REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET DE l'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : [réseaux et télécommunication]

Par :

Boumediene hadjer
Boumediene chaimaa

Sur le thème

Proposal of a new approach for efficient intrusion detection systems in SDN networks

---

Soutenu publiquement le 12/ 06 / 2024 à Tiaret devant le jury composé de :

Mr DJAAFARI Laouni	MCA	Université de Tiaret	Président
Mr DAOUD Mohamed_Amine	MCB	Université de Tiaret	Encadrant
Mr MOSTEFAOUI Kadda	MAA	Université de Tiaret	Examinateur

2023-2024

Avant propos

Louange à Allah, on le glorifie, on lui demande de l'aide et on lui demande pardon contre le mal de nos péchés, celui qui fut guider personne ne peut l'égarer et celui qui est égaré personne ne peut le guider.

On tient à remercier très vivement notre encadreur le maître de conférence Monsieur DAOUD Mohamed Amine pour avoir accepté de diriger cette thèse avec une souplesse non imposante qui nous a poussé à fournir un grand effort. Sans oublier tous ses encouragements et ses précieux conseils, qu'Allah le récompense.

On est très honoré que Messieurs : DJAAFARI Laouni le maître de conférence et MOSTEFAOUI Kadda le maître Assistant, aient accepté la charge de juger ce mémoire. on les remercie chaleureusement pour leur dévouement.

Mes remerciements vont également : aux enseignants de la faculté des mathématiques et de l'informatique et précisément le département d'informatique

Un grand merci spécial à notre père de nous avoir fait confiance et de nous avoir apporté son soutien inconditionnel et sa contribution pour la réalisation de ce mémoire et au grand cœur de notre mère.

Et toutes les personnes qui ont contribue de près ou de loin au bon déroulement de ce travail, qu'elles voient en ces mots l'expression de notre gratitude pour leur présence, pour leur dévouement et pour l' aide inestimable qu'elles nous ont portées tout au long de ce parcours. Un petit chemin certes, un grand enrichissement

On ne saurait oublier notre famille, pour son soutien indéfectible et pour avoir toujours cru en nous. Leurs encouragements ont été notre refuge et notre motivation durant tout le parcours académique.

Abstract:

In the fast-changing field of network security, Software-Defined Networking (SDN) stands out as a game-changer by providing centralized control and programmability of network resources, which leads to more flexible and efficient management. However, this centralization also brings new vulnerabilities, making SDN environments appealing targets for cyber-attacks. To address these risks, Intrusion Detection Systems (IDS) are essential, as they monitor and analyze network traffic to detect and respond to malicious activities.

Traditional Intrusion Detection Systems (IDS) that depend on centralized data collection and processing encounter major privacy and scalability issues. As network environments become more complex and data volumes increase, these systems often fail to meet modern requirements, resulting in potential single points of failure and heightened privacy risks. This project tackles these challenges by investigating the use of machine learning (ML) and deep learning (DL) models to enhance IDS within Software-Defined Networking (SDN) environments. ML and DL methods can greatly boost the accuracy and efficiency of intrusion detection by analyzing extensive datasets and detecting patterns that suggest malicious activities.

This project focuses on a comprehensive dataset analysis, applying diverse machine learning (ML) and deep learning (DL) models, and evaluating their performance in detecting intrusions within Software-Defined Networking (SDN) environments. The main objective is to improve the effectiveness of Intrusion Detection Systems (IDS) in SDN by utilizing advanced models, recognizing their strengths and weaknesses, and contributing to the creation of more robust, scalable, and privacy-preserving IDS solutions. The insights obtained will ultimately enhance the security of SDN environments.

Keywords: IDS, SND ,classification, Machine learning, Deep learning, Evaluation.

Abréviations

SND	Software-Defined Networking
IDS	Intrusion Detection Systems
IA/AI	Artificial Intelligence
ONF	Open Networking Foundation
APIs	Application Programming Interfaces
SPOF	Single Point Of Failure
DOS	Denial-of-Service
DDOS	Distributed Denial-of-Service
NIDS	Network-based IDS
HIDS	Host-Based IDS
ML	Machine Learning
DL	Deep Learning
(T,P,E)	Task **(T)**, Performance **(P)**, and Experience **(E)**
KNN	K-Nearest Neighbors
CNN	Convolutional Neural Networks
ANNs	Artificial neural networks
RNNs	Recurrent Neural Networks
MLPs	Multilayer Perceptrons
GANs	Generative Adversarial Networks
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
VAE	Variational Autoencoders
VDA	Variational Denoising Autoencoders
VSA	Variational Sparse Autoencoders
FN	False Negatives
TP	True Positives
FP	False Positives
TN	True Negatives
LFA	Link Fabrication Attack

TABLE OF CONTENTS

Figures

Tables

INTRODUCTION

Context

In the rapidly evolving landscape of network security, Software-Defined Networking (SDN) has emerged as a transformative technology. SDN offers centralized control and programmability of network resources, allowing for more flexible and efficient network management. However, this centralization also introduces new vulnerabilities, making SDN environments attractive targets for cyber-attacks. To mitigate these risks, Intrusion Detection Systems (IDS) play a crucial role in monitoring and analyzing network traffic to identify and respond to malicious activities.

Challenges in Traditional Intrusion Detection Systems

Traditional IDS approaches often rely on centralized data collection and processing, which can pose significant privacy and scalability challenges. As network environments grow in complexity and volume, these systems may struggle to keep up with the demands of modern networks. The centralization of data not only increases the risk of a single point of failure but also raises concerns about the privacy of the collected data. Consequently, there is a pressing need for more efficient, scalable, and privacy-preserving solutions.

Contribution

This project explores various machine learning (ML) and deep learning (DL) models to enhance IDS in SDN environments. ML and DL techniques offer the potential to significantly improve the accuracy and efficiency of intrusion detection by learning from large datasets and identifying patterns indicative of malicious activities. By leveraging these advanced models, it is possible to develop IDS that are more adaptable to the dynamic nature of modern networks. The project encompasses several key components:

- *Dataset Analysis*: An in-depth analysis of the dataset used for training and evaluating the models. This includes preprocessing steps, feature selection, and understanding the distribution of normal and malicious traffic.

- *Model Application*: The application of various ML and DL models to the dataset. This involves selecting appropriate algorithms, training the models, and tuning their parameters to achieve optimal performance.

- *Performance Evaluation*: Evaluating the performance of the models in detecting intrusions within SDN environments. This includes assessing metrics such as accuracy, precision, recall, and the trade-offs between different models.

Goals and Objectives

The primary goal of this project is to enhance the effectiveness of IDS in SDN by leveraging ML and DL models. By evaluating these models, we aim to identify their strengths and limitations in the context of network security. The insights gained from this project will contribute to the development of more robust, scalable, and privacy-preserving IDS solutions, ultimately improving the security of SDN environments.

I.STATE OF THE ART

I.1 INTRODUCTION

The rapid evolution is a prominent feature of the introduction information technology sciences, and it would not be an exaggeration to say that networks, like other branches of information technology, have witnessed lots of innovations in devices, applications, services, tools, and so on over the past decades. However, network infrastructure is unique from other branches of information technology because its foundations and ancient structure remain untouched and resistant to tampering. Which the time has come for a revolution against it with modern technology, SDN (Software-Defined Networking).

I.2. DEFINITION OF SDN

Many researchers have different definitions of SDN. Perhaps because SDN evolves as the technology matures and solutions are provided. In general, SDN mostly means that networks are controlled by software applications and SDN controllers rather than traditional network management consoles and commands that require a lot of administrative overhead and can be tedious to manage at scale.[1]

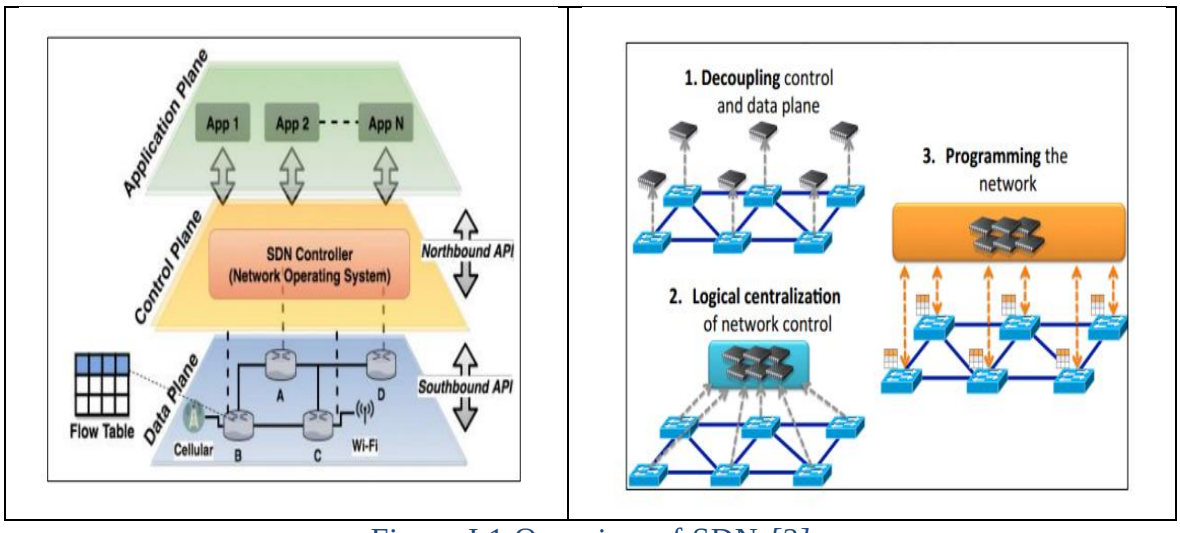


Figure I 1 Overview of SDN *[2]*

Software Defined Networks (SDNs) represent a paradigm shift in the way networks are designed, implemented, and managed. Traditionally, network functionality such as routing, switching, and traffic management has been tightly integrated into hardware

devices like routers and switches. However, with SDNs, this functionality is decoupled from the underlying hardware and abstracted into software, allowing for greater flexibility, scalability, and automation.

The SDN, as defined by the ONF (Open Networking Foundation, 2013), is an architectural approach that divides network management into two main components: the control plane and the data plane.
In a software-defined network, the control plane, which determines how data packets should be forwarded through the network, is separated from the data plane, which actually forwards the packets.

The fundamental concept of SDNs involves separating the data plane from the control plane. Unlike traditional network setups, where here SDNs centralize supervision through SDN controllers. This centralized approach enables configuration and programming from a host, adapting the network to various service requirements across distributed networks. Additionally, the centralized logic and management facilitate comprehensive network monitoring, offering enhanced visibility and control.
The communication between the SDN controller and network devices occurs through the OpenFlow protocol, allowing the controller to regulate network traffic based on predefined rules. Programmers have the flexibility to utilize traditional programming languages, frameworks, APIs, and libraries to develop applications within the SDN environment.[3]
SDN consolidates the control functions of various network devices into a single external software entity known as a "Controller." This Controller has a holistic view of the network and manages its operations through communication interfaces called APIs. By abstracting the physical layer, it allows applications to communicate in a developer language, thereby enabling network programming.
Overall, SDN networks exhibit five key characteristics:
- Segregation of control and data planes.
- Simplified devices.
- Centralized control.
- Network automation and virtualization.
- open-source.[4]

I.3. ARCHITECTURE AND OPERATION

SDN is a three-layer model that contains the following: one, an application layer; two, a control layer; and three, a data layer, communicating with each other via API interfaces.

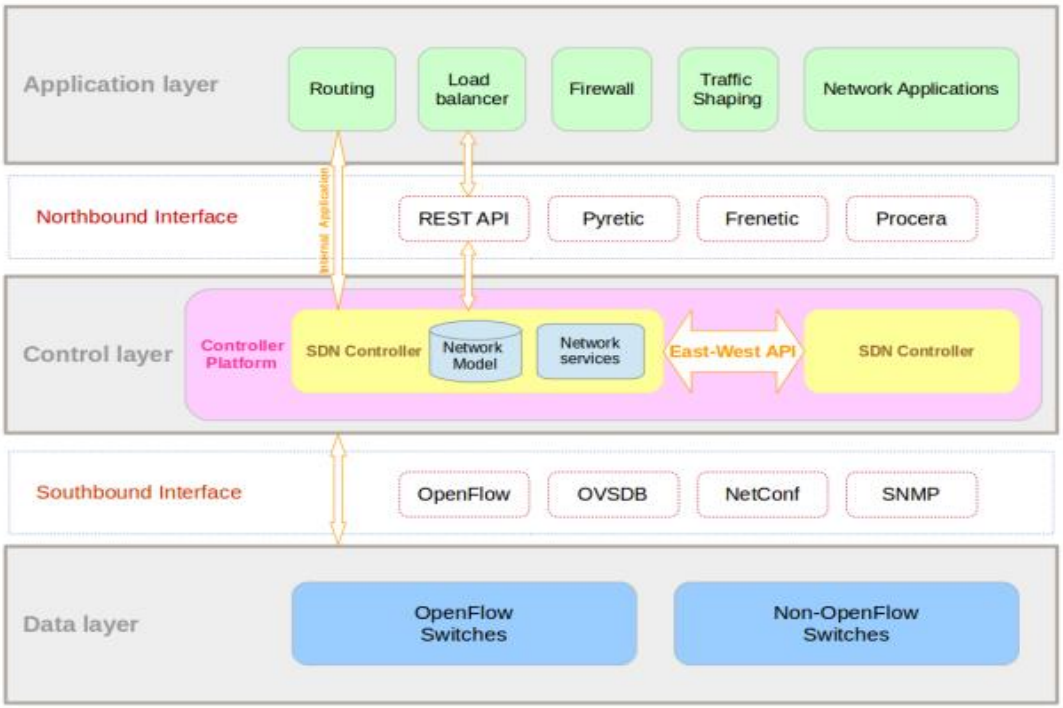


Figure I 2 SDN Architecture *[5]*

In this figure, from highest to lowest we have:

I.3.1 Application Layer (the management plane)

The application layer, also known as Tier-3 in SDN architecture, is the highest layer where applications and services interact with the underlying control and infrastructure layers. The application layer sits above the control layer and facilitates the development of network applications.

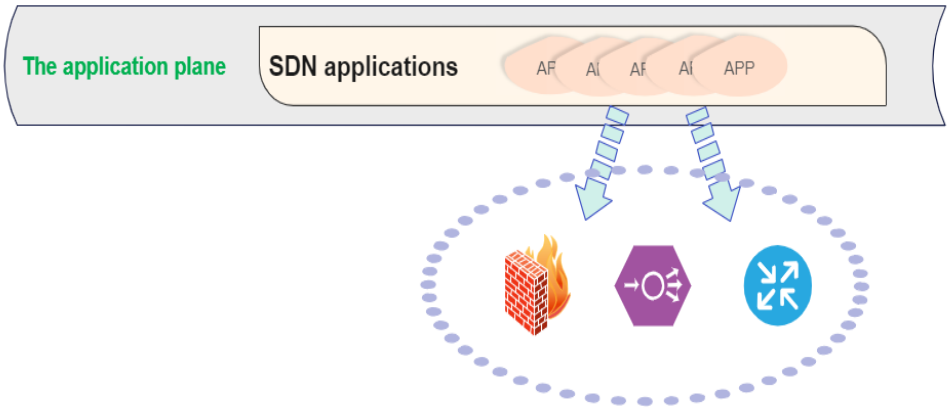


Figure I 3 Application layer

Applications at this layer are responsible for performing various network management tasks, leveraging the capabilities provided by the lower layers. For example, they can

utilize the programmable nature of SDN to dynamically allocate resources, optimize traffic flow, and enhance network security[6].
Some common examples of SDN applications include:

- **Load Balancers:** These distribute work evenly across multiple servers, preventing any one from getting overloaded.
- **Network Monitors:** These keep an eye on how your network is functioning.
- **Intrusion Detection Systems (IDS):** These act like security guards for your network, watching for any unauthorized access attempts, where IDS applications detect and respond to suspicious or malicious activities within the network, helping to enhance security by identifying and mitigating potential threats.

Overall, the application layer in SDN enables the development of custom network applications tailored to specific organizational needs, providing greater flexibility, scalability, and control over network operations.
The application layer is where the true power of SDN shines; The application layer in SDN serves as a platform for hosting applications that leverage the advantages offered by the architecture, enabling the development of new network functionalities.

Positioned above the controller layer, SDN applications are responsible for executing tasks, no matter how small, within the network. Unlike traditional networks where devices handle specific tasks, SDN applications take control of the entire network operation, from the smallest configurations to complex services[4]. A wide spectrum of applications has been developed to cater to diverse network and management functions such as:

- **Managing network traffic flow:** These applications optimize packet routing by determining the most efficient path between two points in the network.
- **Load balancing:** Applications in this category evenly distribute network traffic across the infrastructure and available resources to prevent bottlenecks, ensuring optimal performance and resource utilization.
- **Adapting to changes:** Responding to network events; applications monitor the network for any alterations, such as link failures or equipment additions, and promptly adjust network configurations to accommodate these changes.
- **Traffic redirection for security purposes:** Certain applications are designed to redirect network traffic for security-related reasons, such as mitigating cyber threats and isolating suspicious activity or enforcing access control policies.

I.3.2 Control layer:

The control plane is a crucial component of Software Defined Networking (SDN) networks. It is responsible for managing and controlling the data plane, which consists of the network's switches and routers.

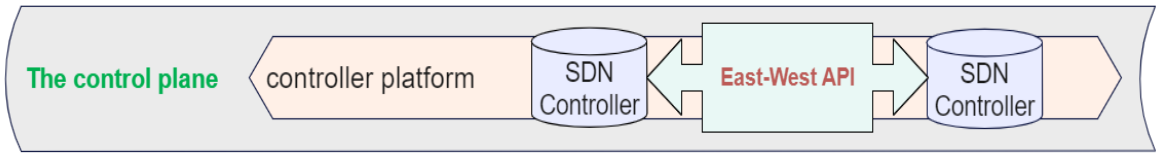


Figure I 4 Control layer

The control layer in an SDN network relies on the SDN controller, which is responsible for managing the data plane reactively or proactively by applying various forwarding policies. This layer is often regarded as the network's brain, handling most of the computational operations[4].

I.3.2.1 The SDN Controller

The SDN controller plays a crucial role in managing the network infrastructure equipment and connecting them with applications. It can consist of one or multiple controllers and is often likened to an operating system for the network.
The controller acts as the brain of the SDN network. It centralizes network intelligence and allows for centralized configuration and management[4]. The controller is responsible for:

- **Defining traffic policies:** The controller defines rules that tell switches how to handle packets.
- **Managing network resources:** The controller can monitor network resource usage and allocate them optimally.
- **Ensuring network security:** The controller can implement security policies to protect the network from attacks.

A controller's performance is commonly measured in terms of throughput, representing the amount of data processed per second, and latency, indicating the time taken to install a new forwarding rule. The SDN controller can operate in two modes[4]:

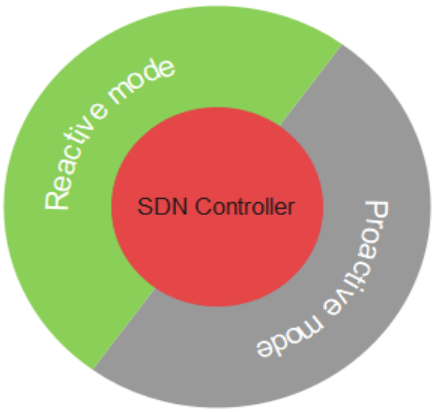


Figure I 5 Modes of operation in SDN Controller

- **Reactive mode/approach:** In this mode, the controller is consulted every time a switch receives a packet it does not know how to handle. So, the switch waits for the controller's response before processing the packet.
- **Proactive mode/approach:** Here, the controller preconfigures forwarding rules on the switches to enable them to process packets locally, without needing to consult the controller.

The control plane plays a critical role in SDN networks. It centralizes network intelligence and enables flexible and efficient network management. Choosing the right SDN controller is crucial for ensuring the network's smooth operation.

Numerous SDN controllers are available, this table summarizes some of them[7]:

Controller Name	Architecture	Northbound API (NBAPI)	Southbound API (SBAPI)	Programming Language	Commercial/Open Source
Beacon	Centralized, multi-threaded	Ad-hoc API	OpenFlow	Java	Open Source
DISCO	Distributed	REST	OpenFlow	Java	Open Source
Floodlight	Centralized, multi-threaded	RESTful API	OpenFlow	Java	Open Source
HP VAN SDN	Distributed	RESTful API	OpenFlow	Java	Commercial
Onix	Distributed	NVP NBI API	OpenFlow	Python, C	Commercial
Maestro	Centralized, multi-threaded	Ad-hoc API	OpenFlow	Java	Open Source
Meridian	Centralized, multi-threaded	Extensible API layer	OpenFlow	Java	Open Source

MobileFlow	-	SDMN API	-	-	Commercial
MuL	Centralized, multi-threaded	Multi-level interface	OpenFlow	C	Open Source
NOX	Centralized	Ad-hoc API	OpenFlow	C++	Open Source
NOX-MT	Centralized, multi-threaded	Ad-hoc API	OpenFlow	C++	Open Source
OpenDaylight	Distributed	REST, RESTCONF	OpenFlow, others (e.g., NETCONF, gRPC)	Java	Open Source
ONOS	Distributed	RESTful API	OpenFlow, others (e.g., ODL Network Programming Language)	Java	Open Source
PANE	Distributed	PANE API	OpenFlow	-	Commercial
POX	Centralized	Ad-hoc API	OpenFlow	Python	Open Source
SMaRTLight	Distributed	RESTful API	OpenFlow	Java	Open Source
SNAC	Centralized	Ad-hoc API	OpenFlow	C++	Open Source

Table I 1 SDNs controllers

The control plane, which is considered the most intelligent and vital layer of an SDN architecture, encompasses a multitude of controllers responsible for transmitting a diverse array of rules and policies to the data layer via the southbound interface.

I.3.3 SDN data plane (forwarding plane infrastructure Layer)

The infrastructure layer includes the devices that host the network data planes. These are the SDN switches responsible for directing network traffic.

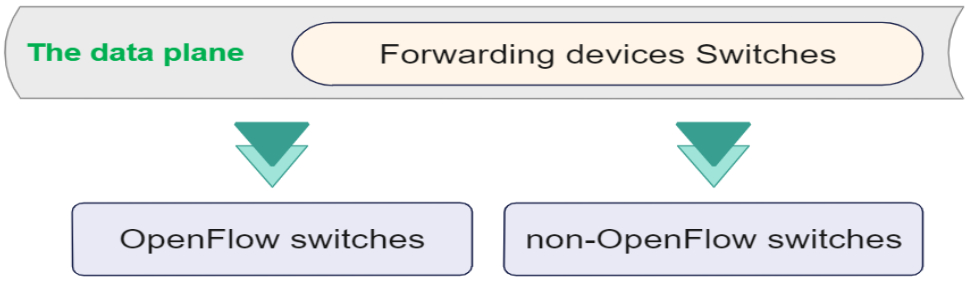


Figure I 6 Infrastructure layer

All SDN transmission equipment is called switches. In this context, the term switch is applied to any transmission equipment that compares packet headers to flow tables.

