PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF OUM EL BOUAGHI
FACULTY OF EXACT SCIENCES AND NATURAL AND LIFE SCIENCES
Department of Mathematics and Computer Science
Research Laboratory on Computer Science's Complex Systems ReLa(CS)^2
MoVéVaSiS Research Team

MASTER THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER IN COMPUTER SCIENCE
SPECIALTY : DISTRIBUTED ARCHITECTURES

Intrusion Detection in Wireless Sensor Networks using Deep Learning

Prepared by :

TOUFIK HAMEL AND BASSEM MAKHLOUF

Presented for defense in a public examination on: June 24th 2023

SUPERVISED BY :

PROF. ABDELHABIB BOUROUIS
UNIVERSITY OF OUM EL BOUAGHI

In front of the jury composed of :

DR. MARIR TOUFIK	UNIVERSITY OF OUM EL BOUAGHI	PRESIDENT
PR. BOUROUIS ABDELHABIB	UNIVERSITY OF OUM EL BOUAGHI	SUPERVISOR
MR. BOULKAMH CHOUAIB	UNIVERSITY OF OUM EL BOUAGHI	EXAMINER

Academic year 2022-2023

Contents

Contents vi

List of Figures vii

List of Tables ix

List of listings xi

Dedications 3

Acknowledgements 5

Abstract 7

Résumé 9

Moulakhas 11

General introduction 13

1 Wireless Sensor Networks 17
- 1.1 Introduction . . . 19
- 1.2 Sensor nodes . . . 19
  - 1.2.1 Components of a sensor node . . . 20
  - 1.2.2 Sensor types . . . 20
- 1.3 Wireless Sensor Network . . . 21
  - 1.3.1 Network Architecture of WSNs . . . 21
  - 1.3.2 WSN routing protocols . . . 22
    - 1.3.2.1 Direct transmission protocol . . . 22
    - 1.3.2.2 Minimum transfer energy protocols . . . 22
    - 1.3.2.3 Clustering protocol . . . 23
    - 1.3.2.4 LEACH (Low Energy Adaptive Clustering Hierarchy) . . . 23
    - 1.3.2.5 Stable Election Protocol . . . 23
    - 1.3.2.6 Hierarchical Cluster-Based Routing (HCR) Protocol . . . 23
    - 1.3.2.7 Genetic Algorithm . . . 23
  - 1.3.3 WSNs topologies . . . 24
    - 1.3.3.1 Point-to-point topology . . . 24
    - 1.3.3.2 Bus topology . . . 24
    - 1.3.3.3 Tree topology . . . 25
    - 1.3.3.4 Star Topology . . . 25
    - 1.3.3.5 Ring topology . . . 25
    - 1.3.3.6 Mesh topology . . . 26

1.3.3.7 Circular topology . . . 27
1.3.3.8 Grid topology . . . 27
1.4 Types of WSNs . . . 27
1.5 WSN applications . . . 28
1.6 WSNs constraints . . . 30
1.6.1 Unreliable Communication . . . 30
1.6.2 Energy constraints . . . 30
1.6.3 Memory limitations . . . 30
1.6.4 Higher latency in communication . . . 30
1.6.5 Unattended operation of networks . . . 31
1.7 Security in WSNs . . . 31
1.8 Conclusion . . . 32

2 Deep learning **35**
2.1 Introduction . . . 37
2.2 Definition . . . 37
2.3 Machine learning . . . 37
2.3.1 Machine learning life cycle . . . 38
2.3.2 Machine learning applications . . . 39
2.4 Basic types of machine learning . . . 40
2.4.1 Supervised learning . . . 40
2.4.2 Unsupervised learning . . . 40
2.4.3 Semi-supervised learning . . . 41
2.4.4 Reinforcement learning . . . 42
2.4.5 Transfer learning . . . 42
2.5 Deep learning . . . 43
2.5.1 Introduction . . . 43
2.5.2 Definition . . . 43
2.5.3 Applications of deep learning . . . 43
2.5.4 Advantages of Deep Learning . . . 44
2.6 Common issues . . . 44
2.6.1 Overfitting . . . 44
2.6.2 Underfitting . . . 45
2.6.3 Insufficient Training Data (Data scarcity) . . . 45
2.6.4 Nonrepresentative Training Data . . . 46
2.6.5 Poor-Quality Data . . . 46
2.6.6 Irrelevant Features . . . 46
2.6.7 Vanishing gradients . . . 46
2.6.8 Exploding gradients . . . 46
2.6.9 Computational complexity . . . 46
2.6.10 Interpretability . . . 46
2.7 Neural networks . . . 47
2.7.1 Feed-Forward network . . . 47
2.7.2 Deep neural network (DNN) . . . 47
2.8 Deep Learning Architectures . . . 48
2.8.1 Discriminative Architectures . . . 48
2.8.1.1 Convolutional neural networks . . . 48
2.8.1.2 Recurrent Neural Network (RNN) and Recursive Neural Network (RvNN) . . . 49
2.8.2 Generative Architectures . . . 50
2.8.2.1 Auto-Encoder (AE) . . . 50
2.8.2.2 Restricted Boltzmann Machine (RBM) . . . 51

2.8.2.3 Deep Belief Network (DBN) . . . 52
2.8.3 Hybrid Architectures . . . 52
2.8.3.1 Generative Adversarial Network (GAN) . . . 53
2.9 Conclusion . . . 53

3 Intrusion detection systems (IDS) 57
3.1 Introduction . . . 58
3.2 Network intrusion . . . 58
3.3 Types of Wireless Sensor Networks attacks . . . 59
3.3.1 Active Attacks . . . 60
3.3.2 Passive Attacks: . . . 63
3.4 Detection methodology . . . 63
3.4.1 Signature-based detection . . . 63
3.4.2 Anomaly-based detection . . . 64
3.5 General architecture . . . 65
3.6 Types of intrusion detection systems . . . 66
3.6.1 Network-based Intrusion Detection Systems (NIDS) . . . 67
3.6.2 Network Node Intrusion Detection System (NNIDS) . . . 67
3.6.3 Host-Based Intrusion Detection Systems (HIDS) . . . 68
3.6.4 Hybrid Intrusion Detection Systems . . . 69
3.7 Conclusion . . . 69

4 Deep Learning based Intrusion detection systems (DL-IDS) 71
4.1 Datasets for Intrusion Detection System . . . 72
4.1.1 KDD Cup 99 . . . 73
4.1.2 NSL-KDD . . . 73
4.2 Performance metrics used for DL-based IDS . . . 74
4.2.1 Classification metrics . . . 74
4.2.2 Regression metrics . . . 76
4.2.3 General metrics . . . 77
4.3 Related Work . . . 77
4.4 Feature selection . . . 80
4.5 Data preprocessing . . . 80
4.6 Implementation and experimentation on NSL-KDD . . . 81
4.6.1 NSL-KDD Preprocessing . . . 81
4.6.2 Design of the suggested deep neural network (DNN) . . . 82
4.6.2.1 DNN model generation . . . 82
4.6.2.2 Activation functions . . . 83
4.6.2.3 Optimization functions . . . 84
4.7 Environment . . . 85
4.7.1 Python . . . 85
4.7.2 Anaconda . . . 85
4.7.3 Jupyter Notebook . . . 86
4.7.4 Numpy . . . 86
4.7.5 Pandas . . . 86
4.7.6 Scikit-learn . . . 86
4.7.7 TensorFlow . . . 86
4.7.8 Keras . . . 87
4.8 DNN classification results . . . 87
4.8.1 The most important metrics . . . 88
4.8.2 Discussion . . . 89
4.9 Conclusion . . . 89

General conclusion **93**

Bibliography **95**

List of Figures

1.1 A typical WSN mote with its fundamental units [15]. . . . 19
1.2 Typical Wireless Sensor Network [77]. . . . 21
1.3 The sensor networks protocol stack [70]. . . . 22
1.4 Point to Point Technology [104]. . . . 24
1.5 Bus Topology [28]. . . . 24
1.6 Cluster Tree Topology [28]. . . . 25
1.7 Star Topology [28]. . . . 25
1.8 Ring Topology [28]. . . . 26
1.9 Mesh Topology [28]. . . . 26
1.10 Circular Topology [101]. . . . 27
1.11 Grid Topology [101]. . . . 28

2.1 Machine learning life cycle. . . . 39
2.2 Supervised Learning. . . . 40
2.3 Unsupervised Learning. . . . 41
2.4 Semi-supervised Learning. . . . 41
2.5 The reinforcement learning framework [51]. . . . 42
2.6 Deep Learning neural network [115]. . . . 44
2.7 Classification of deep learning architectures [52]. . . . 48
2.8 Difference between hidden units in RNN and feed-forward neural networks [7]. . . . 50
2.9 The conceptual structure of an AE [7]. . . . 51
2.10 Difference between BM and RBM architectures. . . . 52
2.11 The conceptual structure of a DBN [7]. . . . 52
2.12 Generative Adversarial Network framework [108]. . . . 53

3.1 Classification of WSN security attacks [58] 59
3.2 Architecture of signature-based IDS [100]. . . . 64
3.3 Anomaly based IDS [52]. . . . 65
3.4 Basic architecture of intrusion detection system (IDS). [66] 66
3.5 Architecture of a network-based intrusion detection system (NIDS) [2]. . . . 67

4.1 model DNN 83
4.2 Python version. . . . 85
4.3 Anaconda environment. . . . 85
4.4 Accuracy evolution in terms of epochs on training and test data. . . . 87
4.5 Loss evolution in terms of epochs on training and test data. . . . 88

List of Tables

3.1 Comparisons of intrusion detection methodologies based on [59]. 66
3.2 Different types of IDS [80]. 69

4.1 Most used Datasets for intrusion detection. 72
4.2 NSL-KDD train and test data distribution [84]. 74
4.3 Confusion Matrix for IDS System. 74
4.4 Example table 87
4.5 The proposed model's performance metrics on NSL-KDD dataset. 88
4.6 results Comparison with literature 89

Code Listings

4.1 NSL-KDD features naming 81
4.2 DNN Model code 82


بسم الله الرحمن الرحيم

Dedications

"When you want something, the whole universe conspires to help you achieve it." - Paulo Coelho

We would like to dedicate this humble work

To all those who have given us reasons for hope and faith.

For everyone who was inspired and designed.

To my dear parents for their support, love, patience and encouragement.

To my sister, to my brothers. To our teacher professor Bourouis abdelhabib for his guidance. To my uncle Professor Makhlouf Seddik for his help, guidance and support.

To my wonderful teacher Mrs. Dahimi Nour Al-Huda.

To all of my friends

Thank you very much all of you.

Acknowledgements

In the Name of Allah, the Most Merciful, the Most Compassionate, all praise be to Allah, the Lord of the worlds, and prayers and peace be upon Mohamed, His servant and messenger. First and foremost, I must acknowledge my limitless thanks to Allah, the Ever-Magnificent; the Ever-Thankful, for His help and blessing. I am totally sure that this work would have never become truth without His guidance.

We would like to warmly and successively thank all those who have contributed from near and far to the realization of this final study project. We thank Prof. Abdelhabib Bourouis for his confidence and his effective supervision and we express to him our deepest gratitude for sharing his knowledge and his kindness, as well as his working methods, especially for his scientific rigor.

We would like to thank the members of the jury for accepting to judge our work.

Abstract

The human species, driven by the desire for a more adequate life, has constantly strived to advance and create a modern civilization. Technology has been an unstoppable force pushing us into a brighter future, marked by great innovations and new challenges. Wireless Sensor Networks (WSN) have emerged as a prominent topic in our contemporary society, serving as a gateway to realizing the vision of global smart cities through Internet of Things (IoT) devices. These networks are finding applications in fields as diverse as telemedicine and smart agriculture, offering exciting opportunities.

However, WSNs face ongoing cybersecurity threats. Whether it is deliberate actions by enemies or mismanagement of the system, the security of wireless networks is of paramount importance, presenting significant challenges. The limitations inherent in sensors, including limited memory and power consumption, make security measures a complex task. It is critical to design security solutions that consider these constraints, ensuring optimal network performance without delays, packet loss, or abnormal functionality.

This thesis proposes an intrusion detection system (IDS) engine based on deep learning techniques, with the aim of achieving superior packet classification results. The NSL-KDD dataset is used to evaluate the performance of the model. The data undergoes pre-processing before being fed into the deep learning (DNN) model. Binary classification is used to distinguish between normal and abnormal traffic using a six-layer neural network consisting of an input layer, four hidden layers, and an output layer.

The results obtained in this study showed high precision and accuracy compared to what the researchers achieved in their published papers. The model achieves an accuracy of 99.65% and an F1 score of 99.67%.

Keywords: Wireless Sensor Networks, Intrusion Detection, Cybersecurity, Deep Neural Network, Deep Learning.

Résumé

L'homme est une créature en évolution qui s'efforce constamment de rendre sa vie plus facile, ce qui a conduit à l'établissement de la civilisation moderne. Ses contributions exceptionnelles se retrouvent dans la technologie, qui est une flèche sans cesse en mouvement, avançant vers un avenir prometteur avec des innovations étonnantes et de nouveaux défis. Les réseaux de capteurs sans fil (WSN) font partie des sujets phares de la technologie moderne ; ils sont la clé pour concrétiser le rêve de villes intelligentes mondiales grâce à l'utilisation d'appareils IoT (Internet des objets), ainsi que la clé de nombreux domaines passionnants tels que la pratique de la télémédecine et l'ingénierie agricole.

Les WSN sont exposés à des menaces cybernétiques à tout moment, qu'elles soient intentionnelles de la part d'adversaires ou dues à une mauvaise gestion du système. Par conséquent, la sécurité des WSN est à la fois critique et complexe en raison des limitations spécifiques des capteurs et des WSN en général, comme une mémoire relativement réduite et la nécessité de maintenir la consommation d'énergie aussi basse que possible. Ainsi, tout protocole de sécurité doit prendre en compte ces facteurs afin d'offrir au réseau des performances maximales sans aucun retard, perte de paquets ou tout autre comportement anormal.

Dans cette thèse, nous proposons un système de détection d'intrusions (IDS) basé sur l'apprentissage automatique profond. En plus de l'ingénierie des caractéristiques pour obtenir des résultats précis dans la classification du jeu de données NSL-KDD, utilisé pour tester le modèle, le jeu de données a subi des prétraitements. Étant donné que la puissance des WSNs est limitée, une classification binaire a été utilisée pour classer les paquets comme normaux ou anormaux, en utilisant un réseau profond composé de six couches : une couche d'entrée, quatre couches cachées et une couche de sortie.

Les résultats obtenus au cours de cette étude ont montré une grande précision et exactitude par rapport aux travaux similaires récents. À la fin de cette recherche, nous avons obtenu une précision de 99,65% et un score F1 de 99,67%

Mots clés: Réseaux de capteurs sans fil, Détection d'intrusion, Cybersécurité, Réseau de Neurones Profonds, Apprentissage Profond.

ملخص

الإنسان مخلوق متطور يسعى دوما لجعل حياته أسهل مما أدى به إلى إنشاء الحضارة الحديثة. إسهاماته المتميزة التكنولوجيا التي هي عبارة عن سهم لا يتوقف؛ يستمر في المضي قدمًا نحو مستقبل مشرق وابتكارات مذهلة وتحديات جديدة. تعد شبكات الاستشعار اللاسلكية (WSN) أحد الميادين الرائدة في التكنولوجيا الحديثة؛ فهي المفتاح لتحقيق حلم المدن الذكية العالمية باستخدام أجهزة إنترنت الأشياء (IoT)، كما هي المفتاح للعديد من المجالات المثيرة مثل ممارسة الطب عن بعد والهندسة الزراعية. مع ذلك؛ تتعرض هذه الشبكات للتهديدات السيبرانية في كل لحظة؛ سواء كانت عن قصد من قبل خصم ما أو بسبب سوء إدارة للنظام. فبالتالي يعد الأمن أمرًا بالغ الأهمية وصعبًا في نفس الوقت بسبب القيود المحددة في أجهزة الاستشعار وشبكاتها بشكل عام نظرا للقدرات المحدودة والضئيلة مثل الذاكرة الصغيرة نسبيا ووجوب الحفاظ على أدني استهلاك ممكن للطاقة. لذلك. يجب أن تأخذ أي حلول أمنية في الاعتبار هذه العوامل من أجل منح الشبكة أقصى أداء يمكن أن تحصل عليه دون أي تأخير أو فقدان للحزم أو أي نوع من الأداء الغير الطبيعي.

حاولنا من خلال هذه الأطروحة اقتراح محرك نظام كشف التسلل أو التطفل (IDS) بناءً على التعلم الآلي وتقنيات التعلم العميق بالإضافة الى هندسة الميزات للحصول على نتائج عالية الدقة في تصنيف الحزم. تم استخدام مجموعة بيانات NSL-KDD لتقييم أداء النموذج. خضعت البيانات للمعالجة المسبقة قبل إدخالها في نموذج التعلم العميق (DNN). نظرًا لأن قوة عناصر شبكات الاستشعار اللاسلكية محدودة تم استخدام التصنيف الثنائي للتمييز بين حركة البيانات العادية المريبة باستخدام شبكة عصبية من ست طبقات واحدة للإدخال وواحدة للإخراج واربعة طبقات مخفية.

النتائج المتحصل عليها خلال هذه الدراسة دقيقة وجيدة مقارنة بما أنجزه الباحثون في نتشوراتهم الحديثة حيث تحصلنا في نهاية هذا البحث على accuracy = 99.65% و F1-score = 99.67%.

الكلمات المفتاحية: شبكات الاستشعار اللاسلكية، كشف التسلل، الأمن السيبراني، الشبكة العصبية العميقة، التعلم العميق.

General Introduction

General introduction

Wireless sensor networks (WSNs) are networks comprised of small, low-power devices called sensor nodes. These nodes are equipped with sensors to monitor various physical or environmental conditions. WSNs are typically connected wirelessly, enabling communication between nodes and forming a network. They find applications in environmental monitoring, industrial automation, healthcare, and smart homes.

Deep learning, a subset of machine learning, utilizes multi-layered artificial neural networks to solve complex problems. Deep learning algorithms have demonstrated remarkable success in areas such as security.

However, WSNs are susceptible to different types of attacks, including denial-of-service attacks, spoofing attacks, sewer attacks, and selective redirect attacks. Traditional intrusion detection systems (IDS) for WSNs rely on predefined rules or signatures, which are ineffective against new or unknown attacks.

To address this challenge, IDS based on deep learning in WSNs offers a promising solution. These algorithms can recognize complex patterns in network traffic and detect anomalies in real-time. By training on large amounts of data, deep learning algorithms can continuously improve their accuracy and performance.

One significant advantage of deep learning-based IDS is its ability to adapt to new and unknown attacks. The algorithms can learn from new data and update their models over time, enhancing accuracy. Additionally, deep learning-based IDSs can operate in a distributed manner, making them well-suited for WSNs.

In our thesis, we aim to tackle the security problem in WSNs using deep learning. Our research will follow the following scheme:

Chapter 1: Wireless Sensor Networks - We will provide a detailed explanation of WSNs, including their topologies, types, applications, and constraints.

Chapter 2: Introduction to Machine Learning and Deep Learning - We will define machine learning, highlight its advantages, and then delve into the definition and basic types of deep learning. Furthermore, we will discuss neural networks.

Chapter 3: Network Intrusion and Wireless Sensor Attacks - We will define network intrusion and discuss wireless sensor attacks, along with various detection methodologies.

Chapter 4: Our Approach and Results - In this final chapter, we will present our approach to addressing the security problem in WSNs using deep learning. We will also provide our research result and compare them with previous findings.

CHAPTER 1

WIRELESS SENSOR NETWORKS

Chapter 1

Wireless Sensor Networks

Contents

1.1 Introduction . . . **19**
1.2 Sensor nodes . . . **19**
1.2.1 Components of a sensor node . . . 20
1.2.2 Sensor types . . . 20
1.3 Wireless Sensor Network . . . **21**
1.3.1 Network Architecture of WSNs . . . 21
1.3.2 WSN routing protocols . . . 22
1.3.2.1 Direct transmission protocol . . . 22
1.3.2.2 Minimum transfer energy protocols . . . 22
1.3.2.3 Clustering protocol . . . 23
1.3.2.4 LEACH (Low Energy Adaptive Clustering Hierarchy) . . . 23
1.3.2.5 Stable Election Protocol . . . 23
1.3.2.6 Hierarchical Cluster-Based Routing (HCR) Protocol . . . 23
1.3.2.7 Genetic Algorithm . . . 23
1.3.3 WSNs topologies . . . 24
1.3.3.1 Point-to-point topology . . . 24
1.3.3.2 Bus topology . . . 24
1.3.3.3 Tree topology . . . 25
1.3.3.4 Star Topology . . . 25
1.3.3.5 Ring topology . . . 25
1.3.3.6 Mesh topology . . . 26
1.3.3.7 Circular topology . . . 27
1.3.3.8 Grid topology . . . 27
1.4 Types of WSNs . . . **27**
1.5 WSN applications . . . **28**
1.6 WSNs constraints . . . **30**
1.6.1 Unreliable Communication . . . 30
1.6.2 Energy constraints . . . 30
1.6.3 Memory limitations . . . 30
1.6.4 Higher latency in communication . . . 30
1.6.5 Unattended operation of networks . . . 31

1.7 Security in WSNs 31
1.8 Conclusion 32

1.1 Introduction

Sensors have a long history, originating as electromechanical detectors utilized for measuring physical quantities. Their earliest application dates back to 1933, when they were first employed in room thermostats. Early microelectromechanical systems (MEMS) consisted of separate chips and packages housing the sensor, electronics, and mechanics. This led to larger sizes, increased costs, and a lower sensor yield. Nevertheless, advancements in MEMS and integrated circuits (IC) have enabled the development of compact and cost-effective sensors by integrating actuators and electronics into a single chip. These advanced sensors, commonly known as smart sensors, now incorporate on-board processors, memory, and transceivers in small form factors, powered by batteries. They serve as nodes within Wireless Sensor Networks (WSNs). The advantage of these compact nodes lies in their low production and installation costs [15].

Presently, WSN nodes can range from hundreds to thousands of dollars, but it is anticipated that the cost will decrease significantly, reaching just a few dollars, due to ongoing technological advancements and mass production. The small size and affordability of these nodes make installation convenient, whether they are randomly deployed or strategically positioned to meet specific application requirements. Equipped with processing, storage, and sensing capabilities, along with the infrastructure-less networking capabilities of the wireless transceiver, these low-cost nodes contribute to the powerful and cost-effective nature of WSNs, offering solutions to a wide range of research fields. WSNs enable distributed collaboration for sensing and processing information, where vital data is transmitted through multi-hop ad hoc networks to WSN sinks (data collectors) or base stations, which act as gateways to fixed infrastructures.

Various types of WSNs exist, including those with actuators forming a Wireless Sensor and Actuator Network (WSAN), mobile nodes within WSNs, or nodes capable of handling multimedia content such as audio, video streaming, or still images in Wireless Multimedia Sensor Networks (WMSNs). WSNs find applications in diverse fields, providing cost-effective solutions for environmental observation, military and civilian surveillance, target detection and tracking, industrial process monitoring and control, precision farming and agriculture, environmental and habitat monitoring, patient monitoring in healthcare, residential applications like energy management, vehicular networks for safety and efficiency, and even outer space exploration. Figure 1.1 represents a typical WSN mote with its fundamental units.

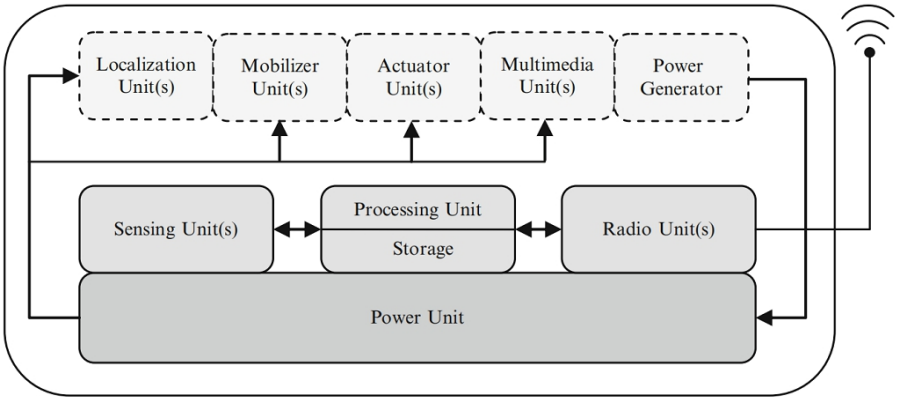


Figure 1.1: A typical WSN mote with its fundamental units [15].

1.2 Sensor nodes

In a sensor field, there are several sensor nodes that serve as transceivers. These nodes have the ability to collect and process data, and transmit it back to the sink or gateway and end-users through

a multi-hop infrastructure-less architecture. The nodes carry out local computations to reduce the amount of data transmitted. The sink can be connected to the end-users directly or through different types of wireless networks like WiFi, mesh networks, cellular systems, WiMAX, or satellite systems, enabling the Internet of Things. It is important to note that there can be multiple sinks (gateways) and multiple end-users in this architecture [33].

1.2.1 Components of a sensor node

A sensor node is a small electronic device that is designed to gather data from its environment, process it, and transmit it to a central system for further analysis [17, 122]. The architecture of a sensor node typically consists of four basic components, as shown in Figure 1.1, a sensing unit, a processing unit, a transceiver unit, and a power unit:

1. **A sensing unit:** Typically, sensing units consist of two main components: sensors and analog-to-digital converters (ADCs). The sensors capture the analog signals related to the observed phenomenon, which are subsequently transformed into digital signals by the ADCs. These digital signals are then directed to the processing unit for further analysis and interpretation.
2. **A processing unit:** The processing unit, often accompanied by a compact storage unit, oversees the operations that enable the sensor node to cooperate with other nodes in executing the designated sensing tasks.
3. **A communication unit (transceiver unit):** The transceiver unit establishes the connectivity of the node with the network.
4. **A power unit:** The power unit is a crucial component of a sensor node, and it can be supplemented by power scavenging units like solar cells to generate energy.

1.2.2 Sensor types

In a WSN, sensors are of different types and play different roles [120]:

- ☞ **Ordinary sensor node:** The source node possesses the ability to perform data processing, data gathering, and communication with other associated nodes within the network.
- ☞ **Sink node:** This node is a particular node that has higher resources than the other nodes in the network. It is responsible for receiving data from all the nodes in the network and can communicate with an external network such as the internet.
- ☞ **Cluster head:** The high-bandwidth sensing node serves the purpose of executing data fusion and aggregation functions. Depending on the application, a cluster may consist of multiple cluster heads. The gateway acts as an interface between the sensor networks and external networks. It is characterized by its high program memory and data memory capacity, processor utilization, transceiver range, and the potential for extension through external memory.
- ☞ **Actor node:** The actor node is a high-end node specifically designed to execute tasks and make decisions based on the specific requirements of the application. It possesses advanced capabilities that enable it to perform complex actions and contribute to the decision-making process within the network.
- ☞ **Relay node:** The intermediary node acts as a relay between neighboring nodes, playing a crucial role in improving network reliability. It is a unique type of field device that lacks process sensors or control equipment. Its primary function is to facilitate communication within the network.

