**People's Democratic Republic of Algeria**
**Ministry of Higher Education and Scientific Research**
**YAHIA FARES UNIVERSITY OF MEDEA**



**Faculty of Sciences**

Department of Mathematics and Computer Science

# Final Master Project

**Branch:** Computer science

**Specialty**: Systems Engineering and Web Technologies

**Theme**

# Intrusion Detection System based on Fire Hawk Optimizer Algorithm and Deep Learning

**Prepared and presented by:**

Ms. FERIAL BEN YAHIA

Ms. IBTIHEL BOUHALLA

**Proposed and directed by:**

Dr. BRAHIM SAHMADI

Class: 2023/2024

# *Acknowledgment*

*As a preamble to this dissertation, we thank **ALLAH**, who helps us and gives us patience, courage, strength, and perseverance throughout our lives and our years of study.*

*We extend our heartfelt thanks to my supervisor, **Dr. BRAHIM SAHMADI**, who played a major role in guiding and directing me during the preparation of this thesis. He did not withhold his advice and guidance, and his encouragement and motivation greatly impactedmy continued hard work and diligence.*

*Furthermore, we sincerely acknowledge and appreciate the professors who imparted theirwisdom and shared their extensive expertise throughout my academic journey.*

*We would like to express our sincere gratitude to the esteemed jury, who dedicated their timeand expertise to evaluate my work.*

# *Dedication*

*The journey was neither short nor was the path paved with ease, but I did it. Praise be to* ***Allah*** *who made the beginnings easy and helped us reach the ends.*

***I*** *dedicate this success first to my ambitious self, which began with ambition and ended insuccess, and then to everyone who strove with me to complete my university journey.*

***To*** *the one who has always been my role model and guide in life, to the one who has overwhelmed me with his love, care, and protection, to the one who taught me the meaning ofpatience, perseverance, and hard work:*

*(beloved father).*

***To*** *the hidden hand that removed thorns from my path, who endured every moment of pain Iwent through, supported me, stayed up many nights for my comfort, and woke up at dawn topray for me:*

*(beloved mother).*

***To*** *those who have always been by my side, encouraging and supporting me in tough timesand joyous moments. You have been the strength I drew my determination from and the source I turned to in critical times. Your love and support are priceless:*

*(dear siblings).*

***To*** *those who shared the journey with me and were the best companions. You have always been by my side, offering support and assistance to each other, sharing dreams and ambitions. Your friendship is a great blessing in my life, and I cannot express how grateful Iam to you:*

*(dear friends)*

***To*** *all these dear ones,*

***I*** *dedicate this humble work with all my love and appreciation. Thanks to your support and assistance, I was able to reach this achievement and realize my dream.*

*Thank you all.*

***BEN YAHIA Ferial and BOUHALLA Ibtihel***

# Abstract

In light of the increasing cyber threats, intrusion detection systems (IDS) have become critical for protecting computer systems. This thesis introduces an advanced IDS that combines the Fire Hawk Optimizer (FHO) algorithm with a one-dimensional Convolutional Neural Network (CNN-1D) to improve accuracy and efficiency in detecting intrusions. The FHO algorithm is used for feature selection, reducing input data dimensionality while retaining essential features. These optimized features are then analyzed by the CNN-1D model, which identifies temporal and spatial patterns in network traffic. The system's performance is evaluated using the CSE-CIC-IDS2018 dataset, demonstrating high accuracy, recall, precision, and F1 scores in detecting various network attacks. The FHO algorithm enhances efficiency by reducing computational overhead and training time without sacrificing detection performance. Future research will involve testing the system on additional datasets and exploring its deployment in real-world network environments.



**Keywords**

Intrusion detection systems (IDS), cyber threats, deep learning, Fire Hank Optimizer (FHO), CSE-CIC-IDS2018 dataset, Random Forest, convolutional neural network (CNN).