The use of the term switch is justified by the main function of the equipment, which is transmission.[4]
The SDN infrastructure layer consists of networking equipment such as switches, routers, and middlebox appliances. However, unlike traditional networks, the key distinction lies in the fact that these traditional physical devices now function solely as forwarding elements, these devices no longer have embedded control or software to make independent decisions. Instead, the network intelligence has been shifted from the data plane devices to a logically-centralized control system. [8]
Furthermore, one key aspect of these new networks is that they are built on open and standard interfaces, such as OpenFlow. This approach is crucial for ensuring compatibility and interoperability in terms of configuration and communication among different data and control plane devices. In other words, these open interfaces allow controller entities to dynamically program diverse forwarding devices, which was challenging in traditional networks due to the use of proprietary and closed interfaces, and the decentralized nature of the control plane.[8]
These devices handle the actual transportation of data packets across the network.[9]
But how do these devices know where to send the data? They rely on the control plane, which acts like a central traffic control center. The control plane creates rules and policies for data flow. To communicate these instructions to the data plane, southbound APIs come into play. The control plane sends the routing rules (like traffic light signals) through these APIs, and the data plane devices (like the cars) receive and follow them to forward the data packets efficiently.[9]
This separation of tasks between the data plane (moving data) and the control plane (deciding how to move it) makes networks function smoothly. The data plane can focus on speed and efficiency, while the control plane can strategize on optimizing traffic flow. It's like having dedicated teams for each role, ensuring a well-functioning network.[9]

I.3.4 SDN Communication interfaces

Communication interfaces, also called APIs, allow the controller to interact with the different layers of the SDN network. These interfaces are classified according to the North/South/East/West notation, which is determined by the position of the layer with which the controller communicates in the architectural hierarchy. When communicating with a lower layer (sublayer), the controller uses a southbound interface. Conversely, for the upper layers, a North interface is used. Communication between controllers takes place via East/West interfaces.

I.3.4.1(southbound) interfaces

The communication process between the controller and the switches/routers and other elements of the network infrastructure layer is facilitated by the (southbound) interfaces. The controller injects the different policies into the equipment and retrieves

the information allowing applications to build a global view of the network via this interface, in particular by using the OpenFlow protocol in the case of the Open SDN standard.
Besides the OpenFlow standard, various protocols functioning as a southbound interface have been designed and are used in specific implementations called SDN based on APIs[4].

These interfaces play a crucial role in transmitting instructions and policies from the SDN controller to the forwarding plane. By utilizing these interfaces, the controller is able to push configurations, rules, and policies to the network devices, thereby determining how data packets should be forwarded, processed, or managed within the network [8].
The most prominent and widely embraced protocol for southbound communication in SDN environments is OpenFlow. OpenFlow establishes a standardized protocol for communication between the SDN controller and network switches, enabling the controller to dynamically program the behavior of the switches based on network conditions, policies, and requirements.[9]

I.3.4.2 northbound interfaces

Northbound interfaces allow transmission elements to be programmed using the network abstraction offered by the control plane. These interfaces facilitate communication between the controller and the application layer, resembling APIs rather than programming and networking protocols. The lack of standard between the control layer and the application layer means that there are multiple levels of abstraction and various use cases, allowing for multiple North interfaces catering to all of these use cases.[4]
Specifically, northbound interfaces consist primarily of open-source Application Programming Interfaces (APIs). APIs define the methods and protocols through which software components can interact with each other.
The utilization of open-source APIs in SDN systems offers advantages such as increased flexibility, interoperability, and opportunities for innovation.[9]

I.3.4.3 East/West interface

East/West interface These are inter-controller interfaces, they are found in distributed architectures (multi-controllers). These interfaces facilitate communication between controllers to synchronize network states.
East-west interfaces in networking pertain to the communication channels connecting multiple controllers within a Software-Defined Networking (SDN) environment. In contrast to northbound interfaces linking the control layer to the management layer,

east/west interfaces enable controllers to communicate and coordinate with each other viz between controllers themselves.[9]
East/West interfaces are not yet standardized, meaning there isn't a universally accepted protocol or set of protocols for this purpose. Instead, they rely on various mechanisms for communication.[9]

I.4. ADVANTAGES AND DISADVANTAGES

I.4.1 Advantages of SDN

Software-Defined Networking (SDN) offers lots of advantages here are some of them

- **Network Management Simplicity**

SDN simplifies network management by allowing networks to be viewed and managed as a single entity. This abstraction of complex tasks into easy-to-use interfaces streamlines network management and troubleshooting, this translates to improved network reliability.

- **Reducing Operational Expense (Cost Savings)**

Software-Defined Networking (SDN) offers substantial cost-saving benefits by revolutionizing network infrastructure and administration

  - **Reduced Hardware Costs**

SDN simplifies switch hardware by separating control plane functions, leading to cheaper switch hardware that only requires a data plane.

  - **Centralized and Automated Administration**

SDN centralizes and automates network administration tasks, reducing the need for manual intervention and costly specialized hardware.

  - **Improved Server Utilization and Virtualization**

SDN optimizes server usage and virtualization by dynamically allocating network resources based on demand, minimizing wasted capacity, and reducing the need for additional hardware investments.

- **Enhancing network security: Enhanced Security.**

Although SDN is not a security solution in itself, it can significantly contribute to improving network security in several ways:

  - **Centralized Security Policies**

SDN allows centralized management and enforcement of security policies across the network. By consolidating security controls into a central console, administrators can ensure consistent application of security measures across the network infrastructure.

- **Real-time monitoring and analysis**

SDN controllers have visibility into network traffic and can analyze it in real-time. This allows for the detection of anomalies or suspicious activities within network traffic, allowing immediate response and mitigation of security threats. Because once detected, the controller can quickly deploy security policies to mitigate these threats. For example, it can redirect or drop packets associated with malicious activity, thus preventing potential security breaches.

- **Dynamic Threat Adaptation**

The programmable nature of SDN allows for dynamic adaptation to emerging security threats. Administrators can modify security policies and configurations in response to changing threat landscapes, ensuring the network remains resilient in the face of evolving cyber threats.

- **Isolation and segmentation**

SDN technology facilitates network segmentation and isolation, which can help contain security breaches and limit the impact of cyberattacks. By dividing the network into separate virtual networks or segments, SDNs reduce the attack surface and reduce lateral movement of threats within the network.

- **Integration with Security Services**

SDN software can be integrated with many security services and solutions, such as firewalls, intrusion detection systems (IDS), and security analytics platforms. This integration enhances the overall security posture of the network by leveraging specialized security functions combined with SDN management capabilities.

- **Simplified management and visualization**

SDN's centralized management and visualization capabilities make it easier for administrators to monitor and manage network security. By providing a unified view of the network and its security state, SDN simplifies security management and facilitates more effective decision-making.

➢ **Directly Programmable**

In SDN, Network devices can be directly programmed through the controller, providing fine-grained control over network behavior.

➢ **Centralized Control and Management**

SDN decouples the control plane from the data plane, allowing for a centralized view of the network and optimal forwarding and routing decisions. This centralized control simplifies network management and enables administrators to configure and manage the entire network from a single point, providing a higher level of control and management over network resources.

➢ **Programmability**

SDN facilitates programmable network management, where network administrators can write programs to control how the network operates, empowering administrators to

adjust network behavior swiftly via software applications. This capability enables seamless deployment of new services and applications without the hassle of reconfiguring individual network devices.

- **Flexibility**

SDN offers a highly adaptable network infrastructure, where the network can be easily reconfigured through software to meet changing needs. This allows for dynamic adjustments based on specific requirements or traffic conditions, or with other words allowing for dynamic configurations tailored to specific needs or changing circumstances.

- **Network Virtualization**

SDN facilitates the creation of multiple virtual networks atop physical infrastructure, fostering isolation between network slices (virtual networks are isolated from each other), ensuring secure and separate operation for different purposes. and accommodating guest network operating systems and diverse vendors (Vendor Neutrality) where SDN promotes using equipment from various vendors within the same physical network through virtualization

- **High Performance and Granular Traffic Control**

SDN, particularly through the OpenFlow standard interface, provides high performance and granular traffic control across multiple networking devices.

- **Scalability**

SDN simplifies network scaling to accommodate evolving demands. Through software reconfiguration, networks can easily adjust to fluctuating traffic patterns or integrate more resources, ensuring seamless scalability in response to changing requirements.

- **Innovation and Experimentation**

SDN opens up the possibility of experimentation and innovation by allowing researchers and engineers to build custom protocols in software, which can be easily deployed in the network. This flexibility encourages the development of new networking solutions and approaches.

- **Interoperability**

SDN promotes open interfaces between network devices, allowing for interoperability between different vendors' equipment. This interoperability simplifies network integration and management in heterogeneous environments.

- **Fast Service Deployment**

With SDN, new features and applications can be deployed rapidly, within hours instead of days, enhancing agility and responsiveness to business needs.

I.4.2. Disadvantage of SDN

there are several disadvantages and challenges associated with Software-Defined Networking (SDN). Here are some key ones:

- **Increased Complexity**

SDN's strength, its programmability, and control, creates a double-edged sword; The complexity of Software-Defined Networking (SDN) arises from several factors, including the lack of standardized security protocols, the introduction of a centralized controller and southbound APIs, and the requirement for specialized expertise in network management. While SDN aims to simplify network operations, its adoption may present challenges during the transition from traditional networking architectures, necessitating additional training and skills development, compared to traditional networks. Managing an SDN environment effectively demands a deep understanding of both networking principles and software programming, highlighting the need for specialized expertise to address security concerns and troubleshoot complexities inherent in SDN deployment.
This complexity can strain organizations with limited IT resources, making secure SDN implementation a challenge.

- **Centralized Control = Security Risks**

While SDN can enhance network security, it can also introduce new security risks.
SDN's reliance on a central controller for network management and control creates a critical vulnerability. Because, a single point of control could be an attractive target for attackers, if compromised, this controller can become a single point of failure, bringing down the entire network. Additionally, attackers may target vulnerabilities in the controller software or communication protocols to gain unauthorized access or disrupt network operations. The programmability of the network could make it easier for attackers to manipulate traffic. This emphasizes the importance of robust security measures for SDN controllers and the APIs they use.

- **Hurdles and Challenges in Software-Defined Networking Adoption**

The immaturity and rapid evolution of Software-Defined Networking (SDN) present challenges such as compatibility issues between different solutions and a shortage of skilled professionals. Being a relatively new technology, SDN standards and protocols are still evolving, complicating the integration and interoperability of diverse SDN solutions. Additionally, migrating existing network infrastructure to SDN entails complexity and time-consuming processes due to legacy systems, proprietary protocols, and compatibility concerns. Moreover, the revolutionary nature of SDN, coupled with its perceived risks, may provoke resistance from network engineers and IT staff accustomed to traditional networking technologies, further hindering adoption efforts. Overall, these factors underscore the need for careful planning and strategic

implementation to overcome challenges and realize the potential benefits of SDN adoption.

- **Reactive Control Model**

With The Reactive Control Model in Software-Defined Networking (SDN) OpenFlow switches consult an OpenFlow controller each time a forwarding decision needs to be made. This model can lead to degraded forwarding performance, as switches must wait for instructions from the controller, potentially causing delays, especially if the controller is geographically distant or if most flows are short-lived. This delay can result in higher latency, impacting network performance, particularly in scenarios where low latency is crucial.

- **Interoperability Challenges**

Integrating SDN solutions from different vendors or with existing network infrastructure can be challenging due to interoperability issues. Ensuring seamless communication between various network elements and controllers may require additional effort and resources.

- **SDN's Costs: Beyond the Initial Investment**

Achieving reduced equipment costs through SDN migration may be elusive, especially in existing data center environments, it might promise reduced equipment costs in the long run, where legacy networking equipment may still need to be maintained alongside SDN components, delaying potential cost savings.

I.5. SECURITY IN SDNS

refers to the measures and protocols implemented to protect Software-Defined Networking environments from unauthorized access, data breaches, and cyber threats. It involves ensuring the confidentiality, integrity, and availability of network resources and data within the SDN architecture.[10]
Security in SDN is crucial due to the vulnerabilities present in the control plane, the application plane and data plane which can lead to various attacks

I.5.1. vulnerability

Vulnerability is a weakness point or flaw in a system, application, network, or process that could be exploited by attackers to compromise the security of the system or cause harm.

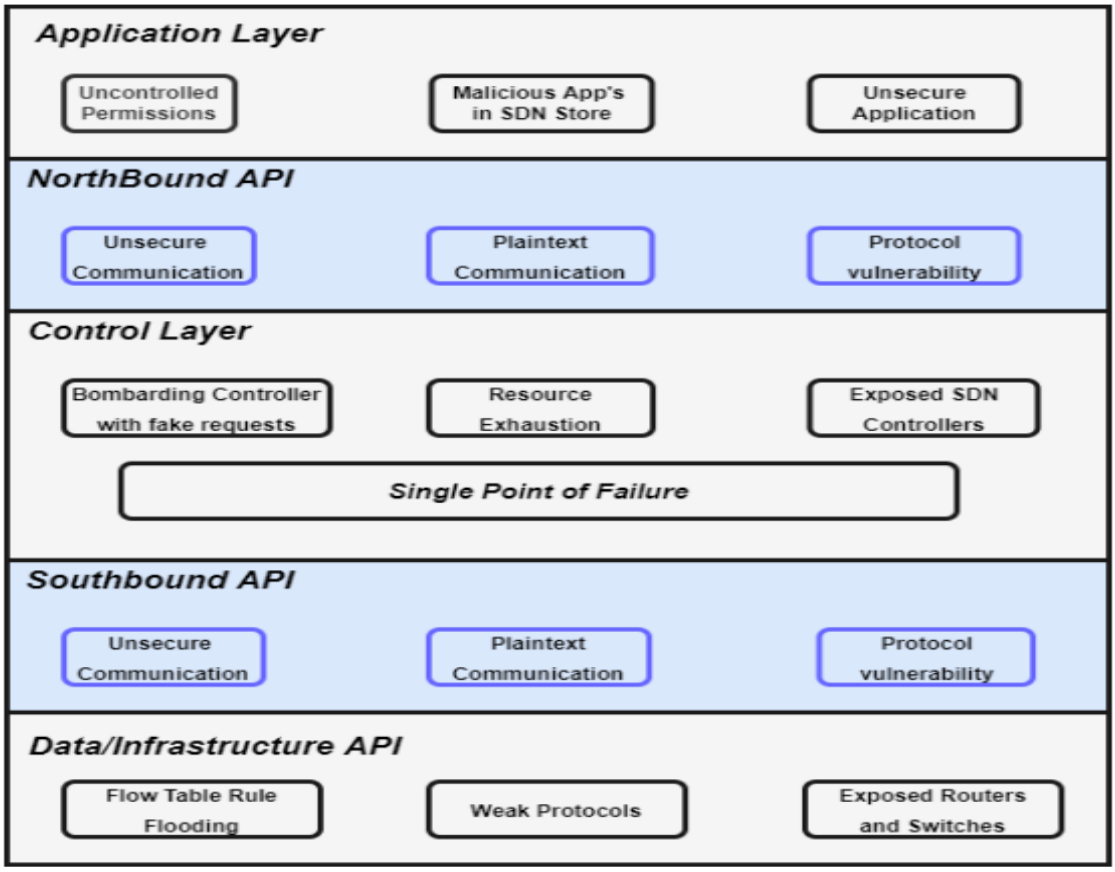


Figure I 7 Vulnerabilities present in different layers of SDN

I.5.1.1. Specific vulnerabilities and layer

I.5.1.1.1 Application Plane vulnerabilities

The application layer within SDN offers a range of network application services utilizing programming and management techniques. However, because of the programmable nature of this layer, vulnerabilities arise in the following aspects:

- **Accountability and access control vulnerabilities**

It is challenging to enforce policies for accountability and access control on third-party and nested applications that utilize network resources [11].

- **Authentication vulnerabilities**

There are no strong authentication and authorization mechanisms for applications, especially concerning a large number of third-party applications [11].

- **Fake insertion of traffic/flow rules**

Compromised applications can introduce false flow rules, posing difficulties in verifying if a specific application has been compromised [11].

- **Lack of Standardization and Open Specifications**

One of the challenges in SDN is the lack of standard or open specifications that would allow applications to easily control and manage the services and functions of the network through the control layer.
Without standard or open specifications, it becomes difficult for applications to interact with the control layer and have control over the network services and functions.

- **Third-party applications**

Third-party applications refer to software or programs that are developed by external entities or individuals, rather than the original creators or owners of the network or system.
These applications are often used in software-defined networking (SDN) environments to enhance functionality or provide additional services. [12]
The attacker can target the application layer to carry out various malicious activities like: Network Configuration Changes, Information Access, Resource Lockout, and Malicious Code Injection [12].

- **Illegal access**

Occurs as a result of vulnerabilities or weaknesses in the software of the controller [12].

- **Intrusion attack**

In an intrusion attack, the attacker manages to penetrate the system's defenses and gain entry into the network or a specific computer system. This access is typically obtained through exploiting vulnerabilities in the system's security measures, such as weak passwords, unpatched software vulnerabilities [12].

- **Anomaly attack**

An anomaly attack is a type of cyber-attack where an attacker gains access to unauthorized users and injects a malicious application into a system or network. This attack is difficult to trace and detect, allowing the attacker to carry out harmful activities without being noticed [12].
The lack of traceability and detection makes anomaly attacks particularly dangerous, as they can cause significant damage before being discovered and mitigated [12].

- **Altering SDN database**

By virtue of specific privileges granted to applications operating atop SDN, they gain access to the underlying storage infrastructure. This access enables them to manipulate and potentially exploit the SDN database, thereby exerting control over network behavior [12].

- **App Manipulation**

Refers to a type of attack in Software Defined Networking (SDN) where an attacker gains unauthorized access to an SDN application and manipulates it for their own malicious purposes [13].

- **Malicious application**

This refers to the use of worms, spyware, and other malicious software at the application layer to steal network information, change network configurations, and occupy network resources. These actions can disrupt the normal functioning of the control plane, causing confusion in network control by the controller.

I.5.1.1.2 Control Plane Vulnerabilities

Centralized control plane is a crucial component of network services in Software Defined Network (SDN). It is responsible for managing and controlling the network, and it directly affects the availability, reliability, and data security of the network services.

This centralized control plane simplifies network management and allows for more efficient control over the network but vulnerabilities in the control layer can lead to network confusion [14].

- **Weak authentication and incomplete encryption**

Weak authentication refers to inadequate methods for verifying user identities, making systems vulnerable to unauthorized access.

- **Centralization and Single Point of Failure (SPOF)**

Due to its centralized nature, the SDN controller becomes a single point of failure. If the controller is compromised, it can lead to a network outage or disruption. This centralized nature makes it a prime target for attackers aiming to disrupt network operations [12].

- **Scalability Limitations**

Another challenge in the control layer is the increasing number of switches connected to the controller. As the number of switches increases, the time taken by the controller to respond to requests from each switch also increases. This increased load on the controller can potentially causing delays in network operations or even controller failure under high loads [12].

- **Policy Enforcement and Configuration Conflicts**

With multiple controllers in a network domain, conflicting configurations can arise. These conflicts can lead to unpredictable network behavior and potential security vulnerabilities and security breaches.

Additionally, there may be semantic gaps between the controller and switches, complicating access control and policy distribution [12].

- **Denial-of-Service (DoS) Attacks**

The controller is susceptible to DoS attacks that flood it with requests, making it unavailable for legitimate users.
A surge in data flow through the network can overwhelm the controller's processing capabilities [13].

- **Enormous Number of Flows**

the flow table, which is responsible for storing and managing the flow rules in an SDN controller, has a limited capacity, as a result, when the flow table receives an overwhelming number of packets, it becomes full or, this overflow of the flow table can have a significant impact on the performance of the SDN controller [15].

- **DDoS Attack**

The attack floods the controller with a large volume of bogus requests or traffic, which makes it difficult for the controller to respond to legitimate requests from network devices and users, consuming its processing power (CPU) and memory.
And there are some types of DDoS Attacks : (SYN Flood); (HTTP Flood); (Ping of Death); (Low-rate DDoS)

- **Replay attacks**

Replay attacks involve the interception and retransmission of valid data packets, with the aim of impersonating a legitimate user or gaining unauthorized access to a system.

- **Network Manipulation**

Network manipulation is a type of attack in which an intruder gains control over the controller in a software-defined network (SDN) and that poses a significant threat to the security of a network [16].

- **Host Hijacking Attack**

The HTS is a service provided by the SDN controller that allows for network-wide visibility and is an essential component of the SDN architecture.
The issue with the HTS is that it can be compromised through various attacks, such as host impersonation, man-in-the-middle, or denial-of-service attacks.

- **Port amnesia**

Port amnesia attacks in SDN networks involve bypassing port labels to impersonate hosts and disrupt network flow rates. [11]

- **Port Probing**

Port probing is a significant attack in SDN where the attacker bypasses security mechanisms and confuses the host location. [11]

- **Persona Hijacking**

Persona hijacking exploits vulnerabilities in SDN's identifier binding mechanisms, allowing attackers to pose as network owners and gain unauthorized access to network resources [11].

- **Reverse Loop**

Reverse loop attacks target the SDN controller's view of the network by exploiting vulnerabilities in the topology discovery mechanisms. These attacks involve identifying and removing existing inter-switch links within a specified time interval by reversing the links [11].

- **Topology freezing**

Topology freezing is an attack that impacts the controller's visualization of the network by exploiting weaknesses in the topology services modules .This attack freezes the controller's view of the network, preventing it from updating dynamic changes [11].

- **Traffic sniffing**

involves capturing and inspecting data packets as they traverse a network [15].

- **Link Fabrication Attack**

An attacker introduces a new malicious link in the network to control traffic, known as a link fabrication attack (LFA) [11].

- **flow table rules conflict**

Attackers can target the flow table rules by using malicious applications that can create or add new rules.[13].

- **Side-channel attack**

A side-channel attack occurs when an attacker gains access to sensitive information, such as energy consumption patterns or timing variations, which are inadvertently leaked by a system or device [12].

- **Intrusion attack**

Intrusion attacks involve unauthorized access to a system without proper permissions, thereby compromising network security and stability [12].

- **Anomaly attack**

Anomaly attacks involve unauthorized users infiltrating the system and introducing malicious applications, making it challenging to trace and detect [12].

- **Scanning attack**

During scanning attacks, perpetrators scan the entire network to gather information and intercept network data through the controller [12].

- **Spoofing attack**

Spoofing attacks occur when attackers insert false rules into switch flow tables, granting them full control over the network [12].
For example: IP address spoofing. [17]

I.5.1.1.3 Data Plane Vulnerabilities

the data forwarding layer in a network infrastructure is crucial for packet forwarding, but switches with in this layer are vulnerable to security attacks

- **False Flow Rules**

One security challenge is the need to differentiate between true flow rules and false rules, because Malicious actors can inject fake flow rules into the tables.

- **Configuration Complexity and TLS Usage**

The configuration complexity coupled with the optional use of Transport Layer Security (TLS), introduces vulnerabilities to different types of attacks [12].

- **Limited Switch's Buffering Capacity**

the reactive caching mechanism makes switches vulnerable to DoS attacks.[13]

- **Side Channel Attack**

Side channel attacks can be used to gather information about the network's structure, configuration, or even the presence of specific security measures, which can aid the attacker in planning further attacks or exploiting vulnerabilities. [12]

- **DoS Attack**

This attack targeting the flow table and flow buffer in SDN switches can disrupt the normal functioning of the network by overwhelming these components with a large volume of packets. [12]

- **Spoofing attacks**

The attacker uses the identity of a legal user to send fake packets and malicious applications into the network [12].