1.3 Wireless Sensor Network

Advancements in MEMS, wireless communications, and integrated digital electronics have facilitated the development of cost-effective, low-power micro sensors. These sensors communicate over short distances and have multifunctional capabilities [5]. Sensor nodes play a crucial role in sensing, data processing, and data delivery to the base station (BS) in a wireless sensor network (WSN). WSNs consist of numerous sensor nodes deployed in a designated coverage area, where they collect local physical information, process it, and transmit it to the BS or sink as illustrated in Figure 1.2. Collaboration among WSN nodes is another significant characteristic. Rather than sending raw data, sensor nodes can perform local calculations and fusion operations to transmit only the necessary information, reducing data transmission [6]. The versatile nature of wireless sensors makes them suitable for various applications, particularly in surveillance and monitoring fields [34].

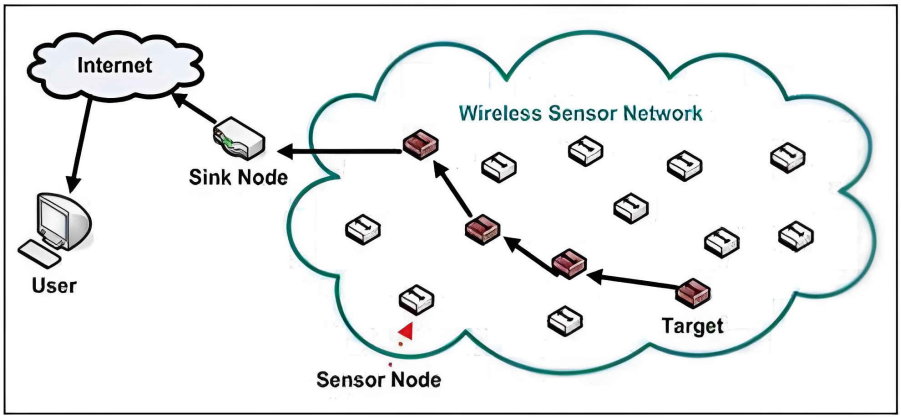


Figure 1.2: Typical Wireless Sensor Network [77].

1.3.1 Network Architecture of WSNs

A wireless sensor network (WSN) typically adopts a five-layered network architecture based on the OSI reference model as illustrated in Figure 1.3. To ensure efficient operation, three cross layers are employed to manage various aspects of the network. These layers include power management, connection/mobility management, and task management. These cross layers play a critical role in optimizing power consumption, managing node connectivity and mobility, and coordinating task execution within the network [122]:

1. **Application Layer:**
   The application layer, situated at the topmost level of the WSN architecture, is responsible for traffic management and software provisioning for various applications. Applications within this layer send queries to retrieve system information and interact with the network for specific purposes.

2. **Transport Layer:**
   The transport layer in a WSN plays a crucial role in facilitating internetwork communication. It employs a range of protocols to ensure system reliability and prevent network congestion. Unlike conventional TCP-based connections, WSNs utilize the user datagram protocol (UDP) for efficient node-to-node communication, enabling reliable data transmission within the network.

3. **Network Layer:**
   The network layer in a WSN handles routing protocols, which are essential for managing parameters such as power consumption, reliability, memory utilization, and redundancy. These

protocols can be categorized into flat routing, hierarchical routing, and event-driven, query-driven, or time-driven routing, depending on the application type or deployment scenario. To ensure efficiency and energy savings, routing protocols may employ techniques like data aggregation for redundancy and noise removal from aggregated data through data fusion.

4. **Data Link Layer:**
   The data link layer in a WSN ensures reliable data transmission from point to point or multipoint. It handles error control and data multiplexing. Additionally, this layer incorporates a media access control (MAC) address, which serves as a unique identifier for nodes. Its primary objectives are to enhance reliability, minimize latency, and optimize efficiency and throughput.

5. **Physical Layer:**
   The physical layer in a WSN acts as an interface for transmitting data over a physical medium. It facilitates the transmission frequency and manages parameters such as carrier frequency generation for modulation, signal detection, and security measures.

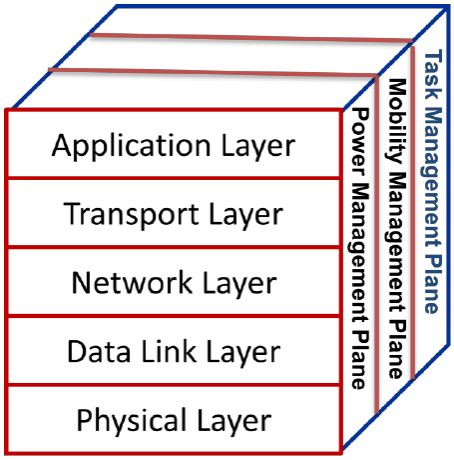


Figure 1.3: The sensor networks protocol stack [70].

1.3.2 WSN routing protocols

Different types of protocols for WSN are following [3]:

1.3.2.1 Direct transmission protocol

In a direct communication protocol, sensors transmit their data directly to the base station. However, if the base station is located far away from the nodes, this method requires a significant amount of transmit power from each node. Consequently, the nodes' batteries are quickly depleted, leading to a reduction in the overall system lifetime. Nevertheless, since only the base station receives data in this protocol, it may be considered acceptable if the base station is in close proximity to the nodes or if the energy required for data reception is substantial.

1.3.2.2 Minimum transfer energy protocols

Sensor nodes forward their data through a series of intermediate nodes before reaching the base station, intermediate nodes get to act like routers for other nodes without losing the capacity of sensing the environment. This will reduce the transmit power required for each node, as they only need to send data to their neighboring nodes, which are likely to be closer than the base station.

1.3.2.3 Clustering protocol

A commonly used protocol in wireless networks is clustering, which involves organizing nodes into clusters that communicate with a local base station. The local base stations then transmit the data to a global base station, which is accessible to end-users. This clustering approach significantly reduces the transmission distance for nodes since the local base station is typically in close proximity to all the nodes within the cluster. Consequently, clustering is often regarded as an energy-efficient communication protocol. However, it is important to consider that the local base station is assumed to have ample energy resources. In scenarios where the base station has limited energy capacity, it would experience rapid energy depletion due to its heavy utilization. As a result, conventional clustering may not perform well within our model of micro sensor networks.

1.3.2.4 LEACH (Low Energy Adaptive Clustering Hierarchy)

LEACH is a clustering protocol designed to achieve self-organization and adaptability in wireless sensor networks. It employs a randomization strategy to evenly distribute the energy load among the sensors in the network. In LEACH, nodes autonomously form local clusters, and one node is selected as the cluster head or local base station. The cluster head position is randomly rotated among different sensors to prevent excessive energy depletion in a single sensor. Furthermore, LEACH incorporates local data fusion techniques to reduce the amount of data transmitted from the clusters to the base station. This data compression approach minimizes energy dissipation and extends the overall lifetime of the system.

1.3.2.5 Stable Election Protocol

The Stable Election Protocol (SEP) proposes a weighted election mechanism to select cluster heads (CHs) in wireless sensor networks based on the energy levels of individual nodes. This ensures a fair distribution of CH roles by considering the fraction of energy available to each node. SEP also introduces a hierarchical structure with two types of nodes and two levels of hierarchy. In our analysis, we investigated a three-level hierarchical clustered heterogeneous sensor network, comprising three types of nodes: advanced, moderate, and normal. Advanced and moderate nodes possess higher energy reserves, longer transmission ranges, and faster data rates compared to normal nodes. Consequently, advanced and moderate nodes have a greater probability of becoming cluster heads in each round, which significantly extends the operational lifespan of the sensor network.

1.3.2.6 Hierarchical Cluster-Based Routing (HCR) Protocol

HCR (Hierarchical Cluster Routing) is a protocol where nodes autonomously form clusters, and each cluster is supervised by a group of associates known as the head-set. Through a round-robin mechanism, the associates take turns serving as cluster heads (CHs) [3]. Sensor nodes send their data to their respective cluster heads, which then aggregate and transmit the data to the base station. Additionally, the protocol employs heuristics-based techniques to identify energy-efficient clusters, allowing them to operate for extended durations.

1.3.2.7 Genetic Algorithm

A genetic algorithm (GA) is utilized to optimize energy efficiency in wireless sensor networks by creating clusters for data dissemination. The GA operates at the base station, generating energy-efficient solutions to optimize cluster formation and minimize energy consumption during runtime. Through analysis of the network condition, the base station applies the GA after each iteration. The optimizer at the base station evaluates different solutions using a fitness function based on parameters

such as energy consumption, number of clusters, cluster size, direct distance to the sink, and cluster distance. The optimizer receives feedback at the end of each iteration to adjust the weights of the fitness function's parameters for improved decision-making in subsequent iterations. In this approach, the GA represents each node as a chromosome bit, distinguishing between head nodes (represented as 1s) and member nodes (represented as 0s). A population of chromosomes is evaluated, and the best chromosome is used to generate the next population. Simulation results demonstrate that employing GA-based hierarchical clusters significantly increases the network's lifetime. Furthermore, future research can explore cross-layer optimization through query and routing strategies.

1.3.3 WSNs topologies

The development and deployment of WSNs have taken traditional network topologies in new directions. Different Wireless sensor network topologies are Bus, Tree, Star, Ring, Mesh, Circular and Grid.

1.3.3.1 Point-to-point topology

The point-to-point topology involves a dedicated wireless link between two sensor nodes, enabling long-range and high-capacity communication as illustrated in Figure 1.4. It is a conventional model commonly used in wireless sensor networks. This topology has both advantages and disadvantages. One advantage is the use of a single data communication channel, which ensures secure communication between the nodes. However, a disadvantage is that if this channel fails, communication between the two nodes will be disrupted [104].

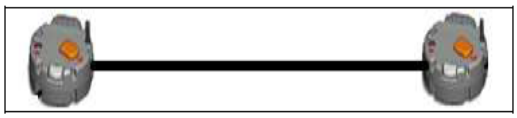


Figure 1.4: Point to Point Technology [104].

1.3.3.2 Bus topology

The broadcast topology is commonly employed for message dissemination within a network. In this topology, a node broadcasts a message intended for another node, and all nodes in the network receive the message as illustrated in Figure 1.5. However, only the intended recipient processes the message, while the other nodes discard it. Although this topology can lead to traffic congestion due to the single communication path, it is relatively easy to install. It is particularly suitable when the number of nodes is limited. The lines shown in figure depict wireless connectivity [28].

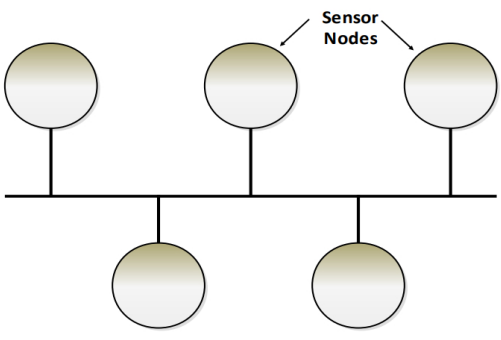


Figure 1.5: Bus Topology [28].

1.3.3.3 Tree topology

In this topology, the central hub serves as the main communication router and acts as the root node. It can be described as a combination of the Peer-to-Peer and Star topologies. As illustrated in Figure 1.6, at a lower level, it resembles a star network. The parent node is responsible for managing the children nodes in this topology [28]. Communication paths can be either single hop or multi hop. Sensor nodes gather data or information and transmit it to the sink node through their parent node. The parent node forwards the data received from its children to the upper level. To maximize the sensor's lifetime, it is important to find the shortest path that optimizes factors such as shorter time delay and proper node distribution. The main challenge lies in load balancing, which involves distributing the workload among siblings to conserve power and extend the network's lifetime.

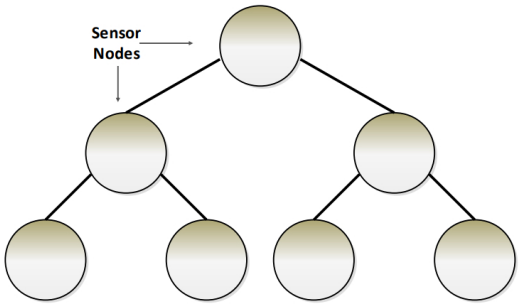


Figure 1.6: Cluster Tree Topology [28].

1.3.3.4 Star Topology

In a star topology, network nodes do not communicate directly with each other or exchange data. Instead, they are connected to a central communication hub (sink) through which they communicate with one another as illustrated in Figure 1.7. The centralized hub acts as a router for the entire communication network. The central hub functions as a server or sink, while the surrounding connected nodes act as "clients" [28].

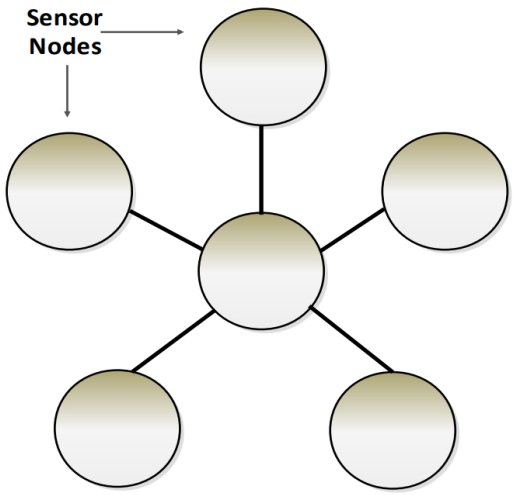


Figure 1.7: Star Topology [28].

1.3.3.5 Ring topology

In a ring topology, each node is connected to exactly two neighboring nodes for communication as illustrated in Figure 1.8. The flow of messages or information is unidirectional, either clockwise or

counterclockwise. This topology can experience traffic congestion, and if any node fails, it breaks the loop, resulting in the failure of the entire network [28].

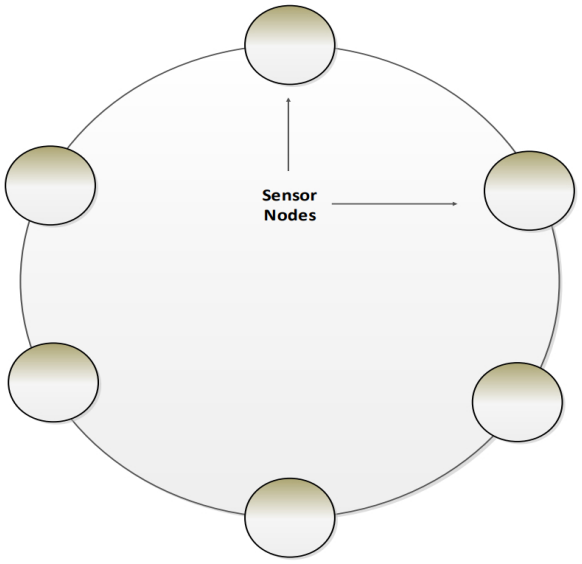


Figure 1.8: Ring Topology [28].

1.3.3.6 Mesh topology

It is a multi-hop system where nodes can communicate with each other directly. The visual representation of this topology is shown in Figure 1.9.

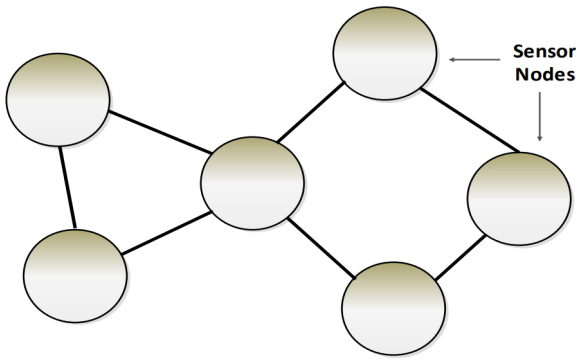


Figure 1.9: Mesh Topology [28].

The mesh topology in wireless sensor networks offers several advantages. Firstly, it eliminates the presence of a single point of failure, making it a highly reliable communication network structure. Additionally, it provides scalability, allowing for easy expansion and addition of nodes as the network grows. Another benefit is the presence of alternate paths, which reduces the chances of data loss and enhances data reliability. However, there are some disadvantages to consider with mesh technology in wireless sensor networks. One notable drawback is the higher power consumption associated with this topology. This increased power usage is a concern for wireless sensor networks, which strive to be energy-efficient and have prolonged battery life. Efforts should be directed towards addressing the power consumption issue and optimizing the energy efficiency of mesh-based wireless sensor networks to overcome this drawback.

1.3.3.7 Circular topology

The circular topology in wireless sensor networks offers a circular sensing area with a central sink. Sensor nodes detect and transmit data to the sink, either through single or multiple hops as illustrated in Figure 1.10. This topology provides benefits such as ease of establishment and maintenance, efficiency, and energy efficiency. Self-configuring algorithms and integrated MAC and routing protocols further enhance the performance of circular topology-based networks. The use of multiple tiers and the selection of routing paths based on energy levels contribute to energy conservation and improved packet reception at the sink [101].

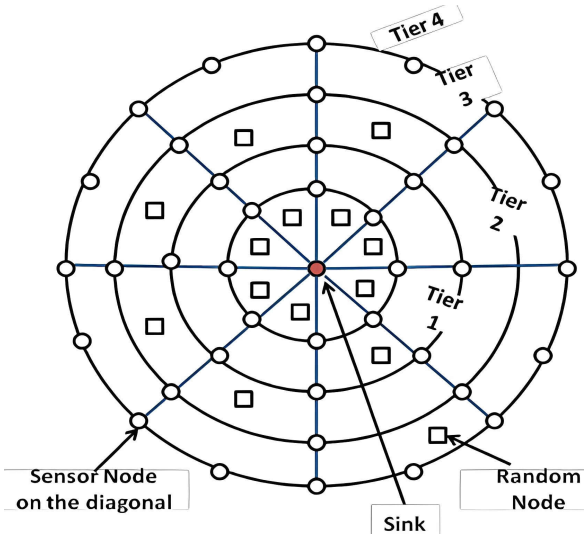


Figure 1.10: Circular Topology [101].

1.3.3.8 Grid topology

Grid-based wireless sensor networks offer a structured approach to organizing sensor nodes, where the network area is divided into square grids as illustrated in Figure 1.11. Each grid contains at least one active node, ensuring coverage throughout the network. To prolong the network's lifespan and optimize energy usage, nodes within a grid take turns operating, with a designated grid head responsible for routing and data transmission. This grid-based multi-path routing protocol enables fast packet routing and efficient energy utilization. Additionally, clustering techniques can be employed within each grid to further enhance energy efficiency and load balancing. Congestion control mechanisms play a crucial role in maintaining network performance, while joint priority-based algorithms help alleviate congestion and achieve fairness in data transmission. By leveraging grid-based structures, these approaches contribute to the longevity, energy efficiency, and effective management of wireless sensor networks [101].

1.4 Types of WSNs

There are many different types of WSNs, each with its own unique set of characteristics and applications. Some of the most common types of WSNs include:

- **Terrestrial WSNs:** these are the most common type of WSN, and they are used in a wide variety of applications, including environmental monitoring, industrial control, and military surveillance.
- **Underground WSNs:** these networks are used to collect data from underground environments, such as mines and oil fields.

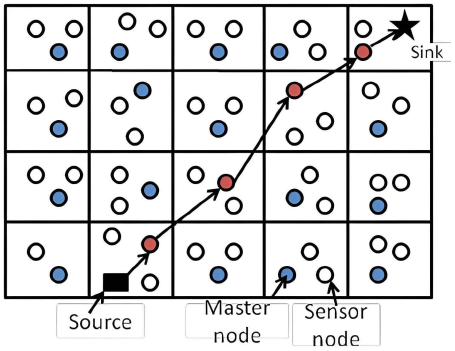


Figure 1.11: Grid Topology [101].

- **Underwater WSNs:** these networks are used to collect data from underwater environments, such as oceans and lakes.
- **Multimedia WSNs:** these networks are used to collect and transmit multimedia data, such as video and audio.
- **Mobile WSNs:** these networks are used to collect data from moving objects, such as vehicles and animals.

Each of these types of WSNs has its own unique set of challenges and requirements. For example, terrestrial WSNs must be able to deal with the challenges of operating in a noisy RF environment, while underwater WSNs must be able to deal with the challenges of operating in a harsh and corrosive environment.

Despite the challenges, WSNs are a rapidly growing technology with a wide range of potential applications. As the technology continues to evolve, we can expect to see even more innovative and groundbreaking applications for WSNs in the years to come.

1.5 WSN applications

Wireless Sensor Networks (WSNs) have a wide range of applications across various industries and fields [120]. Their applications include:

- **Environmental Applications:**
  Environmental applications of WSNs involve monitoring and studying the movements and patterns of insects, birds, or small animals. This allows for valuable insights into the behavior and habitats of these species in their natural environments.
- **Military application:**
  WSNs play a crucial role in military command, control, communication, and intelligence systems. They are deployed in battlefield scenarios to monitor vehicle presence and track their movements, providing valuable surveillance capabilities for observing opposing forces.
- **Healthcare:**
  The utilization of wireless sensor networks in health care enables the monitoring and tracking of patients, providing a potential solution to address the shortage of health care personnel and reduce healthcare expenses in existing healthcare systems. This technology allows for remote patient monitoring, facilitating timely interventions and personalized care.
- **Structural Monitoring:**
  Wireless sensors are employed to monitor and track the movement within various structures such as buildings, bridges, flyovers, tunnels, and more. This technology enables engineering

practices to remotely monitor assets and infrastructure. With its high accuracy, wireless sensor networks provide a more reliable alternative to visual inspections for detecting and analyzing structural changes or anomalies. This facilitates proactive maintenance and enhances overall safety and efficiency in managing critical infrastructure.

- **Industrial Applications:**
  Industries extensively utilize wireless sensor networks (WSNs) to monitor manufacturing processes and equipment. For instance, chemical plants and oil refiners make use of sensors to monitor the condition of their extensive pipeline networks. By deploying sensors strategically, they can detect any failures or abnormalities in real-time. This proactive monitoring enables timely alerts and facilitates prompt maintenance and intervention, minimizing the risk of accidents and optimizing operational efficiency. WSNs play a crucial role in ensuring the smooth functioning and safety of industrial operations.
- **Agriculture:**
  Sensors are widely deployed both on the ground and underwater to monitor the quality of air and water. For instance, pressure transmitters can be used to observe gravity feed water systems and monitor water levels. Wireless I/O devices can control pumps based on the data collected by these sensors. By implementing irrigation automation techniques, the usage of water can be optimized, leading to more efficient irrigation practices and reduced wastage. These wireless sensor networks enable continuous monitoring, allowing for timely interventions and resource management to ensure optimal environmental conditions and resource utilization.

There are many advantages of using WSNs:

- **Low cost**: WSNs are relatively inexpensive to deploy and maintain.
- **Scalability:** WSNs can be scaled to meet the needs of any application.
- **Flexibility:** WSNs can be deployed in a variety of environments.
- **Real-time data collection:** WSNs can collect data in real time, which can be used to make timely decisions.
- **Enhanced security:** WSNs can be used to improve security by monitoring for intrusions and other threats.

There are also some disadvantages of using WSNs:

- **Limited bandwidth:** WSNs have limited bandwidth, which can restrict the amount of data that can be collected and transmitted.
- **Security:** WSNs are vulnerable to security attacks, such as data theft and denial-of-service attacks.
- **Power consumption:** WSNs are battery-powered, so they have limited power resources.
- **Deployment:** WSNs can be difficult to deploy and maintain, especially in large or remote areas.

WSNs are a promising technology with a vast array of potential applications despite their drawbacks. In the future, we could expect even more innovative and ground-breaking applications for WSNs as the technology continues to advance.

1.6 WSNs constraints

Wireless Sensor Networks (WSNs) have certain constraints that make it challenging to deploy security measures. These networks primarily rely on low-power devices that have limited capabilities in terms of performance, energy, and storage. Due to these constraints, it is difficult to implement robust security measures in WSNs [122].

1.6.1 Unreliable Communication

Unreliable communication poses a significant security threat to sensor networks. Typically, sensor networks rely on connectionless protocols for packet-based routing, which inherently lack reliability. Channel errors or congestion can result in packet damage or dropping. Additionally, the wireless communication channel itself can introduce packet corruption. To address these challenges, robust error handling schemes with increased overhead are necessary. Even in cases where the channel is considered reliable, communication may still be unreliable due to the broadcast nature of wireless communication. Collisions during packet transit may necessitate retransmission, further impacting reliability.

1.6.2 Energy constraints

Energy is a critical constraint for wireless sensor capabilities. Once sensor nodes are deployed in a network, they are difficult to replace or recharge, leading to the need for conserving battery life. Therefore, when implementing security measures in sensor nodes, the impact on energy consumption must be carefully considered. The power consumption of security functions, such as encryption, decryption, signing, and verification, as well as the energy required for transmitting security-related data and storing security parameters, all contribute to the additional power consumption. Key establishment is particularly energy-intensive in the realm of security, while the energy consumed for protecting each message is relatively small. Therefore, WSNs may need to be divided into different security levels based on energy costs.

1.6.3 Memory limitations

A sensor is a small device with limited memory and storage capacity. Sensor processors require different types of memory to perform various processing tasks. Typically, sensor nodes include flash memory and RAM for memory storage. ROM or EPROM is used to store general-purpose programming, such as embedded operating systems, security functions, and basic networking capabilities. RAM is utilized for storing application programs, sensor data, and intermediate computations. Programmable memory, such as EEPROM and FLASH, is used for storing downloaded application code and data during sleep periods.For instance, in the SmartDust project, TinyOS consumes approximately 4K bytes of instructions, leaving only 4,500 bytes for running security algorithms and applications . As an example, the TelosB sensor, which is a common sensor type, features a 16-bit, 8 MHz RISC CPU with limited resources including 10K RAM, 48K program memory, and 1024K flash storage. Due to these resource constraints, current security algorithms are impractical to implement on such sensors .

1.6.4 Higher latency in communication

In a wireless sensor network (WSN), several factors contribute to increased latency in packet transmission, including multi-hop routing, network congestion, and processing delays in intermediate nodes. These latency issues can pose challenges in achieving synchronization within the network. Synchronization becomes particularly crucial in security applications where timely critical event reports

and cryptographic key distribution are necessary . The presence of higher latency in communication can complicate the implementation of synchronization mechanisms and affect the overall efficiency and reliability of security protocols in the WSN.

1.6.5 Unattended operation of networks

In WSNs, sensor nodes are typically placed in remote locations and are often unattended, increasing the likelihood of physical attacks on the nodes. Managing the security of these networks remotely can make it difficult to detect physical tampering or breaches, making security in WSNs a challenging task.

