and defense mechanisms. This groundwork underscored the critical role of intrusion detection systems in safeguarding network infrastructure. In the second part of the thesis, we delved into the state-of-the-art in artificial intelligence intrusion detection systems for various datasets, focusing on machine learning and deep learning techniques. We reviewed key methodologies, including convolutional neural networks (CNNs) and their applications in intrusion detection. This provided a robust framework for understanding the potential and limitations of traditional, centralized learning models.

We used the intrusion detection system dataset UNSW-NB15 to validate and evaluate the accuracy of our proposal. The dataset contains various indicators for multiple attacks blended in with normal traffic. The core of our research focuses on federated deep learning for intrusion detection. We utilized both centralized and decentralized learning approaches, with a particular emphasis on the decentralized approach through federated learning. By leveraging CNNs within a federated learning framework, we aimed to create a scalable and privacy-preserving NIDS. We validated our approach through extensive experiments using both IID and non-IID data distributions, demonstrating significant improvements in accuracy and robustness while maintaining data security and privacy.

Our experimental results highlighted the effectiveness of federated learning in dealing with non-IID data distributions, a common scenario in real-world networking environments. The decentralized federated learning model showed similar performance to centralized models while providing improved data privacy and security. Our findings also indicate that federated learning not only maintains high accuracy but also addresses key privacy concerns associated with centralized data processing.

The future direction of this work aims towards, we plan to study other models in federated deep learning, including split learning. Other than improving the accuracy of the proposed model, we want to include more features to the work such as feature selection, and client selection, and we are also looking to use our proposed method in another dataset to create a robustness analysis of our approach.

# Bibliography

- [1] M. J. Idrissi, H. Alami, A. El Mahdaouy ,A. El Mekki ,S. Oualil ‘Fed-ANIDS: Federated learning for anomaly-based network intrusion detection systems’, *Expert Systems with Applications*, vol. 234, p. 121000, Dec. 2023, doi: 10.1016/j.eswa.2023.121000.
- [2] A. A. Mishra, K. Surve, U. Patidar, and R. K. Rambola, ‘Effectiveness of Confidentiality, Integrity and Availability in the Security of Cloud Computing: A Review’, in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, India: IEEE, Dec. 2018, pp. 1–5. doi: 10.1109/CCAA.2018.8777537.
- [3] K. Rihaczek, ‘The harmonized ITSEC evaluation criteria’, *Computers & Security*, vol. 10, no. 2, pp. 101–110, 1991.
- [4] Qadir, Suhail, and Syed Mohammad Khurshaid Quadri. ”Information availability: An insight into the most important attribute of information security.” *Journal of Information Security* 7.3 (2016): 185-194. National Security Agency US.
- [5] Pfleeger, Charles P., and Shari Lawrence Pfleeger. *Analyzing computer security: A threat/vulnerability/countermeasure approach*. Prentice Hall Professional, 2012.
- [6] Menkus, Belden. ”Introduction to computer security.” *Computers & Security* 11.2 (1992): 121-127.
- [7] Stallings, W., & Brown, L. (2018). *Computer Security: Principles and Practice* (4th ed.). Pearson.
- [8] Aissaoui Sihem, *automatic operation and security of systems information: Application: un Intrusion detection system based on SVM*, University of Oran, 2008
- [9] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, ‘A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection’, *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 686–728, 2019, doi: 10.1109/COMST.2018.2847722.
- [10] William Stallings, *network security essentials: applications and standards fourth edition*,2011.
- [11] A. S. Ashoor and S. Gore, ‘Importance of Intrusion Detection System’, *International Journal of Scientific & Engineering Research*, 2010, ISSN: 2229-5518.