- **Scanning attacks**

A scanning attack allows the attacker to obtain essential information about the entire network, including details such as host features, topology, and communication specifics between switches and hosts [12].

- **Hijacking attacks**

During a hijacking attack, the attacker seizes control of communication elements, potentially accessing communication data and flow rules when a switch is compromised.[12]

- **Traffic sniffing**

Is a method utilized by hackers or sniffers to gain unauthorized access to critical data within a network.by intercepting data packets, they can analyze the information for their own purposes, often targeting key components. Eavesdropping can occur anywhere there is continuous traffic flow within the network [13].

- **Flow-table flooding attack**

is common in SDN networks, where a large number of packets are sent randomly, causing the flow table to overflow [13].

- **Password guessing or Brute Force**

Unauthorized individuals may gain access to SDN components or non-SDN elements through password guessing or brute force attacks. Brute force attacks involve software attempting to decode encrypted data, like passwords or encryption keys, through exhaustive trial and error methods [16].

When a system is connected to the Internet, it becomes vulnerable to a multitude of attacks. These attacks aim to compromise the confidentiality, integrity, or availability of resources, constituting what we term as an intrusion. With the pervasive use of computer networks, the Internet, and online transactions, the risk of such intrusions has escalated, necessitating robust protective measures.

Given this scenario, Intrusion Detection has emerged as a pivotal domain in both commercial and academic spheres.

I.6. INTRUSION DETECTION SYSTEMS (IDS)

I.6.1 Introduction

In today's rapidly evolving digital landscape, the proliferation of Internet-based threats has escalated, posing a significant risk to the security of sensitive information. To

counter these threats, Intrusion Detection has emerged as a critical process for safeguarding networks and systems.

Initially introduced by James Anderson J. P. in 1980 [18] IDS has gained paramount importance due to recent cyberattacks on IT infrastructure, highlighting the escalating risks to information security. With threats evolving in sophistication and frequency, the imperative for secure networks has never been greater. Information security administrators now prioritize IDS implementation to detect and respond to unauthorized access and malicious activities, mitigating risks and ensuring the integrity of organizational data and operations in the face of ever-present cyber threats.

This shift is evident in the increasing sophistication of attacks, coupled with a diminishing requirement for intruder expertise as shown in the figure below. Attacks range from simple viruses and worms to more insidious malwares, the spectrum of threats continues to expand, encompassing a range of tactics such as Denial of Service (DOS) and Ransomware Attacks and network-based attacks like flooding.

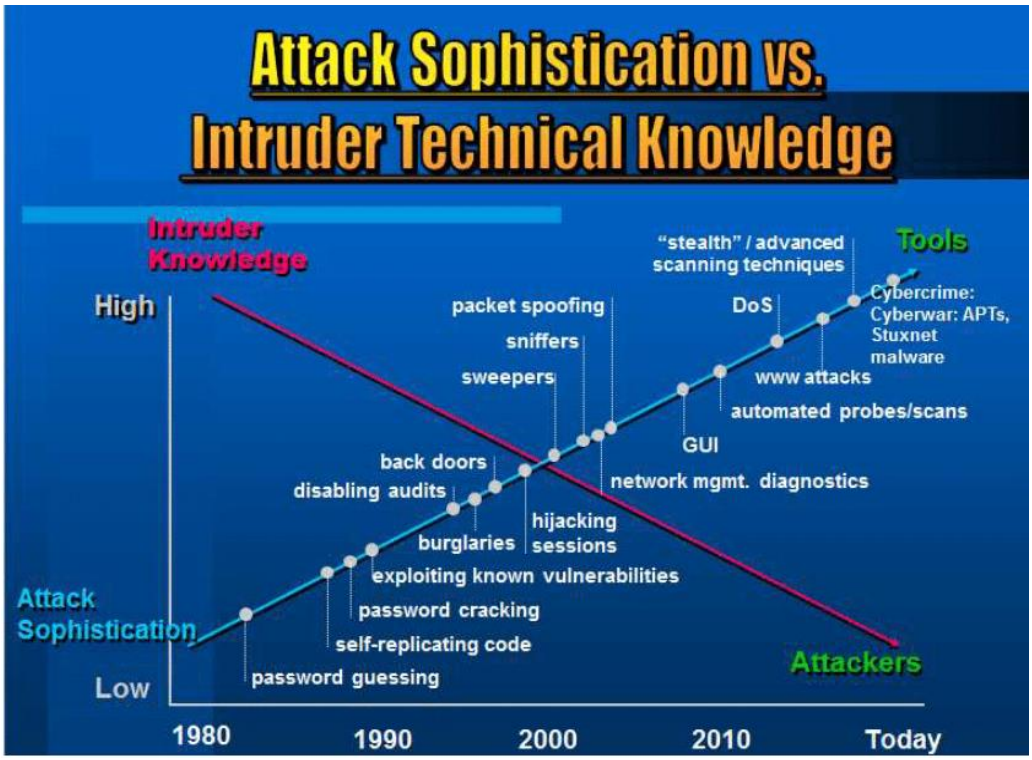


Figure I 8 Attack sophistication Vs Intruder Knowledge [19]

the escalating frequency and sophistication of cyber-attacks underscore the critical importance of effective Intrusion Detection Systems. As organizations strive to safeguard their digital assets, investing in robust IDS capabilities remains paramount in the ongoing battle against cyber threats.[20]

I.6.2 Need for intrusion detection systems

In the realm of organizational Information Security, Intrusion Detection Systems (IDS) play a crucial role. These systems, comprising both software and hardware components, are seamlessly integrated into an organization's Information/Data policies and practices. Their purpose is to fortify the organization's defenses against various threats. Researchers are increasingly focusing on enhancing the detection capabilities of IDS, particularly in terms of detection rates and response times. Techniques such as Data Mining and Machine Learning are being employed to bolster these capabilities.

Moreover, the reliability of a computer system or network hinges on its ability to uphold the principles of confidentiality, integrity, and availability. Incorporating IDS into the security framework of an organization not only strengthens its defenses but also ensures adherence to these fundamental security requirements.

In essence, IDS serve as the linchpin of an organization's Information Security strategy, providing comprehensive protection against evolving threats and safeguarding its valuable assets.

I.6.3 Definition of IDS

Intrusion detection involves monitoring events within a computer system or network and analyzing them for signs of unauthorized access or malicious activities. Intruders may include external attackers attempting to compromise the system, authorized users seeking unauthorized privileges, or authorized users misusing the. IDS automates the process of monitoring and analyzing these events, providing timely alerts to security personnel or administrators.

Think of an IDS as a security guard that constantly patrols your network, looking for anything out of the ordinary. It doesn't physically stop intruders, but it raises the alarm so you can take appropriate action.

I.6.4 The history of IDS

The history of Intrusion Detection Systems (IDS) traces back to the early 1980s following the evolution of the internet, with a focus on surveillance and monitoring to identify threats. The concept emerged from the need to detect unauthorized access and malicious activities within computer networks. James Anderson J. P. is often credited with introducing the concept of IDS in 1980 James Anderson's seminal paper for a government organization in the same decade underscored the importance of audit trails in tracking misuse and understanding user behavior, laying the foundation for modern IDS. As networks expanded and diversified, so did the capabilities of IDS, with the development of network-based IDS (NIDS) and host-based IDS (HIDS) to monitor

network traffic and individual host systems, respectively in 1983,[21] SRI International and Dorothy Denning initiated a government project that catalyzed the development of intrusion detection systems. Over time, IDS technologies have evolved to encompass both signature-based and anomaly-based detection methods, playing a crucial role in safeguarding network assets against evolving cyber threats.
The history of IDS reflects the continuous evolution of cyber threats. As attackers develop new tactics, IDS technology adapts to stay ahead of the curve.

I.6.5. Major types of IDSs

I.6.5.1 Host-Based IDS (HIDS):

Host-Based Intrusion Detection Systems (HIDS) focus on the host system itself, analyzing activities directly on the host to detect potential security threats. A key component in HIDS is the host-based sensor or handler, which collects data from various sources within the host environment for intrusion detection purposes.

I.6.5.2 Network-based IDS (NIDS)

Diverging from host-based counterparts, NIDS gather data directly from the network rather than individual hosts. They conduct real-time audits of network attacks as packets traverse the network, enhancing overall network security.

I.6.5.3 Hybrid IDS

Hybrid IDS ingeniously combines NIDS and HIDS, yielding more pertinent alerts. By blending elements from both network-based and host-based detection techniques, it forms a robust defense mechanism.
This amalgamated approach harnesses insights from network traffic and individual host activities, presenting a comprehensive picture of potential threats and intrusions. [22]

I.6.6 Detection methods

In the realm of intrusion detection, there are two main categories of techniques: signature-based detection, also known as misuse-based detection, and anomaly-based detection, also referred to as behavior-based detection. Each method has its own unique strengths and weaknesses.

I.6.6.1 Signature-based (or Misuse-based) detection.

Signature-based (or Misuse-based) detection is a method that relies on predefined patterns or signatures of known attacks to identify and detect intrusions. By comparing

network or system activity with these known signatures or indicators of misuse, it can detect and identify malicious behavior this technique of identifying intrusions by comparing network activity to known attack patterns, is particularly effective in detecting known threats, although it does require regular updates to its signature database.

I.6.6.1.1 Some advantages include

- **Effective Detection of Known Attack Patterns**

Misuse detection is highly effective at identifying known attack patterns and vulnerabilities in the system

- **Early Warning for Intrusion Attempts**

It can detect intrusion attempts based on partial signatures or indicators, providing early warnings of potential security breaches.

- **Familiar Operation**

The operation of misuse detection systems is similar to that of antivirus software, making them familiar and relatively easy to implement for organizations already using antivirus solutions.

Misuse detection systems tend to have lower false positive rates compared to anomaly-based detection systems. This means they are less likely to raise alarms for benign activities or deviations from normal behavior, reducing the burden on security personnel to investigate false alarms.

However, there are also limitations to signature-based detection:

I.6.6.1.2 Some disadvantages include

- **Limited to Known Threats**

Misuse detection relies on predefined signatures or patterns of known attacks. As a result, it may struggle to detect new or unknown threats that do not match any existing signatures. This limitation makes misuse detection less effective against zero-day attacks or sophisticated, previously unseen threats.

- **Dependency on Signature Updates**

Regular updates of signatures are necessary to keep misuse detection systems effective against evolving threats. Without frequent updates, the system may miss newly emerged attack patterns or vulnerabilities, leaving the network vulnerable to emerging threats.

- **High Rate of Missing Reports**

If an attack does not match any known signature, misuse detection systems may fail to detect it, leading to missed reports or false negatives. This can result in undetected security breaches and increased risk to the organization.

- **Inability to Detect Polymorphic Malware**

Signature-based detection may struggle to identify polymorphic malware, which constantly changes its code to evade detection by traditional signature-based antivirus software.

- **Inflexibility to Legitimate Deviations**

Misuse detection systems may generate false alarms for legitimate activities that deviate from normal behavior but are not malicious.

I.6.6.2 Anomaly based (or Behavior-based) Detection

Anomaly-based (or Behavior-based) Detection: This technique focuses on identifying deviations from normal behavior in network traffic. It establishes a baseline of normal activity and raises alarms when deviations are detected.
Anomaly detection systems have the advantage of being able to detect unknown intrusions, leading to a lower rate of missed reports.
However, the domain and nature of anomalies evolve over time, and intruders continuously adapt their network attacks to bypass existing intrusion detection solutions. [18]

Anomaly detection is effective for detecting various attacks, including misuse of protocol and service ports, DoS attacks, DDoS, buffer overflow, and anomalies in application payloads. By identifying such deviations from normal behavior, anomaly detection enhances security by enabling timely response to potential threats.[23]

I.6.6.2.1 Some advantages include:

- **Detection of Unknown Threats**

Anomaly-based detection is adept at identifying unknown or previously unseen threats by focusing on deviations from normal behavior. Unlike signature-based detection, which relies on predefined patterns of known attacks, anomaly detection can identify novel intrusion attempts that do not match any known signatures.

- **Lower False Positive Rate**

Anomaly-based detection tends to have a lower false positive rate compared to signature-based detection. By focusing on deviations from normal behavior, it can distinguish between genuine threats and benign anomalies in network traffic, reducing the number of false alarms generated.

- **Adaptability to Changing Threat Landscape**

Anomaly-based detection systems can adapt to changes in the threat landscape over time. They do not rely on static signatures and can dynamically adjust their detection algorithms to detect emerging threats without requiring frequent updates.

- **Potential for Early Detection**

By identifying deviations from normal behavior, anomaly-based detection can potentially detect intrusions at an early stage, allowing organizations to respond proactively to security threats before they escalate into full-scale attacks.

I.6.6.2.2. Some disadvantages include:

- **High False Positive Rate**

Anomaly detection systems can generate a significant number of false alarms, flagging normal behavior as anomalous.

- **Complexity of Implementation**

Implementing anomaly detection systems can be complex and resource intensive. These systems often require extensive training data to establish baseline behavior, and fine-tuning parameters to reduce false positives can be challenging.

- **Limited Effectiveness Against Targeted Attacks**

Anomaly detection systems may struggle to detect sophisticated, targeted attacks designed to blend in with normal behavior. Attackers may intentionally mimic legitimate activity to evade detection, making it challenging for anomaly detection systems to identify such threats.

- **Computational Overhead**

Some anomaly detection techniques, particularly those involving complex statistical analysis or machine learning algorithms, can impose a significant computational burden on the system. This can affect system performance and scalability, especially in high-volume network environments.

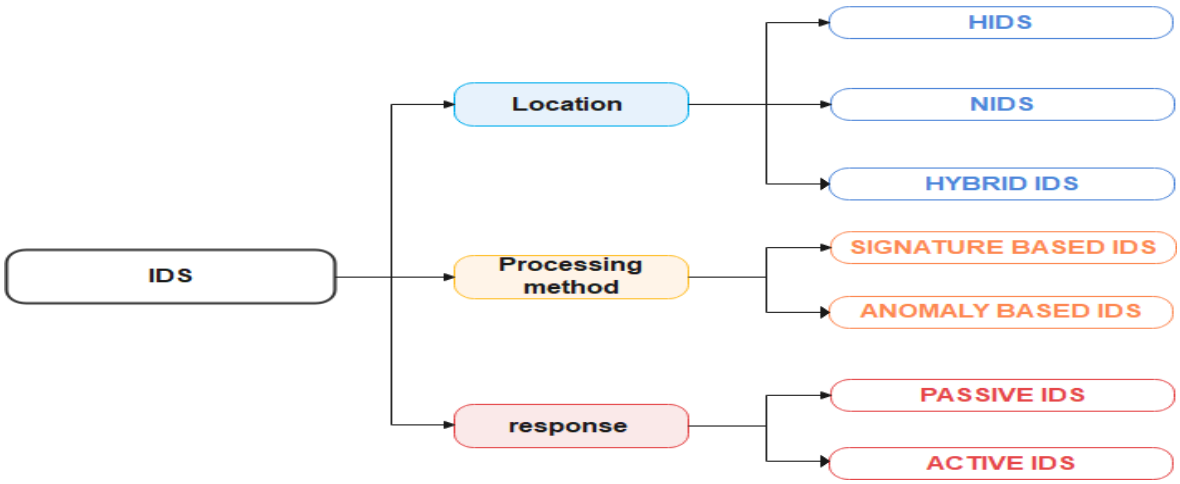


Figure I 9 SUMMRY OF IDS

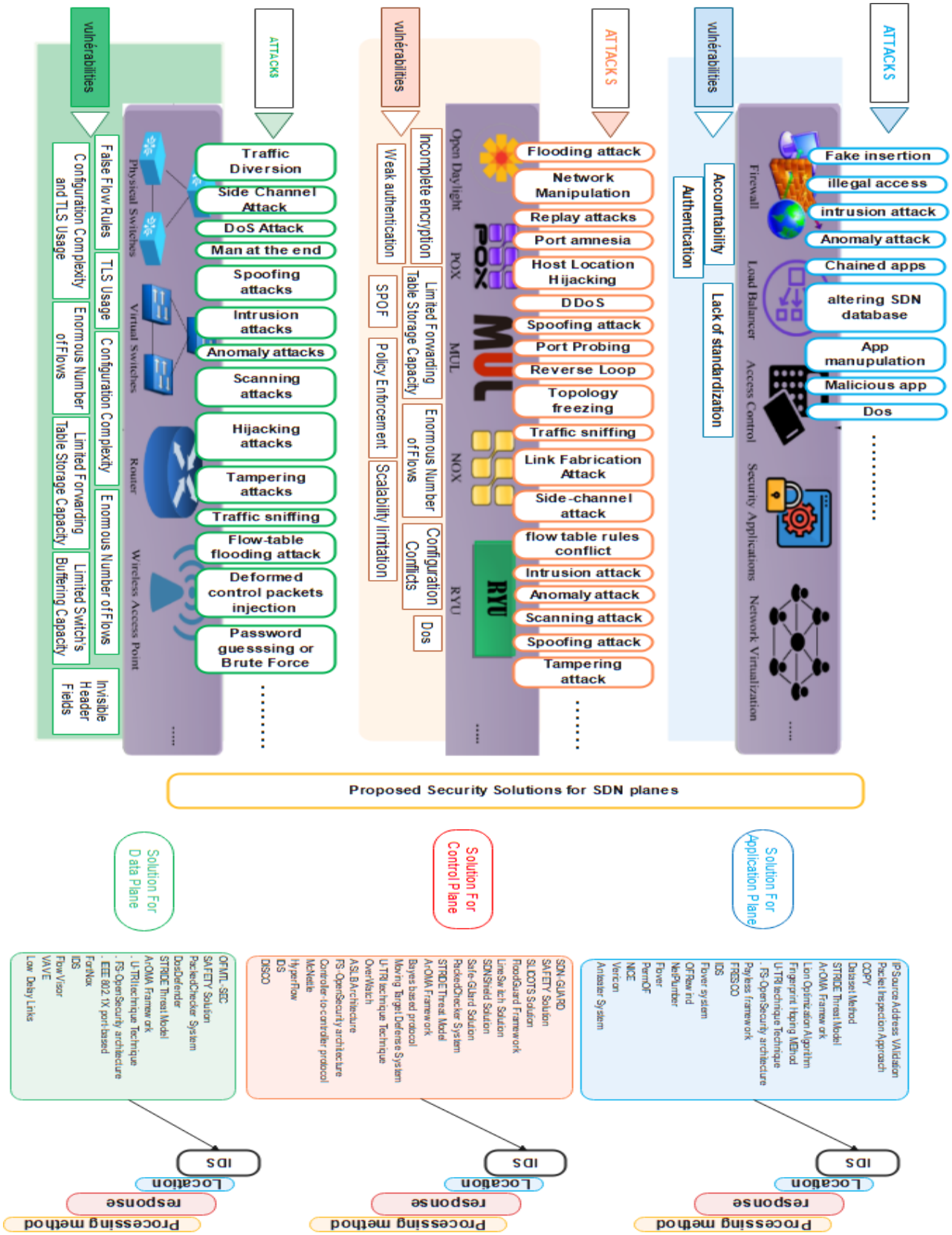


Figure I 10 SDN Security Architecture

I.7. CONCLUSION:

Now most of the attacks/intrusions are network based and the network needs to be protected. Researchers have used various approaches, including data mining, machine learning, soft computing, and statistical techniques, Bayesian Techniques, Artificial Neural Networks and Evolutionary Computing etc. to enhance the efficacy of IDS and for Network Anomaly Detection. By leveraging these advanced methodologies, organizations can bolster their defenses and stay ahead when facing evolving threats.

II.MACHINE and DEEP LEARNING

II.1 INTRODUCTION.

Computer science, data science, and artificial intelligence (AI) are closely intertwined disciplines that propel technological advancement. Computer science lays the groundwork, encompassing algorithms, programming, and system design. data science extracts insights from data. AI aims to create intelligent machines. Data science leverages computer science to analyze data, Artificial intelligence (AI) merges principles from computer science and data science to construct intelligent systems capable of understanding, reasoning, and learning. Machine learning (ML) and deep learning (DL) within AI employ algorithms enabling machines to autonomously learn from data, identifying patterns and making decisions without explicit programming. AI integrates ML and DL to create sophisticated systems capable of advanced tasks like natural language understanding and image recognition.

II.1.1. Definition of artificial intelligence.

Understanding intelligence is crucial before delving into artificial intelligence (AI). Different perspectives exist on what constitutes intelligence, ranging from knowledge and mental speed to reasoning and adaptability. Animal intelligence offers diverse examples, challenging our notions of intelligence. Bees communicate through intricate dances, while water spiders craft air-filled diving bells for hunting. Octopuses exhibit learning by imitating others, and chimpanzees display complex behaviors like communication and tool use. Evaluating intelligence across species requires recognizing unique abilities beyond human-centric criteria.[1]

Artificial Intelligence, is a branch of computer science focused on creating machines capable of performing tasks that typically require human intelligence. These tasks range from understanding natural language to recognizing objects in images and making complex decisions. Essentially, AI involves programming machines to think and learn like humans. There are various types of AI [2].

- **Reactive Machines:**

  These are the basic form of AI, reacting to their environment without memory of past events.

- **Limited Memory:**

These AI systems can learn from past experiences but are limited to specific tasks or contexts, like self-driving cars recalling traffic conditions.

- **Theory of Mind:**
  This AI attempts to understand human mental states and emotions, enabling it to respond accordingly.
- **Self-Aware:**
  The most advanced type, still in research, would possess self-awareness and consciousness.

"*Artificial intelligence is the ability of machines to do things that people would say require intelligence. Artificial intelligence research is an attempt to discover and describe aspects of human intelligence that can be simulated by machines*".
Artificial intelligence (AI) involves machines performing tasks traditionally requiring human intelligence. AI research aims to replicate aspects of human intelligence in machines. [3]

II.1.2. History

The history of AI begins long before the advent of modern computers. Philosophers like Descartes pondered the machine-like aspects of animals[24],while tales like the Prague Golem and Greek myths hinted at artificial beings. The groundwork for AI was laid in 1943 by McCulloch and Pitts, who devised mathematical models of brain neurons called perceptrons, showcasing how neurons operate and learn. Alan Turing's seminal paper in the 1950s posed the question[24], "Can a machine think?" and introduced the Turing Test to evaluate it.

The 1960s saw significant developments with the General Problem Solver by Newell and Simon and Lotfi Zadeh's concept of 'fuzzy' sets, allowing computers to mimic human-like reasoning.[24] However, ambitious claims about replicating human intelligence led to disappointment in the 1970s, termed the **"Dark Ages"** of AI research, marked by limited computing power and philosophical debates, including John Searle's Chinese room argument.

The 1980s witnessed a resurgence in AI, driven by practical applications like expert systems and robotics, diverging from debates about human-like intelligence. The period since the 1990s has seen AI's steady advancement, evidenced by achievements like Deep Blue defeating chess champion Garry Kasparov and the development of intelligent agents and neural networks. Moore's Law, predicting the doubling of computing power every two years, has fueled AI's progress, alongside novel approaches like probability theory and genetic algorithms.