1.7 Security in WSNs

A Wireless Sensor Network (WSN) is a specific network that has similarities to a conventional computer network but also possesses distinct features. Security services in a WSN should ensure the protection of transmitted data and resources against potential attacks and misbehavior of nodes. According to [98], the primary security requirements for a WSN are enumerated below:

1. **Data confidentiality:** to maintain data confidentiality in a WSN, the security mechanism must guarantee that only the intended recipient can comprehend the messages transmitted across the network. The confidentiality requirements for a WSN should take into account the following aspects: a sensor node should not permit its readings to be accessed by unauthorized neighboring nodes, the key distribution mechanism should be highly reliable, and in some cases, public information such as sensor identities and public keys of nodes should also be encrypted to prevent traffic analysis attacks.

2. **Data integrity:** the security mechanism in a WSN must ensure that no entity can modify any message as it travels from the sender to the intended recipient, which guarantees the data integrity.

3. **Availability:** the availability requirement of a WSN ensures that its services remain accessible even during internal or external attacks, such as a denial of service attack (DoS). Researchers have proposed various approaches to achieve this objective, including utilizing additional communication among nodes or using a central access control system to ensure the successful delivery of each message to its intended recipient.

4. **Data freshness:** data freshness is a security requirement that guarantees the data's recentness and prevents adversaries from replaying old messages. This is particularly crucial in WSNs where shared keys are used for message communication because an attacker can launch a replay attack using the old key while the new key is being refreshed and distributed to all nodes in the network. To verify the freshness of a packet, anonce or time-specific counter can be added to each packet.

5. **Self-organization:** in a WSN, each node must be capable of self-organizing and self-healing, which is a challenging task for security. Due to the dynamic nature of a WSN, it is often impossible to deploy pre-installed shared key mechanisms between nodes and the base station. Several key pre-distribution schemes have been proposed for symmetric encryption, but for the application of public-key cryptographic techniques, an efficient key distribution mechanism is crucial. It is desirable for nodes in a WSN to self-organize not only for multi-hop routing but also for key management and developing trust relations.

6. **Secure localization:** in a WSN, it is often necessary to accurately and automatically locate each sensor node for fault detection and other purposes. However, accurate location information can be manipulated by potential adversaries, making it necessary to secure location information. Two techniques for securing location information are verifiable multi-lateration (VM) and secure range-independent localization (SeRLoC). VM uses authenticated ranging and distance bounding to ensure accurate location of a node, while SeRLoC is a decentralized range-independent localization scheme that uses a shared global symmetric key pre-distributed in the sensor nodes. Both techniques ensure accurate location information without compromising security.

7. **Time synchronization:** time synchronization is essential for many applications in WSN, and any security mechanism for WSN must also be time-synchronized. In some cases, a group of sensors may require synchronization, which poses a challenge to the security mechanism.

8. **Authentication:** authentication is a crucial security requirement in WSN, as it ensures that the communicating node is genuine and not an adversary trying to modify or inject fabricated packets. To achieve this, a mechanism such as message authentication code (MAC) can be used to compute a code from a shared secret key between two nodes. Many authentication schemes have been proposed by researchers for secure routing in WSNs.

1.8 Conclusion

In this chapter, we have provided definitions for a sensor node and a wireless sensor network architecture. We have also explored different WSN topologies, including Bus, tree, and mesh. Additionally, we have discussed various communication protocols like LEACH and HCR that govern traffic flow within WSNs. These networks have wide-ranging applications in fields such as Industrial Applications, Structural Monitoring, and Agriculture .

However Similar to any technology, WSNs face certain limitations such as energy constraints, limited memory, and unreliable communication inherent to wireless nature. Ensuring data confidentiality, integrity, availability, and freshness are crucial considerations for WSNs. Moreover, Also they must be self- organized and synchronized.

CHAPTER 2

DEEP LEARNING

Chapter 2

Deep learning

Contents

2.1 Introduction **37**
2.2 Definition **37**
2.3 Machine learning **37**
2.3.1 Machine learning life cycle 38
2.3.2 Machine learning applications 39
2.4 Basic types of machine learning **40**
2.4.1 Supervised learning 40
2.4.2 Unsupervised learning 40
2.4.3 Semi-supervised learning 41
2.4.4 Reinforcement learning 42
2.4.5 Transfer learning 42
2.5 Deep learning **43**
2.5.1 Introduction 43
2.5.2 Definition 43
2.5.3 Applications of deep learning 43
2.5.4 Advantages of Deep Learning 44
2.6 Common issues **44**
2.6.1 Overfitting 44
2.6.2 Underfitting 45
2.6.3 Insufficient Training Data (Data scarcity) 45
2.6.4 Nonrepresentative Training Data 46
2.6.5 Poor-Quality Data 46
2.6.6 Irrelevant Features 46
2.6.7 Vanishing gradients 46
2.6.8 Exploding gradients 46
2.6.9 Computational complexity 46
2.6.10 Interpretability 46
2.7 Neural networks **47**
2.7.1 Feed-Forward network 47
2.7.2 Deep neural network (DNN) 47

2.8 Deep Learning Architectures **48**
2.8.1 Discriminative Architectures 48
2.8.1.1 Convolutional neural networks 48
2.8.1.2 Recurrent Neural Network (RNN) and Recursive Neural Network (RvNN) 49
2.8.2 Generative Architectures 50
2.8.2.1 Auto-Encoder (AE) 50
2.8.2.2 Restricted Boltzmann Machine (RBM) 51
2.8.2.3 Deep Belief Network (DBN) 52
2.8.3 Hybrid Architectures 52
2.8.3.1 Generative Adversarial Network (GAN) 53
2.9 Conclusion **53**

2.1 Introduction

Artificial intelligence (AI) refers to the development of intelligent computer systems that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and natural language processing. AI algorithms are designed to enable machines to learn from experience, adjust to new inputs, and perform tasks that were previously the sole domain of humans.

The history of AI dates back to the 1940s, when computer pioneer Alan Turing proposed the concept of a machine that could exhibit intelligence comparable to that of a human. The term "artificial intelligence" was first coined in 1956 at a conference at Dartmouth College. Since then, AI has grown and evolved significantly and has become an interdisciplinary field that draws upon computer science, mathematics, psychology, philosophy, and linguistics, among other areas.

AI is becoming increasingly important in today's world, as it has a wide range of applications across various industries, including healthcare, finance, manufacturing, and transportation. AI technologies such as machine learning, natural language processing, and computer vision are enabling organizations to automate tasks, gain insights from data, and make more informed decisions. AI is also being used to develop intelligent systems that can assist humans in various ways, such as in medical diagnosis, autonomous driving, and virtual personal assistants. Overall, AI is expected to play a critical role in shaping the future of many industries and society as a whole [7].

2.2 Definition

With access to large data sets of cyber resources, networks, operating systems or information systems and meeting challenges cybersecurity, methods and techniques such as machine learning (machine learning), data mining, statistics and other social skills used [29]. Deep learning which is part of machine learning can be used for signature-based IDS or anomaly detection.

Classification and predictive methods can be used to determine The unique principles and practices of various cyber attacks allow for a good cyber response. They have the ability to detect attacks when they occur products and the ability to predict potential future attacks [72]. A deep learning approach based on learning can help overcome challenges related to the development of effective IDS [32, 93]. On the other hand, data collection and network traffic caused problems of large data security professionals still need to perform better IDS which has the highest detection rate and the lowest false alarm rate. From Therefore, deep learning approaches the scale well to large sums of data. These are introduced for network anomaly detection to distinguish normal behavior from abnormal behavior to detect bad or suspicious behavior [111].

Deep learning (DL) belongs to a class of methods learning (machine learning or ML), it achieves great success and many artificial intelligence (AI) operations compared to ML algorithms classical times. Deep modeling architecture is a recent phenomenon that exploits many non-realistic information processing techniques, which Information is processed in a series of levels, each of which is received and translated information from the previous layer for learning the representation of data [25].

2.3 Machine learning

Machine learning is a specialized area of artificial intelligence (AI), which focuses on constructing models and algorithms that facilitate computers to learn from data and make decisions or predictions. It is a continual process where a computer program enhances its performance on a particular task without explicit programming.

The core concept of machine learning revolves around enabling machines to autonomously analyze and learn from data, identify patterns, make informed decisions, and adapt to changes. This feature enables the development of intelligent systems that can improve their performance with the availability of additional data and may even outperform human-level performance in some tasks [9].

2.3.1 Machine learning life cycle

The machine learning life cycle refers to the series of steps involved in developing and deploying a machine learning model as illustrated in Figure 2.1 It typically consists of the following stages:

1. **Problem Definition:**
   Clearly define the problem you want to solve using machine learning. Understand the business objectives, available data, and desired outcomes.

2. **Data Collection:**
   Gather the relevant data required for training and evaluating the machine learning model. This may involve data acquisition, data cleaning, data integration, and data preprocessing steps.

3. **Data Exploration and Analysis:**
   Explore and analyze the collected data to gain insights and understanding. Perform descriptive statistics, data visualization, and data quality checks. Identify patterns, trends, and relationships in the data.

4. **Feature Engineering:**
   Select or create meaningful features from the available data that will be used as inputs for the machine learning model. This step may involve feature selection, feature extraction, dimensionality reduction, and transforming data into a suitable format.

5. **Model Selection and Training:**
   Choose an appropriate machine learning algorithm or model based on the problem type and data characteristics. Split the data into training and validation sets. Train the model using the training data and tune its hyperparameters to optimize its performance.

6. **Model Evaluation:**
   Evaluate the trained model's performance using appropriate evaluation metrics and techniques. Assess how well the model generalizes to unseen data and whether it meets the desired business objectives. Adjust the model if necessary.

7. **Model Deployment:**
   Once the model has been trained and evaluated, deploy it into a production environment where it can be used to make predictions or provide insights. This may involve integrating the model into an application or system and setting up the necessary infrastructure.

8. **Monitoring and Maintenance:**
   Continuously monitor the model's performance in the production environment. Collect feedback and evaluate its performance over time. Retrain or update the model periodically to keep it up to date and accurate. Handle issues such as concept drift or changing data distributions.

9. **Model Interpretability and Explainability:**
   Understand and interpret the decisions made by the model. Use techniques such as feature importance analysis, model explainability methods, or interpretability tools to gain insights into the model's inner workings.

10. **Model Iteration and Improvement:**
    Iterate on the machine learning life cycle based on feedback, new data, or changing business requirements. Refine the model, incorporate new features, explore different algorithms, or apply advanced techniques to improve model performance and address limitations.

It's important to note that the machine learning life cycle is an iterative process, where different stages may be revisited multiple times to improve the model's performance and address challenges.

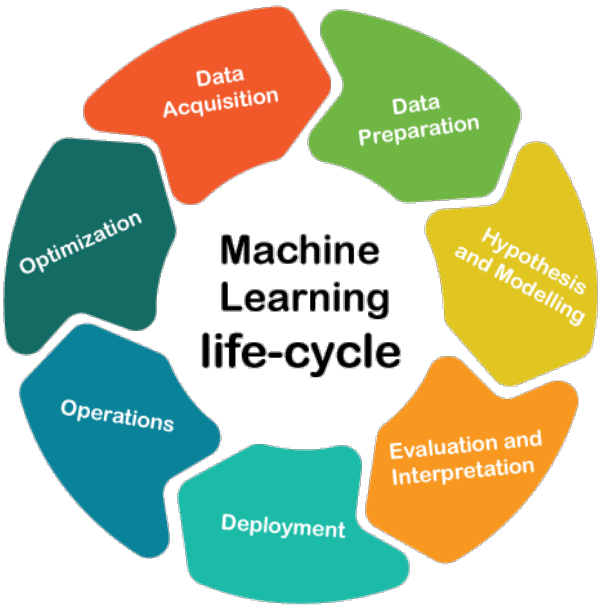


Figure 2.1: Machine learning life cycle.

2.3.2 Machine learning applications

AI and ML are being increasingly used in a wide range of industries, including healthcare, finance, transportation, and education, among others. For example, AI and ML are being used to develop personalized medicine, autonomous vehicles, fraud detection systems, and intelligent tutoring systems [55]. A non-exhaustive list of applications includes:

- ❏ Computer vision uses machine learning to identify objects, people, and other elements in images and videos.
- ❏ Natural language processing uses machine learning to understand and generate human language, including translation and text classification.
- ❏ Recommendation systems use machine learning to suggest products or content based on past user behavior and preferences.
- ❏ Fraud detection can benefit from machine learning algorithms to identify and prevent fraudulent activity in credit card transactions and insurance claims.
- ❏ Healthcare can benefit from machine learning by predicting disease outbreaks, identifying potential outbreaks, and predicting patient outcomes.
- ❏ Finance can benefit from machine learning by predicting stock prices, detecting fraudulent activity, and identifying potential investment opportunities.

2.4 Basic types of machine learning

2.4.1 Supervised learning

Supervised learning is a type of ML in which the computer is provided with labeled example inputs to learn from as illustrated in Figure 2.2. After the algorithm has been trained on this data, it can make predictions on new, unlabeled data by using the patterns it has learned to predict label values. If the model makes errors, it can be adjusted to correct them.

For example, a supervised learning algorithm could be fed data containing images of sharks labeled as "fish" and images of oceans labeled as "water." After being trained on this data, the algorithm should be able to identify unlabeled shark images as fish and unlabeled ocean images as water. This type of machine learning is commonly used to predict statistically likely future events using historical data, such as predicting market trends or filtering spam emails. Another application of supervised learning is in image recognition, where tagged photos of an object can be used to classify untagged photos of the same object. Overall, supervised learning is a powerful tool that can be applied to a wide range of problems, allowing computers to make accurate predictions and decisions based on labeled data [13].

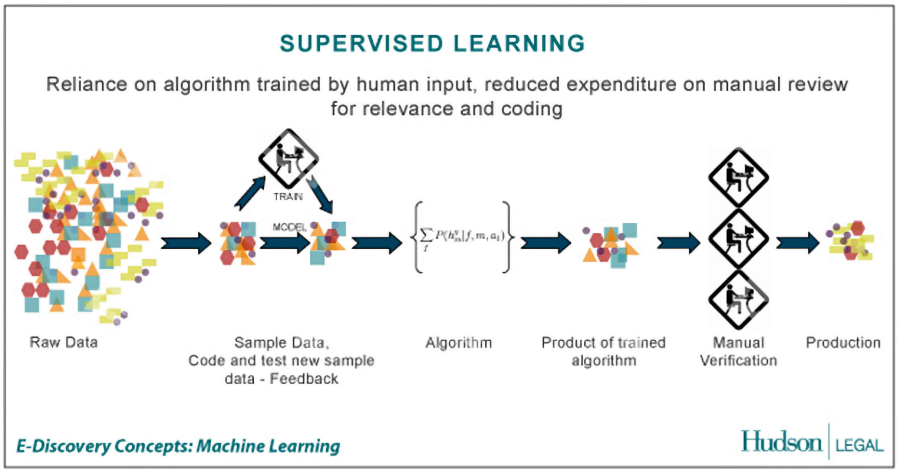


Figure 2.2: Supervised Learning.

2.4.2 Unsupervised learning

Unsupervised learning is a type of ML algorithm where the input data is unlabeled, meaning that there are no corresponding output labels as shown in Figure 2.3. The goal of unsupervised learning is to find patterns and structures in the input data without the help of any external supervision or guidance. Unsupervised learning can be used for a variety of tasks such as clustering, dimensionality reduction, and anomaly detection. In clustering tasks, the algorithm groups similar data points into clusters based on their similarity. In dimensionality reduction tasks, the algorithm reduces the number of features or variables in the input data while preserving as much information as possible. In anomaly detection tasks, the algorithm identifies data points that deviate significantly from the norm or the expected behavior [95].

Unsupervised learning algorithms use different techniques to find patterns and structures in the input data, such as clustering algorithms, principal component analysis (PCA), and autoencoders. These algorithms try to find hidden structures in the input data by reducing the dimensionality, finding similarities or dissimilarities between data points, or reconstructing the input data from a compressed representation. Once the unsupervised learning algorithm has found patterns and structures in the input data, it can be used for various downstream tasks such as classification, anomaly detection, and data visualization.

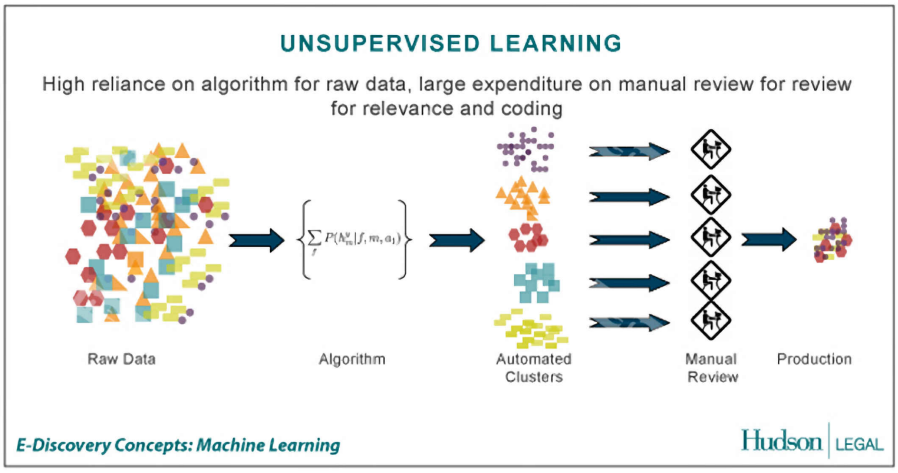


Figure 2.3: Unsupervised Learning.

2.4.3 Semi-supervised learning

Semi-supervised learning is a machine learning technique that resides in between supervised and unsupervised learning. During the training phase, it uses a mix of labeled and unlabeled data to enhance model performance and generalization to new samples as illustrated in Figure 2.4. It is a type of ML that utilizes a small amount of labeled data and a large amount of unlabeled data to train a model [121]. It bridges the gap between unsupervised learning, where there is no labeled data, and supervised learning, where only labeled data is used.

In semi-supervised learning, labeled data contributes to the learning process by delivering explicit information, whilst unlabeled data allows for capturing the underlying data distribution and detecting latent patterns. Semi-supervised learning attempts to enhance model performance by incorporating both kinds of data, particularly when obtaining labeled data can be costly or time-consuming. Semi-supervised learning algorithms can improve the accuracy of predictions and decision-making while reducing the need for human intervention [29]. Various techniques for semi-supervised learning have been developed, including self-training, co-training, and multi-view learning. These approaches make use of the connections between labeled and unlabeled data in order to improve the learning process and obtain better outcomes.

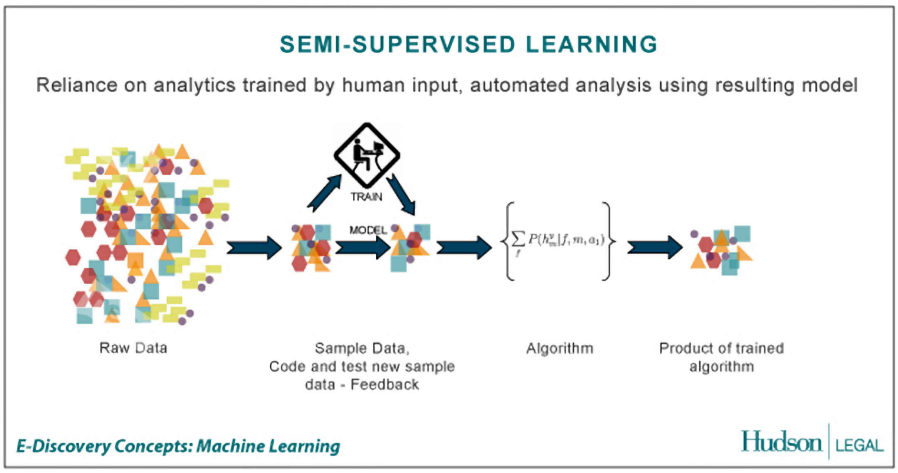


Figure 2.4: Semi-supervised Learning.

2.4.4 Reinforcement learning

Reinforcement Learning(RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences. Though both supervised and reinforcement learning use mapping between input and output, unlike supervised learning where the feedback provided to the agent is correct set of actions for performing a task, reinforcement learning uses rewards and punishments as signals for positive and negative behavior [11, 51].

As compared to unsupervised learning, reinforcement learning is different in terms of goals. While the goal in unsupervised learning is to find similarities and differences between data points, in the case of reinforcement learning the goal is to find a suitable action model that would maximize the total cumulative reward of the agent. The figure 2.5 illustrates the action-reward feedback loop of a generic RL model.

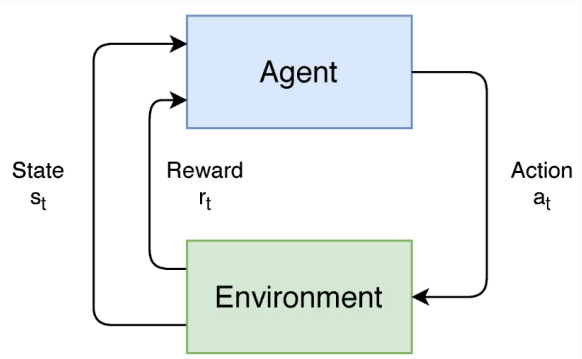


Figure 2.5: The reinforcement learning framework [51].

2.4.5 Transfer learning

Transfer learning is a machine learning technique in which the knowledge acquired from training one model on a specific task is used to enhance the performance of a different but related task. Instead of training a model from scratch on a new task, transfer learning uses the learned representations and knowledge from a previously-trained model. Transfer learning can be both supervised and unsupervised, depending on the availability of labeled data for the target task.

In an untrained deep learning model, the nodes are randomly initialized with weights, which are then optimized using an algorithm specific to the task and dataset during training. However, the authors in [134] demonstrated that initializing the weights based on a trained network with a distant dataset improves the training performance compared to random initialization.

Deep transfer learning differs from semi-supervised learning in that the source and target datasets can have different distributions and may only be related, while in semi-supervised learning, the source and target data are from the same dataset, with the target set lacking labels [137]. Multiview learning, on the other hand, utilizes two or more distinct datasets to enhance the quality of a single task. In Multitask learning, the focus is on using interconnections between tasks to enhance each other, whereas in deep transfer learning, the focus is on the target domain, with the knowledge having already been acquired from source data and not necessarily related or functioning simultaneously [137].

2.5 Deep learning

2.5.1 Introduction

Deep Learning, also known as hierarchical or deep-structured learning, encompasses a range of machine learning techniques that can be supervised or unsupervised. Inspired by the structure and function of the human brain, as well as the processing of signals through neurons, Deep Learning relies on artificial neural networks with multiple hidden layers, each of which responds nonlinearly to input data. Over the past few years, Deep Learning has found widespread applications in areas such as voice and image recognition, object detection, drug discovery, and genomics. Its ability to extract meaningful patterns from complex data has made it a powerful tool for solving a variety of real-world problems [65, 106].

Deep Learning has developed a structure that enables it to handle large datasets by using a backpropagation algorithm to identify how the device modifies its core parameters to calculate representations in each successive layer based on the previous layer [19].

Despite their enormous complexity, successful Deep Neural Networks can achieve only a slight difference between their performance on training data and on test data. According to conventional wisdom, the minor discrepancies in performance can be attributed to the inherent characteristics of the network or to the training techniques used [96].

2.5.2 Definition

The original text is discussing deep learning, which is a high-level abstraction algorithm used to model data from large datasets. The concept of abstraction assumes that there is no simple relationship between the input and output data. The goal is to create a realistic scenario that leads to a realistic classification or result. The process of deep learning typically involves detecting important properties of the input data during the learning process.

The term "deep" comes from neuroscience, specifically from the idea of a neural network. A neural network is a kind of software brain composed of thousands of units (neurons) that perform calculations. These networks were originally called artificial neural networks (ANNs) to distinguish them from biological systems. They typically consist of input and output layers, a narrow network of neurons, and several hidden layers as illustrated in Figure 2.6. These intermediate layers enable the network to process complex problems. The number of layers is a decisive factor for the complexity of the system and the learning process. Data is passed from layer to layer, with the results of one layer serving as the input for the next, and so on, until a complex decision is reached. This layered approach gives depth to the network and the learning process.

2.5.3 Applications of deep learning

Deep Learning has revolutionized the way we approach technology, and its impact on Artificial Intelligence (AI) and its subfields, such as Machine Learning (ML), has been significant. The excitement surrounding Deep Learning is well-justified, as it has already transformed our lives and will continue to do so in the near future. DL has been gaining market share rapidly, and it is expected that DL tools, techniques, and libraries will become ubiquitous in development toolkits within the next five to ten years. In this section, we will discuss some of the Deep Learning applications that have captured the market's attention in 2019 and beyond.

Self-driving cars, powered by digital sensor systems trained through massive amounts of unstructured data, have been a major breakthrough in the automotive industry. DL has also made significant

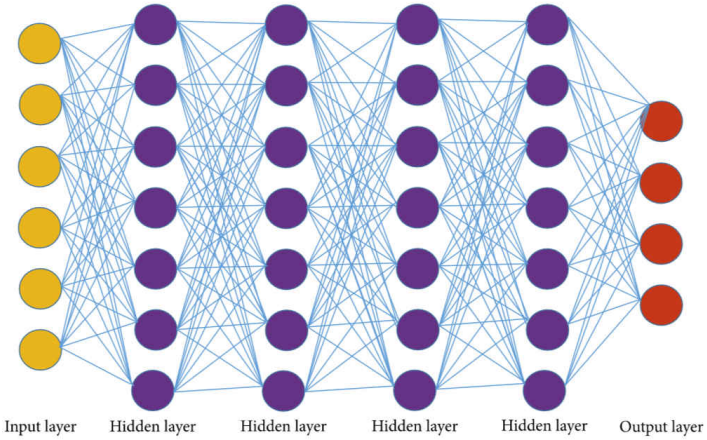


Figure 2.6: Deep Learning neural network [115].

contributions to the field of healthcare, particularly in breast cancer diagnostics and personalized medicine based on Biobank data. Voice recognition and activation, a feature already available on every smartphone, is another famous example of Deep Learning's capabilities, with Google, Apple, and Microsoft Cortana offering voice assistants. The Google Translator is a well-known example of machine translation, with DL improving its results and expanding its capabilities to translate images. Automatic handwriting generation is another area where Deep Learning has made significant contributions, enabling the generation of new writing styles.

There are numerous other applications of Deep Learning, such as image and face recognition, automatic colorization, image captioning, advertising, earthquake prediction, brain cancer detection, price forecasting, natural language processing, gaming, and cybersecurity, among others [115].

2.5.4 Advantages of Deep Learning

Deep learning represents a significant breakthrough in the field of AI and extends beyond machine learning. Its applications encompass a broad range of AI challenges, including but not limited to:

- Advancing conventional AI development tasks.
- Harnessing vast amounts of data, such as big data.
- Adapting to diverse problem domains.
- Automatically extracting relevant features.