## ملخص

في ظل تزايد التهديدات الإلكترونية، تكتسب أنظمة كشف التسلل (IDS) أهمية قصوى لحماية أمان الأنظمة الحاسوبية. تقدم هذه الأطروحة نظام IDS متقدمًا يجمع بين خوارزمية Fire Hawk Optimizer (FHO) والشبكة العصبية التلافيفية أحادية البعد (CNN-1D) لتحسين دقة وكفاءة كشف التسلل. يتم استخدام خوارزمية FHO لاختيار الميزات، مما يقلل من أبعاد البيانات المدخلة مع الاحتفاظ بالميزات الأساسية. يتم بعد ذلك تحليل هذه الميزات المحسنة بواسطة نموذج CNN-1D، الذي يحدد الأنماط الزمنية والمكانية ضمن بيانات التدفق الشبكي. تم تقييم أداء النظام باستخدام مجموعة بيانات CSE-CIC-IDS2018، واعطت نتائج جيدة وفق معايير precision ،accuracy، recall وF1-score في اكتشاف هجمات الشبكة المختلفة. تعمل خوارزمية FHO على تحسين الكفاءة عن طريق تقليل العبء الحسابي ووقت التدريب دون التضحية بأداء الكشف. ستتضمن الأبحاث المستقبلية اختبار أداء النظام على مجموعات بيانات إضافية واستكشاف امكانيات تطبيقه في بيئات شبكات حقيقية.



**الكلمات المفتاحية**

أنظمة كشف التسلل (IDS)، التهديدات الإلكترونية، التعلم العميق، محسن صقر النار (FHO)، الشبكة العصبية التلافيفية (CNN)، مجموعة بيانات CSE-CIC-IDS2018، الغابة العشوائية.

# Résumé

Face à la multiplication des cyber menaces, les systèmes de détection d'intrusion (IDS) sont devenus essentiels pour protéger les systèmes informatiques. Ce mémoire présente un IDS avancé qui combine l'algorithme Fire Hawk Optimizer (FHO) avec un réseau neuronal convolutif unidimensionnel (CNN-1D) pour améliorer la précision et l'efficacité de la détection d'intrusion. L'algorithme FHO est utilisé pour la sélection des attributs, réduisant la dimensionnalité des données d'entrée tout en conservant les caractéristiques essentielles. Cet ensemble d'attributs optimisé est ensuite analysé par le modèle CNN-1D, qui identifie les modèles temporels et spatiaux du trafic réseau. Les performances du système sont évaluées à l'aide de l'ensemble de données CSE-CICIDS2018, démontrant une exactitude, un rappel, une précision et des scores F1 élevés dans la détection de diverses attaques réseau. L'algorithme FHO améliore l'efficacité en réduisant les frais de calcul et le temps de formation de modèle sans sacrifier les performances de détection. Les recherches futures impliqueront des tests sur des ensembles de données supplémentaires et un déploiement dans le monde réel.



**Mots clés**

Systèmes de détection d'intrusions (IDS), cybermenaces, apprentissage profond, algorithme Fire Hawk Optimiser (FHO), base de données CSE-CIC-IDS2018, Random Forest, réseau de neurones convolutifs (CNN).

# Table of contents

**List of Acronyms** ..... **i**
**List of figures** ..... **iii**
**List of tables** ..... **vi**
**General Introduction** ..... **1**

**Chapter I: Intrusion Detection Systems (IDS)**
**1.Introduction** ..... **5**
**2.Definition** ..... **5**
**3.General architecture** ..... **5**
**4.Types of intrusion detection systems** ..... **6**
**4.1.Host-based IDS (HIDS)** ..... **7**
**4.2.Network-based IDS (NIDS)** ..... **7**
**4.3.Application-based IDS** ..... **7**
**4.4.Wireless IDS (WIDS)** ..... **8**
**5.Methodology of intrusion detection** ..... **8**
**5.1.Signature-based detection** ..... **9**
**5.2.Anomaly-based detection** ..... **9**
**5.3.Hybrid-Based detection** ..... **10**
**6.IDS Efficiency** ..... **11**
**7.Limitations and vulnerabilities of IDS** ..... **11**
**8.Advantages of using IDS** ..... **12**
**9.Enhancing Reliability in Intrusion Detection** ..... **13**
**10.Conclusion** ..... **14**