II.2. MACHINE LEARNING (ML)

Automated learning, or as we commonly refer to it, machine learning (ML). Essentially, it's about teaching computers to learn from the data they receive. in other words: to program computers so that they can "learn" from input available to them, Learning here means turning that data into knowledge or skills. Roughly speaking, learning is the process of converting experience into expertise or knowledge. The input for this learning process is called training data, which represents the computer's experiences, and the outcome is usually some kind of expertise, like a program that can do a specific task. [25]

Machine learning, as defined by Arthur Samuel in 1959 [26] is a subfield of computer science where computers are empowered to learn without explicit programming. This process involves the application of statistical modeling to detect patterns and improve performance based on data and empirical information. Unlike traditional programming, where commands lead to direct actions, Instead, Samuel observed that machines don't require a direct input command to perform a set task but rather input data. Machine learning operates through a three-step process: Data > Model > Action.[26]

Machine learning differs from traditional programming in its adaptability and ability to modify assumptions through continuous learning and testing. Data is split into training and test datasets, with the model refined based on performance on both datasets. This iterative process allows for the improvement of the model's accuracy over time.[26]

II .2.1. Why machine learning

Machine Learning (ML) is preferred over manual rule-based systems for decision-making in applications due to several reasons ("if" and "else" decisions to process data or adjust to user input). Initially, intelligent applications relied on handcrafted rules, such as blacklisting words for spam filtering. However, this approach has limitations[27]:

- **Specificity and Rigidity:**

Hand coded rules are tailored to a single domain and task, making them inflexible to changes. Even slight alterations may necessitate a complete system rewrite.

- **Expert Knowledge Dependency:**

Crafting decision rules requires deep understanding from human experts, making it impractical for complex tasks.

For instance, detecting faces in images posed a challenge until recently due to the disparity between how computers perceive pixels and how humans perceive faces. This difference in representation makes it basically impossible for a human to come up with a good set of rules to describe what constitutes a face in a digital image. ML, on the other hand, overcomes these limitations. By presenting algorithms with a diverse set of

face images, they autonomously discern the necessary characteristics for face identification, making ML a more adaptable and efficient approach.[27]

Two aspects of a given problem may call for the use of programs that learn and improve based on their "experience": the problem's complexity and the need for adaptivity.
Machine Learning is indispensable when tasks are too complex to be directly programmed or when adaptability is required.[25]

II.2.1.1. Tasks That Are Too Complex to Program.

Firstly, for tasks beyond human capabilities, such as driving, speech recognition, and analyzing vast data sets like astronomical data or genomic data, Machine Learning programs excel by learning from extensive training examples, leveraging the combination of learning algorithms and computing power.[25]

II.2.1.2. Adaptivity

Secondly, Machine Learning provides adaptability, a feature lacking in traditional programmed tools. Unlike rigid programs, Machine Learning tools adapt to changes in tasks or environments. For instance, they decode handwritten text with variations between users' handwriting, detect evolving spam emails, and improve speech recognition accuracy over time. Thus, Machine Learning is essential for tackling complex problems and achieving adaptability in ever-changing scenarios.[25]

II.2.2. Machine Learning Model

Before delving into the specifics of machine learning model, it's crucial to grasp Professor Mitchell's formal definition of ML[28]:

"A computer program is said to learn from ***experience E*** *with respect to some class of* ***tasks T*** *and* ***performance*** *measure* ***P****, if its performance at* ***tasks*** *in* ***T,*** *as measured by* ***P****, improves with* ***experience E****."*

This definition revolves around three key parameters inherent to any learning algorithm: Task **(T)**, Performance **(P)**, and Experience **(E)**.
In this context, we can simplify this definition to understand that ML, as a field of AI, comprises learning algorithms that:
improve their **performance (P)** at executing some **tasks (T)** over time with **experience (E)**.

- **Task (T):**

In problem-solving, the task (T) represents the real-world problem to be addressed. This could involve something like determining the optimal house price in a specific location or finding the optimal marketing strategy.

ML tasks include classification (sorting things into categories), regression (predicting a value), clustering (finding groups of similar data points), Structured annotation (not widely used), etc.[28]

- **Experience (E):**

Experience (E) refers to the knowledge gained from the data points provided to the ML algorithm or model. The model iteratively learns inherent patterns from the dataset, which constitutes its experience. The learning thus acquired is called experience(E). like how human being, learn or gain some experiences from various attributes like situations, relationships etc.[28]

- **Performance (P):**

The performance (P) of an ML algorithm measures how effectively it performs the task (T) using its experience (E), in other words, this is how well the machine learning model performs the task after being trained on the data (experience).

P is basically a quantitative metric that tells how a model is performing the task, T, using its experience, E.[28]

II.2.3. The different types of machine learning

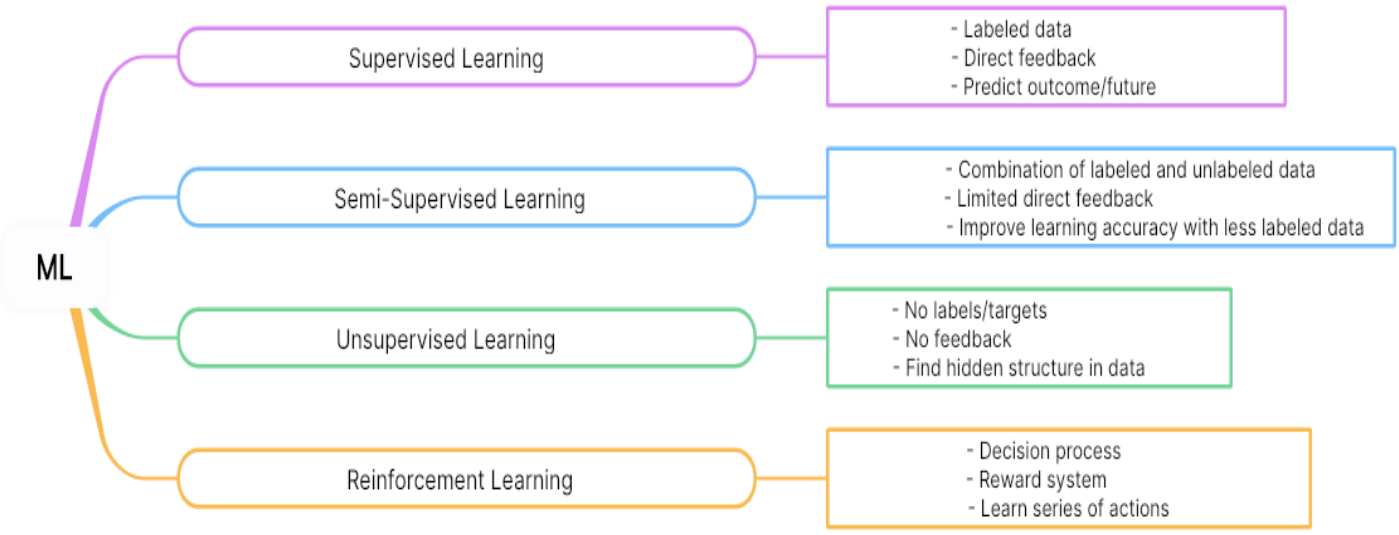


Figure II 1 types of machine learning

II.2.3.1. Supervised Learning

Supervised learning is a type and a fundamental branch of machine learning where the algorithm is trained on labeled data. It revolves around learning patterns by establishing connections between variables and known outcomes using labeled datasets. Labeled

data means that the data has a known outcome. The algorithm learns the relationship between the input and output data and then uses this knowledge to predict new, unseen data.[26]

So, it operates by providing the machine with labeled sample data, where **features (X)** and corresponding **output values (y)** are known. The algorithm then discerns underlying patterns in the data to create a model, an algorithmic equation capable of reproducing these patterns with new data. Common supervised learning algorithms include regression analysis, decision trees, k-nearest neighbors, neural networks, and support vector machines.[26]

There are two main types of supervised learning: classification and regression. Classification is used when the output is a category. Regression is used when the output is a continuous value.[29]

Examples of supervised learning algorithms include Decision Trees, Random Forest, KNN, and Logistic Regression.[28]

We give the computer a bunch of data with two parts: **"features (X)"** and **"labels (Y)"**.

The computer uses an algorithm to learn the **"rule"** that connects the features to the labels. We can write this rule as a function: **Y = f(X)**

Once the computer learns the rule, it can be used to predict **the outcome (Y)** for entirely **new data (X)** it hasn't seen before.

This way, supervised learning lets computers learn from examples and make predictions on new data, useful for tasks like spam filtering or image recognition.

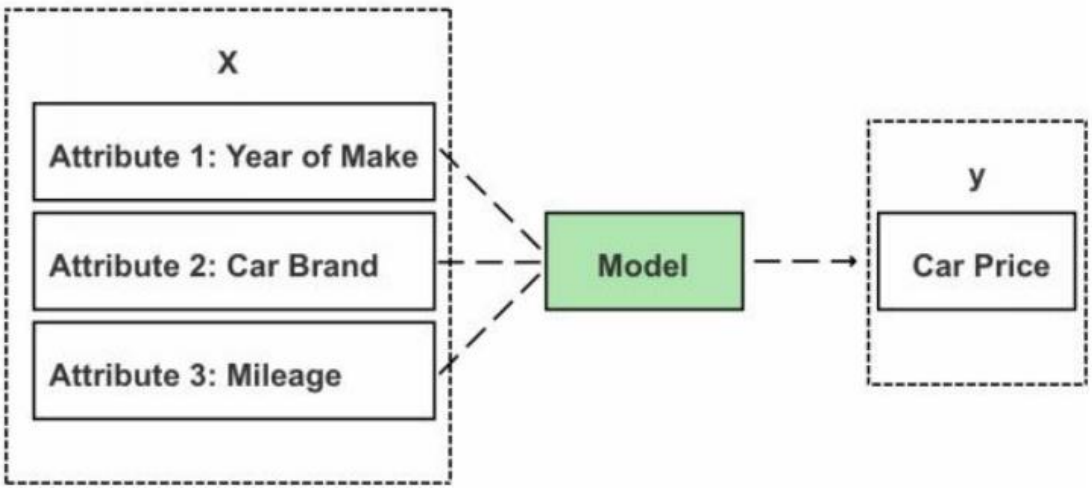


Figure II 2 the function algorithm

II.2.3.1.1. Types of supervised learning:

1- **Classification**

Classification is a type of supervised learning aimed at predicting categorical class labels for new data based on past observations.

These labels represent groups or categories that the data belongs to. Where these class labels are discrete and unordered (e.g., not "more spammy" or "less spammy"), representing group memberships.[29]

Classification is sometimes divided into binary classification, which is like trying to answer a yes/no question. For example, email spam detection is a binary classification task where the model distinguishes between spam and non-spam emails.[29]

Classification can also involve multiple classes, as in handwritten character/digit recognition and it is called multiclass classification, where the model predicts letters of the alphabet based on training data. The process involves training a model to learn decision boundaries that separate different classes based on the training data, enabling the classification of new instances.

- **Algorithms**

a- Decision trees

Decision Tree analysis is a versatile predictive modeling tool used across various fields. As a powerful supervised algorithm, it can handle both classification and regression tasks. Decision trees split datasets into subsets based on different conditions. They consist of decision nodes, where data is split, and leaves, which represent the outcomes.[28]

b- Random forest

Random forest is a supervised learning technique used for both classification (predicting categories) and regression (predicting continuous values). It's particularly strong in classification.

The algorithm works by creating a bunch of decision trees from random subsets of the data. Each tree votes on the outcome, and the most popular vote wins. This approach reduces overfitting compared to a single decision tree.[28]

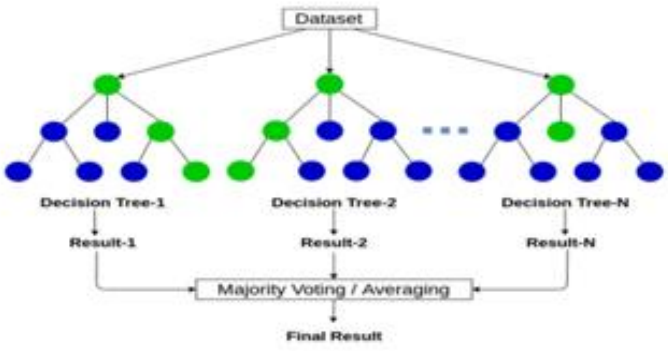


Figure II 3 Algorithm - *Random forest*

c- k-Nearest Neighbors (k-NN) Algorithm

The k-nearest neighbors (k-NN) algorithm is a simple, instance-based learning method used for classification and regression in supervised machine learning. It classifies a data point by examining the `k` closest data points in the feature space and assigning the most common label (for classification) or averaging the values (for regression) among those neighbors. The algorithm relies on calculating distances, typically using the Euclidean distance, between the data points and does not require an explicit training phase beyond storing the dataset.[27]

2-Regression

Regression is a supervised learning technique used to predict continuous numerical values outcomes based on input data.in other words, Predict continuous numeric outputs from input data. leveraging patterns learned during training. It involves using predictor variables to establish relationships with continuous response variables, enabling accurate outcome predictions.

The term regression was devised by Francis Galton in his article Regression towards Mediocrity in Hereditary Stature in 1886,[29] who observed that children's heights tend to regress towards the population mean, rather than exactly matching their parents' heights.

The primary goal of regression tasks is to predict continuous numeric values for given input data ; such as a person's annual income based on their education, age, and location. The predicted values can be any number within a range. The output is based on patterns the model has learned during training. Regression models use input features (independent variables) and their corresponding continuous output values (dependent or outcome variables) to identify associations between the inputs and outputs. [28]

d- Algorithms

a- Linear regression

Linear regression is one of the simplest and most widely used regression algorithms. It models the relationship between a **dependent variable (target)** and one or more **independent variables (features)** by fitting a linear equation to the observed data.

Linear regression is a statistical model that examines the linear relationship between a dependent variable and one or more independent variables. In a linear relationship, changes in the independent variable(s) result in corresponding changes in the dependent variable. The relationship is represented by the equation **Y = m.X + b**, where:

- **Y** is the dependent variable.
- **X** is the independent variable.
- **m** is the slope of the regression line (indicating the effect of **X** on **Y**).
- **b** is the **Y** -intercept- the point where the line crosses the **Y-axis** (when **X** is zero).

The linear relationship can be **positive** (both variables increase together) or **negative** (one variable increases while the other decreases).[28]

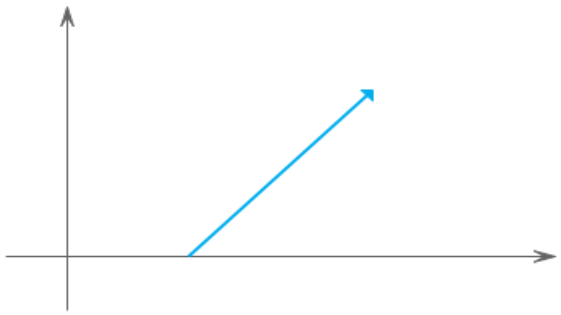
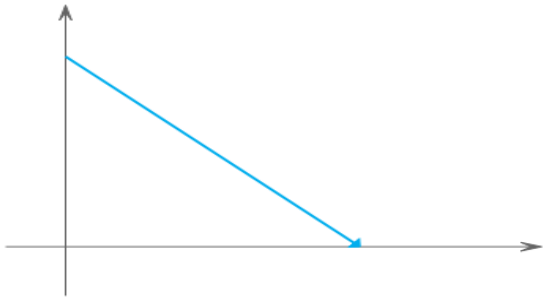
Positive Linear Relationship	Negative Linear Relationship
	

Figure II 4 linear relationship

II.2.3.2. Semi-supervised Learning: The Middle Ground

Semi-supervised learning is a branch of machine learning that combines elements of both supervised and unsupervised learning. It utilizes a small amount of labeled data alongside a large amount of unlabeled data. The goal of this approach is to Leverage the unlabeled data to improve the model's performance beyond what's possible with just labeled data.

Semi-supervised learning bridges the gap between supervised and unsupervised learning by using both labeled and unlabeled data for training. Various strategies can be adopted to implement semi-supervised learning techniques.[28]

- **First strategy:**
  - Build a supervised model using the labeled data.
  - Build the unsupervised model by applying the same to the unlabeled data to get more labeled samples.
  - Train the model on them and repeat the process.
- **Second strategies**
  - Group similar data samples together using unsupervised clustering techniques.
  - Annotate the clusters.
  - Train the model using a combination of this information.

II.2.3.2.1. Types of Semi-Supervised Learning

- **Self-Training**

Self-training is a wrapper method that iteratively uses a supervised learning model to label the unlabeled data. The model is initially trained on the labeled data, and then it predicts labels for the unlabeled data; where, unsupervised learning techniques are used primarily in the steps involving the unlabeled data. The most confident predictions are added to the labeled dataset, and the model is retrained.

II.2.3.3. Unsupervised learning

Unsupervised learning is a type of machine learning where the algorithms are not given labeled data. Instead, the algorithms are left to find patterns and structures in the data on their own.

Unsupervised learning flips the script on traditional machine learning by tackling datasets without predefined labels. Instead of being spoon-fed patterns, the algorithm has to play detective, uncovering hidden structures and patterns autonomously. Take k-means clustering, for instance.

It's like organizing a messy room by grouping similar items without any prior instructions.

This approach is particularly potent for tasks such as fraud detection, where the most insidious threats lurk in the shadows, where the most dangerous attacks are often those that have not yet been classified.

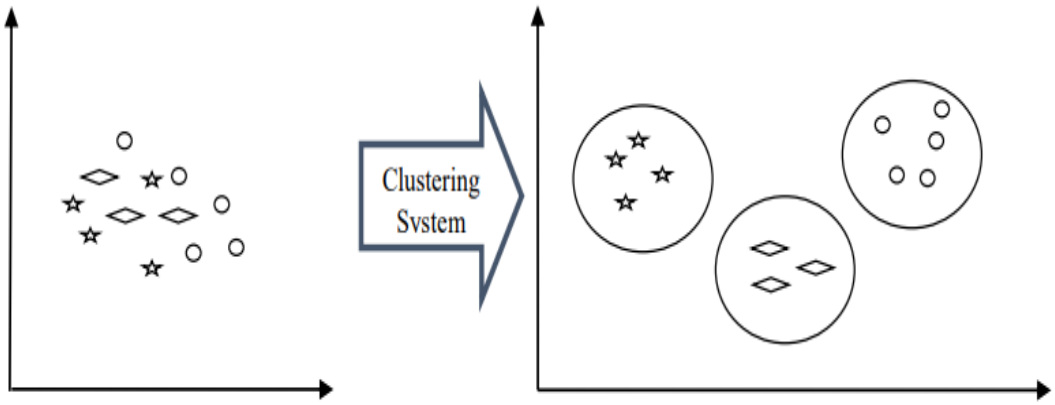


Figure II 5 Clustering system

- **Algorithms:**

1- **K-means Clustering**

K-Means is one of the simplest and most widely used clustering algorithms, it is an unsupervised learning method used to partition a dataset into **`k' clusters**. Each cluster is represented by its centroid, which is the mean of the data points within the cluster, where each data point belongs to the cluster with the nearest mean. The goal is to minimize the variance within each cluster.

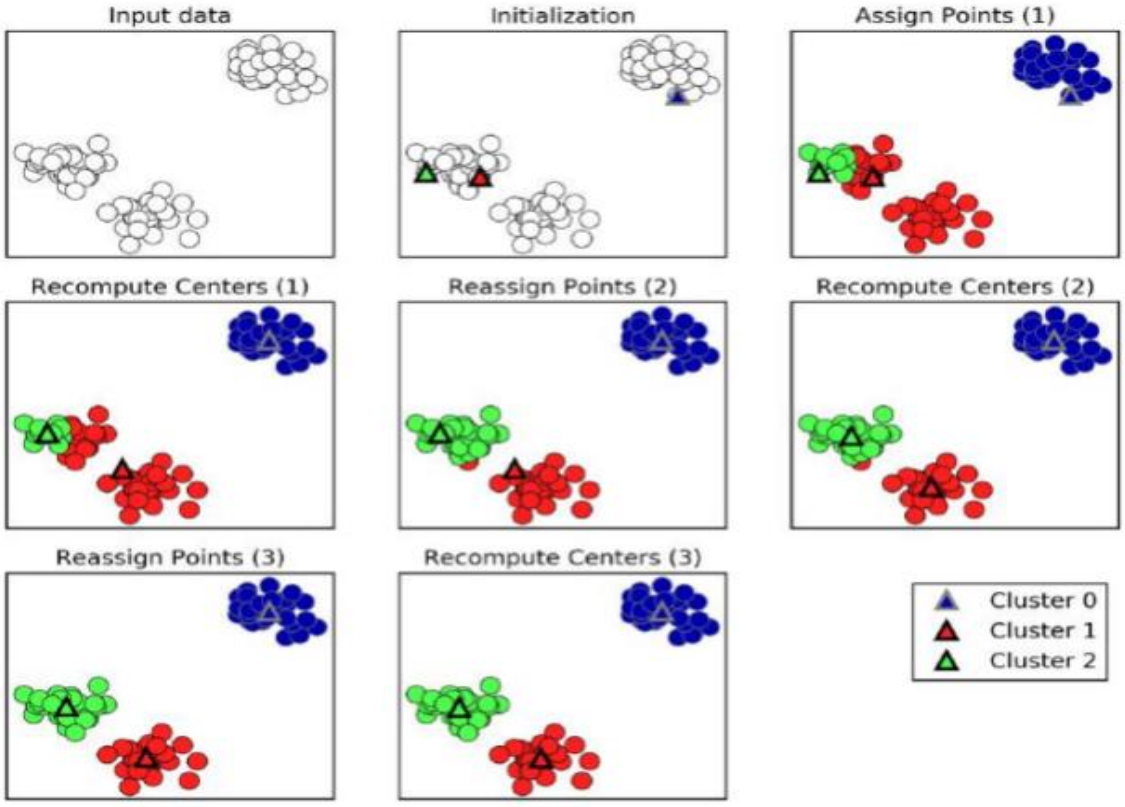


Figure II 6 Input data and three steps of the k-means algorithm

2- Dimensionality Reduction

Dimensionality reduction is a subfield of unsupervised learning that addresses the challenge of high-dimensional data, which can hinder storage capacity and computational efficiency. This method reduces the number of features **(dimensions)** in the data by removing noise and compressing the data onto a smaller subspace while retaining most of the relevant information; in other words, Reducing the number of features while retaining most of the important information.

Techniques like **Principal Component Analysis (PCA)**, K-nearest neighbors, and discriminant analysis are some of the common algorithms used for this purpose, helping to manage the **"curse of dimensionality"** which is the issue of feature space complexity (which emerges once the analysis and extraction of millions of features from data samples begin) by simplifying the feature space.

Often data has many features, which can be cumbersome to store, analyze and visualize, dimensionality reduction helps you work with data more effectively by finding a more compact representation that preserves its essence.

3- Reinforcement learning

Reinforcement learning is a type of machine learning where an agent improves its performance through interactions with its environment. Thus, the Goal is to Develop an agent (system) that gets better at a task by interacting with its environment.

The key difference from Supervised Learning is that Feedback comes in the form of rewards (positive or negative) instead of labeled data. In other words, unlike supervised learning, which uses predefined labels, reinforcement learning relies on a reward signal that measures the success of an action. The agent learns by trial and error, aiming to maximize this reward through a series of actions. The agent explores the environment by taking actions and observing the resulting rewards. It then uses this information to learn a series of actions that maximize its overall reward. This can be done through trial-and-error approach or deliberative planning.