2.6 Common issues

Machine learning and deep learning are powerful tools that can be applied to a broad range of problems. They are not, however, without challenges [37]. Among the most frequent problems encountered by machine learning and deep learning practitioners are:

2.6.1 Overfitting

Overfitting occurs when a model learns to perform extremely well on the training data but fails to generalize to new, unseen data. In other words, the model becomes too complex and captures noise or random variations in the training data, leading to poor performance on unseen examples. Signs of

overfitting include a high training accuracy but a significantly lower accuracy on the validation or test data. Overfitting is characterised by:

- Low bias, high variance: Overfit models have low bias because they can fit the training data very well. However, they have high variance because they are too sensitive to the noise or random fluctuations in the data.
- Captures noise: Overfit models tend to capture noise, outliers, or specific examples in the training data that do not represent the true underlying patterns.
- Memorization: Instead of learning general patterns, overfit models can simply memorize the training data, leading to poor generalization on new data.
- Complex models: Overfitting is more likely to occur with complex models that have a large number of parameters relative to the available training data.

2.6.2 Underfitting

Underfitting, on the other hand, happens when a model fails to capture the underlying patterns and relationships in the training data. It occurs when the model is too simple or lacks the capacity to learn the complexity of the data. As a result, the model performs poorly both on the training data and on new, unseen examples. Underfitting is often indicated by low training and validation/test accuracy. Underfitting is characterised by:

- High bias, low variance: Underfit models have high bias because they fail to capture the complexity of the data. Additionally, they have low variance because they do not vary much between different training runs.
- Oversimplified: Underfit models often make simplistic assumptions or have insufficient capacity to represent the patterns present in the data.
- Limited learning: An underfit model might struggle to learn from the training data and, as a result, fails to achieve good performance on both the training and test data.

Techniques such as regularization, dropout, and reducing model complexity (e.g., using fewer layers or parameters) can be used to combat overfitting. On the other hand, underfitting can be reduced by increasing the model's complexity, collecting more training data, or adopting more expressive model architectures. The objective is to find a balance between the model's ability to capture underlying patterns and its tendency to emphasize noise or oversimplify the data.

2.6.3 Insufficient Training Data (Data scarcity)

The term refers to an issue in machine learning where the quantity of labeled or annotated data available for training a model is inadequate or insufficient. A large and diverse dataset is generally important for training accurate and robust machine learning models. Neural networks require a large amount of data to learn effectively. If the training set is too small, the network may not be able to learn the patterns in the data and may make inaccurate predictions.

Insufficient training data may lead to a limited generalization, overfitting, inaccurate predictions, limited applicability in real-world scenarios, restriction of the model's capacity to capture complex patterns. Many solution could be applied to overcome this issue such as data augmentation, transfer learning, active learning and introducing expert insights or domain knowledge for guiding the learning process of the model.

2.6.4 Nonrepresentative Training Data

It describes the situation in which the available training data does not adequately represent the actual distribution or diversity of the real-world problem being explored. When the training data is nonrepresentative, it may lead to biased models and poor generalization performance.

This issue could be addressed using many methods such as regularization techniques, data augmentation, data balancing or post-hoc bias mitigation.

2.6.5 Poor-Quality Data

This means data that is noisy, incomplete, incorrect, inconsistent, or contains errors (inaccurate Labels or annotations). The performance and reliability of machine learning models may be seriously compromised by poor data quality. Several strategies can be applied to reduce the impact of low-quality data on machine learning including data cleaning (discarding outliers and incomplete instances), feature engineering, data augmentation or bias detection and mitigation.

2.6.6 Irrelevant Features

The system can only be able to learn if the training data comes with a sufficient number of relevant features and a too small amount of irrelevant ones. Developing an adequate set of training features is crucial for the success of any machine learning project and is called **feature engineering**. It consists of feature selection (the most useful features are selected to train on among existing features), feature extraction (combining existing features to create a more appropriate one, dimensionality reduction algorithms may be useful) and creating new features by gathering new data.

2.6.7 Vanishing gradients

It occurs when the gradients of the loss function become too small to adjust the network's weights. This can hinder the network's ability to learn effectively and is more likely to occur in deep neural networks with many layers.

2.6.8 Exploding gradients

Gradient explosion refers to a phenomenon that happens when the gradients of the loss function become excessively large, causing the network's weights to update excessively as well. This can result in an unstable network that is unable to learn effectively. Gradient explosion is more likely to occur in deep neural networks that have numerous layers.

2.6.9 Computational complexity

Training neural networks can be a computationally demanding task. The time required for training can grow exponentially as the network's depth and the number of parameters increase.

2.6.10 Interpretability

Neural networks are often seen as black boxes, meaning that it is hard to understand how they make predictions. This can make it harder to debug the network and to understand why it makes some predictions.

2.7 Neural networks

In 1943, W. McCulloch and W. Pitts pioneered the concept of neural networks, which demonstrated the brain's equivalence to a Turing machine. This suggested that thought could be explained by material and logical mechanisms. Their work was instrumental in the development of cybernetics, leading them to conclude in 1955 that organisms are not simply analogous to machines, but that the machine itself should be considered [73].

D. Hebb introduced a learning rule in his book "The Organization of Behavior" in 1949, which continues to inspire many network models today [45]. In 1958, F. Rosenblatt developed the Perceptron model, a neural network that takes inspiration from the visual system and features two layers of neurons - a perception layer and a decision-making layer. This was the first artificial system capable of learning through experience. During the same period, B. Widrow presented the ADALINE model (ADAptive LINear Element) [129].

An American researcher at Stanford University developed the ADALINE model, which later became the basic model of multilayer networks. In 1969, M. Minsky and S. Papert published a review of the properties of the Perceptron, which had a significant impact on research in this field. Research activity declined until 1972, when T. Kohonen introduced his work on associative memories and its potential applications to pattern recognition [63]. J. Hopfield's analysis of the dynamics of a completely looped network was presented in 1982 [49].

2.7.1 Feed-Forward network

A Feed-Forward network is a type of neural network in which the flow of information moves in one direction only, from the input layer to the output layer, without any loops or cycles. This means that the output of each neuron in one layer serves as input to neurons in the next layer. This creates a hierarchy of representations, with each layer extracting more abstract features from the input data.

The first layer of the network receives the input signals and subsequent layers receive inputs only from the previous layer, along with a bias signal source. The bias signal source allows the network to shift the activation function left or right, which can be useful in certain applications. The neurons in each layer perform a weighted sum of their inputs, followed by an activation function, which determines the output of the neuron.

Feed-Forward networks can be used for a variety of applications, such as ECG abnormality detection, speech recognition, sentiment classification, balancing tasks, sensor signal processing, and plant control. In function approximation tasks, the network is trained to approximate a mathematical function that maps inputs to outputs. In pattern classification tasks, the network is trained to classify inputs into one of several categories, based on their features.

In pattern classification, the network learns to distinguish between different classes of inputs by adjusting the weights and biases of the neurons during training. The training process involves presenting the network with a set of labeled examples, and adjusting the weights and biases to minimize the difference between the network's output and the true label. Once the network has been trained, it can be used to classify new, unseen examples with high accuracy.

2.7.2 Deep neural network (DNN)

A Deep Neural Network (DNN) is a type of artificial neural network that is composed of multiple layers of interconnected nodes, also known as neurons. It is a machine learning model inspired by the structure and functioning of the human brain. DNNs are capable of learning and extracting

hierarchical representations from input data, enabling them to effectively model complex relationships and make predictions or classifications. Each layer in a DNN consists of multiple neurons that apply nonlinear activation functions to the weighted sum of their inputs. The hidden layers between the input and output layers allow the network to learn increasingly abstract and complex features as information propagates through the network. The training process of a DNN involves iteratively adjusting the weights and biases of the neurons to minimize a specific loss function and optimize the network's performance on a given task. DNNs have demonstrated remarkable success in various domains, including computer vision, natural language processing, and speech recognition [14, 30].

2.8 Deep Learning Architectures

This section covers the most famous types of deep learning networks, including recursive neural networks (RvNNs), RNNs, and CNNs. While RvNNs and RNNs are briefly explained, CNNs are given more in-depth coverage due to their significance and widespread use in various applications when compared to other networks. DL architectures are broadly classified as discriminative, generative and hybrid as illustrated in Figure 2.7.

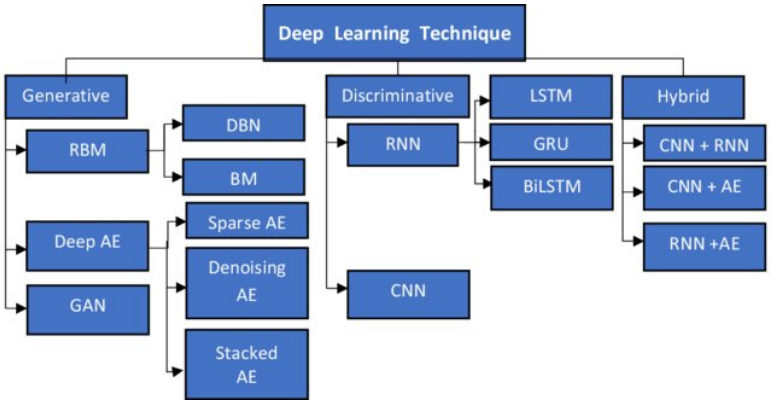


Figure 2.7: Classification of deep learning architectures [52].

2.8.1 Discriminative Architectures

Discriminative architectures, also known as supervised architectures, are often used with labelled data to separate patterns for prediction applications. The most typical discriminative deep learning architectures are listed below:

2.8.1.1 Convolutional neural networks

Convolutional neural networks, also known as convnets or CNNs, have proven to be highly effective for tasks that involve closely related data, particularly in the field of computer vision. A CNN consists of two main parts: the convolutional part and the classification part. The convolutional part of the model is a multilayer neural network that is composed of four types of layers: the convolution layer, the pooling layer, the ReLU correction layer, and the fully connected layer. The second part of the model is a multilayer perceptron (MLP) that performs the classification task [136].

Convolutional Neural Networks for Sentence Classification CNNs are deep neural networks, meaning that they have multiple layers that enable them to learn hierarchical representations of the input data.

2.8.1.1.1 Convolution layer

The convolution layer is considered the most important layer in a CNN, as it performs the most computationally intensive operations. Its main function is to detect the presence of specific features in the input data. The convolution layer achieves this by applying a set of filters to the input data, which are designed to recognize certain patterns or features in the data. These filters are then convolved with the input data, producing a set of feature maps that highlight the areas of the data where the desired features are present. This process allows the network to learn features at different levels of abstraction, starting with simple features such as edges and corners and building up to more complex features as the network gets deeper.

2.8.1.1.2 Pooling layer

The pooling layer is typically placed between two convolution layers in a convolutional neural network. Its function is to receive multiple feature maps as input and apply the pooling operation to each of them. The pooling operation reduces the size of the feature maps while retaining their important characteristics. This is achieved by dividing the feature map into small sub-regions, and computing a summary statistic for each sub-region, such as the maximum or average value. This summary statistic is then used to represent the sub-region in the pooled output. Pooling helps to reduce the dimensionality of the feature maps, making the network more computationally efficient, and also helps to reduce overfitting by introducing a degree of translation invariance into the network.

2.8.1.1.3 The ReLU correction layer

The ReLU correction layer in a convolutional neural network functions as an activation function that sets all negative input values to zero and passes positive values through unchanged. In doing so, it introduces non-linearity into the network, allowing it to model complex relationships between the input and output data. This layer helps to speed up training by reducing the number of parameters that need to be updated during backpropagation.

2.8.1.1.4 The fully-connected layer

The fully-connected layer is typically the final layer of a neural network, including convolutional networks, and is not unique to CNNs. This layer takes a vector as input and produces a new vector as output by applying a linear combination to the input values, followed by an optional activation function. This allows the network to learn complex patterns and relationships in the input data and make predictions about the output.

2.8.1.2 Recurrent Neural Network (RNN) and Recursive Neural Network (RvNN)

A dynamic feed-forward neural network known as an RNN was first developed by Hopfield in 1982. It stands out thanks to its capacity for sequential data learning across timesteps. In typical feed-forward neural networks (FFN), each unit's output is independent of its previous output and only relies on the current input. However, certain applications, like voice recognition, rely on sequential data, while others use time-series data, like sensor data, where each sample relies on the interpretation of earlier examples. Therefore, these applications are not suitable for the traditional feed-forward neural network. RNNs solve this issue by modelling data as time series. Figure 2.8 illustrates the difference between hidden units in RNNs and FFNs.

RNNs have been extended with different memory unit variants, including:

- **Long short time memory (LSTM):** The vanishing gradient issue in a basic RNN is resolved by

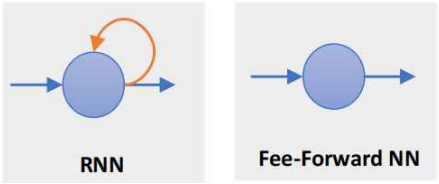


Figure 2.8: Difference between hidden units in RNN and feed-forward neural networks [7].

LSTM. Using the gating mechanism, it has the capacity to learn enduring dependencies. Each LSTM unit has a memory cell that stores previous states.

- **Gated recurrent unit (GRU):** A simplified variant of LSTM is GRU. It is built with a more straightforward design that integrates the states and merges the gates.

RvNNs are capable of making predictions in a hierarchical structure while also classifying outputs using compositional vectors. The development of RvNN architecture was primarily inspired by the recursive auto-associative memory [40] (RAAM) approach. RvNNs are specifically designed for processing objects with randomly shaped structures like graphs or trees, and they generate a fixed-width distributed representation from a variable-sized recursive-data structure. The network is trained using a back-propagation through structure (BTS) learning system [110], which is based on the general back-propagation algorithm and can support a tree-like structure.

RvNNs are highly effective in the context of natural language processing (NLP) [110]. Socher et al introduced an RvNN architecture for processing inputs from a variety of modalities and demonstrated two applications for classifying natural language sentences [69]. RvNN computes a likely pair of scores for merging and constructs a syntactic tree. It then calculates a score related to the merge plausibility for every pair of units and merges the pair with the largest score within a composition vector. After every merge, RvNN generates (a) a larger area of numerous units, (b) a compositional vector of the area, and (c) a label for the class. The compositional vector for the entire area is the root of the RvNN tree structure. RvNNs have been employed in several applications [69, 92, 118].

2.8.2 Generative Architectures

Deep learning architectures that are generative (or unsupervised) may autonomously learn from unlabeled raw data to complete various tasks. The most prevalent architectures in this group are listed below.

2.8.2.1 Auto-Encoder (AE)

An Auto-Encoder (AE) is a deep neural network commonly used for dimensionality reduction by generating improved output data representation than the raw input data. In addition to a hidden layer with low-dimensional feature representation, it has an equal number of feature vectors in both the input and output layers. Backpropagation is used to train an AE, which combines an encoder and a decoder. The encoder converts the input into low-dimensional abstraction in order to retrieve the raw features and learn the data representation. After receiving the low-dimensional representations, the decoder reconstructs the initial features.

Figure 2.9 illustrates the conceptual structure of an AE. There are several AE extensions, such as:

- **Stacked AE (SAE):** The deep network of the SAE is created by cascading many hidden layers. In order to build a new data representation, the input features are progressively learned in in depth.

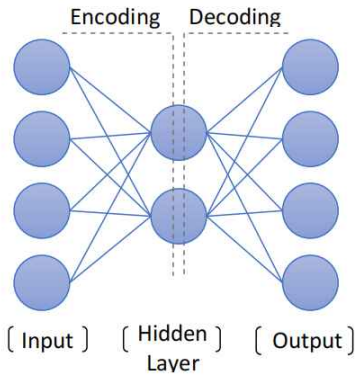


Figure 2.9: The conceptual structure of an AE [7].

- **De-noising AE:** De-noising is the process of producing a cleaner data representation by starting with corrupted data and using only robust feature vectors in the hidden layers.
- **Sparse AE:** In sparse AE, the hidden layers are subject to sparsity constraints. Despite the fact that there are many hidden units, the AE is still important for learning data representations. By turning a large number of neurons inactive most of the time, sparsity constraints attempt to generate low average output.

A deep autoencoder is an autoencoder with multiple hidden layers in both the encoder and decoder. The network can learn more complex representations of the input data by adding more hidden layers. Each hidden layer in the encoder reduces the dimensionality of the data, thereby capturing abstractions and features at a higher level. The decoder layers then expand the representation to the dimensions of the original input.

Deep autoencoders have been applied widely for many applications, including dimensionality reduction, feature learning, data denoising, and outlier detection. The learned representations are useful for further processing, such as classification or clustering, or to generate new samples that are comparable to the input data.

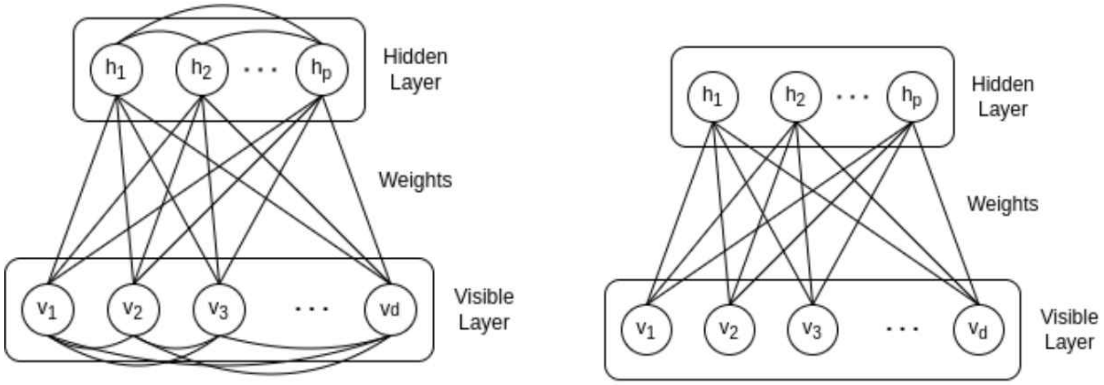
Deep autoencoders are robust models that may identify meaningful representations of complex data by using multiple layers of nonlinear transformations.

2.8.2.2 Restricted Boltzmann Machine (RBM)

Hinton and Sejnowsk proposed the Boltzmann machine (BM), a probabilistic neural network [1]. A BM network determines which binary units are active by pairing them symmetrically. But since there are so many links between the components, the learning process is relatively slow.

Smolensky suggested RBM, a unidirectional model, in 1986 to address problems brought on by BM's complexity. The purpose of RBM is to remove connections between neurons of the same layer. RBM is often used as the preliminary stage of another learning network, either as a feature extractor during preprocessing or to determine the other network's parameters. RBM may also be used as a classification model. A deep Boltzmann machine (DBM) is one that has several cascading Boltzmann machines.

Figure 2.10 illustrates the difference between a BM and an RBM which lies in the fact that the RBM contains fewer connections than the BM. There are no connections between nodes of the same layer in an RBM. The number of nodes in the hidden and visible layers is denoted by p and d, respectively.



(a) Boltzmann Machine (BM). *(b) Restricted Boltzmann Machine (RBM).*

Figure 2.10: Difference between BM and RBM architectures.

2.8.2.3 Deep Belief Network (DBN)

As shown in Figure 2.11, a deep belief network (DBN) is built up of stacked RBMs that have undergone greedy layer-wise training.

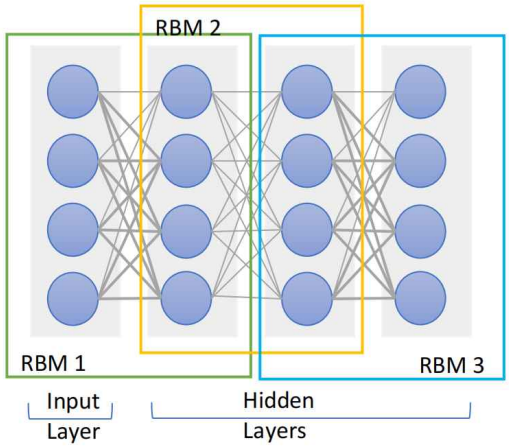


Figure 2.11: The conceptual structure of a DBN [7].

Each of the RBM is trained on the preceding one, with each hidden layer serving as an input for the next RBM. An efficient and fast deep learning algorithm is produced by this training mechanism. When an extra discriminating layer is included, a DBN is used in practical applications for both dimensionality reduction and as a standalone classifier.

2.8.3 Hybrid Architectures

Models from both discriminative and generative domains are included in hybrid architectures. In order to distinguish data, this uses generative features at earlier stages and discriminative features at later stages. In deep learning, hybrid architectures are defined as the integration of different neural network types, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and/or multilayer perceptrons (MLPs), to create a single model. The purpose of these architectures is combining the strengths of each network type to achieve better performance on a given task. For example, a hybrid CNN-RNN model can be used for image captioning, where the CNN extracts features from an image and the RNN generates a textual description based on those features. Another example is a hybrid MLP-CNN model for sentiment analysis, where the MLP performs word embeddings and the CNN extracts features from the text [44].

Hybrid architectures have been shown to outperform single-network models in many tasks, including speech recognition, image classification, and natural language processing. However, designing and training these models can be challenging, and requires careful consideration of network architecture, hyperparameters, and optimization techniques. Some examples of hybrid architectures in deep learning include Deep Residual Learning for Image Recognition (ResNet), Google's Inception-v4, and Neural Machine Translation by Jointly Learning to Align and Translate (GNMT).

2.8.3.1 Generative Adversarial Network (GAN)

Generative Adversarial Networks (GANs) are a category of machine learning models introduced by Ian Goodfellow and his colleagues in 2014 [41]. They are designed to generate new samples that resemble a given training dataset. They consist of two main components: a generator network and a discriminator network as illustrated in Figure 2.12.

A GAN is based on a minimax competition in which one network attempts to maximize the function value and the other network attempts to minimize it. In each adversarial round, the generator network creates synthetic samples using random noise as input. Its purpose is to generate samples that are sufficiently convincing to fool a discriminator into classifying those samples as real. Contrarily, the discriminator network attempts to differentiate between real and generated samples. Its purpose is to accurately determine the input data's source. The generator performs well when it successfully floods the discriminator while the discriminator is trained to be an accurate classifier.

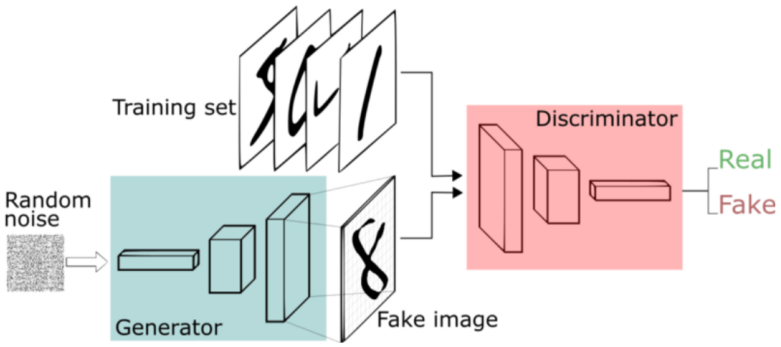


Figure 2.12: Generative Adversarial Network framework [108].

A feedback mechanism is incorporated into the GAN training process. Alternately, gradient-based optimization techniques, such as stochastic gradient descent, are used to update the generator and discriminator. This training process is repeated until the generator produces synthetic samples that are indistinguishable from real samples, or until the desired level of performance is reached.

2.9 Conclusion

The journey through the realm of deep learning has been both challenging and exhilarating. Throughout this chapter, we have explored the foundations, applications, and advancements of deep learning, witnessing its transformative power in various fields like security . Deep learning has reshaped the way we perceive and interact with the world.

As we delved into the intricacies of neural networks, we discovered the immense potential of deep learning algorithms to extract meaningful insights from complex data. We marveled at the capabilities of convolutional neural networks, recurrent neural networks, and generative adversarial networks, among others, in solving intricate problems like detecting intrusions in systems with unprecedented accuracy. We witnessed the impact of deep learning in revolutionizing industries

and fueling technological advancements. From self-driving cars to voice assistants, from medical diagnoses to personalized recommendations, deep learning has proven to be a driving force behind these innovations. Its ability to learn from vast amounts of data and make informed decisions has unlocked endless possibilities for the future.

Also, we must acknowledge the challenges and limitations that accompany this cutting-edge field. The need for substantial computational resources, concerns about data privacy, and the interpretability of deep learning models are just a few areas that demand further exploration and refinement. As we embark on our individual paths, armed with the knowledge and skills acquired during our study of deep learning, let us remember the importance of ethics and responsible AI development. It is our responsibility to ensure that the applications of deep learning are guided by principles of fairness, transparency, and accountability. The world of deep learning is an ever-evolving landscape that holds incredible potential. Our journey through this chapter has provided us with a solid foundation, but it is merely the beginning.

CHAPTER 3

INTRUSION DETECTION SYSTEMS (IDS)

Chapter 3

Intrusion detection systems (IDS)

Contents

3.1 Introduction **58**
3.2 Network intrusion **58**
3.3 Types of Wireless Sensor Networks attacks **59**
3.3.1 Active Attacks 60
3.3.2 Passive Attacks: 63
3.4 Detection methodology **63**
3.4.1 Signature-based detection 63
3.4.2 Anomaly-based detection 64
3.5 General architecture **65**
3.6 Types of intrusion detection systems **66**
3.6.1 Network-based Intrusion Detection Systems (NIDS) 67
3.6.2 Network Node Intrusion Detection System (NNIDS) 67
3.6.3 Host-Based Intrusion Detection Systems (HIDS) 68
3.6.4 Hybrid Intrusion Detection Systems 69
3.7 Conclusion **69**

3.1 Introduction

As the Internet becomes increasingly utilized by individuals and businesses, the frequency of cyber-attacks and intrusions continues to rise. Cybersecurity must prioritize the implementation of Intrusion Detection Systems (IDS) as a crucial measure. IDS is employed to identify successful breaches even after they have occurred [4]. The term "intrusion detection system" was initially coined by James Anderson [10] during the late 1970s and early 1980s. Anderson introduced the concept of identifying misuse and predefined events, laying the foundation for future IDS design and development. An IDS refers to software or hardware designed to detect any malicious activity or attack targeting a system or network. It gathers data from various sources within a computer or network, such as system commands, logs (system, security, and network), system accounting, and network logs. Subsequently, it analyzes this data to identify potential security violations, issuing alerts to system administrators for appropriate response to the intrusion. [12, 78] summarized the functions of IDS as follows: monitoring and analyzing both user and system activities, evaluating system configurations and vulnerabilities, assessing system and file integrity, recognizing patterns typically associated with attacks, analyzing abnormal activity patterns, and tracking violations of user policies. There are two primary types of IDS: Network-based IDS (NIDS) and Host-based IDS (HIDS) [112]. NIDS is deployed along a network to monitor all network traffic [5], while HIDS is placed on a host to scan and monitor all processes and devices within the network [112]. Additionally, there exist other types of IDS, although limited research has been conducted on these variations. The objective of this paper is to compare and present different types of intrusion detection systems based on the platform and data they collect for detecting intrusion analysis. Furthermore, the research will outline the advantages and disadvantages of these IDS types, aiming to identify leading trends, unresolved issues, and potential future directions for further investigation [128]

3.2 Network intrusion

An unauthorized infiltration into a network or a specific computer address within the designated domain is referred to as a "network intrusion." There are two types of intrusions: inactive (where the infiltration occurs discretely and undetected) and active (in which network resources are changed and perceived).