**Chapter II: Fire Hawk Optimizer (FHO)**
**1.Introduction** ..... **16**
**2.Definition** ..... **16**
**3.Fire Hawk optimizer (FHO) algorithm** ..... **17**
**3.1.Inspiration** ..... **17**
**3.2.Mathematical model** ..... **17**
**3.3.FHO algorithm** ..... **21**
**3.4.Fire Hawk Optimizer (FHO) Algorithm Flowchart** ..... **22**
**4.Previous Work on Fire Hawk Optimizer** ..... **23**
**5.Conclusion** ..... **24**

**Chapter III: Deep Learning (DL)**

**1.Introduction ..... 26**

**2.Definition ..... 26**

**3.Deep Learning Classification ..... 26**

**3.1.Deep Networks for Unsupervised or Generative Learning ..... 26**

**3.2.Deep Networks for Supervised Learning ..... 27**

**3.3.Hybrid Deep Networks ..... 27**

**4.Some methods of deep learning ..... 27**

**4.1.Artificial neural network ..... 28**

**4.1.1.Formal Neuron (Perceptron) ..... 28**

**4.1.2.The different layers of a neural network ..... 29**

**4.1.3.Activation function ..... 30**

**4.1.4.The learning rate ..... 30**

**4.1.5.Loss function ..... 31**

**4.1.6.Batch size: ..... 31**

**4.1.7.Propagation ..... 31**

**4.2.Convolutional Neural Networks (CNNs) ..... 32**

**4.3.Recurrent Neural Networks (RNN) ..... 34**

**4.3.1. Long Short-Term Memory Networks (LSTMs) ..... 35**

**4.4. Autoencoders ..... 35**

**4.5. Deep Belief Networks (DBNs) ..... 36**

**5.Evaluation metrics of models ..... 37**

**6.Conclusion ..... 40**

**Chapter IV: Conception of the proposed approach**

**1.Introduction ..... 42**

**2.General Architecture of the proposed approach ..... 42**

**3.Pre-processing of dataset ..... 42**

**3.1.Data cleaning ..... 42**

**3.2.Data Balancing ..... 43**

**3.3.Data Normalization ..... 43**

**4.Feature selection ..... 44**

**5.Data classification ..... 46**

**5.1.Random Forest (RF) Classifier ..... 46**

**5.2.One-dimensional Convolutional Neural Networks (1D CNN) method ..... 46**

**5.3. The Architecture of 1D CNN model ..... 47**

**5.3.1. One-dimensional convolutional layer ..... 47**
**5.3.2.Pooling Layer ..... 48**
**5.3.3.Flatten layer and dropout layer ..... 48**
**5.3.4.Dense fully connected layer ..... 48**
**5.4.Model Architecture of 1D CNN for IDS ..... 49**
**6.Conclusion ..... 50**

**Chapter V: Tests and results**
**1.Introduction ..... 52**
**2.Development environment ..... 52**
**2.1.Hardware specifications ..... 52**
**2.2.Software specifications and libraries ..... 52**
**2.2.1.Software ..... 52**
**2.2.2.Libraries ..... 53**
**3.Description of the database used ..... 54**
**4.Dimensionality Reduction ..... 57**
**4.1.Results of dimensionality reduction ..... 57**
**4.2.Analysis of the results of dimensionality reduction ..... 57**
**5.Classification ..... 60**
**5.1.Evaluation parameters of the CNN network ..... 60**
**5.2.Classification results ..... 60**
**5.3. Analysis of classification results ..... 61**
**5.3.1.Dataset of day 02-15-2018 ..... 61**
**5.3.2.Dataset of day 02-21-2018 ..... 62**
**5.3.3.Dataset of day 02-22-2018 ..... 64**
**5.3.4.Dataset of day 02-23-2018 ..... 65**
**6.Comparison of the CNN model with the Random Forest model ..... 67**
**7.Conclusion ..... 68**
**General Conclusion ..... 69**
**Bibliographic references** ..... 71