A classic example is a chess engine, a chess engine learns by playing many games and receiving a reward **(win)** or penalty **(loss)** for each outcome. Over time, it improves its ability to choose moves that lead to victory; where the agent makes moves based on the board's state, with the reward being winning or losing the game. Each move changes the state, and the agent learns to associate actions with positive or negative outcomes to optimize its strategy. Reinforcement learning continuously refines the agent's behavior, similar to a video game player learning and improving based on past experiences and feedback.

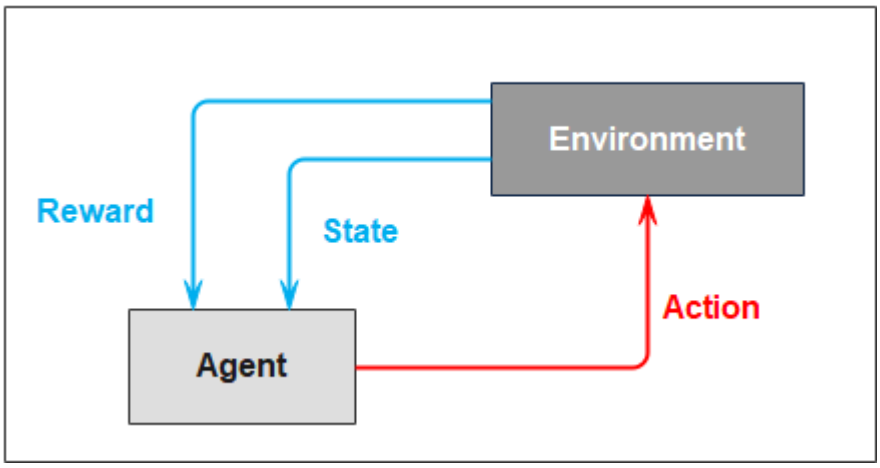


Figure II 7 Reinforcement learning

4- Q-Learning

Q-learning is a reinforcement learning algorithm where an agent learns to navigate an environment by interacting with **states (S)** and taking **actions (A)** to maximize **a value function (Q)**. Initially, **Q-values** are set to zero and are updated based on the rewards or penalties resulting from actions taken in different states. The agent learns through exploration, adjusting **Q-values** to reflect positive and negative outcomes, ultimately optimizing its strategy to maximize rewards and minimize penalties.[26]

II 3. DEEP LEARNING

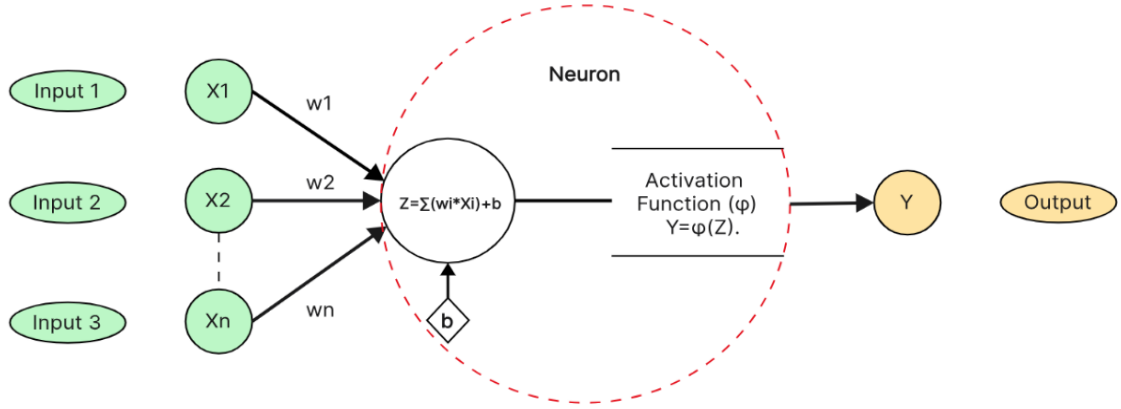


Figure II 8 main components of an ANN

Deep learning is a subfield of machine learning focused on an imitation of the organization and functioning process of the human brain for problem-solving. This is done by the training of artificial neural networks with more than one single layer, so they learn a hierarchical representation of data.[30]
The term "deep" in "deep learning" refers to the number of layers within a neural network (depth of the neural network).[31]
Although it is now most commonly referred to as deep learning, there are other names such as layered representations learning or hierarchical representations learning can also capture well its essence. [31]
However, in order to understand deep learning fully, it is crucial to initially explore its roots in the broader field of artificial neural networks (ANNs).

II.3.1 Artificial neural networks (ANNs)

Artificial Neural Networks are defined as one kind of computational model, and they derive their structure and functionalism from the human brain (biological neural networks) .[32]

The basic building block of every artificial neural network is artificial neuron.

II.3.1.1 Neurons and Layers

- **Neurons**

Neurons; (also known as "neurode"; the basic processing components; or units) are used in an artificial neural network to create a network that replicates a real neuronal network, each neuron receives input signals, perform computations, and produces an output signal

➢ **Components of an Artificial Neuron**

- Inputs (X1, X2, ..., Xn): Data or features that are fed into the neuron.
- Weights (w1, w2, ..., wn): Parameters that signifies the importance or strength of that input.
- Bias (b): is the bias, a parameter added to the weighted sum of inputs.
- Activation Function (φ): enable the network to have non-linearity, which is an essential property for learning complex patterns, determining the neuron's output.

1- Function of a Neuron

- Weighted Sum**:** $\boldsymbol{Z = \sum(w_i . X_i) + b}$
  Calculates a weighted sum of the inputs.
- Activation function : $Y = \phi(Z)$

2- **Types of Activation Functions**

Nature	Activation Function	Formula	Range
Linear Activation Functions	Linear	$\varphi(z) = z$	(-∞,∞)
Non-Linear Activation Functions	Sigmoid	$\varphi(z) = \frac{1}{1+e^{-z}}$	(0, 1)
	Tanh	$\varphi(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	(-1, 1)
	ReLU	$\varphi(z) = \max(0, z)$	$[0, \infty)$
	Softmax	$\varphi(z) = \frac{e^{zi}}{\sum_{j}^{k} e^{zj}}$	(0,1)

Table II 1 Activation functions.

3- Layers:

ANNs are basically built from a layers of connected neurons,

a- Input Layer

This is where the input data is introduced. A neuron in this layer corresponds to one feature of the input.

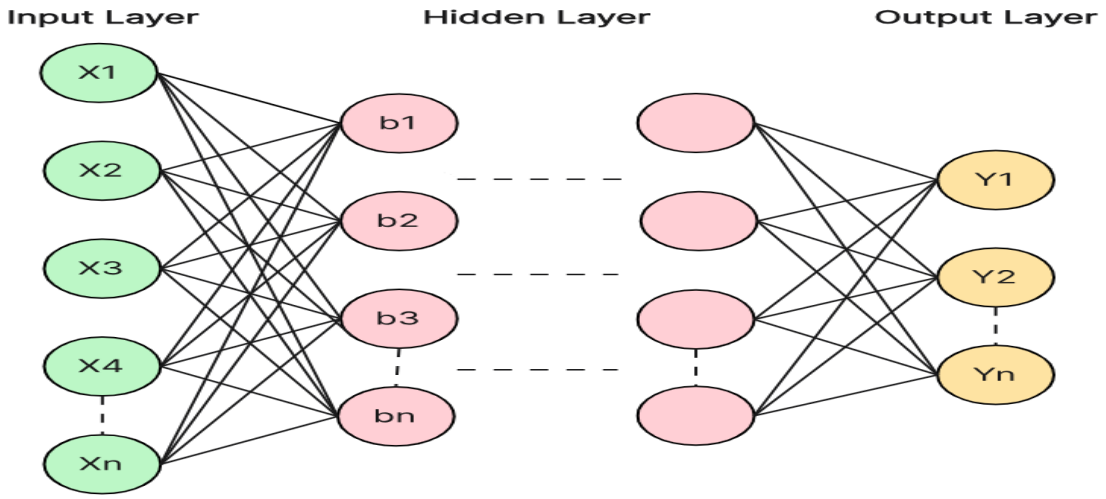


Figure II 9 Layers of ANN

b- Hidden Layer

An intermediate layer where all the computation takes place to derive the features from the input data. ANNs may exist in deep forms with the potential for many hidden layers (e.g., with deep learning, there can be many hidden layers).

c- Output Layer

The final layer that provides a prediction or a classification. The number of neurons in this layer depends on the task, such as classification or regression[32]

II3.1.1. How Artificial Neural Networks Learn

Neural networks are learned through training. This involves:

- **Forward Propagation:**

Input data is passed through the network and then layer by layer until an output has been produced.

- **Calculated Loss:**

The output, using a loss function, measures the error showing the mismatch of the predicted value by the model.

- **Backpropagation:**

Is the process of moving errors through the network in order to update the weights by the optimization algorithm.

II.3.1.2. Types of Learning Paradigms:

In deep learning, as in machine learning in general, there are several types of learning paradigms.

Different learning paradigms are adopted in tackling different types of tasks and data sets. These paradigms define how models are trained, what data to use, and in what manner to learn from that data. Here are the primary learning paradigms in deep learning:

- **Supervised Learning**

Supervised learning includes learning from a labeled dataset where a label is assigned to an output associated with an input. The task is to learn a mapping from inputs to outputs.[33] For this we have Classification and Regression

➢ **Techniques:**

1- Convolutional Neural Networks (CNN).

Is a type of deep learning model designed specifically for processing and analyzing visual data, such as images and handwriting .[33]

1-1- Architecture

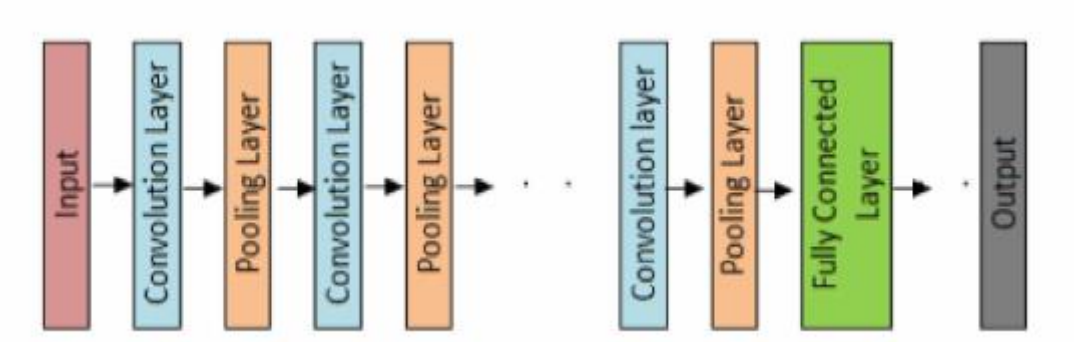


Figure II 10 Conceptual model of CNN *[34]*

Convolutional Layers

Purpose**:** Extract features from the input data.

Pooling Layers.

Purpose**:** Reduce the spatial dimensions of the feature maps, lowering computational complexity and reducing the risk of overfitting. And we have two types : Max Pooling ;Average Pooling)

Fully Connected Layers

Purpose**:** Perform high-level reasoning and classification.

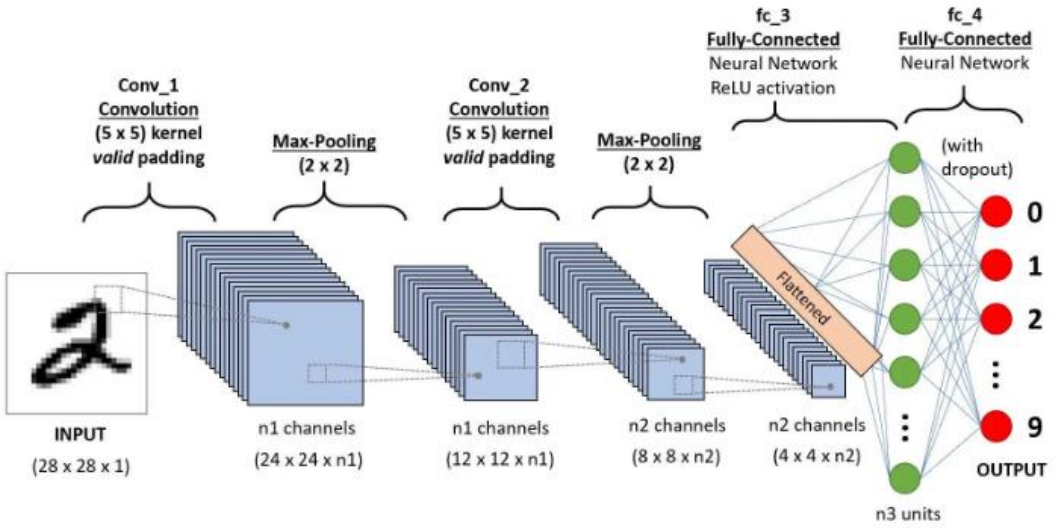


Figure II 11 Convolution Neural Network [34]

2- Recurrent Neural Networks (RNNs)

Are a critical component of deep learning architectures designed to handle sequential of text data and time series.

They form connections so as to make a directed cycle, and so, information gets to persist .[35]

2-1- Variants

- **Long Short-Term Memory (LSTM)**

LSTMs are a type of recurrent neural network (RNN) designed to handle and learn from sequential data. They are particularly effective in capturing long-term dependencies and avoiding issues like the vanishing gradient problem, which RNNs often face.

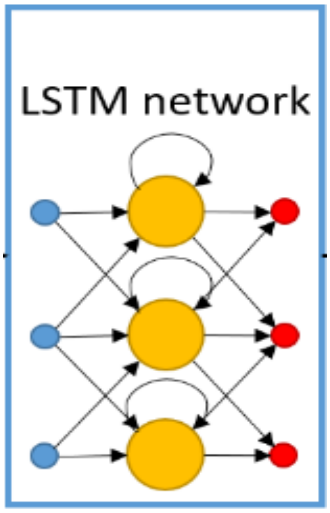


Figure II 12 LSTM network

- **Gated Recurrent Unit (GRU)**

3- Multilayer Perceptrons (MLPs).

is a simple type of artificial neural network, while it can be considered as part of deep learning just in the case of the presence of many hidden layers .[32]

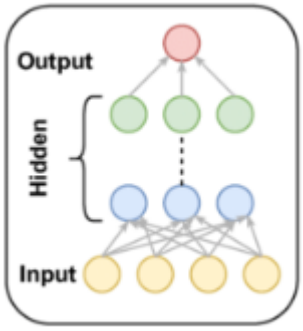


Figure II 13 MLP

4- Unsupervised Learning

Unsupervised learning is concerned with unlabeled data, but in general, the model's task is to identify patterns or similarities in the data without any guidance on what to look for.[33] And we have Clustering Dimensionality Reduction.

Techniques:

1-Autoencoders make learning coding efficient.

Autoencoders are a type of neural network used for unsupervised learning to learn a compressed representation of input data **[23]**.

5- Architecture

- Encoder
- Decoder

6- Variants

- Variational Autoencoders (VAE)
- Denoising Autoencoders (VDA)
- Sparse Autoencoders (VSA)

2- **Generative Adversarial Networks (GANs) for data generation**

GANs are for data generation, they consist of two neural networks: a generator (G) and a discriminator (D) that are trained simultaneously in a competitive manner .[33]

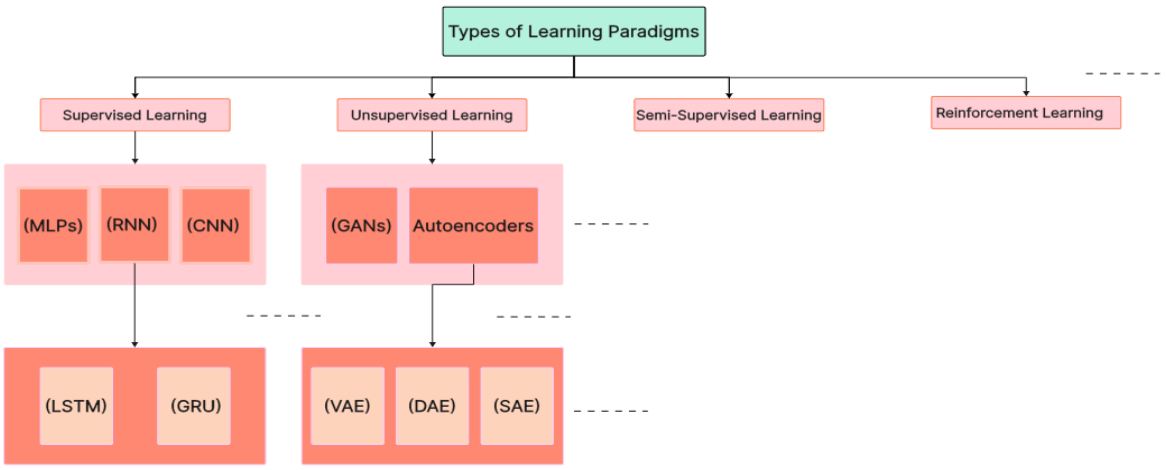


Figure II 14 Deep learning architectures based

II.4. Classification Evaluation Metrics

II.4.1. Confusion Matrix.

A confusion matrix,also called a contingenv table or error matrix ,is used to visualize the performance of a classifer .

The columns of the matrix represent the instances of the predicted classes and the rows represent the instances of the actual class. (Note: It can be the other way around as well.) [36]

A confusion matrix is a simple way to see how well a classification model is doing. It's a table with two sides: actual and predicted. Each side is divided into true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).[28]

Confusion Matrix		Actual classes	
		positive	negative
Predicted classes	positive	TP	FP
	negative	FN	TN

Table II 2 Confusion Matrix

- **TP:** The model correctly predicted a class 1.

- **TN:** The model correctly predicted a class 0.
- **FP:** The model incorrectly predicted a class 1 (it thought something was a 1 when it actually wasn't).
- **FN:** The model incorrectly predicted a class 0 (it missed a 1).

II.4.1.1. Accuracy and Precision

7- **Accuracy.**

A confusion matrix is a great tool to calculate accuracy, and the formula you provided is correct.
Here's the breakdown of how accuracy is calculated using the confusion matrix:

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

This formula essentially calculates the proportion of correct predictions (TP + TN) out of the total number of predictions (all four categories). A higher accuracy value (closer to 1) indicates a better-performing model.[28]

8- Precision.

Precision is a crucial metric in classification problems, particularly in the context of information retrieval, binary classification, and other similar tasks.

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

Precision measures the accuracy of the positive predictions made by a model. It tells us what proportion of the items the model identified as positive are positive.[36]

II4.1.2. Recall or Sensitivity.

9- Recall

Recall, also known as sensitivity, measures the ability of a model to identify all relevant instances within a dataset. It tells us what proportion of the actual positives were correctly identified by the model.

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

Recall may be defined as the number of positives our ML model returns. We can easily calculate it by using a confusion matrix with the help of the above formula .[36]

II.4.1.3. F1-Score

The F1 score is a metric used to evaluate the performance of a classification model. It is the harmonic mean of precision and recall, providing a balance between the two.[28]

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

II.4.1.3. Specificity

Specificity**:** It measures the proportion of negative cases that the model correctly identifies as negative[28]

$$Specificity = \frac{TN}{TN + FP}$$

III. PROPOSED APPROACH

III.1.INTRODUCTION:

In the rapidly evolving landscape of networking and communication, Software-Defined Networking (SDN) has emerged as a transformative paradigm, enabling dynamic, programmable, and efficient network management. At the heart of SDN lies the concept of centralizing network control and intelligence, allowing for flexible network configurations and optimizations. However, as networks grow in complexity and scale, traditional approaches to Intrusion Detection Systems (IDS) face significant challenges in detecting and mitigating security threats effectively.

III.2. DESCRIPTION OF THE DATASET USED

III.2.1. CTU13-CSV-Dataset-main

The CSV files are derived from the original CTU-13[1] dataset, which contains labeled network traffic data, including normal, attack (botnet), and background traffic. The original dataset was captured at CTU University, Czech Republic, in 2011, with the goal of providing a large dataset containing real botnet traffic mixed with normal and background traffic.

III.2.1.1. Data Generation Process

- Data Import:

The original PCAP (packet capture) files from the CTU-13 dataset were imported using the CICFlowMeter tool. This tool processes raw network traffic captured in PCAP files and extracts flow-based features, outputting them in CSV format.

- Conversion to CSV:

The CICFlowMeter tool converts the PCAP files into CSV files based on network flows. Each flow represents a sequence of packets between two endpoints.

`- CTU13_Attack_Traffic.csv`:

Contains network traffic data associated with botnet attacks. Each row in the CSV file represents a network flow that has been labeled as part of an attack.

`- CTU13_Normal_Traffic.csv`:

[1] **Original Dataset:** *The original CTU-13 Dataset is available at [CTU-13 Dataset] (https://www.stratosphereips.org/datasets-ctu13).*

Contains normal network traffic data. Each row in the CSV file represents a network flow that has been labeled as normal or benign.

III.2.2.2. Key Information about CTU-13 Dataset

- Purpose:

- The dataset aims to provide comprehensive data for research in intrusion detection, botnet detection, and network traffic analysis.

- Scenarios:

The CTU-13 dataset includes thirteen different captures (scenarios), each capturing traffic generated by executing specific malware, which used various protocols and performed different actions.

This information clarifies that the `CTU13_Normal_Traffic.csv` and `CTU13_Attack_Traffic.csv` files contain samples of normal and botnet attack traffic, respectively, derived from the larger CTU-13 Dataset.

The columns of the dataset	
Data	**Type**
Unnamed: 0	int64
Flow Duration	int64
Tot Fwd Pkts	int64
Tot Bwd Pkts	int64
TotLen Fwd Pkts	int64
TotLen Bwd Pkts	int64
Fwd Pkt Len Max	int64
Fwd Pkt Len Min	int64
Fwd Pkt Len Mean	float64
Fwd Pkt Len Std	float64
Bwd Pkt Len Max	int64
Bwd Pkt Len Min	int64
Bwd Pkt Len Mean	float64
Bwd Pkt Len Std	float64
Flow Byts/s	float64
Flow Pkts/s	float64
Flow IAT Mean	float64
Flow IAT Std	float64
Flow IAT Max	int64
Flow IAT Min	int64
Fwd IAT Tot	int64
Fwd IAT Mean	float64
Fwd IAT Std	float64
Fwd IAT Max	int64
Fwd IAT Min	int64
Bwd IAT Tot	int64
Bwd IAT Mean	float64
Bwd IAT Std	float64
Bwd IAT Max	int64
Bwd IAT Min	int64
Bwd PSH Flags	int64
Fwd Header Len	int64
Bwd Header Len	int64
Fwd Pkts/s	float64
Bwd Pkts/s	float64
Pkt Len Min	int64
Pkt Len Max	int64
Pkt Len Mean	float64
Pkt Len Std	float64
Pkt Len Var	float64

FIN Flag Cnt	int64
SYN Flag Cnt	int64
RST Flag Cnt	int64
ACK Flag Cnt	int64
Down/Up Ratio	int64
Pkt Size Avg	float64
Fwd Seg Size Avg	float64
Bwd Seg Size Avg	float64
Init Bwd Win Byts	int64
Fwd Act Data Pkts	int64
Active Mean	float64
Active Std	float64
Active Max	int64
Active Min	int64
Idle Mean	float64
Idle Std	float64
Idle Max	int64
Idle Min	int64
Label	object

Table III 1 Dataset CTU-13

III.2.2. InSDN_DatasetCSV

It is a Software-Defined Network (SDN) intrusion dataset designed specifically for intrusion detection systems in SDN environments.