- [12] Kruegel, Christopher, Fredrik Valeur, and Giovanni Vigna. *Intrusion detection and correlation: challenges and solutions*. Vol. 14. Springer Science & Business Media, 2004.
- [13] Halme, Lawrence R., and R. Kenneth Bauer. "AINT misbehaving: A taxonomy of anti-intrusion techniques." *Proceedings of the 18th national information systems security conference*. 2000.
- [14] Richard Heady, George F Luger, Arthur Maccabe, and Mark Servilla. *The architecture of a network level intrusion detection system*. University of New Mexico. Department of Computer Science. College of Engineering, 1990.
- [15] F. Sabahi & A. Movaghar. *Intrusion detection: A survey*. In *Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on*, pages 23–26. IEEE, 2008.
- [16] Eric. Li and M. Datardina, 'Intrusion Detection Systems'.
- [17] Scarfone, Karen, and Peter Mell. "Guide to intrusion detection and prevention systems (IDPS)." *NIST special publication 800.2007 (2007)*: 94.
- [18] Anitha, A., and V. Vaidehi. "Context based application level intrusion detection system." *International conference on Networking and Services (ICNS'06)*. IEEE,pp. 16-21, 2006.
- [19] Kaushik, Sapna S., P. R. Deshmukh. "Detection of attacks in an intrusion detection system." *International Journal of Computer Science and Information Technologies (IJCSIT) 2.3 (2011)*: ISSN:0975-9646.
- [20] Suwannalai, Ekachai, and Chantri Polprasert. "Network intrusion detection systems using adversarial reinforcement learning with deep Q-network." *2020 18th International Conference on ICT and Knowledge Engineering (ICT&KE)*. IEEE,pp. 1–7, Bangkok, Thailand, Nov 2020.
- [21] L. Chen, X. Kuang, A. Xu, S. Suo, and Y. Yang, "A novel network intrusion detection system based on CNN," in *2020 Eighth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 243–247, Taiyuan, China, Dec 2020.
- [22] Zhao, Y., Chen, J., Wu, D., Teng, J., Yu, S.: Multi-task network anomaly detection using federated learning. In: *Proceedings of the Tenth International Symposium on Information and Communication Technology*, pp. 273–279 (2019).
- [23] Man, D., Zeng, F., Yang, W., Yu, M., Lv, J., Wang, Y.: Intelligent intrusion detection based on federated learning for edge-assisted internet of things. *Secur. Commun. Netw.* 2021.
- [24] Y. Liu, Kumar, N., Xiong, Z., Lim, W.Y.B., Kang, J., Niyato, D.: Communication efficient federated learning for anomaly detection in industrial internet of things. In: *GLOBECOM 2020–2020 IEEE Global Communications Conference*, pp. 1–6. IEEE (2020).

- [25] B. Cao, C. Li, Y. Song, Y. Qin, and C. Chen, ‘Network Intrusion Detection Model Based on CNN and GRU’, *Applied Sciences*, vol. 12, no. 9, p. 4184, Apr. 2022, doi: 10.3390/app12094184.
- [26] T.N. Hoang, M. R. Islam, K. Yim, and D. Kim, ‘CANPerFL: Improve In-Vehicle Intrusion Detection Performance by Sharing Knowledge’, *Applied Sciences*, vol. 13, no. 11, Art. no. 11, Jan. 2023, doi: 10.3390/app13116369.
- [27] N. Sultana, N. Chilamkurti, W. Peng, R. Alhadad, ”Survey on SDN based network intrusion detection system using machine learning approaches”, *Peer-To-Peer Netw. Appl.* 12 (2) (2019) 493–501.
- [28] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “D<sup>2</sup>IoT: A federated self-learning anomaly detection system for IoT,” in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 756–767.
- [29] Y. Fan, Y. Li, M. Zhan, et al., Iotdefender: A federated transfer learning intrusion detection framework for 5 g iot, in: *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, IEEE, 2020, pp. 88–95.
- [30] O. Aouedi, K. Piamrat, G. Muller, and K. Singh, ‘Intrusion detection for Softwarized Networks with Semi-supervised Federated Learning’, in *ICC 2022 - IEEE International Conference on Communications*, Seoul, Korea, Republic of: IEEE, May 2022, pp. 5244–5249. doi: 10.1109/ICC45855.2022.9839042.
- [31] J. Shi, B. Ge, Y. Liu, Y. Yan, and S. Li, ‘Data Privacy Security Guaranteed Network Intrusion Detection System Based on Federated Learning’, in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Vancouver, BC, Canada: IEEE, May 2021, pp. 1–6. doi: 10.1109/INFOCOMWKSHPS51825.2021.9484545.
- [32] Koch, Robert, “Towards next-generation intrusion detection.” In: 2011 3rd
- [33] V. V. Platonov et P. O. Semenov, « An adaptive model of a distributed intrusion detection system », *Autom. Control Comput. Sci.*, vol. 51, no 8, p. 894-898, déc. 2017, doi: 10.3103/S0146411617080168.
- [34] M. Tavallaee, E. Bagheri, W. Lu, et A. A. Ghorbani, « A detailed analysis of the KDD CUP 99 data set », in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, juill. 2009, p. 1-6, doi: 10.1109/CISDA.2009.5356528.
- [35] N. Moustafa and J. Slay, ‘UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)’, in *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia: IEEE, Nov. 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942..
- [36] Thakkar, Ankit and Ritika Lohiya. “A Review of the Advancement in Intrusion Detection Datasets.” In: *Procedia Computer Science* 167, pp. 636–645 *International Conference on Cyber Conflict*. IEEE, pp. 1–18.