# LIST OF ACRONYMS

**AI:** Artificial intelligence

**AE:** Autoencoder

**ANN**: Artificial neural network

**ACC:** Accuracy

**CNN**: Convolutional Neural Network

**CNN-1D**: One-Dimensional Convolutional Neural Network

**CM:** Confusion Matrix

**CR:** Classification rate

**DBN**: Deep Belief

**DL:** Deep Learning

**DNN**: Deep Neural Network

**FH:** Fire Hawk

**FHO**: Fire Hawk Optimizer

**FN:** False Negatives

**FP:** False Positives

**GB**: Global Best solution

**HIDS**: Host Intrusion Detection System

**IDS:** Intrusion Detection System

**IP:** Internet Protocol

**LSTM**: Long Short-Term Memory Network

**ML**: Machine Learning

**MLP**: Multi-Layer Perceptron

**NIDS:** Network Intrusion Detection System

**PC**: Personal Computer

**PR:** Prey

## LIST OF ACRONYMS

**PREC**: Precision

**RBM**: Restricted Boltzmann Machine

**REC**: Recall

**ReLU**: Rectified Linear Unit

**RF:** Random Forest

**RNN**: Recurrent Neural Network

**SIEM:** Security Information and Event Management

**SP:** Safe Place

**SN:** Sensitivity

**TN:** True Negative

**TP:** True Positives

**UEBA:** User and Entity Behavior Analytics

**WIDS:** Wireless Intrusion Detection System

# List of figures


| Chapter I: Intrusion Detection Systems (IDS) | | Page |
|---|---|---|
| **Figure I.1** | IDS General Architecture | 6 |
| **Figure I.2** | Classification of IDSs by detection method | 6 |
| **Figure I.3** | Signature-Based Detection (Flow diagram | 7 |
| **Figure I.4** | Anomaly-based detection (Flow diagram) | 8 |
| **Figure I.5** | Anomaly based methodology architecture | 9 |
| **Figure I.6** | Hybrid based methodology architecture | 9 |
| **Figure I.7** | Classification of IDSs by analyzed activities | 10 |
| **Figure I.8** | Host based IDS | 10 |
| **Chapter II: Fire Hawk Optimizer (FHO)** | | |
| **Figure II.1** | Photographs of Fire Hawks' behavior around fires | 17 |
| **Figure II.2** | Schematic presentation of determining fire hawks and prey in the search space | 19 |
| **Figure II.3** | Schematic presentation for measuring the total distance between the Fire Hawks and the prey | 20 |
| **Figure II.4** | Pseudo-code of FHO | 22 |
| **Figure II.5** | Flowchart of FHO | 23 |
| **Chapter III: Deep learning** | | |
| **Figure III.1** | Classification of deep learning techniques | 28 |
| **Figure III.2** | Example of a simple perceptron | 29 |
| **Figure III.3** | Example of fully connected network with tree layers (Input, Hidden and Output). | 30 |
| **Figure III.4** | The structure of a classic CNN model | 32 |
| **Figure III.5** | Application of the convolution process | 33 |
| **Figure III.6** | The process of different pooling layer | 34 |
| **Figure III.7** | Image classification using cross-entropy loss | 34 |
| **Figure III.8** | The structure of a LSTM unit | 35 |
| **Figure III.9** | Deep Autoencoder architecture | 37 |
| **Figure III.10** | The architecture of a DBN (Deep Belief Network) | 38 |