The InSDN dataset collection includes specific datasets such as metasploitable-2.csv, Normal_data.csv, and OVS.csv, each serving a distinct purpose in the context of intrusion detection systems in SDN environments

The columns of the dataset	
Data	**Type**
Flow ID	object
Src IP	object
Src Port	int64
Dst IP	object
Dst Port	int64
Protocol	int64
Timestamp	object
Flow Duration	int64
Tot Fwd Pkts	int64
Tot Bwd Pkts	int64
TotLen Fwd Pkts	float64
TotLen Bwd Pkts	float64
Fwd Pkt Len Max	int64
Fwd Pkt Len Min	int64
Fwd Pkt Len Mean	float64
Fwd Pkt Len Std	float64
Bwd Pkt Len Max	int64
Bwd Pkt Len Min	int64
Bwd Pkt Len Mean	float64
Bwd Pkt Len Std	float64
Flow Byts/s	float64
Flow Pkts/s	float64
Flow IAT Mean	float64
Flow IAT Std	float64
Flow IAT Max	float64
Flow IAT Min	float64
Fwd IAT Tot	float64

Fwd IAT Mean	float64
Fwd IAT Std	float64
Fwd IAT Max	float64
Fwd IAT Min	float64
Bwd IAT Tot	float64
Bwd IAT Mean	float64
Bwd IAT Std	float64
Bwd IAT Max	float64
Bwd IAT Min	float64
Fwd PSH Flags	int64
Bwd PSH Flags	int64
Fwd URG Flags	Int64
Bwd URG Flags	int64
Fwd Header Len	int64
Bwd Header Len	int64
Fwd Pkts/s	float64
Bwd Pkts/s	float64
Pkt Len Min	int64
Pkt Len Max	int64
Pkt Len Mean	float64
Pkt Len Std	float64
Pkt Len Var	float64
FIN Flag Cnt	int64
SYN Flag Cnt	int64
RST Flag Cnt	int64
PSH Flag Cnt	int64
ACK Flag Cnt	int64
URG Flag Cnt	int64
CWE Flag Count	int64
ECE Flag Cnt	int64
Down/Up Ratio	int64
Pkt Size Avg	float64
Fwd Seg Size Avg	float64
Bwd Seg Size Avg	float64
Fwd Byts/b Avg	int64
Fwd Pkts/b Avg	int64
Fwd Blk Rate Avg	int64
Bwd Byts/b Avg	int64
Bwd Pkts/b Avg	int64
Bwd Blk Rate Avg	int64
Subflow Fwd Pkts	int64
Subflow Fwd Byts	int64
Subflow Bwd Pkts	int64
Subflow Bwd Byts	int64
Init Fwd Win Byts	int64
Init Bwd Win Byts	int64
Fwd Act Data Pkts	int64
Fwd Seg Size Min	int64
Active Mean	float64
Active Std	float64
Active Max	float64
Active Min	Float64
Idle Mean	float64
Idle Std	float64
Idle Max	float64
Idle Min	float64
Label	object

Table III 2 Dataset InSDN

III.3.TOOLS:

Python is a versatile and beginner-friendly programming language widely used in data science, machine learning, web development, and more. Its easy-to-read syntax and vast libraries make it a popular choice for various tasks.

Figure III 1 python logo

Anaconda is a free and open-source distribution of Python that simplifies setting up a data science environment. It comes pre-packaged with Python itself, hundreds of popular data science packages (like NumPy, Pandas, Matplotlib, etc.), and a powerful package manager called conda. This eliminates the need to install each package individually, saving you time and effort.

Figure III 2 ANACONDA logo

Jupyter Notebook is an interactive web-based application included in Anaconda's distribution. It allows you to create and share documents called notebooks that combine live code, equations, visualizations, and explanatory text. This makes it a perfect tool for data exploration, prototyping, and creating reports.

Figure III 3 jupyter logo

Jupyter Notebook works by utilizing kernels, which are essentially mini-environments that execute code written in a specific programming language.
This interactive nature makes Jupyter Notebook an excellent tool for learning, experimentation, and data analysis.
Anaconda, Jupyter Notebook, and Python are a powerful trio for data science and scientific computing.

III.4.PROPOSED APPROACH

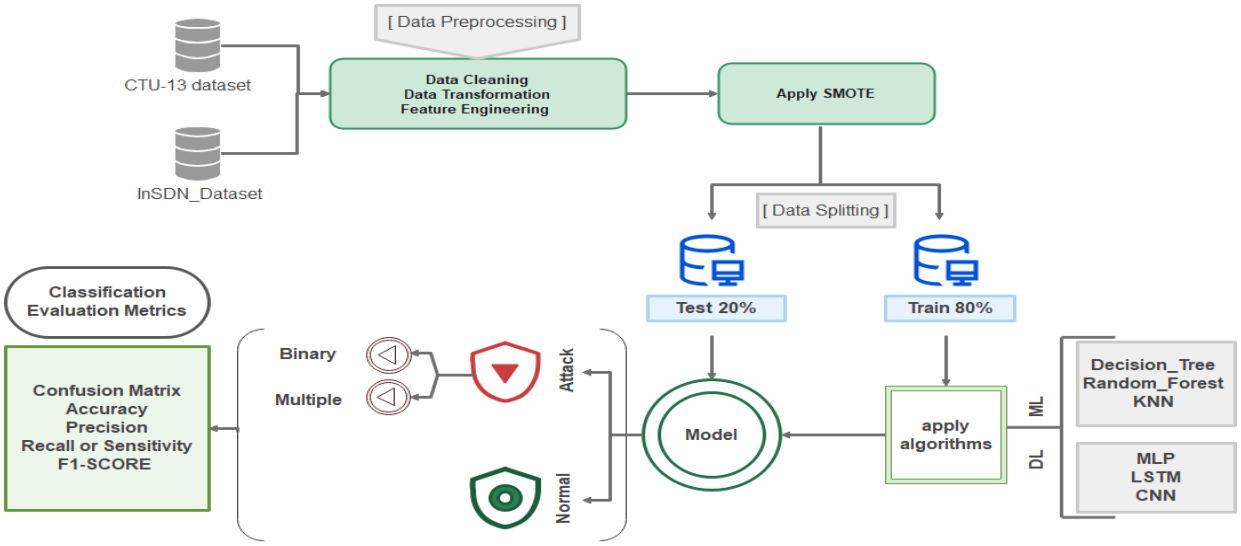


Figure III 4 Proposed approach

III.4.1. Data processing

Data processing is all about taking raw data and turning it into something useful. This involves collecting the data, organizing it, and then cleaning and transforming it so that it can be used to make decisions. Data preprocessing is a critical step in this process, because it ensures that the data is high quality and ready to be used by machines.

III.4.1.1.Preprocessing steps

Data preprocessing plays a crucial role in machine learning and deep learning by ensuring that the data is properly formatted and cleaned before being used to train accurate models. Multiple steps are taken to preprocess the data effectively before feeding the raw dataset into a machine and deep learning algorithms.

1- **Data Cleaning:**
-Handling Missing Values: Replacing or imputing missing values to ensure completeness.
-Handling Duplicate Rows
2- **Data Transformation:**
-Normalization/Standardization: Scaling features to a standard range or distribution, often to improve convergence in optimization.
-Encoding Categorical Variables: Converting categorical data into numerical format (e.g., one-hot encoding, label encoding).

3- **Feature Engineering**

- Elimination of unnecessary columns

4- **Data Preparation**

-Separate features and labels

-Splitting data into Train-Test sets

-Apply SMOTE

Data Cleaning:

Handling Missing Values:

Check for missing values:

```
# Check for missing values
print("\nMissing Values:")
cmbd_df.isnull().sum()
```

CTU13-CSV-Dataset-main	InSDN_DatasetCSV
Series([], dtype: int64)	Series([], dtype: int64)

Handling Duplicate Rows

Check for duplicated rows

CTU13-CSV-Dataset-main	InSDN_DatasetCSV
Number of duplicated rows before removal: 0	Number of duplicated rows before removal: 1

Remove duplicated rows:

```
# Remove duplicated rows
df.drop_duplicates(inplace=True)
```

Check for duplicated rows after removing:

```
# Check for duplicated rows after removing
print("Number of duplicated rows after removing:", df.duplicated().sum())
```

InSDN_DatasetCSV
Number of duplicated rows after removing: 0

Feature Engineering

Elimination of unnecessary columns

Drop the columns

Data Transformation:

Encode Categorical Variables:

Identify categorical columns:

```
# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
categorical_cols
```

Perform label encoding

Data Analysis

Data Visualization:

Plot the distribution of labels

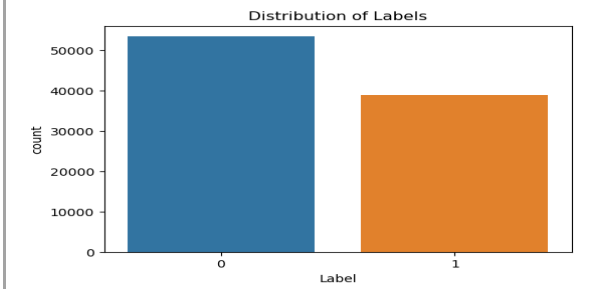
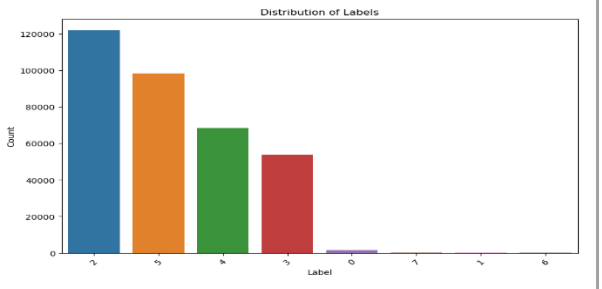
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 5 Data Visualization

Data Preparation:

Separate features and labels:

```
# Split Data into Features (X) and Labels (y)
X = df.drop(columns=['Label'])
y = df['Label']
```

Scale Features:

Normalization or standardization is necessary to ensure that features are on a similar scale, which can improve the performance of machine learning models.

We'll use standardization using Standard Scaler from scikit-learn.

- Standardization:

```
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Convert back to DataFrame for consistency
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)
```

Splitting data into Train-Test sets:

```
# Split the standardized data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Apply Smote:

```
# Apply SMOTE to the training data
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

III.5.MACHINE LEARNING AND DEEP LEARNING MODELS

For modulation we have two learning model

III.5.1. Machine Learning Models

III.5.2.THE EVALUATION OF THE MODELS

Evaluation metrics are used to assess the performance of the models. Here are some common metrics:

- **DecisionTree model**

Classification Report :

CTU13-CSV-Dataset-main

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10651
1	1.00	1.00	1.00	7792
accuracy			1.00	18443
macro avg	1.00	1.00	1.00	18443
weighted avg	1.00	1.00	1.00	18443

InSDN_DatasetCSV

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	279
1	1.00	1.00	1.00	30
2	1.00	1.00	1.00	24426
3	1.00	1.00	1.00	10604
4	1.00	1.00	1.00	13736
5	1.00	1.00	1.00	19671
6	1.00	0.75	0.86	4
7	1.00	1.00	1.00	28
accuracy			1.00	68778
macro avg	1.00	0.97	0.98	68778
weighted avg	1.00	1.00	1.00	68778

Table III 3 Classification Report

Interpretation:

Perfect Performance in Most Classes:
Both datasets show almost perfect precision, recall, and f1-scores in most classes.

Slight Deviation in InSDN_DatasetCSV for Class 6:
The second dataset has slightly lower performance for Class 6, which affects the macro average recall and f1-score.

Consistent High Accuracy:
Both datasets have high accuracy, with the first dataset showing perfect accuracy across all metrics.

Confusion Matrix

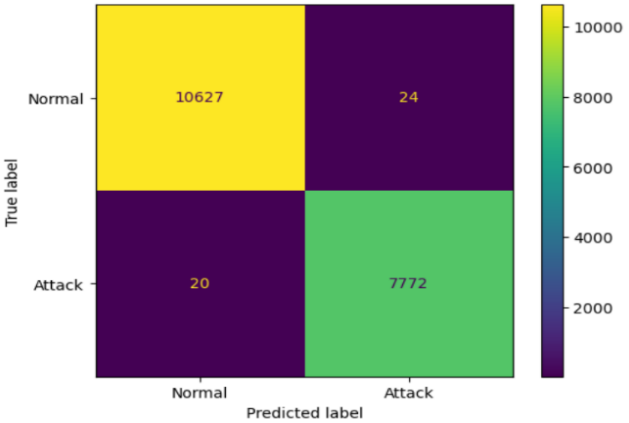
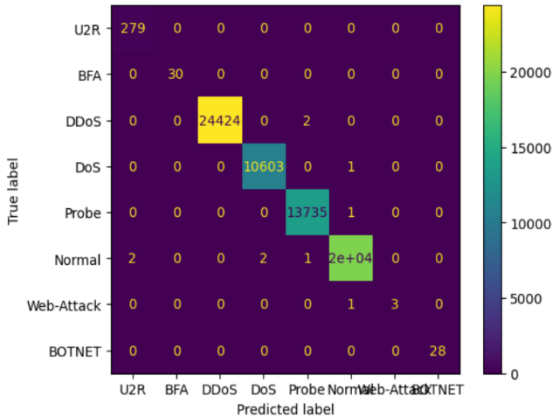
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 6 Confusion Matrix

Interpretation:

CTU13-CSV-Dataset-main Dataset:
The model is highly reliable, making very few errors. The confusion matrix confirms the high scores in the classification report, showing minimal misclassifications.
InSDN_DatasetCSV Dataset: The model also performs extremely well with very few errors. The confusion matrix highlights some misclassifications for the Web-Attack class, which aligns with the slightly lower recall for this class. Despite this, the model maintains high accuracy and robust performance across all other classes.
Both models demonstrate strong performance, with the first dataset's model showing near-perfect results and the second dataset's model showing exceptional performance with minor issues in classifying the Web-Attack class.

Roc curve

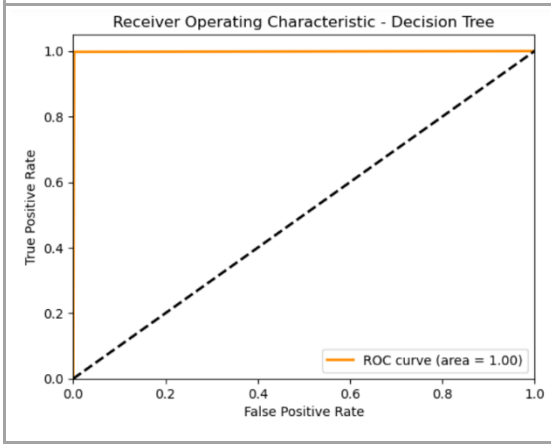
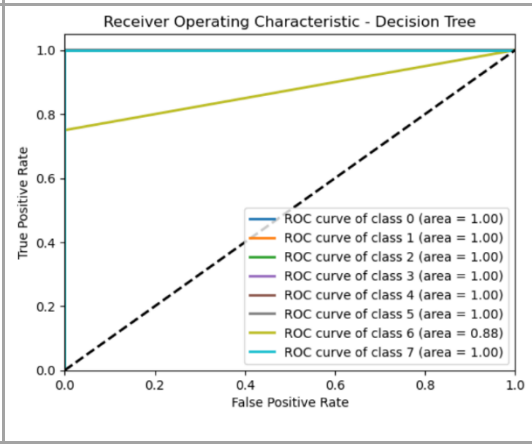
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 7 Roc cuve

Interpretation:

CTU13-CSV-Dataset-main:
The decision tree model has a perfect performance, possibly indicating an easy-to-classify dataset.

InSDN_DatasetCSV:
The decision tree performs perfectly for most classes but has a slight drop in performance for class 6. This could be due to class imbalance, noise, or inherent complexity in distinguishing class 6 from others.

Learning_Curve

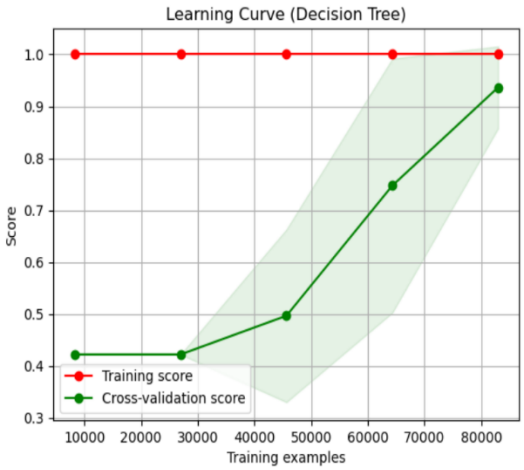
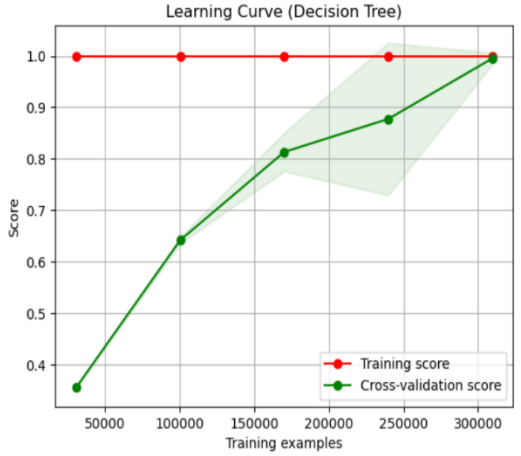
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 8 Learning curve

Interpretation:
CTU13-CSV-Dataset-main:
The Decision Tree model performs very well with enough data, reaching a cross-validation score of around 0.9. This indicates that while the model overfits initially, it generalizes well with more data.

InSDN_DatasetCSV:
The Decision Tree model improves with more data but still shows a significant gap between training and cross-validation scores, stabilizing around 0.85. This suggests that the dataset is more complex and may require more sophisticated modeling techniques.

- **Random_forest**

Classification Report

CTU13-CSV-Dataset-main	InSDN_DatasetCSV

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10651
1	1.00	1.00	1.00	7792
accuracy			1.00	18443
macro avg	1.00	1.00	1.00	18443
weighted avg	1.00	1.00	1.00	18443

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	279
1	1.00	1.00	1.00	30
2	1.00	1.00	1.00	24426
3	1.00	1.00	1.00	10604
4	1.00	1.00	1.00	13736
5	1.00	1.00	1.00	19671
6	1.00	0.75	0.86	4
7	1.00	1.00	1.00	28
accuracy			1.00	68778
macro avg	1.00	0.97	0.98	68778
weighted avg	1.00	1.00	1.00	68778

Table III 4 classification forest

Interpretation

Perfect Performance in First Dataset:

The random forest model performs perfectly on the CTU13-CSV-Dataset-main with no misclassifications.

Near-Perfect Performance in Second Dataset:

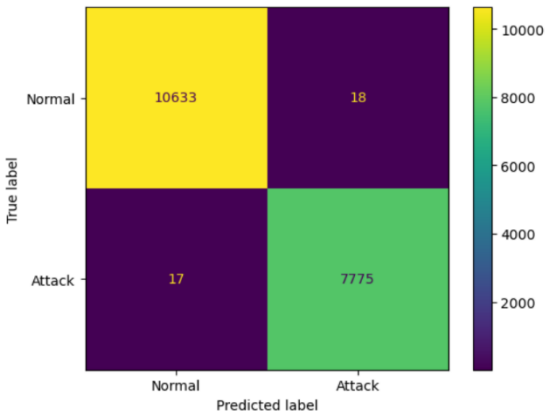
The model performs nearly perfectly on the InSDN_DatasetCSV, with a slight decrease in recall and f1-score for Class 6.

Impact of Class Imbalance:

The lower recall and f1-score for Class 6 in the second dataset might be due to class imbalance or inherent difficulty in distinguishing this class.

Confusion Matrix

CTU13-CSV-Dataset-main	InSDN_DatasetCSV



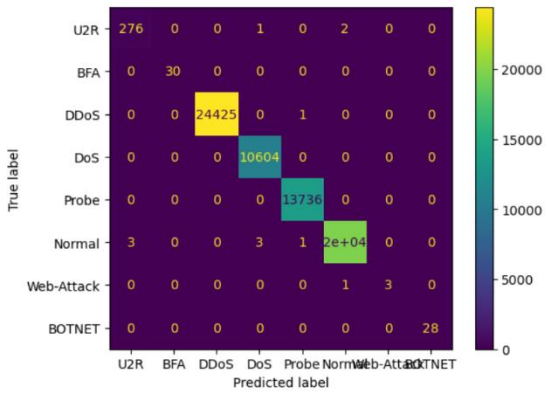


Figure III 9 Confusion matrix

Interpretation:

CTU13-CSV-Dataset-main :

The model is highly reliable with very few errors. The confusion matrix confirms the high scores in the classification report, showing minimal misclassifications. This small number of errors is negligible and does not significantly impact the overall performance.

InSDN_DatasetCSV:

The model performs extremely well with very few errors. The confusion matrix highlights a few misclassifications for the U2R, Normal, and Web-Attack classes. These minor issues align with the slightly lower recall and precision for the U2R and Normal classes but do not significantly impact the overall performance metrics.

Both models demonstrate strong performance, with the first dataset's model showing near-perfect results and the second dataset's model showing exceptional performance with minor issues in classifying specific classes. The updated confusion matrices provide a clear view of the few misclassifications present and confirm the overall high accuracy and reliability of both models.

Roc Curve

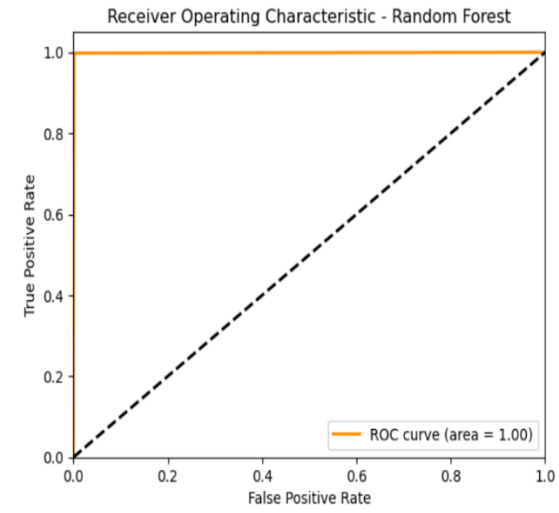
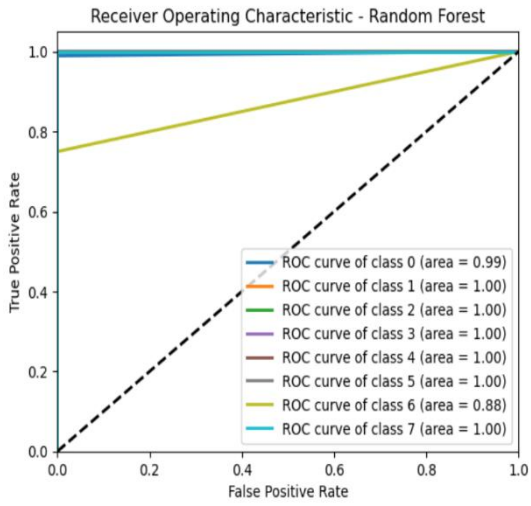
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 10 Roc curve forest

10- Interpretation:

CTU13-CSV-Dataset-main:

The Random Forest model, like the Decision Tree, shows perfect performance on this dataset. This suggests either the dataset is very easy to classify or there's a potential risk of overfitting.