Both internal and external intrusions into the network can occur (such as from an employee, a customer, or a business partner). Some intrusions are just intended to get people's attention by destroying a website with abusive messages or images. Some, with greater malicious purpose, are out stealing sensitive data, either once and for all or as part of a continuing parasitic attachment that siphons off information until it is uncovered. Some intruders inject meticulously written code, such as malware in the form of Trojans, for stealing passwords, keylogging, or unlocking an application's "back door".

Even worse, some skilled hackers can make fake websites that look exactly like a company's website and trick innocent users into going to them. (known as a "man in the browser attack"). Some hackers can steal information quietly until they are found. They do this by becoming passive and becoming part of a network, like parasites.

A system can be compromised physically (by directly getting access to a limited machine's hard disk and/or BIOS), externally (by targeting the web sites or figuring out a means to detour the firewall), or internally (through own users, customers, or partners) [119].

3.3 Types of Wireless Sensor Networks attacks

A comprehensive review has been conducted on the privacy and security concerns in wireless sensor networks (WSNs). In the context of WSNs, two primary types of attacks are primarily attributed to the nature of the transmission medium: Active Attacks and Passive Attacks . Active attacks involve deliberate attempts by attackers to search for and destroy information within the network. On the other hand, passive attacks focus on stealing valuable information such as passwords and confidential data without altering or disrupting the network's normal functioning. To ensure secure communication between sensor nodes in wireless networks, both types of attacks need to be considered by users and organizations.

Figure 3.1 provides a classification of various security attacks in WSN, offering an overview of the different types of attacks that can occur in the network.

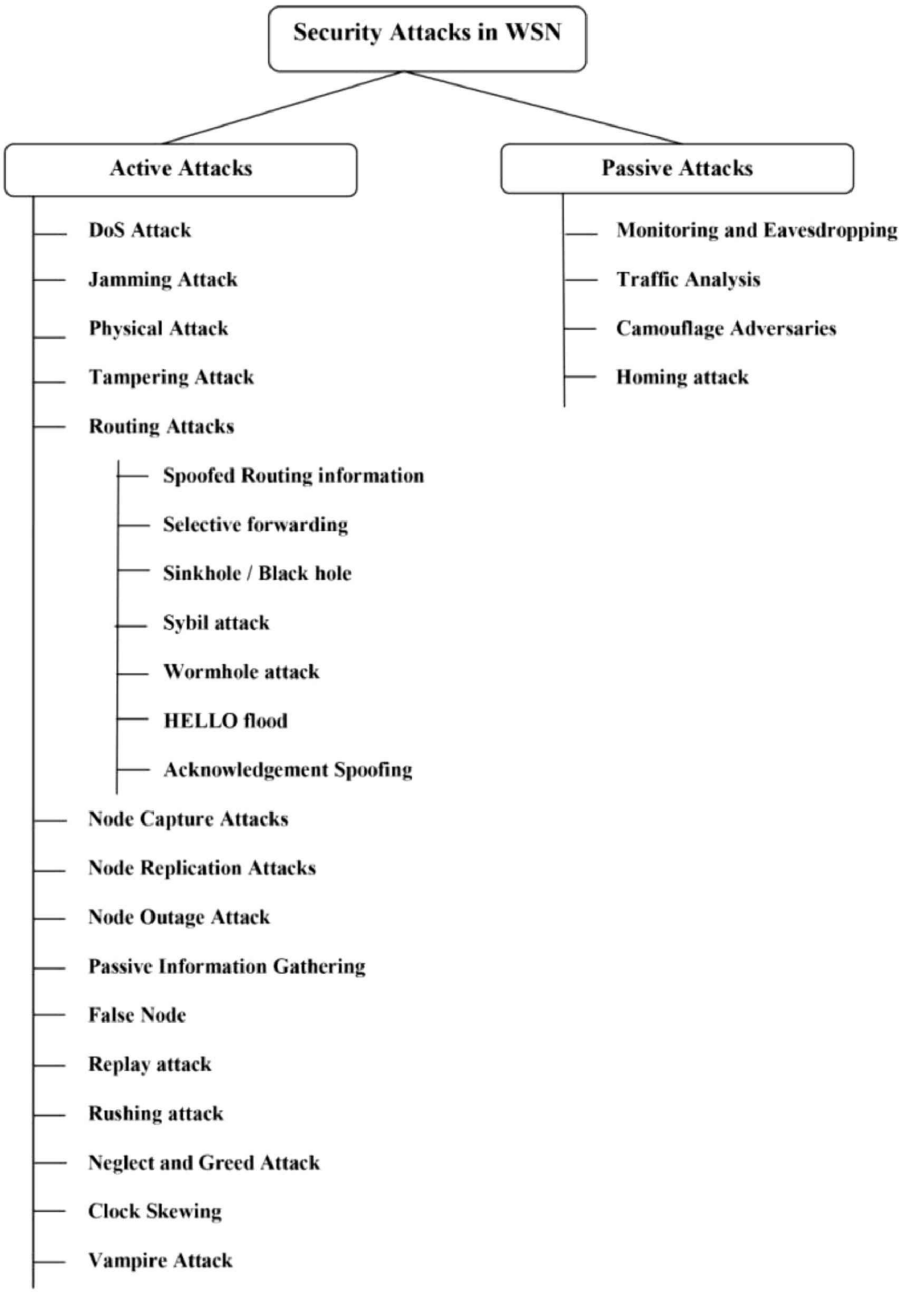


Figure 3.1: Classification of WSN security attacks [58]

3.3.1 Active Attacks

DoS Attack: Attackers disrupt network services by overwhelming the network with malicious traffic or exploiting vulnerabilities in the infrastructure, resulting in a loss of availability and resources [105].

Jamming Attack: Attackers interfere with legitimate wireless communication by using high-power transmitters, preventing the transmission of packets and disrupting the network's normal operation [123].

Physical Attack: WSN nodes are vulnerable to physical attacks due to their distributed and unattended nature. Attackers physically destroy nodes, leading to permanent loss and disruption in the network [20].

Tampering Attack: Attackers modify or damage nodes, gaining control over them and compromising their resources. Proper key management and frequent key changes help prevent this attack [27].

Routing Protocol Attacks:

Spoofed Routing Information: Attackers disrupt network functionality by impersonating entities and misleading legitimate nodes, causing communication disruptions. MAC authentication and the use of secret keys can prevent this attack [83].

Selective Forwarding: Attackers selectively forward specific packets or nodes while dropping others, causing disruptions in multi-hop WSNs. Defense mechanisms include multi-path routing, node monitoring, and finding alternative routing paths [124].

Sinkhole Attack: Compromised nodes divert traffic by advertising false routing metrics, leading neighboring nodes to forward packets to the malicious node. Certificate-based authentication can help prevent this attack [85].

Blackhole Attack: Malicious nodes act as black holes, intercepting and manipulating data packets without forwarding them. Network monitoring, dynamic packet routing, and authentication mechanisms can mitigate this attack [56].

Sybil Attack: Attackers create multiple identities or duplicate themselves to deceive nodes in the network. Authentication and encryption techniques can help detect and prevent this attack [56].

Wormhole Attack: Attackers create tunnels between distant nodes to redirect and manipulate packet flow. Packet leashes and temporal/geographical constraints can help detect and prevent this attack [39].

HELLO Flood Attack: Attackers flood the network with fake "Hello" messages, pretending to be parent nodes. Identity verification protocols and packet leash mechanisms can mitigate this attack [38].

Acknowledgment Spoofing: Attackers spoof acknowledgment messages to deceive neighboring nodes about their status or energy levels. Bi-directional link verification can help prevent this attack [124].

Node Capture Attack: Attackers compromise sensor nodes, perform various operations, and manipulate information in the communication channel, leading to network-wide compromise and attacks [20]. Node Replication Attack:

The node replication attack, also known as a clone attack, occurs when the wireless sensor network (WSN) is exposed to an insecure environment, allowing malicious nodes to be replicated into multiple clones. These replicated nodes are equipped with legitimate IDs and keys, enabling them to communicate with other nodes as if they were normal nodes in the operational network. This attack poses a significant threat to the security and integrity of the network [131].

To counteract the node replication attack, a countermeasure is to provide a unique pair-wise key that supports secure communication between neighboring nodes. By ensuring that each node has a distinct key for communication, it becomes difficult for the attacker to replicate the nodes and compromise the network.

Node Outage Attack:

The node outage attack aims to disrupt the functionality of wireless sensor components such as sensor nodes, communication links, or parent nodes within the network. By completely stopping the operation of these components, communication with other clustered nodes in different areas is interrupted, leading to a loss of connectivity and potential data loss [20].

To mitigate the impact of node outage attacks, time protocols are designed in such a way that they provide packets with alternate routing paths. By establishing redundancy and alternate routes in the network, the communication can be rerouted and maintained even if some nodes or communication links are affected by the attack.

Passive Information Gathering:

Passive information gathering attacks involve the use of powerful algorithms by attackers to intercept messages within the wireless sensor network. Through this interception, the attacker can gather sensitive information, including the physical location of sensor nodes and access to application-specific message contents. This attack exploits unencrypted information within the network [89].

To counteract passive information gathering attacks, several measures can be taken. These include using well-designed antennas with encrypted data transmission to enhance the security of the network. Additionally, the use of powerful encryption algorithms can help protect the confidentiality of the transmitted data, making it difficult for attackers to extract meaningful information from intercepted messages.

False Node Attack:

In a false node attack, a malicious node is inserted into the network with inaccurate data or by impeding the channel of correct data transmission. The false node then sends incorrect data that reaches all nodes in the operational network, potentially compromising the entire network or causing its complete failure [89].

To counteract false node attacks, an en-routing scheme is employed. This scheme involves implementing mechanisms that verify the authenticity and integrity of the nodes within the network, such as using digital signatures or cryptographic techniques. By ensuring that only legitimate nodes are allowed to participate in the network, the risk of false node attacks can be mitigated.

Replay Attack:

In a replay attack, the attacker repeatedly replays malicious nodes within the wireless sensor network, leading to increased energy consumption and domination of communication channels. This attack becomes more challenging to mitigate due to the dynamic nature of the network's topology, influenced by the mobility of sensor nodes. One effective approach to overcome replay attacks is to

introduce session tokens and include timestamps with the messages. Session tokens help ensure the freshness and integrity of the messages, while timestamps provide a means to detect and discard replayed messages [53].

Rushing Attack:

A rushing attack occurs in on-demand routing protocols when a malicious node rushes to obtain data from a neighboring node and deliver it to another destination node through a different tunnel. This attack is similar to a wormhole attack, where packets are tunneled through the network at a faster rate than usual multi-hop routes, potentially bypassing normal routing protocols [16].

To defend against rushing attacks, a countermeasure is to provide route records by embedding a node list. By maintaining a list of nodes traversed by the packets, the network can detect and identify anomalies in the routing paths. This helps ensure that packets are delivered through legitimate routes and prevent malicious nodes from rushing data through unauthorized tunnels

Neglect and Greed Attack:

In a neglect and greed attack, a malicious node deliberately selects the longest path to send packets within the wireless sensor network. By routing the packets to the wrong node, the attacker aims to cause information loss and disrupt the availability of the network. The malicious node receives the packets but refuses to forward them to neighboring nodes, leading to communication failures and potential data loss [124].

To prevent neglect and greed attacks, a combination of authentication mechanisms and malicious node detection techniques can be employed. Authentication mechanisms help ensure that only legitimate nodes are allowed to participate in the network, reducing the chances of malicious nodes performing such attacks. Additionally, implementing mechanisms to detect and identify malicious nodes, such as anomaly detection algorithms or behavior-based analysis, can help mitigate the impact of neglect and greed attacks.

Clock Skewing Attack:

In a clock skewing attack, the attacker manipulates the timestamps of forwarding packets within the wireless sensor network. By altering the timestamps, the attacker can disrupt the synchronization of devices and compromise the accuracy of time-sensitive operations. The clock skew of each physical device in the network may vary based on the application's requirements [50].

To defend against clock skewing attacks, the flooding time synchronization protocol (FTSP) can be utilized. FTSP adjusts the time synchronization period, which helps detect and prevent clock skew attacks. By periodically synchronizing the clocks of sensor nodes and adjusting the synchronization frequency, the network can maintain accurate timekeeping and defend against timestamp manipulation.

Vampire Attack:

A vampire attack is a type of Denial-of-Service (DoS) attack that targets the power consumption of sensor nodes, ultimately disabling the network. In this attack, the attacker exploits vulnerabilities in the network's routing protocols or energy management mechanisms to drain the battery power of sensor nodes, rendering them unable to perform their intended functions [102].

To counteract vampire attacks, various validations and safeguards are implemented. These measures include ensuring that packets are not transmitted in infinite loops, as this can lead to excessive power consumption and network disablement. Additionally, implementing energy-efficient

routing protocols and optimizing energy management strategies can help mitigate the impact of vampire attacks by conserving power and prolonging the network's operational lifespan.

3.3.2 Passive Attacks:

Passive attacks in wireless sensor networks involve unauthorized monitoring and eavesdropping on the communication channels without altering or damaging the messages. These attacks serve as a preliminary stage for potential active attacks [20]. Here are some common passive attacks:

1. **Monitoring and Eavesdropping:**
   Malicious nodes intercept and listen to the communication within the network, violating the privacy of the transmitted data. This attack aims to gather information without modifying the network's integrity.

2. **Traffic Analysis:**
   Attackers analyze communication patterns to gain insights into network behavior and exploit vulnerabilities for potential active attacks. Regular monitoring and detection mechanisms can help mitigate this attack.

3. **Camouflage Adversaries:**
   Attackers hide and mimic legitimate nodes within the network to misroute packets and disrupt communication channels.

4. **Homing Attack:**
   Attackers aim to gain insights into the network's internal resources without modifying the packets. This information can be later utilized to launch active attacks. Traffic pattern analysis and header encryption techniques can help detect and prevent homing attacks [124].

Passive attacks are focused on gathering information and understanding the network's behavior, laying the groundwork for potential active attacks. Implementing security measures such as encryption, traffic monitoring, and anomaly detection can help protect against these attacks and ensure the privacy and integrity of the wireless sensor network.

3.4 Detection methodology

Generally, IDS systems may be classified into two groups: Signature-based Intrusion Detection Systems (SIDS) and Anomaly-based Intrusion Detection Systems (AIDS).

3.4.1 Signature-based detection

Signature detection is a type of intrusion detection that uses pre-defined patterns or signatures to identify known attacks. It is also known as *Knowledge-based Detection* or *Misuse Detection*. In this approach, the deep learning model is trained on a dataset of labeled network traffic containing known attack patterns. The aim of Signature-based Intrusion Detection System (SIDS) is to identify these known attack patterns in real-time when they occur in the network. Signature detection is particularly useful when there is prior knowledge of the types of attacks that might occur on the wireless sensor network.

The steps involved in signature detection using deep learning are:

1. **Data preprocessing:** The data collected from the wireless sensor network is preprocessed in the same way as in anomaly detection.
2. **Feature extraction:** This step involves identifying relevant features from the preprocessed data that can be used to train the deep learning model. However, in signature detection, the features may be specific to the known attack patterns.
3. **Model selection and training:** The deep learning model is selected and trained on the dataset of labeled network traffic containing known attack patterns.
4. **Model evaluation:** The performance of the model is evaluated on a test dataset containing known attack patterns and normal network traffic.
5. **Model deployment:** If the model performs well on the test data, it can be deployed in the wireless sensor network to detect known attacks.

Signature detection using deep learning has several advantages. It can detect known attacks with high accuracy and low false positives. It is also effective in detecting new variants of known attacks.

Both anomaly detection and signature detection using deep learning have their strengths and weaknesses. The choice of which approach to adopt depends on the specific needs of the network. The architecture of a signature-based intrusion detection system is illustrated in figure 3.2.

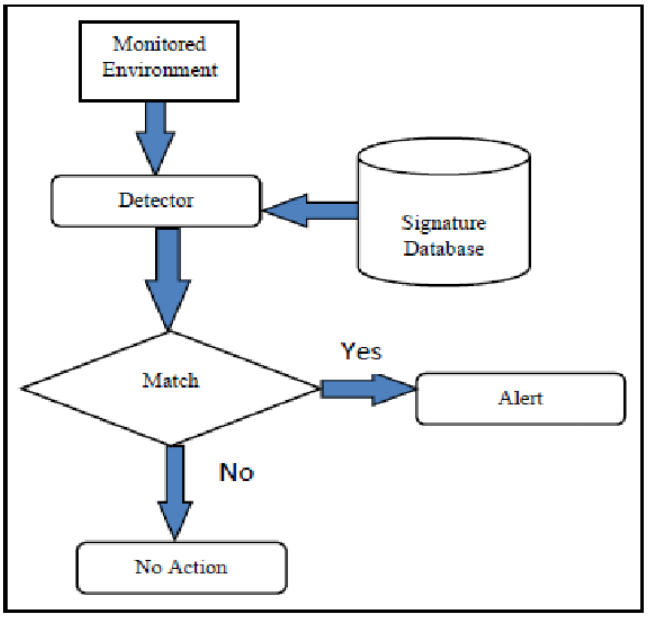


Figure 3.2: Architecture of signature-based IDS [100].

3.4.2 Anomaly-based detection

Anomaly detection is a type of intrusion detection that uses deep learning models to detect deviations from normal behavior. Due of its ability to bypass SIDS's limitations, AIDS has attracted the attention of many academics. The aim of an Anomaly-based Intrusion Detection System (AIDS) is to detect any behavior that deviates from a model's learned normal pattern. To detect such anomalies, the deep learning model is trained on a dataset of normal network traffic to establish a baseline. The model learns the statistical patterns that are inherent in the normal traffic, and when deployed in the network, it can identify any traffic that deviates from these normal patterns. There are different types of deep learning models that can be used for anomaly detection, such as autoencoders, recurrent

neural networks, and convolutional neural networks. Autoencoders are a popular choice for anomaly detection because they can be trained in an unsupervised manner, without the need for labeled data.

The steps involved in anomaly detection using deep learning are:

1. **Data preprocessing**: In this step, the data collected from the wireless sensor network is cleaned, transformed, and normalized to prepare it for deep learning.
2. **Feature extraction:** This step involves identifying relevant features from the preprocessed data that can be used to train a deep learning model. For example, in network traffic data, features such as packet size, time between packets, and number of packets per second may be used.
3. **Model selection and training:** Once the features have been extracted, the deep learning model needs to be selected and trained on the data. There are several deep learning models that can be used for anomaly detection, and the choice will depend on the specific requirements of the system.
4. **Model evaluation:** Once the model has been trained, it needs to be evaluated to determine its performance on detecting anomalies in the network traffic. This involves comparing the model's predictions to a test dataset containing both normal and abnormal traffic.
5. **Model deployment:** If the model performs well on the test data, it can be deployed in the wireless sensor network to monitor network traffic and detect anomalies.

Anomaly detection using deep learning has several advantages. It can detect new or unknown attacks that do not fit any pre-defined signature or rule-based approach. Also, it can adapt to changing network conditions and detect anomalies in real-time.

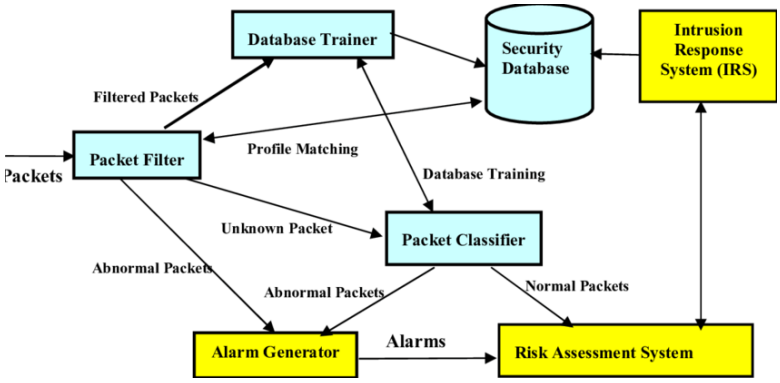


Figure 3.3: Anomaly based IDS [52].

The distinctions between anomaly-based and signature-based detection are summerized in Table 3.1.

3.5 General architecture

There has been many proposed architecture in the literature but they could be generally summarized as bellow :

Sensors: These are responsible for collecting data from various sources in the network, such as routers, switches, firewalls, and servers. The data can include network traffic, system logs, and other system events.

Table 3.1: Comparisons of intrusion detection methodologies based on [59].

Detection method	Advantages	Drawbacks
SAIDS	• Very effective in identifying intrusions with minimum false alarms (FA). • Promptly identifies the intrusions. • Superior for detecting the known attacks. • Simple design.	• Needs frequent updateds for new signatures. • Designed to detect attacks for known signatures. Previous intrusion altered slightly to a new variant would not be detected. • Zero-day attack is not detected. • Inappropriate for multi-step attacks detection. • Little comprehension of the attacks' insights.
AIDS	• Could be used to detect new attacks. • Could be used to create intrusion signature.	• Can't handle encrypted packets (undetected attack: presents a threat). • High false positive alarms. • Hard to build a normal profile for a highly dynamic system. • Unclassified alerts. • Needs initial training.

- **Analysis Engine:** The analysis engine is responsible for processing the data collected by the sensors and generating alerts based on predefined rules or patterns. It uses various techniques such as signature-based detection, anomaly detection, and behavior-based detection to identify potential threats and anomalies.
- **User Interface:** The user interface provides a way for security analysts to view and manage alerts generated by the analysis engine. It can include a dashboard that displays real-time information about the network and system activity, as well as tools for investigating and responding to alerts.
- **Centralized Management:** In some cases, the IDS may be managed centrally, which allows for easier management and coordination of security policies across multiple locations or systems.

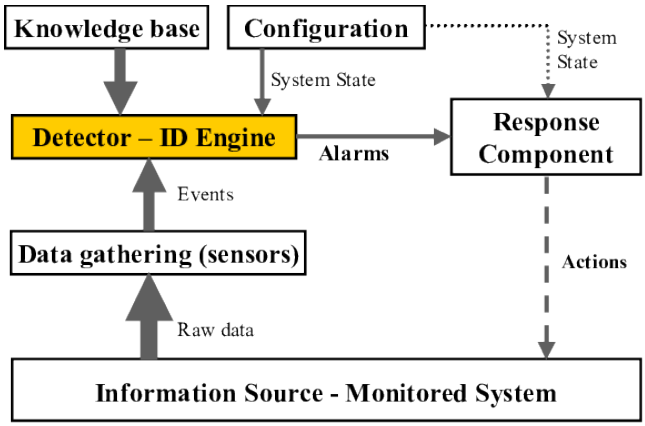


Figure 3.4: Basic architecture of intrusion detection system (IDS). [66]

The figure 3.4 illustrates the basic architecture of intrusion detection system.

3.6 Types of intrusion detection systems

While achieving all the same objective, intrusion detection systems operate slightly in different ways. A comparison of different types of IDS is shown in Table 3.2. According to intrusion data

sources, many IDS types are proposed:

3.6.1 Network-based Intrusion Detection Systems (NIDS)

NIDS, which stands for Network-based Intrusion Detection System, is utilized for monitoring and analyzing network traffic within a specific network section to detect abnormal activities. Figure 3.5 illustrates the architecture of NIDS and the steps involved in detecting attacks. NIDS performs packet-level analysis for all systems within the network segment by examining IP, transport-network, and application protocol-level activities, as well as packet headers, to identify IP-based denial-of-service (DOS) attacks such as TCP SYN attacks and fragment packet attacks [74]. NIDS primarily focuses on the exploitation of vulnerabilities, while HIDS centers around the abuse of privileges [64]. In terms of cost and response time, NIDS is generally more affordable and faster than HIDS since there is no need to maintain sensor programming at the host level, and it monitors network traffic in real-time or near real-time [135]. Consequently, NIDS can detect attacks as they occur. However, NIDS does not provide information on whether these attacks were successful or not, as it does not analyze the system log. The limitation of NIDS lies in its restricted visibility within the host machine and the lack of effective methods to analyze encrypted network traffic for attack detection [74]. To address these challenges, extensive research has been conducted to develop effective approaches for NIDS. Various products for network intrusion detection exist, such as Snort [90] and NetSTAT [125], which are tools designed for real-time NIDS. To date, numerous research studies have proposed methods for NIDS, including:

- SKlavounos et al introduced a new method for DOS attack detection in NIDS based on the tabular cumulative sum (CUSUM) chart and the exponential weighted moving average (EWMA) chart, focusing on UDP and ICMP source bytes using the NSL-KDD experimental dataset [109].
- Suad Othman et al proposed an intrusion detection model for big data environments using the SVM machine learning algorithm for classification and the Chi-selector for feature selection to reduce dimensionality in network traffic [79]. The KDD dataset was employed to evaluate the proposed model.
- Parvat et al proposed a NIDS using deep learning. The method utilized an ensemble of multiple binary classifiers with a deep learning model and employed a divide and conquer strategy [81]. The NSL-KDD dataset was used to evaluate the system.

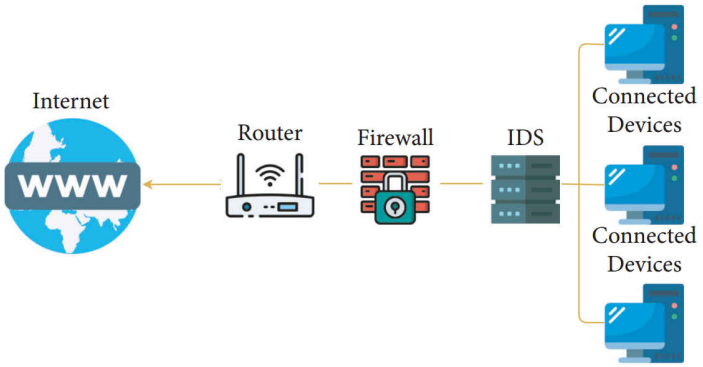


Figure 3.5: Architecture of a network-based intrusion detection system (NIDS) [2].

3.6.2 Network Node Intrusion Detection System (NNIDS)

Technically, a Network Node Intrusion Detection System (NNIDS) is a variant of a NIDS. However, due to its distinct operation, it is regarded as a separate type of intrusion detection system. Unlike a

NIDS, which relies on a central device to monitor all network traffic, a NNIDS scrutinizes each node connected to the network by analyzing the packets that pass through it. This approach yields several benefits, such as :

- Faster processing speeds and lower resource consumption, as each NNIDS agent scrutinizes a smaller volume of traffic.
- Moreover, NNIDS uses fewer system resources and can be easily installed on current servers.

Its primary drawbacks are:

- The need for multiple installations. Unlike a NIDS, which requires only one device, each server you want to monitor with NNIDS necessitates an individual installation.
- Additionally, all NNIDS agents must report to a central dashboard.