| | | |
|---|---|---|
| **Figure III.11** | Elliptical representation of four binary results of the test set classification | 39 |
| **Figure. III.12** | Two ellipses show how accuracy is calculated | 39 |
| **Figure. III.13** | Two ellipses show how the sensitivity is calculated | 39 |
| **Figure. III.14** | Two ellipses show how precision is calculated | 40 |
| **Figure. III.15** | Confusion matrix Table structure | 40 |
| **Chapter IV: Conception of the proposed approach** | | |
| **Figure IV.1** | Diagram of the proposed approach for intrusion detection | 44 |
| **Figure IV.2** | A sample 1D CNN configuration with 3 CNN layers and 2MLP layers | 48 |
| **Figure IV.3** | One-dimensional convolutional neural network (1D CNN) architecture | 49 |
| **Figure IV.4** | Model Architecture of 1D CNN for IDS | 50 |
| **Chapter V: Tests and results** | | |
| **Figure V.1** | Logos of software specifications and libraries used | 54 |
| **Figure V.2** | Frequencies for intrusion categories | 55 |
| **Figure V.3** | Network Topology | 56 |
| **Figure V.4** | Graph of Fitness and the Number of Selected Features on 15-02-2018 | 59 |
| **Figure V.5** | Graph of Fitness and the Number of Selected Features on 21-02-2018 | 59 |
| **Figure V.6** | Graph of Fitness and the Number of Selected Features on 22-02-2018 | 60 |
| **Figure V.7** | Graph of Fitness and the Number of Selected Features on 23-02-2018 | 60 |
| **Figure V.8** | Training and Validation Accuracy of CNN 1D Model on 02-15-2018 dataset | 62 |
| **Figure V.9** | Training and Validation Loss of CNN 1D Model on 02-15-2018 dataset | 62 |
| **Figure V.10** | Confusion Matrix of 02-15-2018 dataset | 63 |
| **Figure V.11** | Training and Validation Accuracy of CNN 1D Model On 02-21-2018 dataset | 63 |
| **Figure V.12** | Training and Validation Loss of CNN-1D Model on 02-21-2018 dataset | 64 |

| | | |
|---|---|---|
| **Figure V.13** | Confusion Matrix of 02-21-2018 dataset | 64 |
| **Figure V.14** | Training and Validation Accuracy of CNN-1D Model On 02-22-2018 dataset | 65 |
| **Figure V.15** | Training and Validation Loss of CNN-1D Model On 02-22-2018 dataset | 65 |
| **Figure V.16** | Confusion Matrix of 02-22-2018 dataset | 66 |
| **Figure V.17** | Training and Validation Accuracy of CNN-1D Model On 02-23-2018 dataset | 66 |
| **Figure V.18** | Training and Validation Loss of CNN-1D Model On 02-23-2018 dataset | 67 |
| **Figure V.19** | Confusion Matrix of 02-23-2018 dataset | 67 |

# List of tables


| | Chapter V: Tests and results | Page |
|---|---|---|
| **TableV.1** | Description of a part of features in CSE-CIC-IDS2018 dataset | 56 |
| **Table V.2** | Physical characteristics of the CSE-CIC-IDS-2018 database | 57 |
| **Table V.3** | Results of dimensionality reduction | 58 |
| **Table V.4** | Classification results of the four files used | 61 |
| **Table V.5** | Comparison of the CNN model with the Random Forest model | 68 |

# General Introduction

In today's interconnected world, cyber threats are evolving at an unprecedented rate, posing significant risks to our digital infrastructure. Network security has become a critical aspect of modern information technology systems due to the increasing reliance on interconnected networks. Unauthorized access, data breaches, and malicious activities threaten the confidentiality, integrity, and availability of network resources. To counter these threats, Intrusion Detection Systems (IDS) have been developed to monitor and analyze network traffic to detect and prevent unauthorized or anomalous behavior.