InSDN_DatasetCSV:

The Random Forest shows high performance for most classes, with a slight drop for class 0 and a more significant drop for class 6, similar to the Decision Tree results. This consistency suggests that class 6 may inherently be more difficult to classify correctly.

Learning_Curve

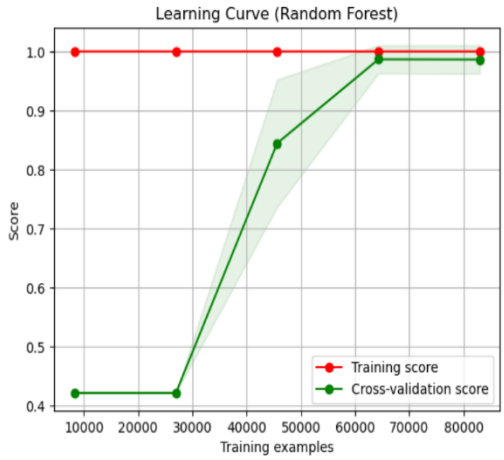
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 11Learning curve forest

11- Interpretation:

CTU13-CSV-Dataset-main:

The Random Forest model performs excellently with enough data, indicating that the dataset is likely easy to classify, but the initial low cross-validation score suggests that the model might need a certain amount of data to generalize well.

InSDN_DatasetCSV:

The Random Forest model improves with more data but still shows a gap between training and cross-validation performance. This suggests the presence of more complex patterns or variability in the data that require more sophisticated modeling or further data preprocessing.

- **KNN**

Classification Report :

CTU13-CSV-Dataset-main

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	10651
1	0.99	0.99	0.99	7792
accuracy			0.99	18443
macro avg	0.99	0.99	0.99	18443
weighted avg	0.99	0.99	0.99	18443

InSDN_DatasetCSV

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	279
1	1.00	1.00	1.00	30
2	1.00	1.00	1.00	24426
3	1.00	1.00	1.00	10604
4	1.00	1.00	1.00	13736
5	1.00	1.00	1.00	19671
6	0.75	0.75	0.75	4
7	0.90	1.00	0.95	28
accuracy			1.00	68778
macro avg	0.95	0.97	0.96	68778
weighted avg	1.00	1.00	1.00	68778

Table III 5 Classification Report KNN

12- Interpretation:

High Performance in CTU13-CSV-Dataset-main:

The KNN model performs exceptionally well on the CTU13-CSV-Dataset-main with only minor misclassifications.

Strong Performance in InSDN_DatasetCSV with Class 6 Issues:

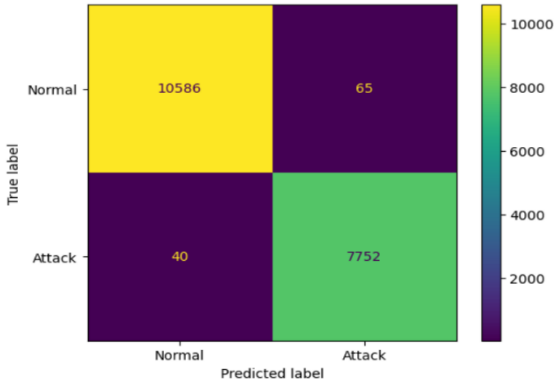
The model performs strongly on the InSDN_DatasetCSV, but struggles with Class 6, as indicated by the lower precision, recall, and f1-score.

Impact of Class Distribution:

The performance issues with Class 6 in the second dataset might be due to its low support (4 instances), suggesting potential class imbalance or difficulty in classifying this class correctly.

Confusion Matrix

CTU13-CSV-Dataset-main



InSDN_DatasetCSV

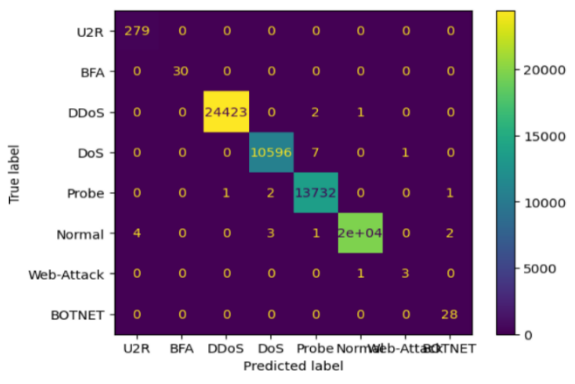


Figure III 12 Confusion Matrix

13- Interpretation:

CTU13-CSV-Dataset-main:

The model is highly reliable with some errors. The confusion matrix confirms the high scores in the classification report, showing several misclassifications but not significantly impacting the overall performance. The slight increase in errors compared to previous confusion matrices may slightly reduce the precision and recall but maintains a strong performance.

InSDN_DatasetCSV:

The model performs extremely well with very few errors. The confusion matrix highlights a few misclassifications for the U2R, Probe, Normal, and Web-Attack classes. These minor issues align with the slightly lower recall and precision for these classes but do not significantly impact the overall performance metrics.

Both models demonstrate strong performance, with the first dataset's model showing high accuracy and the second dataset's model showing exceptional performance with minor issues in classifying specific classes. The updated confusion matrices provide a clear view of the few misclassifications present and confirm the overall high accuracy and reliability of both models.

ROC Curve

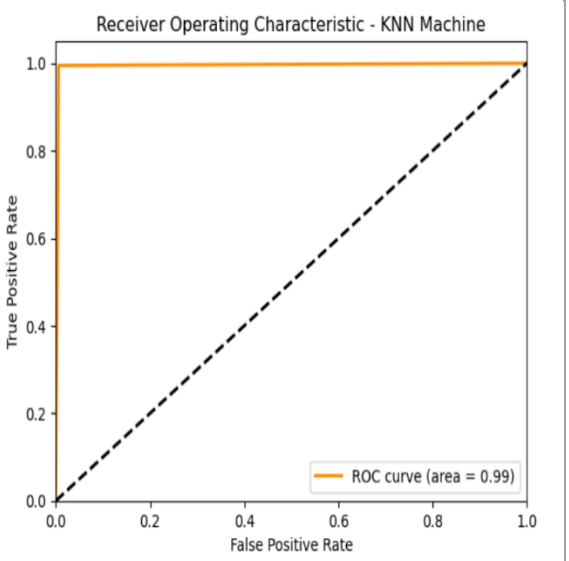
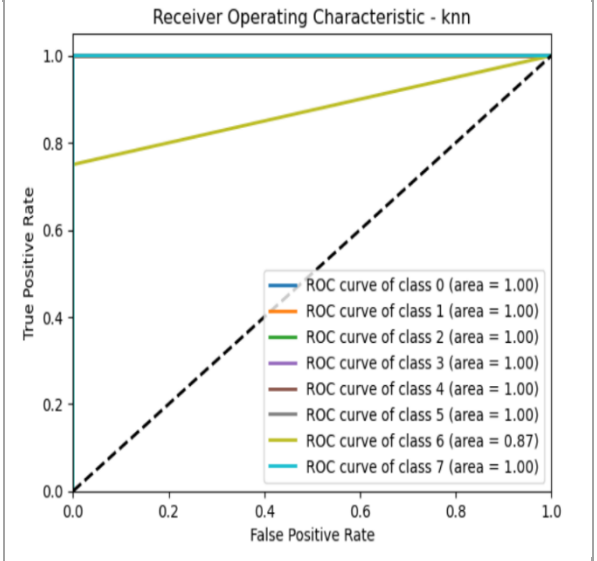
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 13 Roc curve KNN

14- Interpretation:

CTU13-CSV-Dataset-main:

The KNN model shows very high performance (AUC = 0.99), slightly lower than the perfect performance of Decision Tree and Random Forest models. This suggests the

dataset is easy to classify but also indicates that KNN might not be as overfitted as the other models.

InSDN_DatasetCSV:

The KNN model performs perfectly for most classes, but class 6 remains challenging with an AUC of 0.87, slightly lower than previous models. Class 0 shows perfect classification, indicating KNN's robustness for this class.

Learning_Curve

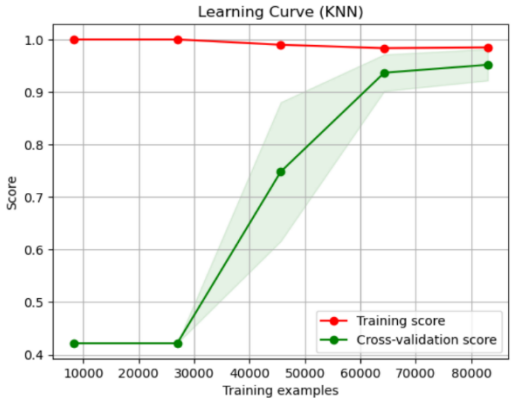
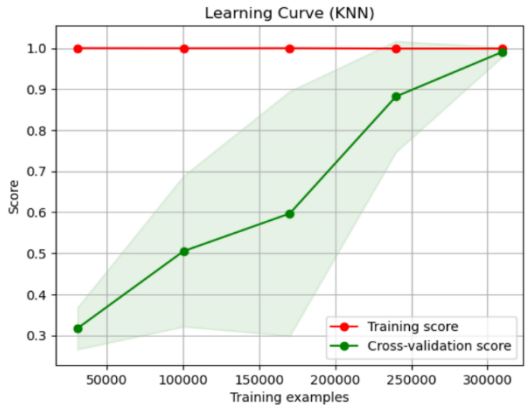
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 14 learning Curve KNN

15- Interpretation:

CTU13-CSV-Dataset-main

- The KNN model performs exceptionally well with a large number of training examples.
- The gap between training and cross-validation scores diminishes as more data is used, indicating better generalization.

InSDN_DatasetCSV

- The KNN model shows a significant improvement in cross-validation scores as the number of training examples increases.
- Despite achieving high training scores, the cross-validation score does not fully converge to 1.0, suggesting some overfitting or data complexity.

Accuracy:

	CTU13-CSV-Dataset-main	InSDN_DatasetCSV
Decision_Tree	0.9976142709971263	0.999854604670098
Random_Forest	0.9981022610204413	0.9998255256041176
KNN	0.994306783061324	0.9996219721422548

Table III 6 Accuracy

16- Interpretation:

High Accuracy Across All Models and Datasets:

All three models achieve very high accuracy on both datasets, with scores nearing 1.0, indicating excellent performance overall.

Slight Differences Among Models:

The Random Forest model has slightly higher accuracy than the Decision Tree and KNN models in both datasets, suggesting it might be slightly better at handling the classification task for these datasets.

KNN Performance:

KNN also performs exceptionally well but tends to have slightly lower accuracy compared to Decision Trees and Random Forests.

III.6. DEEP LEARNING

Deep learning is a subset of machine learning that involves neural networks with many layers, called deep neural networks. It mimics the human brain's structure and function to analyze data and make decisions.

III6.1. Multi-Layer Perceptron (MLP)

MLPs are neural networks with multiple layers of perceptrons, useful for tasks requiring supervised learning.

MLP Architecture:

Layers: The model consists of two MLP layers

Activation Function: Softmax and ReLU.

Regularization: Dropout layers between MLP layers.

Training Configuration:

Optimizer: Adam with a learning rate of 0.001.

Loss Function: sparse_categorical_crossentropy.

Metrics: Accuracy.

Epochs: 50.

Batch Size: 32.

Training and Validation Results:

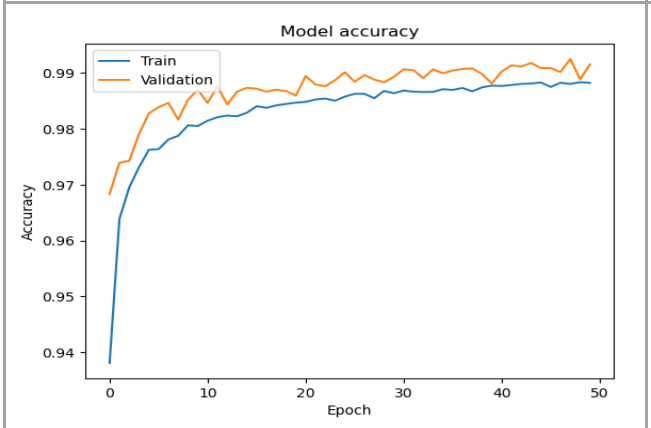
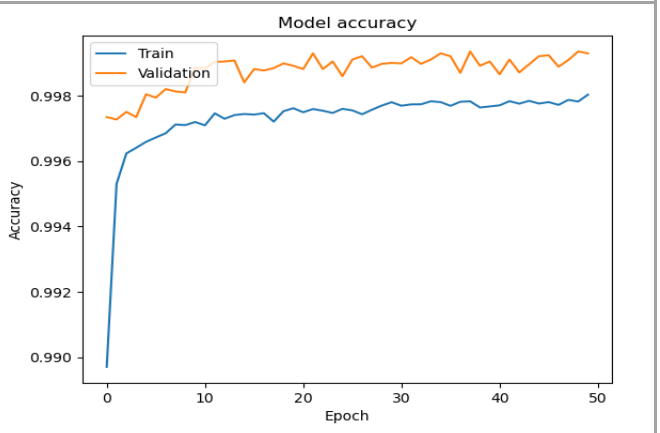
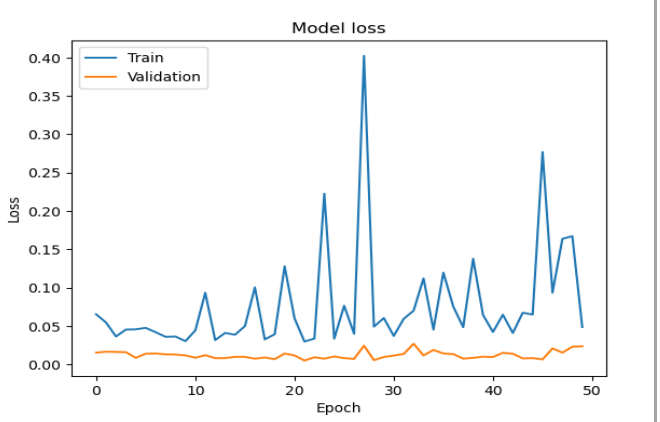
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	
	

Figure III 15 **Training and Validation Results** MLP

CTU13-CSV-Dataset-main

- The training and validation accuracy curves are closely aligned, indicating that the model is learning well
- The training and validation loss curves show a steep decline initially, indicating rapid learning.

InSDN_DatasetCSV

- The accuracy for both training and validation starts at a lower point compared to the first dataset but quickly improves.
- The loss for training decreases initially but shows significant fluctuations. Validation loss also decreases but maintains a more stable trajectory compared to the training loss.

Classification Report:

CTU13-CSV-Dataset-main	InSDN_DatasetCSV

MLP Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	10651
1	0.99	0.99	0.99	7792
accuracy			0.99	18443
macro avg	0.99	0.99	0.99	18443
weighted avg	0.99	0.99	0.99	18443

MLP Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.97	279
1	1.00	1.00	1.00	30
2	1.00	1.00	1.00	24426
3	1.00	1.00	1.00	10604
4	1.00	1.00	1.00	13736
5	1.00	1.00	1.00	19671
6	0.50	0.25	0.33	4
7	1.00	0.86	0.92	28
accuracy			1.00	68778
macro avg	0.93	0.88	0.90	68778
weighted avg	1.00	1.00	1.00	68778

Table III 7 Classification Report MLP

CTU13-CSV-Dataset-main

17- Both classes (0 and 1) have very high precision, recall, and F1-scores (close to 1.00).

InSDN_DatasetCSV

18- Most classes have perfect precision, recall, and F1-scores (1.00). This indicates that the model is highly effective for these classes.

Confusion Matrix:

CTU13-CSV-Dataset-main	InSDN_DatasetCSV

MLP Confusion Matrix:

```
[[10540   111]
 [   45  7747]]
```

MLP Confusion Matrix:

```
[[  269     0     0     0     0     9     1     0]
 [    0    30     0     0     0     0     0     0]
 [    0     0 24425     0     1     0     0     0]
 [    0     0     0 10599     2     3     0     0]
 [    0     0     9     1 13726     0     0     0]
 [    5     0     3     6     2 19655     0     0]
 [    0     0     0     0     0     3     1     0]
 [    3     0     0     0     0     1     0    24]]
```

Table III 8Confusion Matrix MLP

CTU13-CSV-Dataset-main

The confusion matrix reflects the model's high accuracy, with very few misclassifications

InSDN_DatasetCSV

The confusion matrix shows strong performance for most classes, with almost perfect classification for major classes, with very few misclassifications.

(ROC) Curve

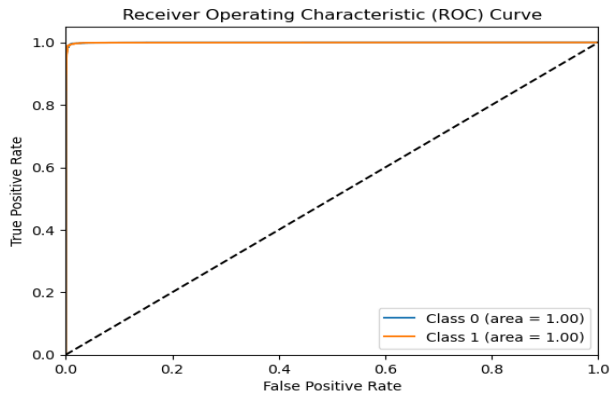
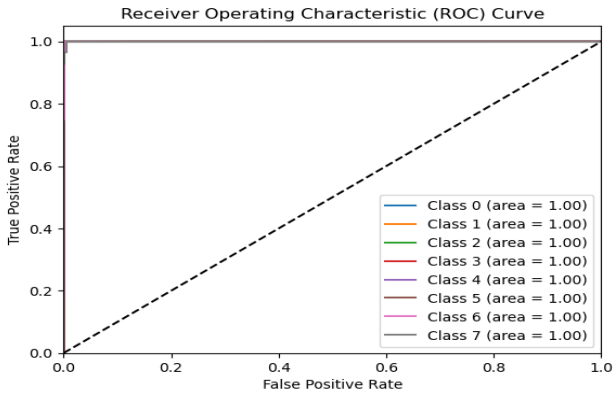
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 16 ROC Curve MLP

III6.2.Deep Learning with 2D CNN

CNNs are designed to process and analyze grid-like data structures, by using convolutional layers to detect spatial hierarchies and patterns.

CNN Architecture:

- Layers: two Convolutional layers for feature extraction, followed by tow pooling layers to reduce dimensionality .
- Activation Function: ReLU and Softmax.
- Regularization: Dropout layers to prevent overfitting.

Training Configuration:

- Loss Function: sparse_categorical_crossentropy.
- Metrics: Accuracy.
- Epochs: 50.
- Batch Size: 32.

Training and Validation Results:

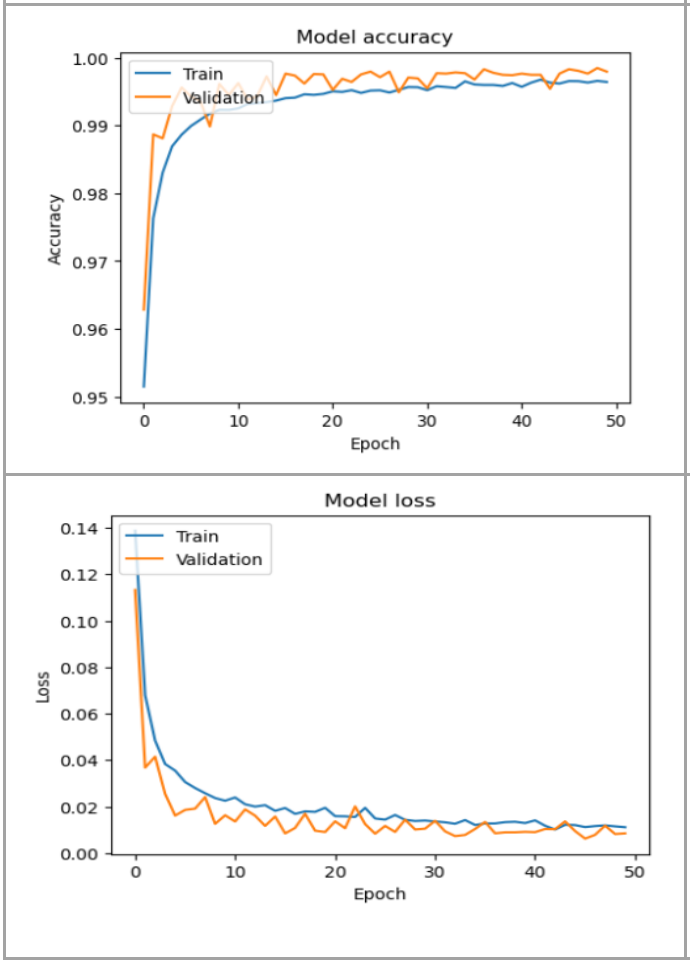
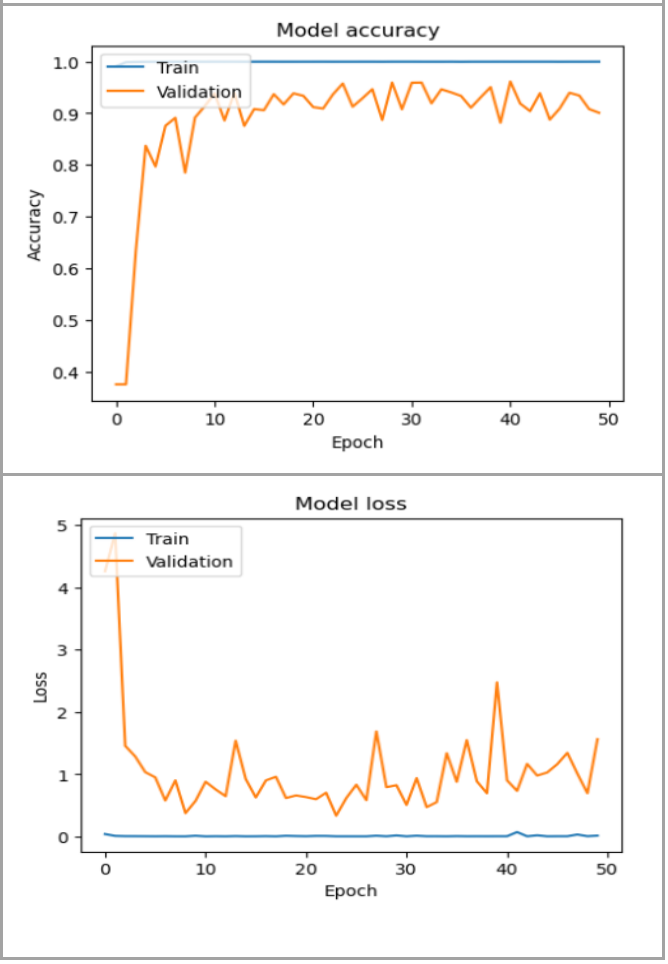
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 17Training and Validation CNN

CTU13-CSV-Dataset-main

- Both accuracy and loss plots indicate excellent performance.