3.6.3 Host-Based Intrusion Detection Systems (HIDS)

HIDS, which stands for Host-based Intrusion Detection System, was the first type of intrusion detection system to be developed. HIDS focuses on monitoring and analyzing the internal activities of a single host, including system configuration, application activity, wireless network traffic (specifically for that host), system logs or audit logs, running user or application processes, and file access and modification. The capabilities of HIDS encompass integrity checking, event correlation, log analysis, policy enforcement, rootkit detection, monitoring processor, memory, hard-disk, and battery utilization, as well as generating alerts [22, 127]. Compared to network-based IDS, HIDS tends to be more accurate and produces fewer false positives because it analyzes log files, allowing it to determine the success or failure of an attack [94]. However, HIDS requires the installation of programs or agents on the system to generate reports indicating any malicious activity. One drawback of host-based systems is their resource-intensive nature since they rely on the host's computer resources and lack operating system independence, unlike other types of IDS [64]. Various host-based intrusion detection systems have been developed, such as OSSEC [18] and Tripwire [60].

HIDS analyzes audit log files to detect and identify any malicious system processes. However, analyzing a large amount of data to differentiate between normal and malicious processes can be time-consuming and resource-intensive. Several research studies have proposed methods to address this issue. For example, Marteau [71] introduced a new similarity measure for symbolic sequential data to detect unknown attacks. The author focused on sequences of system calls and utilized the Sequence Covering algorithm for Intrusion Detection (SC4ID). The SC4ID algorithm aims to optimally cover a sequence by a series of subsequences extracted from a predefined set of sequences. The algorithm was evaluated using well-known system call datasets such as UNM and ADFA-LD.

Subba et al presented a framework to enhance computation efficiency in HIDS [113]. The proposed framework transforms system calls into n-gram vectors and reduces the size of input feature vectors through dimensionality reduction. The reduced feature vectors are then analyzed using various machine learning classifiers, including Naive Bayes, MLP, C4.5 Decision Tree, and SVM, to identify intrusive processes. The benchmark dataset ADFA-LD was utilized to evaluate the proposed model.

Deshpande et al proposed a model that selectively analyzes system call traces to detect malicious activities within the system, providing alerts to the cloud user when malicious processes are identified [26].

Table 3.2: Different types of IDS [80].

IDS	**HIDS** (Host IDS)	**NIDS** (Network IDS)	**WIDS** (Wireless IDS)	**NBA** Network Behavior Analysis
Components	Agent, Management server, Database server	Sensor: (inline/passive), Management server, Database server	Sensor: (passive), Management server, Database server	Sensor: (most passive), Management server, Database server
Detection scope	Host	Network or Host	WLAN, WLAN client	Network or Host
Network Architecture	Managed networks or Standard networks	Managed networks	Managed networks or Standard networks	Managed networks or Standard networks

3.6.4 Hybrid Intrusion Detection Systems

HIDS, which stands for Hybrid Intrusion Detection System, combines two or more types of IDS to leverage the advantages of each and achieve accurate detection [67]. An example of a HIDS is Double Guard [97], which utilizes both host-based IDS (HIDS) and network-based IDS (NIDS). By integrating these two types of IDS, HIDS aims to enhance the overall detection capabilities. However, one drawback of HIDS is that it often requires a significant amount of time for data analysis.

3.7 Conclusion

Intrusion detection systems (IDSs) are an important part of the security of wireless sensor networks (WSNs). IDSs can detect attacks on WSNs, which can help to protect the data and infrastructure that WSNs collect and control.There are two main types of IDSs for WSNs: signature-based IDSs and anomaly-based IDSs. Signature-based IDSs look for known attack patterns, while anomaly-based IDSs look for unusual behavior.

Deep learning is a promising new approach for building IDSs for WSNs. Deep learning models can learn to recognize attack patterns from data, even if the attack patterns are not known in advance. Deep learning based IDSs are still under development, but they have the potential to be more effective than traditional IDSs

CHAPTER 4

DEEP LEARNING BASED INTRUSION DETECTION SYSTEMS (DL-IDS)

Chapter 4

Deep Learning based Intrusion detection systems (DL-IDS)

Contents

4.1 Datasets for Intrusion Detection System . . . **72**
4.1.1 KDD Cup 99 . . . 73
4.1.2 NSL-KDD . . . 73
4.2 Performance metrics used for DL-based IDS . . . **74**
4.2.1 Classification metrics . . . 74
4.2.2 Regression metrics . . . 76
4.2.3 General metrics . . . 77
4.3 Related Work . . . **77**
4.4 Feature selection . . . **80**
4.5 Data preprocessing . . . **80**
4.6 Implementation and experimentation on NSL-KDD . . . **81**
4.6.1 NSL-KDD Preprocessing . . . 81
4.6.2 Design of the suggested deep neural network (DNN) . . . 82
4.6.2.1 DNN model generation . . . 82
4.6.2.2 Activation functions . . . 83
4.6.2.3 Optimization functions . . . 84
4.7 Environment . . . **85**
4.7.1 Python . . . 85
4.7.2 Anaconda . . . 85
4.7.3 Jupyter Notebook . . . 86
4.7.4 Numpy . . . 86
4.7.5 Pandas . . . 86
4.7.6 Scikit-learn . . . 86
4.7.7 TensorFlow . . . 86
4.7.8 Keras . . . 87
4.8 DNN classification results . . . **87**
4.8.1 The most important metrics . . . 88
4.8.2 Discussion . . . 89
4.9 Conclusion . . . **89**

4.1 Datasets for Intrusion Detection System

A dataset refers to a group of interconnected, distinct pieces of data presented in rows and columns. These columns, often referred to as features, represent different variables that can be of various types, such as numerical or categorical. Each row represents an individual instance of these variables within the dataset.

Evaluation datasets are essential to the validation of any IDS technique since they allow measuring how efficiently the proposed technique can identify intrusive activity. Due to privacy concerns, the datasets used for network packet analysis in commercial solutions are not readily available. However, there are a few publicly accessible datasets that are often used as benchmarks, including DARPA, KDD, NSL-KDD, and ADFA-LD [21].

Table 4.1 summarizes the most used datasets for intrusion detection.

Table 4.1: Most used Datasets for intrusion detection.

Dataset	Developer	Features	Attack types	Description	Year
DARPA 98 [24]	MIT-L Lab	41	Denial of Service, Root to Local, User to Root (U2R) and Probe	- Not an accurate network traffic representation - Irregular data attack instances - No false positive examples	1998
KDD-Cup99 [116]	California University	41	Denial of Service, Root to Local, User to Root (U2R) and Probe	- Includes Repeated and plagiarized samples	1999
CAIDA [46]	Center of Applied Internet Data Analysis	20	Distributed Denial of Service	- Includes for a single attack type, extremely unique cases	2007
NSL-KDD [87]	California University	41	Denial of Service, Root to Local (R2L), User to Root (U2R) and Probe	- Better than KDD-Cup99 - Limited number of attack types	2009
ISCX-2012 [107]	New Brunswick University	IP Flows	Denial of Service, Distributed Denial of Service, Bruteforce and Infiltration	- Labeled data instances and network examples with intrusive activities.	2012
AFDA [23]	University of New South Wales	System call traces	Zero-day attacks, Stealth attack, C100 Webshell attack	- Consists of 10 attack vectors with the traces of other data instances. - Limited range of attacks.	2013
UNSW-NB15 [75]	UNSW Canberra Cyber Range Lab	49	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.	- Created by the IXIA PerfectStorm tool. - Real modern normal activities (32%) and synthetic contemporary attack behaviors (68%).	2015
CICIDS 2017 [99]	Canadian Institute of Cyber Security	80	Brute force, Portscan, Botnet, Denial of Service, Distributed Denial of Service, Web and Infiltration.	- Datasets generated in a particular way from network profiles. - Classes: 15. -Attackers: 4 PCs, 1 router, 1 switch. - Victims: 3 server, 1 firewall, 2 switches, 10 PCs. - Duration: 5 Days of capture.	2017
CSE-CIC-IDS 2018 [42]	Canadian Institute of Cyber Security	80	Brute force, Portscan, Botnet, Denial of Service, Distributed Denial of Service, Web and Infiltration.	- Datasets generated in a particular way from network profiles. - Classes: 18. - Attackers: 50 PCs. - Victims: 420 PCs, 30 servers. - Duration: 10 Days of capture.	2018
Bot-IoT [82]	UNSW Canberra Cyber Range Lab	43	Denial of Service, Distributed Denial of Service, Operating System and Service Scan, Keylogging, and Data Ex-filtration	- Mix of regular and Botnet traffic	2018
TON-IoT [31]	UNSW Canberra Cyber IoT Lab	44	Denial of Service, Distributed Denial of Service, Backdoor, Injection, Man-in-the-middle,,Password, Ransomware, Scanning and Cross-Site - Scripting	- Contains a range of information collected from telemetry datasets for industrial-IoT and IoT sensors	2019
MQTT-IoT-IDS2020 [48]	IEEE Data Port	3 Abstract Features	Aggressive Scan (Scan A), User Datagram Protocol Scan (Scan U), Sparta Secure Shell Brute force (Sparta), MQTT Brute force attack (MQTT-BF)	- Generated using an artificial MQTT network architecture	2020

4.1.1 KDD Cup 99

Since 1999, this dataset has been extensively used for evaluating anomaly detection systems. It was prepared based on the data collected in the DARPA'98 IDS evaluation program by Stolfo et al [116]. The DARPA'98 dataset consists of approximately 4 gigabytes of compressed raw tcpdump data, representing the network traffic of 7 weeks. This data can be represented into 5 million connection records of around 100 bytes per record. The test data specifically spans two weeks and includes approximately 2 million connection records.

The KDD training dataset consists of approximately 4 900 000 individual connection vectors. Each one is composed of 41 features labeled as either attack or normal, with a specific attack type assigned. The dataset's simulated attacks may be divided into four categories:

1. **Denial of Service Attack (DoS):** This type of attack aims to overload or exhaust computing or memory resources, making it difficult for legitimate requests to be processed or denying access to authorized users.
2. **User to Root Attack (U2R):** In this exploit, the attacker initially gains access to a regular user account on the system and then exploits vulnerabilities to escalate privileges and gain root access.
3. **Remote to Local Attack (R2L):** This attack occurs when an attacker, who can send packets over a network to a target machine but does not have an account on that machine, exploits vulnerabilities to gain local access and operate as a user on the target machine.
4. **Probing Attack:** Probing attacks involve attempts to gather information about a network of computers, typically with the intention of identifying vulnerabilities and bypassing security controls.

These attack categories are used to evaluate and classify the connections in the KDD Cup 99 dataset, providing a valuable resource for studying and developing anomaly detection techniques.

4.1.2 NSL-KDD

The older KDD Cup99 dataset was transformed into the publicly accessible NSL-KDD dataset [87]. The KDD Cup99 dataset has significant flaws that have a significant impact on intrusion detection accuracy, as revealed by a statistical examination, and result in an erroneous evaluation of AIDS [116].

Researchers addressed the challenges present in the KDD-99 cup dataset by introducing a new dataset called NSL-KDD. Unlike the complete KDD dataset, NSL-KDD was carefully constructed to include only specific records and does not exhibit the same issues. Tavallaee et al. developed NSL-KDD as a solution to rectify the problems associated with KDD-99 and mitigate its limitations. Nevertheless, it is important to note that NSL-KDD still has certain limitations, such as the omission of low footprint attacks in its representation.

The use of the NSL-KDD dataset according to [84], has the following advantages:

1. The test set has no duplicate records, which offers greater reduction rates.
2. The proportion of records in the original KDD dataset that are chosen from each difficult level category is inversely correlated with the number of records chosen from those categories.
3. Compared to KDD-99, NSL-KDD includes fewer data points, but each of them is unique. Consequently, using it to train machine learning models requires fewer resources.

Table 4.2: NSL-KDD train and test data distribution [84].

Class	Training Set	Percentage	Test Set	Percentage
Normal	67342	53.458%	9710	43.075%
DoS	45927	36.458%	7457	33.080%
Probe	11656	9.253%	2421	10.740%
R2L	995	0.790%	2754	12.217%
U2R	52	0.041%	200	0.887%
Total	125972	100%	22542	100%

4.2 Performance metrics used for DL-based IDS

Every machine learning pipeline has indicators of performance. They indicate whether efforts are making progress and provide a numerical value for it. To measure performance, all machine learning models, whether linear regression or a SOTA approach like BERT, need a metric. Any machine-learning problem may be divided into two categories: regression and classification. Metrics of performance are similarly classified. There are many indicators for both issues, however, we'll experience the most prevalent ones and the details provided about the model's performance. Understanding how the model interprets the data is necessary. Metrics serve for monitoring and evaluating a model's performance (during training and testing).

For ML-based IDS, there are various classification metrics. The confusion matrix for a two-class classifier, which may be used for evaluating the performance of an IDS, is shown in Table 4.3.

Table 4.3: Confusion Matrix for IDS System.

Actual Class	**Predicted Class**	
Class	**Normal**	**Attack**
Normal	True Negative (TN)	False Positive (FP)
Attack	False Negative (FN)	True Positive (TP)

The instances in each row of the matrix reflect the occurrences in the actual class, while each column represents the instances in the predicted class. The following standard performance metrics are commonly used to evaluate ML-based IDSs.

4.2.1 Classification metrics

Because classification models provide discrete output, we require a measure that compares discrete classes in some way. classification Metrics measure the performance of a model and indicate whether it is good or bad, depending on its classification, but they measure it in different ways.

- **True Positive Rate (TPR, Recall, Hit-Rate, Detection Rate or Sensitivity):**
  It is the ratio between the number of properly predicted attacks and the total number of attacks. When all intrusions are correctly detected then the TPR is 1 which is very rare. It explains how sensitive the model is towards identifying the positive class. Accuracy in making positive predictions is measured by a recall. It has values in the range $[0, 1]$ and is expressed mathematically as:
  $$TPR = \frac{TP}{TP + FN} \tag{4.1}$$
- **True Negative Rate (TNR or Specificity):**
  Corresponds to the proportion of normal instances that are correctly predicted, with respect to

all normal instances and have values in the range $[0, 1]$. It is expressed mathematically as:

$$TNR = \frac{TN}{TN + FP} \tag{4.2}$$

- **False Positive Rate (FPR):**
  It is the ratio between the number of normal instances improperly classified as attacks and the total number of normal instances and expressed as:

$$FPR = \frac{FP}{FP + TN} \tag{4.3}$$

  We have :

$$TNR + FPR = 1 \tag{4.4}$$

- **False Negative Rate (FNR):**
  False negative means when a detector fails to identify an anomaly and classifies it as normal. The FNR can be expressed mathematically as:

$$FNR = \frac{FN}{FN + TP} \tag{4.5}$$

  We have :

$$TPR + FNR = 1 \tag{4.6}$$

- **Classification rate (CR) or Accuracy:**
  It measures how accurate the IDS is in detecting normal or anomalous behavior. It is the percentage of all those properly predicted instances to all instances:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of input samples}} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.7}$$

  The balanced accuracy is given by:

$$Balanced\ Accuracy = \frac{TPR + TNR}{2} \tag{4.8}$$

- **Precision:**
  It is the success probability of making a correct positive class classification. It is the ratio between the properly predicted attacks and the total number of instances classified as attacks. Identifying all positive occurrences in the data is quantified by precision. It is expressed as:

$$Precision = \frac{TP}{TP + FP} \tag{4.9}$$

- **Receiver Operating Characteristic (ROC) curve:**
  The ROC curve has TPR and FPR on the axes, respectively. For various cut-off points, the TPR is displayed in ROC curves as a function of the FPR. A FPR and TPR pair corresponding to a certain decision threshold is represented by each point on the ROC curve. A new point on the ROC is chosen with a different False Alarm Rate (FAR) and TPR when the classification threshold is changed. A test with 100% sensitivity and 100% specificity has a ROC curve that goes through the top left corner (no overlap between the two distributions).
- **F-Score:**
  F-Score (also known as the F1-score or F-measure) is the Harmonic Mean between precision and recall in the range of $[0, 1]$. It tries to find the balance between precision and recall and illustrates how precise the classification is (how many instances are correctly classified). It can be expressed mathematically as:

$$\text{F-Score} = 2 \times \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.10}$$

- **F2-Score:**
  It is a variant of the F1-score used to evaluate the performance of a Machine Learning model by combining precision and recall into a single score in the range of $[0, 1]$. Higher values indicate better performance. The F-2 score emphasizes recall compared to the standard F-1 score. The F2-score is defined as:

$$\text{F2-Score} = 5 \times \frac{Precision \times Recall}{4Precision + Recall} \tag{4.11}$$

4.2.2 Regression metrics

The output of regression models is continuous. As a result, we need a metric that is based on computing some form of distance between expected and actual reality.

- **Mean Absolute Error:**
  It is the average of the difference between the Original Values and the Predicted Values. It shows the measure of how far the predictions were from the actual results without giving any idea of the direction of the error (under pridicting or over pridicting). It is expressed as:

$$\text{Mean Absolute Error} = MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4.12}$$

- **Mean Squared Error:**
  The only distinction between Mean Squared Error and Mean Absolute Error is that MSE takes the average of the square of the difference between the original and predicted values. Its main advantage is that it is easier to compute the gradient, whereas Mean Absolute Error requires complicated linear programming tools to compute it. As we square the error, the influence of larger errors gets bigger than that of smaller errors; as a result, the model can now concentrate more on larger errors. Mean Squared Error is expressed mathematically as:

$$\text{Mean Squared Error} = MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.13}$$

- **Root Mean Squared Error (RMSE):**
  It is equal to the square root of the average of the squared difference between the target and predicted values by the regression model. It may be expressed mathematically as:

$$\text{Root Mean Squared Error} = RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{4.14}$$

- **Coefficient of Determination (R^2):**
  It is the best indicator of how good our model is in terms of performance whether it is accuracy, Precision or Recall. It is the measure of the variance in response variable y that can be predicted using predictor variable $\hat{y}$ and is expressed as:

$$\text{Coefficient of Determination} = R^2 = 1 - \frac{MSE}{\text{Variance of inputs}} = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{4.15}$$

4.2.3 General metrics

- **Training time** is the time it takes to train a model on a dataset. This can be much longer than testing time, especially for large datasets or complex models.
- **Testing time** is the time it takes to make a prediction using a trained model. This is typically very fast, as the model has already been trained and does not need to be re-trained every time a prediction is made.

4.3 Related Work

ML-based IDSs have been shown to be effective in detecting a variety of attacks. However, there are still some challenges that need to be addressed. One challenge is that ML algorithms can be computationally expensive. Another challenge is that ML algorithms can be easily fooled by attackers who deliberately try to evade detection .

In the article[100] the authors introduced a network Intrusion Detection System (IDS) using the ANDAE (Autoencoder-based Nonlinear Dimensionality Reduction with Adaptive Neighbors) feature extraction technique. The ANDAE model utilized an Autoencoder (AE) for feature extraction and integrated it with the Random Forest (RF) algorithm for classification. The IDS was evaluated on the NSL-KDD dataset, and it achieved a high recall rate and F1 score for detecting four types of attacks. Compared to the SNDAE model, the ANDAE model demonstrated improved detection accuracy while reducing feature extraction time.

In 2021, researchers utilized the Azure ML (Machine Learning) platform for Intrusion Detection System (IDS) [47]. They employed a meta-classification approach for both binary and multi-classification purposes. The study utilized three datasets: UNSW-NB15, CICIDS2017, and CICIDS2018. The achieved accuracy rates were 99.8% on UNSW-NB15, 99% on CICIDS2017, and 98% on CICIDS2018. The train and test split ratio used in the research was 40:60, indicating that 40% of the data was used for training the models and 60% for testing their performance.

In a study conducted by some researchers used the KDD CUP 99 dataset to train a machine learning model that achieved an accuracy of 95%. The model used a multilayer perceptron (MLP) algorithm and fifteen features. A study conducted by authors in (2022) they used machine learning (ML) techniques to develop an intrusion detection system (IDS). The study used information gain and gain ratios to select features from the IoTID20 and NSL-KDD datasets. The study then evaluated several ML algorithms, including multilayer perceptron (MLP), J48, IBK, and bagging. The study achieved an accuracy of 99%.

In another study, researchers used machine learning (ML) techniques to develop an intrusion detection system (IDS). The study used information gain and gain ratios to select features from the IoTID20 and NSL-KDD datasets. The study then evaluated several ML algorithms, including multilayer perceptron (MLP), J48, IBK, and bagging. The study achieved an accuracy of 99%.

In a 2019 study, the authors evaluated the performance of three machine learning (ML) algorithms on the KDDCUP99 dataset: linear discriminate analysis (LDA), classification and regression tree (CART), and random forest. Random forest achieved the highest accuracy of 99.8%, followed by LDA with 98% and CART with 98.1%.

The authors concluded that random forest is the most accurate ML algorithm for the KDDCUP99 dataset. They also noted that LDA and CART are both effective algorithms, but they are not as accurate as random forest.

Researchers have moved from machine learning to deep learning for intrusion detection systems (IDSs) for a number of reasons:

- **Deep learning models can learn more complex features from data than machine learning models**:This is because deep learning models use multiple layers of artificial neurons to learn features, while machine learning models typically only use a single layer.
- **Deep learning models can be trained on larger datasets than machine learning models**:This is because deep learning models are more computationally expensive to train, so they require more data to learn effectively.
- **Deep learning models can be more accurate than machine learning models**:This is because deep learning models can learn more complex features from data, and they can be trained on larger datasets.

Deep learning models offer a number of advantages over machine learning models for intrusion detection. These advantages include the ability to learn more complex features from data, the ability to be trained on larger datasets, and the ability to be more accurate. As a result, deep learning models are becoming increasingly popular for IDSs.

Previous work has shown that deep learning (DL) methods outperform other machine learning algorithms, such as Support Vector Machine (SVM) and Traditional Neural Network (ANN), in anomaly detection [57].

Research on deep learning for intrusion detection began in 2011 when Salama et al. presented a hybrid method that combines Deep Belief Network (DBN) and SVM to classify network intrusion into two categories: normal or attack [93]. The DBN network utilizes a restricted Boltzman machine (RBM) as a method of minimization to enhance the learning experience. The SVM classifier is then applied to the DBNs to perform binary classification (normal/attack) on NSL-KDD data records. The proposed DBN-SVM method achieves over 90% accuracy and outperforms other data analysis methods such as ACP, Gain Ratio, and Chi-Square.

Kang et al proposed an intrusion detection system based on Deep Neural Networks (DNN) for securing vehicular networks [57]. The system addresses conflict situations caused by malicious data packets injected into the network bus of the bus controller. The proposed method uses a DBN algorithm to train the hidden layers' parameters, with subsequent layers combining the results using the ReLU activation function. The method achieves a false positive rate (TPR) of less than 1% or 2% and a detection rate (DR) of 99%.

Sandee et al developed a DNN for Network Intrusion Detection Systems (NIDS) with unsupervised plan learning and regression analysis for binary classification on the NSL-KDD dataset [43]. By considering 115 input features, a sparse auto-encoder is used to generate and learn new characteristics, reducing them to 50 and then 10. These characteristics are assigned to the regression classifier for classification. The model achieves an overall accuracy of 87.2%.

Tang et al applied a deep neural network (DNN) in the implementation of an IDS for Device Defined Network (SDN) management, capable of monitoring all traffic passing through OpenFlow switches [114]. They trained the model on the NSL-KDD dataset for binary classification (normal/anomalous), considering four categories: DoS attack, R2L attack, U2R attack, and honest fighting. Only six main features out of the 41 are used. The model is optimized by finding the best hyperparameter, 'ce', for classification results. The authors experiment with different learning coefficients ranging from 0.1 to 0.0001, finding the best result with a learning rate of 0.001. The model achieves

a performance and accuracy of 75.75%. This experience demonstrates the strong potential of deep DNNs in anomaly detection and NIDS.

Network intrusion detection systems (NIDS) based on deep learning have been studied by various researchers. Javaid et al utilized self-directed learning (Selftaught Learning STL) for classification [54]. The approach involved two steps: unsupervised learning to obtain representations from unlabeled data and applying the learned representations to labeled data for classification. STL was tested on the NSL-KDD dataset, achieving an accuracy of 85.44% and a recall of 95.95% for binary classification (normal traffic/bad traffic). Compared to simple soft-max regression and other previous works, STL showed promising results.

In the realm of intrusion detection, there have been few studies using Convolutional Neural Networks (CNN). CNN offers advantages such as shared convolutional kernels, reducing parameters and computational training.

Vinayakumar et al explored the performance of CNN for anomaly detection by modeling network traffic as a time series of data [126]. They used a 1D Convolution layer to process the temporal data. The authors found that CNN-LSTM performed well, achieving 99% accuracy on the KDDCup 99 dataset.

Kim et al proposed an RNN-based IDS, representing TCP/IP network traffic as a system. They employed the Hessian-Free method to address the bursting/disappearing gradient problem of RNN [61]. Using the DARPA database with 41 features and 22 different attacks, their approach achieved a detection rate of 95.37% and a false alarm rate of 2.1%.

GAO et al applied the Deep Belief DBF network, combining unsupervised multi-layered learning networks (Blocked Boltzmann Machine) with a supervised learning network (back-propagation network) for intrusion detection [35]. The DBN model outperformed SVM and ANN when tested on the KDD CUP 99 dataset.

Rezvy et al employed the AutoEncoder method, where the model first trained to transform features and then used a DNN classifier for differentiation. The authors tested the model for intrusion detection, but further details were not provided in the given text [88].

Alom et al utilized Deep Belief Network for input detection. The proposed method successfully detected attacks and classified network activities into five groups based on finite, incomplete, and non-linear data sources [8]. It achieved a high detection accuracy of 97.5% with only 50 iterations.

Wu et al introduced an IDS model based on a Convolutional Neural Network (CNN) that automatically selected features from raw traffic data [130]. They employed the NSL-KDD dataset for monitoring and addressed the issue of imbalanced datasets by adjusting the cost function weight for each class based on its number. To reduce computational costs, they converted the raw traffic data into a vector format. The proposed model improved classifier accuracy, particularly for small numbers, and reduced the false alarm rate (FAR).

Torres et al investigated the effectiveness of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM), for identifying network traffic behavior modeled as a dynamic system [117]. They proposed an LSTM detection method and analyzed two different types of network traffic data, CTU13-42 and CTU13-47, which contained two classes: botnet connections and normal connections. Botnet connections refer to a group of compromised computers controlled remotely to execute coordinated attacks or maneuvers. The authors also examined the number of states per link for the model. The experimental results demonstrated that the RNN-based approach achieved a high detection rate (ADR) of 97.0% and a low false alarm rate (FAR) of 0.018%. The impact of sampling

methods (over-sampling/under-sampling) on misclassification was also investigated, revealing no significant differences in ADR values across the three cases. However, the experiments highlighted the challenges faced by RNN detection models in handling diverse and redundant network behaviors.

4.4 Feature selection

Enhancing the effectiveness of Feature selection algorithms requires careful consideration of feature selection. This crucial process involves choosing a subset of relevant features based on specific criteria. Feature selection plays a significant role in reducing dimensionality and is widely employed in data mining. Many datasets contain superfluous or irrelevant features, which can hinder analysis. By reducing the number of features and eliminating irrelevant, redundant, or noisy ones, feature selection yields tangible benefits in various applications. It accelerates data mining algorithms, improves learning accuracy, and enhances model interpretability [68].