Intrusion Detection Systems are crucial in identifying and responding to security breaches in real time by monitoring network traffic and analyzing activity patterns. They play a crucial role in maintaining the security and integrity of computer networks by detecting various types of attacks, such as port scanning and malware infiltration. Traditional IDS approaches rely on predefined rules and signatures to identify known attack patterns, but theyoften struggle to detect novel or sophisticated attacks.

To overcome the limitations of traditional IDS methods, advanced techniques such as deep learning have emerged. Deep learning algorithms, particularly Convolutional Neural Networks (CNNs), shown remarkable capabilities in automatically extracting intricate patterns and features from complex data, including detecting abnormal network traffic.

The motivation behind our research is to develop a deep learning-based method for abnormal network traffic detection using an Improved One-Dimensional Convolutional Neural Network (CNN-1D). This method aims to significantly enhance the extraction accurate trafficfeatures and improve the identification of attack traffic. CNN-1D applies multiple filters to the raw network traffic data to extract and learn higher-level features.

The design of IDS exploiting deep learning for network cybersecurity involves several key components. Firstly, a robust and comprehensive dataset is required for training and evaluating the Deep Learning models. The CSE-CIC-IDS2018 dataset that encompasses a wide range of network traffic patterns, including both normal and malicious activities is usedin this study to enable effective learning. Secondly, we focus on feature selection technique tocapture relevant information from the network data through the integration of the Fire Hawk Optimizer (FHO) algorithm.

The Fire Hawk Optimizer (FHO) is a novel algorithm inspired by collaborative hunting behaviors of fire hawks. These birds are known for their exceptional ability to hunt cooperatively, often spreading fires deliberately to flush out prey. This collaborative and strategic behavior forms the basis of the FHO algorithm, making it a powerful tool for featureselection in high-dimensional datasets. Feature selection is a crucial process in deep learning as it involves identifying the most informative features that contribute to the predictive accuracy of models. By applying the FHO algorithm, we select relevant features, which improves the performance of the deep learning model by reducing data dimensionality, increasing computational efficiency, and reducing the risk of overfitting.

This thesis is structured into five chapters as follows:

The first chapter is dedicated to the presentation of an intrusion detection system, their operating principles, and the various concepts related to intrusion detection.

The second chapter provides an in-depth exploration of the Fire Hawk Optimizer (FHO)algorithm. It gives a precise definition and detailed mathematical modeling of the algorithm. Key aspects such as the initialization process, and exploration and exploitation procedures are thoroughly explained.

The third chapter presents an overview of deep learning, its types and its various methods. It also introduces evaluation metrics for assessing the performance of deep learning models.

The fourth chapter details the development of the proposed learning method for intrusion detection, integrating FHO algorithm and deep learning.

The fifth and final chapter presents the details of the dataset used in the tests and discusses the results obtained by the proposed method.

Finaly, we end this thesis by a general conclusion, summarizes the key findings, and presents perspectives and future directions for research based on the outcomes of the study.

# Chapter I

## Intrusion Detection Systems (IDS)

# 1. Introduction

Security is a major concern for all networks in today's enterprise environment. Hackers and intruders have made numerous successful attempts to disrupt high-profile company networks and web services.

To secure network infrastructure and communication over the Internet, various methods have been developed including firewalls, encryption, and virtual private networks. Intrusion detection is a relatively new addition to these techniques, emerging in recent years. Intrusion Detection Systems (IDS) play a crucial role in cybersecurity by identifying potential threats and unauthorized access to computer networks.

This chapter gives a comprehensive understanding of how IDS function to identify and mitigate potential security breaches, by examining their various types, their detection techniques, and the components that comprise these systems.

# 2. Definition

Intrusion detection is the process of monitoring and analyzing network traffic or system activities for signs of malicious behavior or policy violations. The primary purpose of An Intrusion Detection System (IDS) is to detect potential security breaches in real-time and alert administrators to take appropriate action. Unlike preventive measures such as firewalls, IDS are designed to identify and respond to threats that have penetrated initial defenses [1].