- [37] P. Ongsulee, ‘Artificial intelligence, machine learning and deep learning’, in 2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE), Nov. 2017, pp. 1–6. doi: 10.1109/ICTKE.2017.8259629.
- [38] Das, S., Dey, A., Pal, A., Roy, N.: Applications of artificial intelligence in machine learning: review and prospect. *Int. J. Comput. Appl.* 115(9) (2015).
- [39] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, ‘Machine learning: a review of classification and combining techniques’, *Artif Intell Rev*, vol. 26, no. 3, pp. 159–190, Nov. 2006, doi: 10.1007/s10462-007-9052-3.
- [40] Simon, A., Singh, M.: An overview of Machine learning and its Ap. *Int. J. Electr. Sci. Electr. Sci. Eng. (IJESE)* 22 (2015).
- [41] Dey, A.: Machine learning algorithms: a review. *Int. J. Comput. Sci. Inf. Technol.* 7(3), 1174–1179 (2016).
- [42] M. Ahmad, S. Aftab, S. S. Muhammad, and U. Waheed, “Tools and Techniques for Lexicon Driven Sentiment Analysis: A Review,” *Int. J. Multidiscip. Sci. Eng.*, vol. 8, no. 1, pp. 17–23, 2017.
- [43] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* 521(7553), 436 (2015).
- [44] Deng, L., Yu, D.: Deep learning: methods and applications. *Found. Trends® Signal Process.* 7(3–4):197–387 (2014).
- [45] Deng, L.: Three classes of deep learning architectures and their applications: a tutorial survey. *APSIPA Trans. Signal Inf. Process.* (2012).
- [46] Schmidhuber, J.: Deep learning in neural networks: an overview. *Neural Netw.* 61, 85–117 (2015).
- [47] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.
- [48] M. Sewak, S. K. Sahay, and H. Rathore, ‘An Overview of Deep Learning Architecture of Deep Neural Networks and Autoencoders’, *j comput theor nanosci*, vol. 17, no. 1, pp. 182–188, Jan. 2020, doi: 10.1166/jctn.2020.8648.
- [49] D. Mishra, B. Naik, R. M. Sahoo, and J. Nayak, ‘Deep Recurrent Neural Network (Deep-RNN) for Classification of Nonlinear Data’, in *Computational Intelligence in Pattern Recognition*, A. K. Das, J. Nayak, B. Naik, S. Dutta, and D. Pelusi, Eds., Singapore: Springer, 2020, pp. 207–215. doi: 10.1007/978-981-15-2449-3\_17.
- [50] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a Convolutional Neural Network.” In 2017 International Conference on Engineering and Technology (ICET), 1–6. Ieee, 2017.
- [51] Sharma, Siddharth, Simone Sharma, and Anidhya Athaiya. “ACTIVATION FUNCTIONS IN NEURAL NETWORKS.” *International Journal of Engineering Applied Sciences and Technology* 04, no. 12 (May 10, 2020): 310–16.