4.5 Data preprocessing

Data preprocessing is an essential step in data mining known as the Knowledge Discovery from Data process. It involves conducting preliminary procedures before applying data mining techniques. Data is prone to imperfections like inconsistencies and redundancies, making it unsuitable for direct analysis. Moreover, with the increasing generation rate and size of data in various domains, advanced mechanisms are needed to handle larger volumes of data for analysis. Data preprocessing plays a crucial role in adapting the data to meet the specific requirements of each data mining algorithm, allowing the processing of otherwise impractical data [36].

- Data cleaning
- **Data cleaning** involves various procedures to enhance the quality of data. These procedures include filling in missing values, reducing noise in the data, identifying and eliminating outliers, and resolving any inconsistencies present [132].
- **Data transformation** Normalization and aggregation [132].
- **Label encoding** Label encoding, also known as integer encoding, is applied to nominal variables that contain a limited number of distinct classes. These classes do not possess any inherent relationship with each other. In label encoding, each category within the variable is assigned a numerical value ranging from 1 to N, where N represents the total number of unique classes present in the feature [86].
- **One-hot encoding** One-hot encoding is commonly employed when dealing with nominal categorical variables that lack any ordinal relationship, and when label (integer) encoding is insufficient. Suppose we have a random categorical variable X with n distinct values $x_1, x_2, \ldots, x_n$. In one-hot encoding, a specific value x_i is represented by a vector v, where all components are zero except for the ith component, which is assigned a value of 1. For instance, if X can take values from the set $S = a, b, c$, and $x_1 = a$, $x_2 = b$, $x_3 = c$, the corresponding one-hot encodings would be $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ respectively [86].
- **Classification** The classification stage holds significant importance among all the proposed methods as its objective is to categorize (identify) packets into two groups: normal and malicious. Hence, binary classification is employed for this purpose.

4.6 Implementation and experimentation on NSL-KDD

4.6.1 NSL-KDD Preprocessing

- **NSL-KDD features:** naming every features :

Code Listing 4.1: NSL-KDD features naming

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

- **Data cleaning:** we removed the attribute named "*difficulty level*" because This column was created by the authors of the dataset to help researchers assess the difficulty of detecting attacks, this column is not used by most intrusion detection systems (IDSs). This is because IDSs are typically designed to detect attacks regardless of their difficulty. The difficulty level column is not very reliable. This is because it is subjective and it can be difficult for human experts to agree on the difficulty of an attack .

  There are other reasons why we delete the difficulty level column:

  It is not used by most IDSs. It is subjective and unreliable. It can introduce bias into the dataset. Deleting the difficulty level column does not affect the accuracy of IDSs. This is because the difficulty level column does not contain any information that is not already contained in the other columns of the dataset.

  Deleting the difficulty level column can actually improve the accuracy of IDSs. This is because it can help to reduce the number of false positives.

- **Binary Label Conversion**
  Transforming Attack Labels into **Normal** and **Abnormal** Categories

- **Data transformation** we did normalization for the numeric columns using the '*StandardScaler*' class from $scikit-learn$.

- **Label encoding:** we change column called *label* (abnormal,normal) to another column named *intrusion* contain the values (0,1) so that normal turned 1 and abnormal turned 0.

- **One-hot encoding** in this step, We transformed categorical columns ($protocol_type$, *service*, and *flag*) into numerical representations using one-hot encoding. Each unique value in the categorical columns will be transformed into a separate binary column, with a value of 1 indicating the presence of that category and 0 indicating its absence.

- **Binary classification** The aim of this task is to divide elements into two categories based on a classification rule. In the context of this thesis, the goal is to classify traffic as either "normal" or "abnormal" [1]. One question that arises is whether it would be more effective for a wireless sensor network IDS to detect the specific nature and details of each attempted attack. This could be achieved by implementing a multi-classification model capable of distinguishing different types of attacks individually. While this hypothesis seems reasonable and would enable prompt responses to detected intrusions, it is important to consider the constraints inherent in wireless sensor networks, as discussed in the first chapter. These constraints include limited power consumption, low computing capacity, and low memory performance. Therefore, installing a complex and resource-intensive IDS within the network can negatively impact the overall system quality, leading to higher latency in communication and a decrease in the accuracy of network services. As a result, binary classification is the preferred solution for such systems. Alternatively, an additional IDS could be installed on an external server separate from the WSN. This external IDS would be responsible for identifying the nature of attacks on the network without impacting the overall functioning of the network and while adhering to the constraints of the WSN.

4.6.2 Design of the suggested deep neural network (DNN)

4.6.2.1 DNN model generation

The architecture of the proposed DNN is illustrated in figure 4.1 and the Python code that creates this DNN is provided in listing 4.2.

Code Listing 4.2: DNN Model code

```
mlp = Sequential()
mlp.add(Dense(units=256, input_dim=X_train.shape[1], activation='relu'))
mlp.add(Dropout(0.2))
mlp.add(Dense(units=128, activation='relu'))
mlp.add(Dropout(0.2))
mlp.add(BatchNormalization())
mlp.add(Dense(units=64, activation='relu'))
mlp.add(Dropout(0.2))
mlp.add(BatchNormalization())
mlp.add(Dense(units=32, activation='relu'))
mlp.add(Dropout(0.2))
mlp.add(Dense(units=16, activation='relu'))
mlp.add(Dropout(0.2))
mlp.add(Dense(units=1, activation='sigmoid'))
optimizer = Adamax(learning_rate=0.001)
mlp.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
mlp.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

1. **Input Layer:** The first layer added is a Dense layer with 256 units. The input shape is determined by the number of features in the training data. The activation function used is ReLU (relu).

2. **Dropout:** A Dropout layer with a dropout rate of 0.2 is added after the first Dense layer. Dropout randomly sets a fraction of input units to 0 during training to reduce overfitting.

3. **Hidden Layers:** Additional Dense layers with 128, 64, 32, and 16 units are added, each followed by a Dropout layer with a dropout rate of 0.2. These layers introduce non-linearity and help the model learn complex patterns in the data.

[1] Remark: for each instance in the dataset, the label column is set to "1" in case of **normal** traffic and "0" in case of any attack of any category.

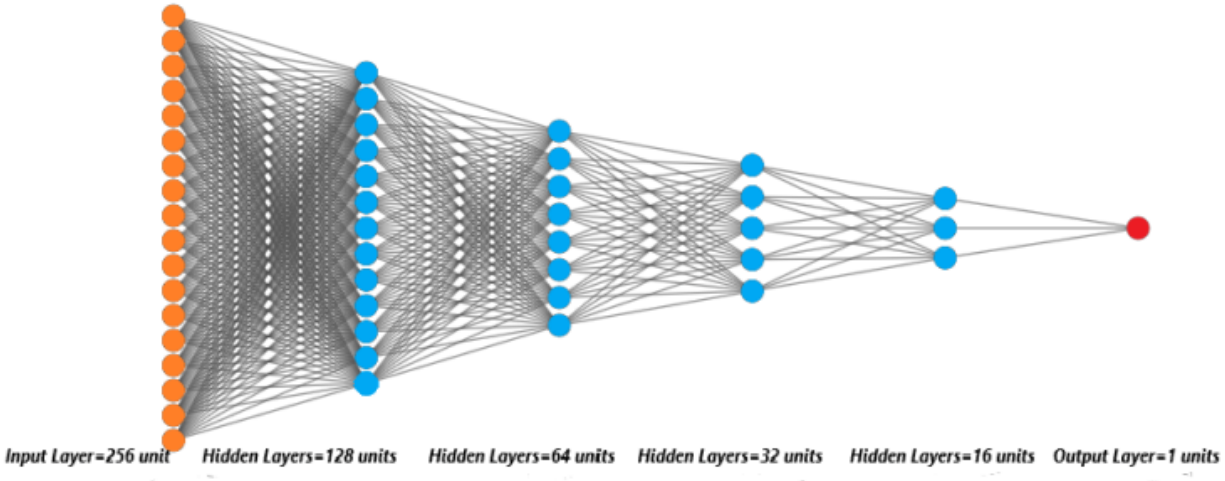


Figure 4.1: model DNN

4. **Batch Normalization:** BatchNormalization layers are inserted after the second and fourth Dense layers. Batch normalization helps normalize the inputs to each layer, stabilizing the learning process and potentially improving performance.

5. **Output Layer:** The final Dense layer consists of 1 unit and uses the sigmoid activation function ("sigmoid").

6. **Optimization:** Two compilation statements are present, The initial compilation statement sets the optimizer as Adamax with a learning rate of 0.001. The subsequent compilation statement the optimizer as adam, which is a different optimization algorithm. The model is compiled with the binary cross-entropy loss function binary crossentropy suitable for binary classification problems. The model's performance is evaluated using the accuracy metric.

4.6.2.2 Activation functions

Activation functions are utilized in artificial neural networks to convert input signals into output signals, which are then passed as input to the subsequent layer in the network [103]. Within an artificial neural network, the inputs and their respective weights are multiplied and summed, and an activation function is subsequently applied to obtain the output of the current layer. This output is then forwarded as input to the next layer in the network.

- **ReLU** One advantage of utilizing the ReLU function is that it ensures not all neurons are activated simultaneously. This means that a neuron is deactivated only when the output of the linear transformation is zero. ReLU is considered more efficient compared to other functions because it activates only a specific number of neurons at a given time.

- **sigmoid** The sigmoid function is highly popular as an activation function due to its non-linear nature. It transforms input values into a range between 0 and 1. Mathematically, it can be represented as $f(x) = \frac{1}{1+e^{-x}}$. The sigmoid function is characterized by its continuous differentiability and smooth, S-shaped curve. Its derivative is given by $f'(x) = 1 - sigmoid(x)$.

However, it should be noted that the sigmoid function is not symmetric around zero, resulting in all output values of neurons sharing the same sign. This limitation can be addressed by scaling the sigmoid function to improve its behavior.

4.6.2.3 Optimization functions

The optimization function, in the context of machine learning and deep learning, refers to the algorithm or method used to update the parameters of a model during the training process in order to minimize the loss function and improve the model's performance. The optimization function determines how the model's parameters are updated based on the computed gradients of the loss function with respect to those parameters. It aims to find the optimal set of parameter values that minimize the loss and improve the model's ability to generalize to new data:

- **Adam (Adaptive Moment Estimation):** Adam is an optimization algorithm that combines the benefits of both adaptive learning rates and momentum-based optimization. It computes adaptive learning rates for each parameter based on estimates of first and second moments of the gradients. Adam adjusts the learning rates for different parameters individually and allows the model to converge quickly and handle sparse gradients effectively ().

  Adaptive Learning Rates: Adam computes adaptive learning rates for each parameter based on estimates of the first and second moments of the gradients.

  First Moment (Mean) : Adam calculates the first moment, which is the mean of the gradients, using an exponential moving average. It keeps track of the past gradients to adapt the learning rate for each parameter individually.

  Second Moment (Variance): Adam calculates the second moment, which is the variance of the gradients, also using an exponential moving average. It provides information about the magnitude of the gradients.

  Bias Correction: Since the first and second moments are initialized as zero, there could be bias towards zero at the beginning of training. Adam applies bias correction to address this issue.

  Update Step: Adam combines the first and second moments to calculate the update step for each parameter. It also introduces a hyperparameter called the learning rate to control the step size [62].

- **Adamax:** Adamax is a variant of the Adam optimizer that replaces the second moment estimation with the infinity norm. It uses the infinity norm to normalize the gradient instead of the L2 norm, which allows it to handle very large gradients. Adamax is particularly useful when dealing with sparse gradients or in models with a large number of parameters().

  Infinity Norm: Adamax replaces the second moment estimation of Adam with the infinity norm (maximum norm) of the gradients. It uses the infinity norm to normalize the gradient values.

  Handling Large Gradients: The infinity norm allows Adamax to handle very large gradients effectively, making it suitable for models with sparse gradients or a large number of parameters.

  Update Step: Similar to Adam, Adamax calculates the update step using the first moment (mean) and the infinity norm of the gradients. It also incorporates the learning rate hyperparameter.

The main idea behind both Adam and Adamax is to adaptively adjust the learning rates for different parameters based on the gradients' characteristics. By considering the past gradients and their magnitudes, these algorithms improve the optimization process by converging faster and avoiding getting stuck in local minima[62].

4.7 Environment

4.7.1 Python

One of the most widely used and popular programming languages in the field of data science is Python. Python is a high-level programming language that emphasizes code readability and has a syntax that enables programmers to express concepts concisely. It was developed under an OSI-approved open source license, which means it is freely available for use and distribution, including commercial use. Python has been successfully employed in numerous real-world applications across various industries, including large-scale and mission-critical systems in the security field. Its versatility and extensive ecosystem of libraries and frameworks make it a preferred choice for data scientists and developers working on data analysis, machine learning, and other data-intensive tasks. We used Python version 3.10.11.

```
Entrée [1]: import sys

            print(sys.version)

            3.10.11 | packaged by Anaconda, Inc. | (main, Apr 20 2023, 18:56:50) [MSC v.1916 64 bit (AMD64)]