InSDN_DatasetCSV

- The training accuracy is high and stable, indicating effective learning. And validation accuracy shows more fluctuations.
- Training loss remains low, but validation loss has more variability

Classification Report:

CTU13-CSV-Dataset-main	InSDN_DatasetCSV
CNN Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 10651 1 1.00 1.00 1.00 7792 accuracy 1.00 18443 macro avg 1.00 1.00 1.00 18443 weighted avg 1.00 1.00 1.00 18443	CNN Classification Report: precision recall f1-score support 0 0.98 1.00 0.99 279 1 1.00 1.00 1.00 30 2 1.00 1.00 1.00 24426 3 1.00 1.00 1.00 10604 4 1.00 1.00 1.00 13736 5 1.00 1.00 1.00 19671 6 1.00 0.75 0.86 4 7 1.00 0.89 0.94 28 accuracy 1.00 68778 macro avg 1.00 0.96 0.97 68778 weighted avg 1.00 1.00 1.00 68778

Table III 9Classification Report CNN

CTU13-CSV-Dataset-main

- The CNN model demonstrates perfect classification performance across all metrics, indicating it is very well-suited for this dataset.

InSDN_DatasetCSV

- The CNN model performs exceptionally well, with nearly perfect scores for most classes.
- Some misclassifications occur in classes with fewer instances (Class 6 and Class 7)

Confusion Matrix:

CTU13-CSV-Dataset-main	InSDN_DatasetCSV
CNN Confusion Matrix: [[10616 35] [24 7768]]	CNN Confusion Matrix: [[279 0 0 0 0 0 0 0] [0 30 0 0 0 0 0 0] [0 0 24425 0 0 1 0 0] [1 0 0 10602 0 1 0 0] [0 0 0 0 13736 0 0 0] [5 0 3 3 0 19660 0 0] [0 0 0 0 0 1 3 0] [1 0 0 0 0 2 0 25]]

Table III 10Confusion Matrix CNN

CTU13-CSV-Dataset-main And InSDN_DatasetCSV Datasets

-Both models shows high accuracy and precision, suggesting effective learning and good generalization on this dataset.

(ROC) Curve

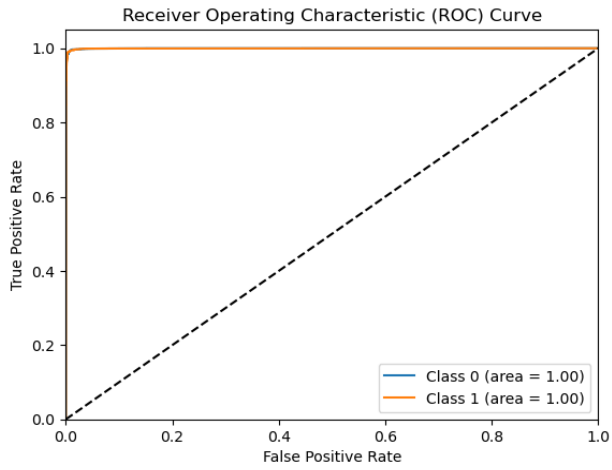
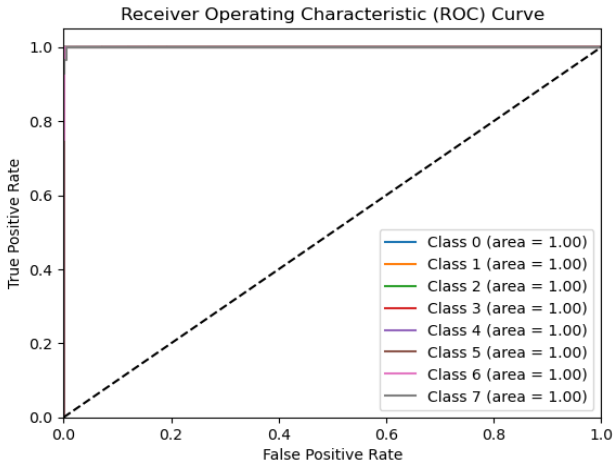
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 18 ROC Curve CNN

III.6.3. LSTM (RNN) Model

LSTMs are a type of recurrent neural network that capture long-term dependencies in sequential data, ideal for analyzing time-series network data to detect intrusions in SDN.

LSTM Architecture:

- Layers: The model consists of two LSTM layers
- Activation Function: Softmax and ReLU.
- Regularization: Dropout layers between LSTM layers.
-

Training Configuration:

- Optimizer: Adam with a learning rate of 0.001.
- Loss Function: sparse_categorical_crossentropy.
- Metrics: Accuracy.
- Epochs: 50.
- Batch Size: 32.

Training and Validation Results:

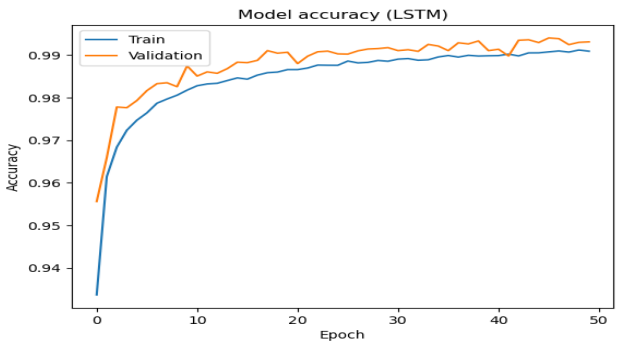
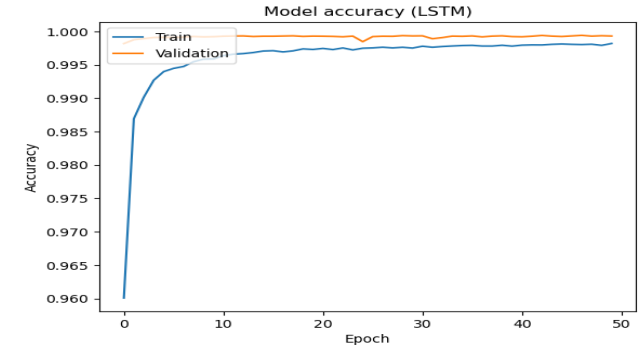
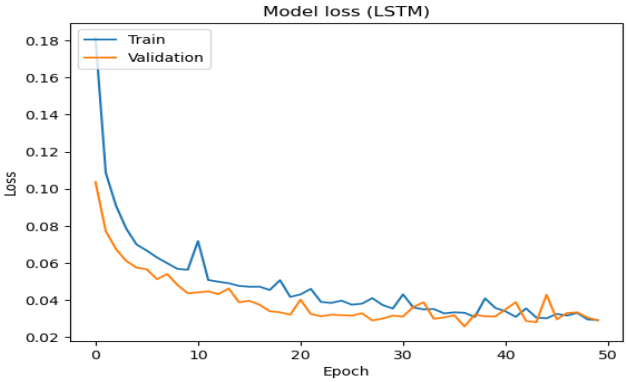
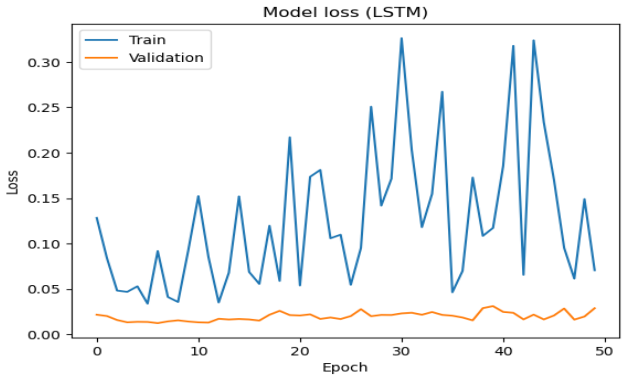
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	
	

Figure III 19Training and Validation LSTM

CTU13-CSV-Dataset-main And InSDN_DatasetCSV

- The training accuracy and validation accuracy are both very high.
- The training loss and validation loss also show a trend of decreasing over epochs.

Classification Report:

CTU13-CSV-Dataset-main

lstm Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	10651
1	0.99	1.00	0.99	7792
accuracy			0.99	18443
macro avg	0.99	0.99	0.99	18443
weighted avg	0.99	0.99	0.99	18443

InSDN_DatasetCSV

Classification Report (LSTM):

	precision	recall	f1-score	support
BFA	0.98	1.00	0.99	279
BOTNET	1.00	1.00	1.00	30
DDoS	1.00	1.00	1.00	24426
DoS	1.00	1.00	1.00	10604
Normal	1.00	1.00	1.00	13736
Probe	1.00	1.00	1.00	19671
U2R	1.00	0.50	0.67	4
Web-Attack	0.80	1.00	0.89	28
accuracy			1.00	68778
macro avg	0.97	0.94	0.94	68778
weighted avg	1.00	1.00	1.00	68778

Table III 11CLASSIFICATION REPORT LSTM

CTU13-CSV-Dataset-main

- The model performs well for both classes (0 and 1) with very high precision, recall, and F1-scores (all at or very close to 1.00).

InSDN_DatasetCSV

- The model performs exceptionally well on most classes, with perfect scores (precision, recall, and F1-score) for BOTNET, DDoS, DoS, Normal, and Probe. However, it shows slightly lower performance for the U2R and Web-Attack classes.

Confusion Matrix:

CTU13-CSV-Dataset-main	InSDN_DatasetCSV
lstm Confusion Matrix: [[10557 94] [34 7758]]	Confusion Matrix (LSTM): [[278 0 0 0 0 1 0 0] [0 30 0 0 0 0 0 0] [0 0 24424 0 1 0 0 1] [1 0 0 10596 6 1 0 0] [0 0 9 0 13725 2 0 0] [6 0 4 7 1 19647 0 6] [0 0 0 0 0 2 2 0] [0 0 0 0 0 0 0 28]]

Table III 12Confusion Matrix LSTM

CTU13-CSV-Dataset-main

The model shows high accuracy and a very low number of misclassifications, which aligns with the high precision, recall, and F1-scores observed in the classification report.

InSDN_DatasetCSV

The model exhibits excellent performance, with nearly perfect classification for most classes. Misclassifications are very low and occur.

ROC Curve

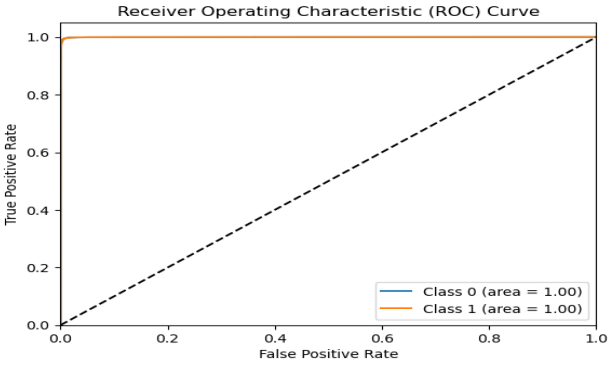
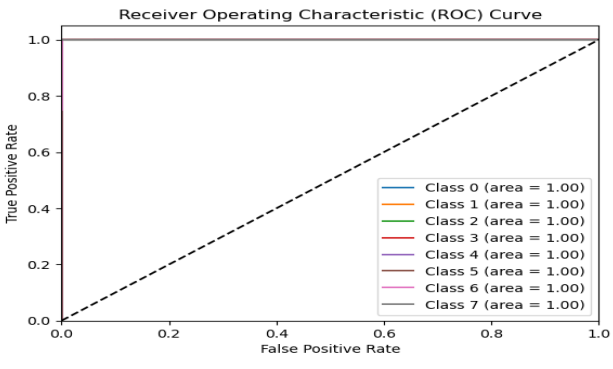
CTU13-CSV-Dataset-main	InSDN_DatasetCSV
	

Figure III 20 ROC Curve LSTM

Accuracy:

	CTU13-CSV-Dataset-main	InSDN_DatasetCSV
MLP	99.15%	99.93%
CNN	99.68%	99.97%
LSTM	99.68%	99.93%

Table III 13Accuracy

III.7. CONCLUSION

In this chapter, we explored an approach to enhancing Intrusion Detection Systems (IDS) within Software-Defined Networking (SDN) environments Applying some ML and DL algorithms.

To assess the performance of the proposed algorithms, we compared the same ML and DL models on two datasets: one distinguishing between normal and attack traffic, and the other categorizing specific attack types such as DDoS and DoS. This comparison helps evaluate the effectiveness and robustness of the models across different types of data, demonstrating their capability to accurately identify and classify various cyber threats

Conclusion

In this project, we have demonstrated a comprehensive methodology for enhancing Intrusion Detection Systems (IDS) within Software-Defined Networking (SDN) environments using various machine learning (ML) and deep learning (DL) models. Our approach encompassed meticulous data collection, preprocessing, and analysis to ensure the quality and relevance of the dataset for training the models.

We implemented and evaluated several ML and DL algorithms, including Decision Trees, Knn, Random Forest, MLP, CNN, LSTM. These models were assessed based on key performance metrics such as accuracy, precision, recall, and F1-score, offering valuable insights into their respective strengths in the context of IDS for SDN.

Despite achieving promising results, we identified significant challenges related to data privacy, scalability, and the need for continuous model updates in dynamic network environments. To address these challenges, we proposed the integration of Federated Learning (FL) as a future direction. FL provides a decentralized approach to model training, preserving data privacy by keeping data localized and only sharing model updates. This method enhances scalability and enables continuous and dynamic updating of intrusion detection models.

In conclusion, while our work with ML and DL models has substantially advanced the state of IDS in SDN, the incorporation of Federated Learning has the potential to further revolutionize the field. By leveraging FL, we enhance the capabilities of AI-driven IDS in SDN environments by enabling collaborative model training across distributed network nodes, accelerating model convergence, and ensuring that sensitive network information remains localized and protected.

Bibliography

[1] J. Wiley, « Software Defined Networking for Dummies », 2015.
[2] « 5e1b00c9d7ae2.pdf ».
[3] « Jankowski et Amanowicz - Intrusion Detection in Software Defined Networks wi.pdf ».
[4] « AichaouiAnis_AitbelkacemYanis.pdf ».
[5] « Bannour - Extending SDN control to large-scale networks Tax.pdf ».
[6] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, et M. Ghogho, « Intrusion Detection in SDN-Based Networks: Deep Recurrent Neural Network Approach », in *Deep Learning Applications for Cyber Security*, M. Alazab et M. Tang, Éd., in Advanced Sciences and Technologies for Security Applications. , Cham: Springer International Publishing, 2019, p. 175-195. doi: 10.1007/978-3-030-13057-2_8.
[7] « Kreutz et al. - 2015 - Software-Defined Networking A Comprehensive Surve.pdf ».
[8] « Kreutz et al. - 2014 - Software-Defined Networking A Comprehensive Surve.pdf ».
[9] « Blial et al. - 2016 - An Overview on SDN Architectures with Multiple Con.pdf ».
[10] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, et M. Shaheed, « SDN Security Review: Threat Taxonomy, Implications, and Open Challenges », *IEEE Access*, vol. 10, p. 45820-45854, 2022, doi: 10.1109/ACCESS.2022.3168972.
[11] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, et M. Shaheed, « SDN Security Review: Threat Taxonomy, Implications, and Open Challenges », *IEEE Access*, vol. 10, p. 45820-45854, 2022, doi: 10.1109/ACCESS.2022.3168972.
[12] N. M. Abd Elazim, M. A. Sobh, et A. M. Bahaa-Eldin, « Software Defined Networking: Attacks and Countermeasures », in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, Cairo, Egypt: IEEE, déc. 2018, p. 555-567. doi: 10.1109/ICCES.2018.8639429.
[13] A. N. Alhaj et N. Dutta, « Analysis of Security Attacks in SDN Network: A Comprehensive Survey », in *Contemporary Issues in Communication, Cloud and Big Data Analytics*, vol. 281, H. K. D. Sarma, V. E. Balas, B. Bhuyan, et N. Dutta, Éd., in Lecture Notes in Networks and Systems, vol. 281. , Singapore: Springer Singapore, 2022, p. 27-37. doi: 10.1007/978-981-16-4244-9_3.
[14] Y. Tseng, « Securing network applications in software defined networking ».
[15] M. Dabbagh, B. Hamdaoui, M. Guizani, et A. Rayes, « Software-defined networking security: pros and cons », *IEEE Commun. Mag.*, vol. 53, nº 6, p. 73-79, juin 2015, doi: 10.1109/MCOM.2015.7120048.
[16] A. Pradhan et R. Mathew, « Solutions to Vulnerabilities and Threats in Software Defined Networking (SDN) », *Procedia Computer Science*, vol. 171, p. 2581-2589, 2020, doi: 10.1016/j.procs.2020.04.280.
[17] B. Lin, X. Zhu, et Z. Ding, « Research on the Vulnerability of Software Defined Network », in *Proceedings of the 3rd Workshop on Advanced Research and Technology in Industry (WARTIA 2017)*, Guilin, China: Atlantis Press, 2017. doi: 10.2991/wartia-17.2017.49.

[18] « Associate Professor, Department of Computer Science Government Arts College, Thuvakudimalai, Tiruchirappalli, India et Kumar - 2017 - INTRUSION DETECTION SYSTEMS A REVIEW.pdf ».
[19] « Attack sophistication vs. intruder technical knowledge | Download Scientific Diagram ». Consulté le: 9 juin 2024. [En ligne]. Disponible sur: https://www.researchgate.net/figure/Attack-sophistication-vs-intruder-technical-knowledge_fig2_326786055
[20] Associate Professor, Department of Computer Science Government Arts College, Thuvakudimalai, Tiruchirappalli, India et D. A. Kumar, « INTRUSION DETECTION SYSTEMS: A REVIEW », *ijarcs*, vol. 8, nº 8, p. 356-370, août 2017, doi: 10.26483/ijarcs.v8i8.4703.
[21] « 4115ijsptm04.pdf ».
[22] « IntrusionDetectionTechniquesAReview.pdf ».
[23] « 4115ijsptm.pdf ».
[24] « Warwick - 2012 - Artificial intelligence the basics.crdownload ».
[25] « Shalev-Shwartz et Ben-David - 2014 - Understanding Machine Learning From Theory to Alg.pdf ».
[26] « Theobald - Machine Learning For Absolute Beginners.pdf ».
[27] « Mu - Introduction to Machine Learning with Python.pdf ».
[28] « Prasanna - Machine Learning with Python.pdf ».
[29] « Raschka et Mirjalili - Python machine learning machine learning and deep.pdf ».
[30] « Ketkar - 2017 - Deep Learning with Python.pdf ».
[31] « Chollet - 2021 - Deep learning with Python.pdf ».
[32] « Neural-Networks.pdf ».
[33] « Hosseini et al. - 2020 - Deep Learning Architectures.pdf ».
[34] « Full Text PDF ». Consulté le: 9 juin 2024. [En ligne]. Disponible sur: https://www.researchgate.net/profile/A-Sufian/publication/337401161_Fundamental_Concepts_of_Convolutional_Neural_Network/links/5ed612a7299bf1c67d3292e6/Fundamental-Concepts-of-Convolutional-Neural-Network.pdf
[35] « Polamuri - INTRODUCTION TO DEEP LEARNING.pdf ».
[36] « bernd klein python and machine learning_a4.pdf ».

الملخص
في مجال أمن الشبكات سريع التغير، تبرز الشبكات المحددة بالبرمجيات (SDN) كعامل تغيير من خلال توفير التحكم المركزي وقابلية برمجة موارد الشبكة، مما يؤدي إلى إدارة أكثر مرونة وكفاءة. ومع ذلك، فإن هذه المركزية تجلب أيضًا نقاط ضعف جديدة، مما يجعل بيئات SDN أهدافًا جذابة للهجمات الإلكترونية. لمعالجة هذه المخاطر، تعد أنظمة اكتشاف التطفل (IDS) ضرورية، حيث تراقب وتحلل حركة المرور على الشبكة للكشف عن الأنشطة الضارة والاستجابة لها.
تواجه أنظمة اكتشاف التطفل التقليدية (IDS) التي تعتمد على جمع البيانات ومعالجتها مركزيًا مشكلات كبيرة تتعلق بالخصوصية وقابلية التوسع. مع تزايد تعقيد بيئات الشبكة وزيادة أحجام البيانات، غالبًا ما تفشل هذه الأنظمة في تلبية المتطلبات الحديثة، مما يؤدي إلى نقاط فشل واحدة محتملة ومخاطر خصوصية متزايدة. يعالج هذا المشروع هذه التحديات من خلال التحقيق في استخدام نماذج التعلم الآلي (ML) والتعلم العميق (DL) لتعزيز أنظمة اكتشاف التطفل داخل بيئات الشبكات المحددة بالبرمجيات (SDN). يمكن لأساليب التعلم الآلي والتعلم العميق أن تعزز بشكل كبير من دقة وكفاءة اكتشاف التطفل من خلال تحليل مجموعات البيانات الشاملة واكتشاف الأنماط التي تشير إلى أنشطة ضارة.
يركز هذا المشروع على تحليل شامل لمجموعة البيانات، وتطبيق نماذج التعلم الآلي والتعلم العميق المتنوعة، وتقييم أدائها في اكتشاف التطفلات داخل بيئات الشبكات المحددة بالبرمجيات (SDN). الهدف الرئيسي هو تحسين فعالية أنظمة اكتشاف التطفل (IDS) في SDN من خلال الاستفادة من النماذج المتقدمة، والتعرف على نقاط القوة والضعف لديها، والمساهمة في إنشاء حلول اكتشاف تطفل أكثر قوة وقابلية للتطوير والحفاظ على الخصوصية. ستعمل الرؤى التي تم الحصول عليها في النهاية على تعزيز أمان بيئات SDN.

الكلمات الرئيسية: IDS، SND، التصنيف، التعلم الآلي، التعلم العميق، التقييم.

Resumé:
Dans le domaine en constante évolution de la sécurité des réseaux, le Software-Defined Networking (SDN) se distingue par son rôle de pionnier en offrant un contrôle centralisé et une programmabilité des ressources réseau, ce qui conduit à une gestion plus flexible et plus efficace. Cependant, cette centralisation entraîne également de nouvelles vulnérabilités, faisant des environnements SDN des cibles attrayantes pour les cyberattaques. Pour faire face à ces risques, les systèmes de détection d'intrusion (IDS) sont essentiels, car ils surveillent et analysent le trafic réseau pour détecter et répondre aux activités malveillantes.
Les systèmes de détection d'intrusion (IDS) traditionnels qui dépendent de la collecte et du traitement centralisés des données rencontrent des problèmes majeurs de confidentialité et d'évolutivité. À mesure que les environnements réseau deviennent plus complexes et que les volumes de données augmentent, ces systèmes ne parviennent souvent pas à répondre aux exigences modernes, ce qui entraîne des points de défaillance uniques potentiels et des risques accrus pour la confidentialité. Ce projet s'attaque à ces défis en étudiant l'utilisation de modèles d'apprentissage automatique (ML) et d'apprentissage profond (DL) pour améliorer les IDS dans les environnements de réseau défini par logiciel (SDN). Les méthodes ML et DL peuvent considérablement améliorer la précision et l'efficacité de la détection d'intrusion en analysant de vastes ensembles de données et en détectant des modèles suggérant des activités malveillantes.
Ce projet se concentre sur une analyse complète des ensembles de données, en appliquant divers modèles d'apprentissage automatique (ML) et d'apprentissage profond (DL), et en évaluant leurs performances dans la détection des intrusions dans les environnements de réseau défini par logiciel (SDN). L'objectif principal est d'améliorer l'efficacité des systèmes de détection d'intrusion (IDS) dans les SDN en utilisant des modèles avancés, en reconnaissant leurs forces et leurs faiblesses et en contribuant à la création de solutions IDS plus robustes, évolutives et préservant la confidentialité. Les informations obtenues amélioreront à terme la sécurité des environnements SDN.

Mots-clés : IDS, SND , classification, apprentissage automatique, apprentissage profond, évaluation.