- [52] Lee, Chen-Yu, Patrick W. Gallagher, and Zhuowen Tu. "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree." *Artificial intelligence and statistics*. PMLR, pp 464–472 ,2016.
- [53] Srivastava, Nitish et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *The journal of machine learning research* 15.1, pp. 1929– 1958.(2014).
- [54] Khan, A., Sohail, A., Zahoor, U. et al."A survey of the recent architectures of deep convolutional neural networks." *Artificial intelligence review* 53 (2020): 5455-5516.
- [55] Rawat W, Wang Z (2016) Deep convolutional neural networks for image classification: a comprehensive review. *Neural Comput* 61:1120–1132.
- [56] Khan, Asifullah, Anabia Sohail, Umme Zahoor, and Aqsa Saeed Qureshi. "A Survey of the Recent Architectures of Deep Convolutional Neural Networks." *Artificial Intelligence Review* 53, no. 8 (December 2020): 5455–5516.
- [57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi:10.1162/neco.1997.9.8.1735.
- [58] Nardi, M., Valerio, L., Passarella, A.: Centralised vs decentralised anomaly detection: when local and imbalanced data are beneficial. In: *Third International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pp. 7–20, PMLR (2021).
- [59] Abbasi, M., Shahraki, A., Taherkordi, A.: Deep learning for network traffic monitoring and analysis (NTMA): a survey. *Comput. Commun.* (2021)
- [60] Yang, Q., Liu, Y., Cheng, Y., Kang, Y., Chen, T., Yu, H.: Federated learning. *Synth. Lect. Artif. Intell. Mach. Learn.* 13(3), 1–207 (2019).
- [61] S. Bharati, M. R. H. Mondal, P. Podder, and V. B. S. Prasath, 'Federated learning: Applications, challenges and future directions', *HIS*, vol. 18, no. 1–2, pp. 19–35, May 2022, doi: 10.3233/HIS-220006.
- [62] Tianyang Li, Ankit Kumar Sahu, Manzil Zaheer, Mehrdad Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Mach. Learn. Syst.*, 2:429–450, 2020.
- [63] Ramin Taheri, Mohammad Shojafar, Mamoun Alazab, and Rahim Tafazolli. Fed iiot: A robust federated malware detection architecture in industrial iot. *IEEE Transactions on Industrial Informatics*, 17:8442–8452, 2020.
- [64] Mittal S. Gupta, A. and V. Chaudhari. Federated learning: A survey of security and privacy landscape. *arXiv preprint arXiv:2005.14165*, 2020.
- [65] Feki, I., Ammar, S., Kessentini, Y., and Muhammad, K. (2021). Federated learning for covid-19 screening from chest x-ray images. *Applied Soft Computing*, 106:107330.
- [66] P. M. Mammen, 'Federated Learning: Opportunities and Challenges'. *arXiv*, Jan. 13, 2021.: [arxiv.org/abs/2101.05428](https://arxiv.org/abs/2101.05428).

- [67] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.
- [68] Konecny, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016). Federated ‘optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.
- [69] Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konecny, J., Kumar, ‘ S., and McMahan, H. B. (2020). Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.
- [70] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. (2018). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*
- [71] Neel Guha, Ameet Talwalkar, and Virginia Smith. 2019. One-shot federated learning. *arXiv preprint arXiv:1902.11175* (2019).
- [72] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. 2020. Distilled One-Shot Federated Learning. *arXiv preprint arXiv:2009.07999* (2020).
- [73] Jiawen Kang, Zehui Xiong, Dusit Niyato, Han Yu, Ying-Chang Liang, and Dong In Kim. 2019. Incentive design for efficient federated learning in mobile networks: A contract theory approach. In *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. IEEE, 1–5.
- [74] Latif U Khan, Shashi Raj Pandey, Nguyen H Tran, Walid Saad, Zhu Han, Minh NH Nguyen, and Choong Seon Hong. 2020. Federated learning for edge networks: Resource optimization and incentive mechanism. *IEEE Communications Magazine* 58, 10 (2020), 88–93.
- [75] Nicolas Kourtellis, Kleomenis Katevas, and Diego Perino. 2020. FLaaS: Federated Learning as a Service. *arXiv preprint arXiv:2011.09359* (2020).
- [76] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).
- [77] Michael R Sprague, Amir Jalalirad, Marco Scavuzzo, Catalin Capota, Moritz Neun, Lyman Do, and Michael Kopp. 2018. Asynchronous federated learning for geospatial applications. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 21–28.
- [78] arten van Dijk, Nhuong V Nguyen, Toan N Nguyen, Lam M Nguyen, Quoc TranDinh, and Phuong Ha Nguyen. 2020. Asynchronous Federated Learning with Reduced Number of Rounds and with Differential Privacy from Less Aggregated Gaussian Noise. *arXiv preprint arXiv:2007.09208* (2020)
- [79] Xianglin Bao, Cheng Su, Yan Xiong, Wenchao Huang, and Yifei Hu. 2019. Flchain: A blockchain for auditable federated learning with trust and incentive. In *2019 5th*