Entrée [ ]:
```

Figure 4.2: Python version.

4.7.2 Anaconda

Anaconda is a free software package tailored for research and science. It provides access to environments for coding in Python or R, known as integrated development environments (IDEs), which simplify the coding process. These IDEs offer features such as code editing, debugging, data visualization, and collaboration. The syntax of Python remains consistent across IDEs, making it easy to switch between them. The choice of IDE depends on personal preference, as each has its own pros and cons [91].

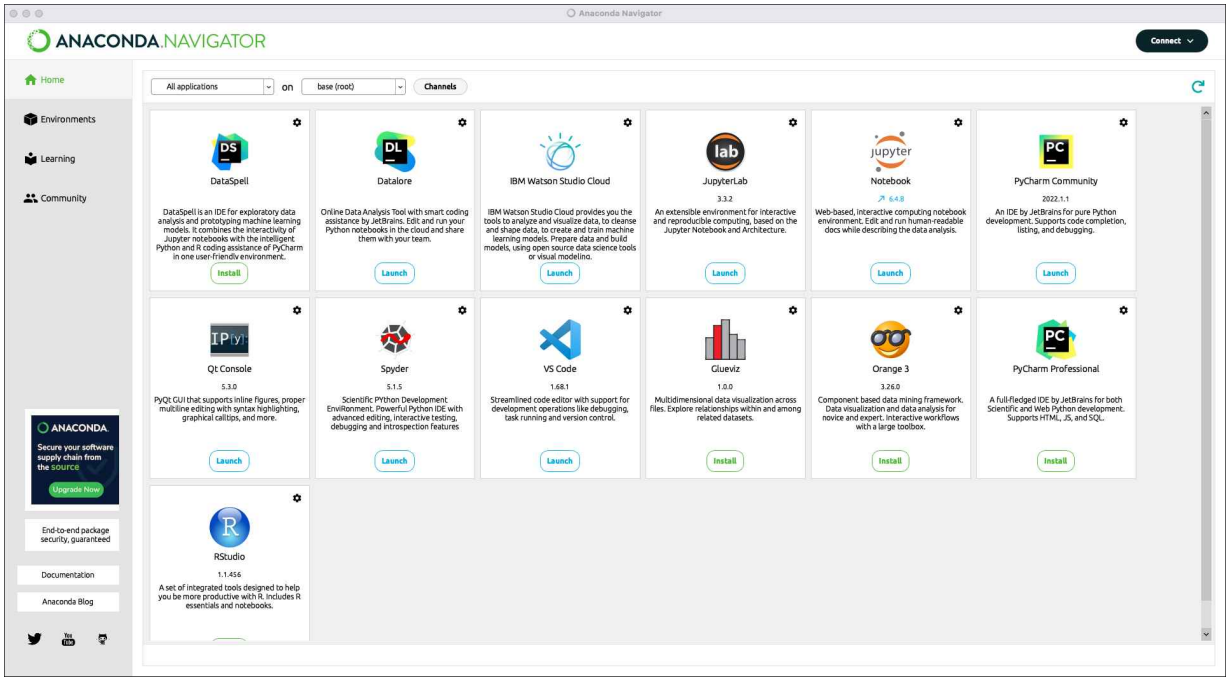


Figure 4.3: Anaconda environment.

4.7.3 Jupyter Notebook

The Jupyter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Once saved, you can share these files with others[2].

This page provides a brief introduction to Jupyter Notebooks for AEN users. For the official Jupyter Notebook user instructions, see Jupyter documentation [3].

For information on the notebook extensions available in AEN, see Using Jupyter Notebook extensions.

4.7.4 Numpy

NumPy is a scientific package for Python that enables efficient scientific computing. It offers N-dimensional array objects for handling large datasets and supports mathematical functions for advanced computations. NumPy's broadcasting feature allows convenient operations on arrays of different shapes. It also allows integration with code from other languages. NumPy is widely used in data science for numerical computations, data analysis, and machine learning[4].

4.7.5 Pandas

Pandas is a Python library for data analysis that simplifies the data analysis workflow by providing convenient data structures and functions. It eliminates the need to switch to other data analysis languages and offers tools for data manipulation and analysis, making it popular among data scientists and analysts[5].

4.7.6 Scikit-learn

Scikit-learn, also known as Sklearn, is a widely used machine learning library developed by David Cournapeau as a Google summer of code project in 2007. It provides a range of simple and efficient tools for predictive data analysis, making it accessible to everyone and applicable in various contexts. Sklearn is built on top of popular scientific computing libraries such as NumPy, SciPy, and matplotlib. It is an open-source library that can be freely used for both personal and commercial purposes [6].

4.7.7 TensorFlow

TensorFlow is a popular open-source library for numerical computation, released by Google on November 15, 2015. It gained rapid success within the Machine Learning community and received significant support from Google and other community projects. TensorFlow's unique feature is its workflow based on data flow graphs, where nodes represent mathematical operations and edges represent the multidimensional data arrays exchanged between them. This data structure is referred to as a tensor, which inspired the library's name. In May 2016, TensorFlow was utilized in the AlphaGo project, and Google even developed specialized hardware to enhance its performance. The user experience of TensorFlow was prioritized, leading to a strong foundation of basic support and a thriving GitHub community. Despite being relatively new compared to other established communities,

[2]NOTE: Python and R language are included by default, but with customization, Notebook can run several other kernel environments.

[3]https://docs.anaconda.com/ae-notebooks/user-guide/basic-tasks/apps/jupyter/index.html/

[4]https://numpy.org/

[5]https://pandas.pydata.org/

[6]https://scikit-learn.org/

TensorFlow has made significant disruptions in the field of Deep Learning, demonstrating its immense potential for the future [7].

4.7.8 Keras

The utilization of Keras has greatly simplified the development workflow for ML practitioners at Waymo. Keras provides numerous benefits, including a significantly simplified API, standardized interface and behaviors, the ability to easily share model building components, and improved debuggability. These advantages have enhanced the efficiency and effectiveness of the ML development process at Waymo [8].

4.8 DNN classification results

In this section, we will discuss the results obtained from the epochs as shown in the table 4.4:

Table 4.4: Example table

Epoch	Time	Loss	Accuracy	Epoch	Time	Loss	Accuracy	Epoch	Time	Loss	Accuracy
1	5s	0.4409	0.8260	45	2s	0.0115	0.9959	91	2s	0.0077	0.9974
2	2s	0.1777	0.9500	46	2s	0.0116	0.9957	92	2s	0.0070	0.9976
3	2s	0.1080	0.9702	47	2s	0.0116	0.9958	93	2s	0.0076	0.9974
4	2s	0.0777	0.9763	48	2s	0.0116	0.9961	94	2s	0.0072	0.9974
5	2s	0.0585	0.9800	49	2s	0.0109	0.9960	95	2s	0.0072	0.9975
6	2s	0.0472	0.9828	50	2s	0.0110	0.9962	96	2s	0.0067	0.9978
7	2s	0.0388	0.9874	51	2s	0.0114	0.9958	97	2s	0.0064	0.9977
8	2s	0.0335	0.9893	52	2s	0.0113	0.9956	98	2s	0.0072	0.9975
9	2s	0.0299	0.9898	53	2s	0.0108	0.9961	99	2s	0.0066	0.9979
10	2s	0.0284	0.9905	54	2s	0.0109	0.9964	100	2s	0.0063	0.9978

Table 4.4 shows the loss, accuracy, and time for 30 selected epochs out of the 100 epochs (10 first, 10 middle, and 10 last)Our DNN that we used for the NSL-KDD data set. The highest accuracy is in epoch 99 of 0.9979 and the lowest is in epoch 1 of 0.8260. The highest loss rate occurred in era 1 of 0.4409 and the lowest in era 100 of 0.0063.

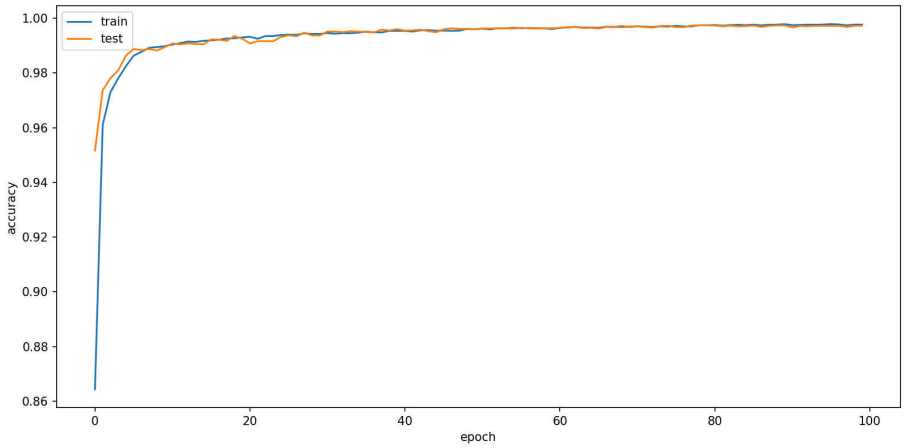


Figure 4.4: Accuracy evolution in terms of epochs on training and test data.

The curve in the figure 4.4 represents the evolution of *Accuracy* over time from an epoch to another.

[7] https://www.tensorflow.org/
[8] https://keras.io/

It shows two lines: one for the accuracy on the train data (blue) and the other for the accuracy on the test data set (orange).

The train accuracy line shows how well the model performs on the training data. The test accuracy line indicates the model's ability to generalize to unseen data. A small gap between training and testing accuracy indicates better generalizability.

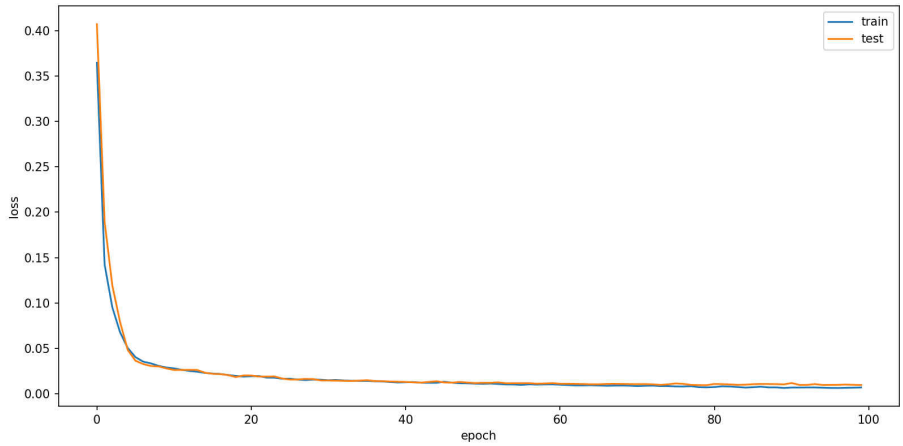


Figure 4.5: Loss evolution in terms of epochs on training and test data.

The curve in the figure 4.5 represents the *Loss* evolution over time for each Epoch and consists of two lines: one representing the Loss in the train data set and the other representing the loss in the test data set.The blue line corresponds to the loss on the train dataset, showing how the model's loss decreases over epochs during training,The orange line represents the loss on the test dataset, which measures the model's performance on unseen data. the train and test losses decrease and converge, it indicates that the model is learning effectively and generalizing well.

4.8.1 The most important metrics

The table 4.5 shows the obtained results of the proposed approach when applied on the NSL-KDD dataset:

Table 4.5: The proposed model's performance metrics on NSL-KDD dataset.

Accuracy	F1 score	Precision	False Positives	False Negative
0.9965	0.9967	0.9973	35	52
Loss	Recall	True Positive	True Negative	R2 Score
0.01390	0.9961	13370	11738	0.9888

Recall is the most critical performance metric in the context of network intrusion detection. This is so that intrusion detection can detect as many attacks as it can, even though some normal traffic may also be considered an attack.

While recall is more important in the case of intrusion detection, precision is still a key factor. This is because it is more essential to recognize all attacks than to prevent identifying regular traffic as an attack, even if some of it is. Recall is more important than precision in the context of intrusion detection because:

- The cost of a false negative is high. A false negative is an attack that is not detected by the intrusion detection system. This can allow an attacker to gain access to a network or system,

which can have serious consequences.

- The cost of a false positive is low. A false positive is legitimate traffic that is classified as an attack by the intrusion detection system. This can cause some inconvenience, but it does not have the same serious consequences as a false negative.

In addition to recall and precision, other important performance measures for network intrusion detection include:

- False positive rate: This is the proportion of legitimate traffic that is classified as an attack.
- True positive rate: This is the proportion of attacks that are correctly classified as attacks.

4.8.2 Discussion

The table 4.6 summarizes the results obtained by similar intrusion detection systems studies in the literature.

Table 4.6: results Comparison with literature

Model	Dataset	Performance
ML[47]	UNSW-NB15	Accuracy = 99.2%
ML[47]	CICIDS2017	Accuracy = 99%
ML[47]	CICIDS2018	Accuracy = 98%
ML [47]	KDD CUP 99	Accuracy = 95%
MLP [47]	NSL-KDD	Accuracy = 99%
STL [76]	NSL-KDD	Accuracy = 88.39% F1 score= 90.40% Recall = 95.95%
SMR [76]	NSL-KDD	Accuracy = 78.06% F1 score= 76.80% Recall = 63.73%
2-class RNN [133]	NSL-KDD	Accuracy = 83.28% F1 score= 83.24% Recall = 72.95%
DNN [43]	NSL-KDD	Accuracy = 87.2%
DNN [114]	NSL-KDD	Accuracy = 75.75%
STL [54]	NSL-KDD	Accuracy = 85.44% Recall = 95.95%
RNN [61]	DARPA	Accuracy = 95.37%

By comparing our results with those extracted from previous studies and research, we compare our results with those who worked on NSLKDD datasets and using the DNN model, we found that we achieved a very good accuracy ratio considering available time and hardware resources

4.9 Conclusion

This chapter provides an introduction to deep learning-based intrusion detection systems (DL-IDS) and discusses various aspects related to their implementation and evaluation. The chapter focuses on the NSL-KDD dataset, which is commonly used for training and testing DL-IDS models.

The performance metrics used for evaluating DL-IDS models are categorized into classification metrics, regression metrics, and general metrics. These metrics help assess the effectiveness of the models in detecting and classifying different types of network intrusions accurately.

The chapter also presents related work in the field of DL-IDS, highlighting previous studies and approaches in this area. This helps provide context and understanding of the existing research landscape.

Feature selection and data preprocessing techniques are discussed, as they play a crucial role in improving the performance of DL-IDS models. Proper selection and preprocessing of features can enhance the model's ability to learn meaningful patterns and make accurate predictions.

The implementation and experimentation section specifically focus on the NSL-KDD dataset. The preprocessing steps required for this dataset are explained, followed by the design of a suggested deep neural network (DNN) architecture. The architecture includes details about the algorithm, activation functions (such as ReLU and sigmoid), and optimization functions used for training the DNN model.

The chapter also covers the software and library dependencies required for implementing DL-IDS models, including Python, Anaconda, Jupyter Notebook, NumPy, Pandas, Scikit-learn, TensorFlow, and Keras.

The classification results of the DNN model on the NSL-KDD dataset are presented, along with the evaluation metrics used to assess the model's performance (precision = 99.73%, accuracy = 99.65%, F1 score = 99.67% and recall = 99.61%) . Additionally, a comparison of the obtained results with the existing literature is presented to understand the effectiveness of the proposed approach.

In conclusion, this chapter provides a comprehensive overview of the proposed DL-based IDS, including dataset selection, performance metrics, related work, feature selection, data preprocessing, implementation details, and evaluation results. This information serves as a foundation for building and evaluating DL-based IDS models for intrusion detection in network security.

GENERAL CONCLUSION

General conclusion

The focus of our work was to explore the effectiveness of deep learning techniques in detecting malicious traffic and accurately classifying it into corresponding attack classes. Our study employed a deep learning technique known as a deep neural network (DNN) to construct a flexible and efficient model for Network Intrusion Detection Systems (NIDS).

To validate and evaluate the accuracy of our proposed model, we utilized the NSL-KDD standard network penetration dataset. This dataset contains a variety of indicators representing multiple attacks mixed with normal network traffic. Each sample in the dataset is labeled, enabling the model to recognize attack patterns compared to normal traffic.

In our implementation, we utilized binary classification to assess the efficiency of our DNN-based IDS model. The objective was to evaluate its effectiveness in binary classification scenarios. Our findings demonstrated that our model achieved high accuracy in binary classification and exhibited lower false alarm rates compared to the state of the art in machine learning and deep learning-based methods.

Although the Advanced Intrusion Detection System (DNN-IDS) we developed is designed for deployment in real network environments, time constraints prevented us from realizing this implementation. Nonetheless, we believe this project has been a remarkable and exciting endeavor. It addresses an open-ended problem encompassing a broad range of topics, presenting ample opportunities for expansion and new challenges.

Looking ahead, one of the most significant directions for future research involves designing a solution that does not rely on preexisting datasets like NSL-KDD. Instead, the focus would be on working directly with raw traffic packets to detect malicious packets. This approach would have its own set of complexities but holds great potential for enhancing the accuracy and robustness of intrusion detection systems.

In conclusion, this project has been a rewarding journey, and it has opened doors to further exploration in the field of network security. We are excited about the possibilities that lie ahead and the potential impact our work can have on developing more advanced and resilient intrusion detection systems.

Bibliography

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985. 51

[2] Shaashwat Agrawal, Aditi Chowdhuri, Sagnik Sarkar, Ramani Selvanambi, Thippa Reddy Gadekallu, et al. Temporal weighted averaging for asynchronous federated intrusion detection systems. *Computational Intelligence and Neuroscience*, 2021, 2021. doi:10.1155/2021/5844728. vii, 67

[3] Meena Ahlawat and Ankita Mittal. Different communication protocols for wireless sensor networks: a review. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(3):213–216, 2015. 22

[4] Shaik Akbar, T Srinivasa Rao, and Mohammed Ali Hussain. A hybrid scheme based on big data analytics using intrusion detection system. *Indian Journal of Science and Technology*, 9(33):1–4, 2016. 58

[5] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, 2002. 21

[6] Jamal N Al-Karaki and Ahmed E Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE wireless communications*, 11(6):6–28, 2004. 21

[7] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z Emam. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189:105124, 2020. doi:10.1016/j.knosys.2019.105124. vii, 37, 50, 51, 52

[8] Md Zahangir Alom, VenkataRamesh Bontupalli, and Tarek M Taha. Intrusion detection using deep belief networks. In *2015 National Aerospace and Electronics Conference (NAECON)*, pages 339–344. IEEE, 2015. 79

[9] Massih-Reza Amini. *Apprentissage machine: de la théorie à la pratique*. Editions Eyrolles, 2015. 38

[10] James P Anderson. Computer security threat monitoring and surveillance. 1980. *James P. Anderson Co, Fort Washington, PA*, 2012. 58

[11] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. doi:10.1109/MSP.2017.2743240. 42

[12] Asmaa Shaker Ashoor and Sharad Gore. Importance of intrusion detection system (ids). *International Journal of Scientific and Engineering Research*, 2(1):1–4, 2011. 58

[13] Houssam Ahmed Amin Bahi. Optimization of hyperparameters of anns - application to the second virial coefficient (b) of fluid mixtures. Master's thesis, University of Oum El Bouaghi, 2021. http://bib.univ-oeb.dz:8080/jspui/handle/123456789/11215. 40

[14] Meriem Bahi and Mohamed Batouche. Deep learning for ligand-based virtual screening in drug discovery. In *2018 3rd international conference on pattern analysis and intelligent systems (PAIS)*, pages 1–5. IEEE, 2018. 48

[15] Driss Benhaddou, Ala Al-Fuqaha, and A Rafiq. *Wireless Sensor and Mobile Ad-Hoc Networks*. Springer, 2015. vii, 19

[16] Divya Bharti, Neha Nainta, and Himanshu Monga. Performance analysis of wireless sensor networks under adverse scenario of attack. In *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 826–828. IEEE, 2019. 62

[17] MAHUA Bhowmik, SUMA Dupalli, DHANASHRI SAINDANE, and GITANJALI MURUMKAR. Performance analysis of rf model of wireless sensor networks. *Int. J. Res. Eng. Technol*, 2:1–8, 2014. 20

[18] Rory Bray, Daniel Cid, and Andrew Hay. *OSSEC host-based intrusion detection guide*. Syngress, 2008. 68

[19] Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016. doi:10.1109/COMST.2015.2494502. 43

[20] Ismail Butun, Patrik Österberg, and Houbing Song. Security of the internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys & Tutorials*, 22(1):616–644, 2019. 60, 61, 63

[21] Hee-su Chae, Byung-oh Jo, Sang-Hyun Choi, and Twae-kyung Park. Feature selection for intrusion detection using nsl-kdd. *Recent advances in computer science*, 20132:184–187, 2013. 72

[22] Pavitra Chauhan and Nidhi Chandra. A review on hybrid intrusion detection system using artificial immune system approaches. *International Journal of Computer Applications*, 68(20), 2013. 68

[23] Gideon Creech and Jiankun Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 4487–4492. IEEE, 2013. 72

[24] Robert K Cunningham, Richard P Lippmann, David J Fried, Simson L Garfinkel, Isaac Graf, Kris R Kendall, Seth E Webster, Dan Wyschogrod, and Marc A Zissman. Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. Technical report, Massachusetts Inst Of Tech Lexington Lincoln Lab, 1999. 72

[25] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA transactions on Signal and Information Processing*, 3:e2, 2014. 37

[26] Prachi Deshpande, Subhash Chander Sharma, Sateesh Kumar Peddoju, and S Junaid. Hids: A host based intrusion detection system for cloud computing environment. *International Journal of System Assurance Engineering and Management*, 9:567–576, 2018. 68

[27] Prachi Dewal, Gagandeep Singh Narula, Vishal Jain, and Anupam Baliyan. Security attacks in wireless sensor networks: A survey. In *Cyber Security: Proceedings of CSI 2015*, pages 47–58. Springer, 2018. 60

[28] S Dewangan, A Kumar Pandey, N Verma, and D Xaxa. A comparative assessment of topologies and their issues in wireless sensor networks. *International Journal of Engineering Sciences & Research Technology*, 2015. vii, 24, 25, 26

[29] Sumeet Dua and Xian Du. *Data mining and machine learning in cybersecurity*. CRC press, 2016. 37, 41

[30] Daniel Durstewitz, Georgia Koppe, and Andreas Meyer-Lindenberg. Deep neural networks in psychiatry. *Molecular psychiatry*, 24(11):1583–1598, 2019. 48

[31] Anjum Farah. *Cross Dataset Evaluation for IoT Network Intrusion Detection*. PhD thesis, The University of Wisconsin-Milwaukee, 2020. 72

[32] Ugo Fiore, Francesco Palmieri, Aniello Castiglione, and Alfredo De Santis. Network anomaly detection with the restricted boltzmann machine. *Neurocomputing*, 122:13–23, 2013. 37

[33] Carlo Fischione. An introduction to wireless sensor networks. *Royal Institute of technology. Draft, version*, 1, 2014. 20

[34] SH Gajjar, SN Pradhan, and KS Dasgupta. Wireless sensor network: Application led research perspective. In *2011 IEEE Recent Advances in Intelligent Computational Systems*, pages 025–030. IEEE, 2011. 21

[35] Ni Gao, Ling Gao, Quanli Gao, and Hai Wang. An intrusion detection model based on deep belief networks. In *2014 Second international conference on advanced cloud and big data*, pages 247–252. IEEE, 2014. 79

[36] Salvador García, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez, and Francisco Herrera. Big data preprocessing: methods and prospects. *Big Data Analytics*, 1(1):1–22, 2016. 80

[37] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022. 44

[38] Reenkamal Kaur Gill and Monika Sachdeva. Detection of hello flood attack on leach in wireless sensor networks. In *Next-Generation Networks: Proceedings of CSI-2015*, pages 377–387. Springer, 2018. 60

[39] Diksha Giri, Samarjeet Borah, and Ratika Pradhan. Approaches and measures to detect wormhole attack in wireless sensor networks: a survey. In *Advances in Communication, Devices and Networking: Proceedings of ICCDN 2017*, pages 855–864. Springer, 2018. 60

[40] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 347–352. IEEE, 1996. 50

[41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 53

[42] Subiksha Srinivasa Gopalan, Dharshini Ravikumar, Dino Linekar, Ali Raza, and Maheen Hasib. Balancing approaches towards ml for ids: a survey for the cse-cic ids dataset. In *2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pages 1–6. IEEE, 2021. 72

[43] Sandeep Gurung, Mirnal Kanti Ghose, and Aroj Subedi. Deep learning approach on network intrusion detection system using nsl-kdd dataset. *International Journal of Computer Network and Information Security*, 11(3):8–14, 2019. 78, 89

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 52

[45] Donald O Hebb. The first stage of perception: growth of the assembly. *The Organization of Behavior*, 4(60):78–60, 1949. 47

[46] Paul Hick, Emile Aben, Kc Claffy, and Josh Polterock. the caida ddos attack 2007 dataset. *URL: http://www. caida. org/data/passive/ddos-20070804_ dataset. xml*, 2007. 72

[47] Imran Hidayat, Muhammad Zulfiqar Ali, and Arshad Arshad. Machine learning-based intrusion detection system: An experimental comparison. *Journal of Computational and Cognitive Engineering*, 2(2):88–97, 2023. doi:10.47852/bonviewJCCE2202270. 77, 89

[48] Hanan Hindy, Ethan Bayne, Miroslav Bures, Robert Atkinson, Christos Tachtatzis, and Xavier Bellekens. Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset). In *Selected Papers from the 12th International Networking Conference: INC 2020*, pages 73–84. Springer, 2021. 72

[49] John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984. doi:10.1073/pnas.81.10.3088. 47

[50] Ding-Jie Huang and Wei-Chung Teng. A defense against clock skew replication attacks in wireless sensor networks. *Journal of network and computer applications*, 39:26–37, 2014. 62

[51] Matthias Hutsebaut-Buysse, Kevin Mets, and Steven Latré. Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction*, 4(1):172–221, 2022. doi:10.3390/make4010009. vii, 42

[52] Kai Hwang, Pinalkumar Dave, and Sapon Tanachaiwiwat. Netshield: Protocol anomaly detection with datamining against ddos attacks. 01 2003. vii, 48, 65

[53] Rutuja Jadhav and Vatsala Vatsala. Security issues and solutions in wireless sensor networks. *International Journal of Computer Applications*, 162(2):14–19, 2017. 62

[54] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26, 2016. 79, 89

[55] Mohd Javaid, Abid Haleem, Ravi Pratap Singh, and Rajiv Suman. Artificial intelligence applications for industry 4.0: A literature-based study. *Journal of Industrial Integration and Management*, 7(01):83–111, 2022. doi:10.1142/S2424862221300040. 39

[56] Sayamuddin Ahmed Jilani, Chandan Koner, and Shovon Nandi. Security in wireless sensor networks: attacks and evasion. In *2020 National conference on emerging trends on sustainable technology and engineering applications (NCETSTEA)*, pages 1–5. IEEE, 2020. 60

[57] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6):e0155781, 2016. 78

[58] M Keerthika and D Shanmugapriya. Wireless sensor networks: Active and passive attacks-vulnerabilities and countermeasures. *Global Transitions Proceedings*, 2(2):362–367, 2021. vii, 59

[59] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019. ix, 66

[60] Gene H Kim and Eugene H Spafford. The design and implementation of tripwire: A file system integrity checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 18–29, 1994. 68

[61] Jihyun Kim and Howon Kim. Applying recurrent neural network to intrusion detection with hessian free optimization. In *Information Security Applications: 16th International Workshop, WISA 2015, Jeju Island, Korea, August 20–22, 2015, Revised Selected Papers 16*, pages 357–369. Springer, 2016. 79, 89

[62] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017. doi:10.48550/arXiv.1412.6980. 84

[63] Teuvo Kohonen. Description of input patterns by linear mixtures of som models. In *International Workshop on Self-Organizing Maps: Proceedings (2007)*, 2007. 47

[64] Harley Kozushko. Intrusion detection: Host-based and network-based intrusion detection systems. *Independent study*, 11:1–23, 2003. 67, 68

[65] Jan Lansky, Saqib Ali, Mokhtar Mohammadi, Mohammed Kamal Majeed, Sarkhel H Taher Karim, Shima Rashidi, Mehdi Hosseinzadeh, and Amir Masoud Rahmani. Deep learning-based intrusion detection systems: a systematic review. *IEEE Access*, 9:101574–101599, 2021. doi:10.1109/ACCESS.2021.3097247. 43

[66] Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. *Intrusion Detection: A Survey*, volume 5, pages 19–78. 01 2005. doi:10.1007/0-387-24230-9_2. vii, 66

[67] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013. 69

[68] Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. Feature selection: An ever evolving frontier in data mining. In *Feature selection in data mining*, pages 4–13. PMLR, 2010. 80

[69] Gilles Louppe, Kyunghyun Cho, Cyril Becot, and Kyle Cranmer. Qcd-aware recursive neural networks for jet physics. *Journal of High Energy Physics*, 2019(1):1–23, 2019. 50

[70] Khawar Iqbal Malik and M Mateen Yaqoob. An analytical survey on routing protocols for wireless sensor network (wsn). *International Journal of Computer Applications*, 975:8887, 2004. vii, 22

[71] Pierre-Francois Marteau. Sequence covering for efficient host-based intrusion detection. *IEEE Transactions on Information Forensics and Security*, 14(4):994–1006, 2018. 68

[72] Mehedy Masud, Latifur Khan, and Bhavani Thuraisingham. *Data mining tools for malware detection*. CRC Press, 2011. 37

[73] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943. 47

[74] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of network and computer applications*, 36(1):42–57, 2013. 67

[75] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015. 72

[76] Quamar Niyaz, Weiqing Sun, Ahmad Y Javaid, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS), BICT-15*, volume 15, pages 21–26, 2015. 89

[77] Mohd Fauzi Othman and Khairunnisa Shazali. Wireless sensor network applications: A study in environment monitoring system. *Procedia Engineering*, 41:1204–1210, 2012. vii, 21

[78] Suad Mohammed Othman, Nabeel T Alsohybe, Fadl Mutaher Ba-Alwi, and Ammar Thabit Zahary. Survey on intrusion detection system types. *International Journal of Cyber-Security and Digital Forensics*, 7(4):444–463, 2018. 58

[79] Suad Mohammed Othman, Fadl Mutaher Ba-Alwi, Nabeel T Alsohybe, and Amal Y Al-Hashida. Intrusion detection model using machine learning algorithm on big data environment. *Journal of big data*, 5(1):1–12, 2018. 67

[80] Suad Mohammed Othman and Fadl Mutaher Ba-Alwi Ammar Thabit Zahary Nabeel T.Alsohybe. Survey on intrusion detection system types. *International Journal of Cyber-Security and Digital Forensics*, 7:444–462, 2018. ix, 69

[81] Aniruddha Parvat, Souradeep Dev, Siddhesh Kadam, and Jai Chavan. Network intrusion detection system using ensemble of binary deep learning classifiers. In *Smart Trends in Information Technology and Computer Communications: Second International Conference, SmartCom 2017, Pune, India, August 18-19, 2017, Revised Selected Papers 2*, pages 3–10. Springer, 2018. 67

[82] Jared M Peterson, Joffrey L Leevy, and Taghi M Khoshgoftaar. A review and analysis of the bot-iot dataset. In *2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 20–27. IEEE, 2021. 72

[83] Ahmed Raoof, Ashraf Matrawy, and Chung-Horng Lung. Routing attacks and mitigation methods for rpl-based internet of things. *IEEE Communications Surveys & Tutorials*, 21(2):1582–1606, 2018. 60

[84] Rama Devi Ravipati and Munther Abualkibash. Intrusion detection system classification using different machine learning algorithms on kdd-99 and nsl-kdd datasets-a review paper. *International Journal of Computer Science & Information Technology (IJCSIT) Vol*, 11, 2019. ix, 73, 74

[85] Aqeel-ur Rehman, Sadiq Ur Rehman, and Haris Raheem. Sinkhole attacks in wireless sensor networks: A survey. *Wireless Personal Communications*, 106:2291–2313, 2019. 60

[86] Denis Reilly, Mark Taylor, Paul Fergus, Carl Chalmers, and Steven Thompson. The categorical data conundrum: Heuristics for classification problems—a case study on domestic fire injuries. *IEEE Access*, 10:70113–70125, 2022. 80

[87] Sathyanarayanan Revathi and A Malathi. A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT)*, 2(12):1848–1853, 2013. 72, 73

[88] Shahadate Rezvy, Miltos Petridis, Aboubaker Lasebae, and Tahmina Zebin. Intrusion detection and classification with autoencoded deep neural network. In *Innovative Security Solutions for Information Technology and Communications: 11th International Conference, SecITC 2018, Bucharest, Romania, November 8–9, 2018, Revised Selected Papers 11*, pages 142–156. Springer, 2019. 79

[89] Muhammad Noman Riaz, Attaullah Buriro, and Athar Mahboob. Classification of attacks on wireless sensor networks: A survey. *International Journal of Wireless and Microwave Technologies*, 8(6):15–39, 2018. 61

[90] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX Conference on System Administration*, LISA '99, page 229–238, USA, 1999. USENIX Association. 67

[91] Damien Rolon-Mérette, Matt Ross, Thaddé Rolon-Mérette, and Kinsey Church. Introduction to anaconda and python: Installation and setup. *Quant. Methods Psychol*, 16(5):S3–S11, 2016. 85

[92] Hossein Sadr, Mir Mohsen Pedram, and Mohammad Teshnehlab. A robust sentiment analysis method based on sequential combination of convolutional and recursive neural networks. *Neural Processing Letters*, 50:2745–2761, 2019. 50

[93] Mostafa A Salama, Heba F Eid, Rabie A Ramadan, Ashraf Darwish, and Aboul Ella Hassanien. Hybrid intelligent intrusion detection scheme. In *Soft computing in industrial applications*, pages 293–303. Springer, 2011. 37, 78

[94] Abhijit Sarmah. Intrusion detection systems: Definition, need and challenges. *Rapport Technique, SANS Institute*, 2001. 68

[95] Lars Schmarje, Monty Santarossa, Simon-Martin Schröder, and Reinhard Koch. A survey on semi-, self-and unsupervised learning for image classification. *IEEE Access*, 9:82146–82168, 2021. doi:10.1109/ACCESS.2021.3084358. 40

[96] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 43

[97] D Seethalakshmi and GM Nasira. Detecting and preventing intrusion in multi-tier web applications using double guard. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 3124–3127. IEEE, 2016. 69

[98] Jaydip Sen. Security in wireless sensor networks. *Wireless sensor networks: current status and future trends*, 407:408, 2012. 31

[99] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018. 72

[100] Shaimaa Sharafali, Noor Fallooh, and Mohammed Al-Hayani. Intrusion detection system based on machine learning and deep learning techniques: A review. 03 2023. vii, 64, 77

[101] Divya Sharma, Sandeep Verma, and Kanika Sharma. Network topologies in wireless sensor networks: A review. *International Journal of Electronics & Communication Technology*, 4(3):93–97, 2013. vii, 27, 28

[102] Mayank Kumar Sharma and Brijendra Kumar Joshi. Detection & prevention of vampire attack in wireless sensor networks. In *2017 International conference on information, communication, instrumentation and control (ICICIC)*, pages 1–5. IEEE, 2017. 62

[103] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017. 83

[104] Shamneesh Sharma, Dinesh Kumar, and Keshav Kishore. Wireless sensor networks-a review on topologies and node architecture. *International Journal of Computer Sciences and Engineering*, 1(2):19–25, 2013. vii, 24

[105] Lei Shi, Qingchen Liu, Jinliang Shao, and Yuhua Cheng. Distributed localization in wireless sensor networks under denial-of-service attacks. *IEEE Control Systems Letters*, 5(2):493–498, 2020. 60

[106] Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep ehr: a survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604, 2017. 43

[107] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012. URL: https://www.sciencedirect.com/science/article/pii/S0167404811001672, doi:https://doi.org/10.1016/j.cose.2011.12.012. 72

[108] Thalles Silva. An intuitive introduction to generative adversarial networks (gans) — freecodecamp.org. https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394. [Accessed 05-Jun-2023]. vii, 53

[109] Dimitris Sklavounos, Aloysius Edoh, and George Paraskevopoulos. Utilization of statistical control charts for dos network intrusion detection. *International Journal of Cyber-Security and Digital Forensics*, 7(2):166–175, 2018. 67

[110] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011. 50

[111] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010. 37

[112] S Sobin Soniya and S Maria Celestin Vigila. Intrusion detection system: Classification and techniques. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–7. IEEE, 2016. 58

[113] Basant Subba, Santosh Biswas, and Sushata Karmakar. Host based intrusion detection system using frequency analysis of n-gram terms. In *TENCON 2017-2017 IEEE Region 10 Conference*, pages 2006–2011. IEEE, 2017. 68

[114] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep learning approach for network intrusion detection in software defined networking. In *2016 international conference on wireless networks and mobile communications (WINCOM)*, pages 258–263. IEEE, 2016. 78, 89

[115] Muhammad Imran Tariq, Nisar Ahmed Memon, Shakeel Ahmed, Shahzadi Tayyaba, Muhammad Tahir Mushtaq, Natash Ali Mian, Muhammad Imran, and Muhammad W Ashraf. A review of deep learning security and privacy defensive techniques. *Mobile Information Systems*, 2020:1–18, 2020. doi:10.1155/2020/6535834. vii, 44

[116] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009. 72, 73

[117] Pablo Torres, Carlos Catania, Sebastian Garcia, and Carlos Garcia Garino. An analysis of recurrent neural networks for botnet detection behavior. In *2016 IEEE biennial congress of Argentina (ARGENCON)*, pages 1–6. IEEE, 2016. 79

[118] Gregor Urban, Niranjan Subrahmanya, and Pierre Baldi. Inner and outer recursive neural networks for chemoinformatics applications. *Journal of chemical information and modeling*, 58(2):207–211, 2018. 50

[119] John R Vacca. *Computer and information security handbook*. Newnes, 2012. 58

[120] P Valarmathi. Study of wireless sensor networks with its applications and security. *International Journal of Research in Engineering and Innovation (IJREI)*, 2(4):442–448, 2018. 20, 28

[121] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020. doi:10.1007/s10994-019-05855-6. 41

[122] Parag Verma, Ankur Dumka, Anuj Bhardwaj, Navneet Kaur, Alaknanda Ashok, Anil Kumar Bisht, and Raksh Pal Singh Gangwar. *Security Issues for Wireless Sensor Networks*. CRC Press, 2022. doi:10.1109/CSAC.1998.738566. 20, 21, 30

[123] Rishabh Verma, Sumit J Darak, Vinay Tikkiwal, Himani Joshi, and Rohit Kumar. Countermeasures against jamming attack in sensor networks with timing and power constraints. In *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, pages 485–488. IEEE, 2019. 60

[124] Rishita Verma and Sourabh Bharti. A survey of network attacks in wireless sensor networks. In *Information, Communication and Computing Technology: 5th International Conference, ICICCT 2020, New Delhi, India, May 9, 2020, Revised Selected Papers*, pages 50–63. Springer, 2020. 60, 62, 63

[125] Giovanni Vigna and Richard A Kemmerer. Netstat: A network-based intrusion detection approach. In *Proceedings 14th Annual Computer Security Applications Conference (Cat. No. 98EX217)*, pages 25–34. IEEE, 1998. 67

[126] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. Applying convolutional neural network for network intrusion detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1222–1228. IEEE, 2017. 79

[127] L Vokorokos and A Baláž. Host-based intrusion detection system. In *2010 IEEE 14th International Conference on Intelligent Engineering Systems*, pages 43–47. IEEE, 2010. 68

[128] Min Wei, Chunmeng Rong, Erxiong Liang, and Yuan Zhuang. An intrusion detection mechanism for ipv6-based wireless sensor networks. *International Journal of Distributed Sensor Networks*, 18(3):15501329221077922, 2022. doi:10.1177/15501329221077922. 58

[129] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. Technical report, Stanford University Ca Stanford Electronics Labs, June, 1960. 47

[130] Kehe Wu, Zuge Chen, and Wei Li. A novel intrusion detection model for a massive network using convolutional neural networks. *Ieee Access*, 6:50850–50859, 2018. 79

[131] Haomeng Xie, Zheng Yan, Zhen Yao, and Mohammed Atiquzzaman. Data collection for security measurement in wireless sensor networks: A survey. *IEEE Internet of Things Journal*, 6(2):2205–2224, 2018. 61

[132] Hui Yang. Data preprocessing. *Pennsylvania State University: Citeseer*, 2018. 80

[133] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017. 89

[134] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? corr abs/1411.1792 (2014). *arXiv preprint arXiv:1411.1792*, 2014. 42

[135] Kun-Ming Yu, Ming-Feng Wu, and Wai-Tak Wong. Protocol-based classification for intrusion detection. In *Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science*, ACACOS'08, page 29–34, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS). 67

[136] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015. 48

[137] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. 42