An IDS operates continuously, providing real-time information about the system's state, monitoring user activities, and detect various types of malicious activities that pose a threat to computers and devices. These activities can include network attacks targeting vulnerable services, attempts to escalate privileges, unauthorized access to sensitive files, and the actions of malware such as viruses, Trojans, and worms [1,2].

# 3. General architecture

IDS are composed of several key components that work together to detect and respond to threats. The figure (Figure I.1) below illustrates the general architecture of an IDS.

- Data gathering device (sensor): This component is responsible for collecting data from the monitored system.

- Detector (Intrusion Detection (ID) analysis engine): The detector processes the data collected from sensors to identify intrusive activities.

- Knowledge base (database): This database contains information collected by the sensors but in a preprocessed format. It includes a knowledge base of attacks and their signatures, filtered data, data profiles, etc. This information is typically provided by network and security experts.

- Configuration device: This component provides information about the current state of the IDS.

- Response: This component initiates actions when an intrusion is detected. Responses can be either automated (active) or involve human interaction (inactive).

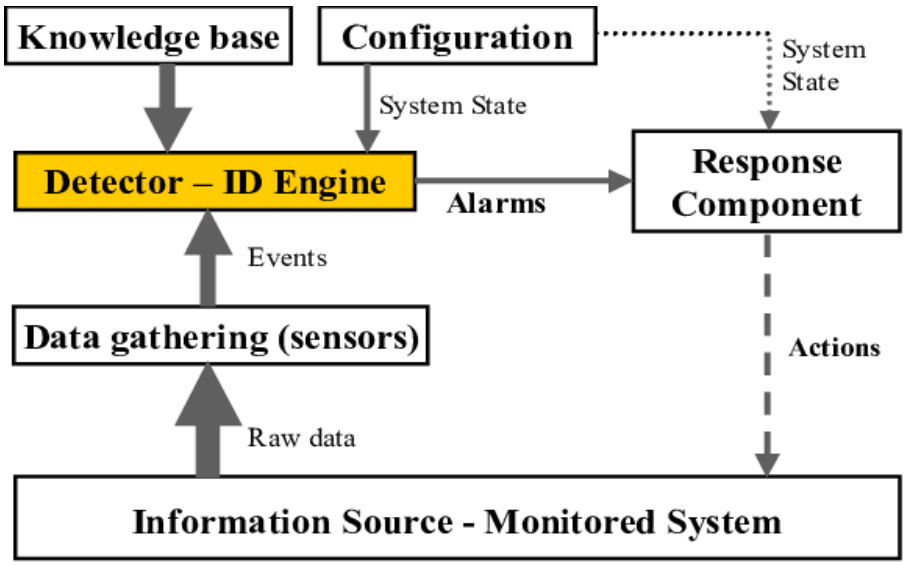



Figure I.1: IDS General Architecture [3].

## 4. Types of intrusion detection systems:

There are many types of IDSs, which can be summarized as follows:

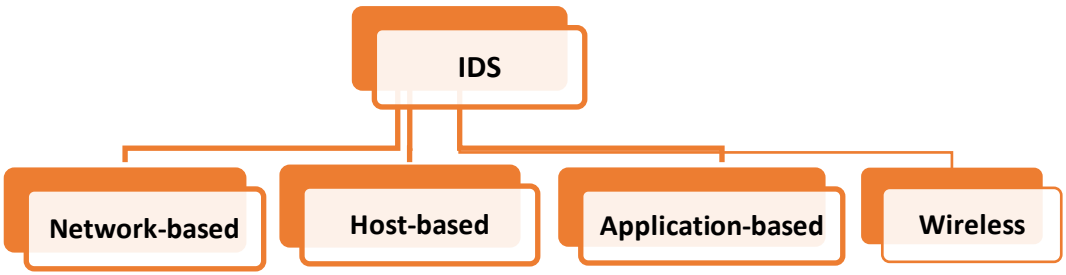



Figure I.2: Classification of IDSs by analyzed activities [5].