- International Conference on Big Data Computing and Communications (BIGCOM). IEEE, 151–159.
- [80] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, ‘Federated Learning in Smart City Sensing: Challenges and Opportunities’, *Sensors*, vol. 20, no. 21, Art. no. 21, Jan. 2020, doi: 10.3390/s20216230.
- [81] ‘Welcome to Python.org’, Python.org. Accessed: May 12, 2024. [Online]. Available: <https://www.python.org>
- [82] ‘TensorFlow’. Accessed: May 13, 2024. [Online]. Available: <https://www.tensorflow.org>
- [83] ‘pandas - Python Data Analysis Library’. Accessed: May 13, 2024. [Online]. Available: <https://pandas.pydata.org/>
- [84] ‘scikit-learn: machine learning in Python — scikit-learn 1.4.2 documentation’. Accessed: May 13, 2024. [Online]. Available: <https://scikit-learn.org>
- [85] ‘NumPy -’. Accessed: May 13, 2024. [Online]. Available: <https://numpy.org>
- [86] ‘Keras: Deep Learning for humans’. Accessed: May 13, 2024. [Online]. Available: <https://keras.io>
- [87] ‘Matplotlib — Visualization with Python’. Accessed: May 13, 2024. [Online]. Available: <https://matplotlib.org>
- [88] ‘Flower: A Friendly Federated Learning Framework’. Accessed: May 13, 2024. [Online]. Available: <https://flower.ai>
- [89] Hsu, T.M.H., Qi, H., Brown, M.: Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335 (2019).
- [90] Hsu, T.M.H., Qi, H., Brown, M.: Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335 (2019).
- [91] Y. Liu, G. Wu, W. Zhang, and J. Li, ‘Federated Learning-Based Intrusion Detection on Non-IID Data’, in *Algorithms and Architectures for Parallel Processing*, W. Meng, R. Lu, G. Min, and J. Vaidya, Eds., Cham: Springer Nature Switzerland, 2023, pp. 313–329. doi: 10.1007/978-3-031-22677-9\_17.
- [92] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, ‘The ‘K’ in K-fold Cross Validation.’, in *ESANN*, 2012, pp. 441–446, ISBN: 978-2-87419-049-0.



# Appendices

# Appendix A

## Acceptance certificates in this work

Our research, titled "An Intrusion Detection System Based on Federated Deep Learning," has been accepted for presentation at the BİLSEL International Scientific Research Conference in Turkey on May 5, 2024. It has also been published in a book with the ISBN: 978-625-6501-77-5, which is announced on the website <https://bilselkongreleri.com/kongre-kitaplari>.

We also present the certificates of participation in this conference:



# CERTIFICATE

of Participation

This is to certify that

**Tedjini Abdeldjalil**


In oral and technical presentation, recognition and appreciation of research contributions to

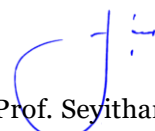
**“2. BİLSEL INTERNATIONAL TURABDİN SCIENTIFIC RESEARCHES AND INNOVATION CONGRESS”**

**04-05 MAY, 2024 - MARDİN/TÜRKİYE**

with the paper entitled

**“AN INTRUSION DETECTION SYSTEM BASED ON FEDERATED DEEP LEARNING”**

  
Dr. Bahar ALTUNOK  
General Coordinator

  
Assoc. Prof. Seyithan CAN  
Head of Organization Committee



# CERTIFICATE

of Participation

This is to certify that

**Chenine Mustapha**

In oral and technical presentation, recognition and appreciation of research contributions to

**“2. BİLSEL INTERNATIONAL TURABDİN SCIENTIFIC RESEARCHES AND INNOVATION CONGRESS”**

**04-05 MAY, 2024 - MARDİN/TÜRKİYE**

with the paper entitled

**“AN INTRUSION DETECTION SYSTEM BASED ON FEDERATED DEEP LEARNING”**

  
Dr. Bahar ALTUNOK  
General Coordinator

  
Assoc. Prof. Seyithan CAN  
Head of Organization Committee

# Appendix B

## Dataset

### B.1 Definition the features of a dataset

A detailed description of the features of the UNSW-NB15 dataset.

No	Name	Type	Description
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
6	state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number

23	smeansz	integer	Mean of the flow packet size transmitted by the src
24	dmeansz	integer	Mean of the flow packet size transmitted by the dst
25	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	integer	Actual uncompressed content size of the data transferred from the server s http service.
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of synack and ackdat .
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
35	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
36	is_sm_ips_ports	Binary	If source (1) and destination (3)IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	integer	No of flows that has a command in ftp session.
41	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).

46	ct_dst_sport_ltm	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).
48	attack_cat	nominal	The name of each attack category. In this data set , nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
49	Label	binary	0 for normal and 1 for attack records