## 4.1. Host-based IDS (HIDS)

Host-based Intrusion Detection Systems (HIDS) are agents installed on individual hosts to analyze their activities, including files, processes, and system logs. HIDSs have various resources at their disposal to monitor the system. They can compare snapshots of the system to detect unauthorized or suspicious activities. Indicators of potential attacks include multiple failed logins attempts and unusually high CPU usage over extended periods. Some HIDSs also perform kernel-based detection by analyzing system calls and changes to system binaries. Additionally, they can monitor user activities, potentially serving as tools for user surveillance.

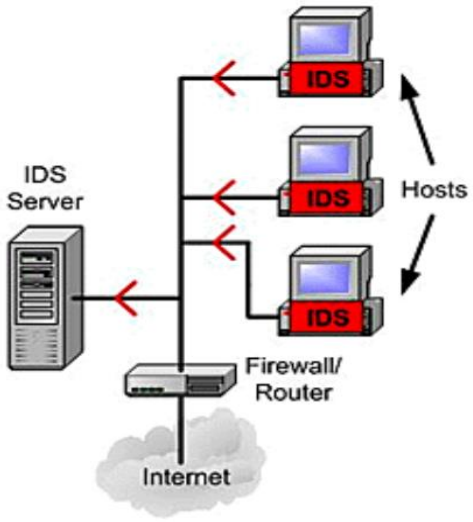



Figure I.3: Host based IDS [4].

## 4.2. Network-based IDS (NIDS)

NIDS typically uses sensors placed at various points within a network. Traffic analysis is performed either directly by the sensor or remotely by a central controller. NIDS are more scalable and cross-platform compatible compared to host-based IDS (HIDS), making them more popular for protecting a company's IT infrastructure. However, using both NIDS and HIDS simultaneously can provide a higher level of security [5] (see Figure I.4).

## 4.3. Application-based IDS

Application-based Intrusion Detection Systems (IDS) are a specialized form of Host-based Intrusion Detection Systems (HIDS) designed to monitor specific applications. These IDSs analyze interactions between users and applications, such as file executions or modifications, logs, authorizations, and any other potential abnormal activities. They can profile specific users to identify suspicious events. Some application-based IDSs also have the capability to access data before it is encrypted, functioning as middleware between the application and the encrypted data for storage [5].

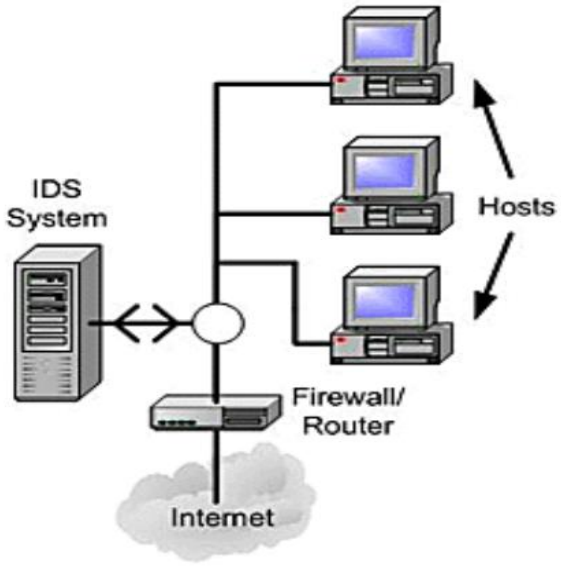



Figure I. 4: Network-based IDS [4].

### 4.4. Wireless IDS (WIDS)

Wireless Intrusion Detection Systems (WIDS) monitor and analyze wireless traffic to identify potential attacks. Wireless networks, including ad-hoc networks, wireless mesh networks, and wireless sensor networks, are susceptible to various types of attacks such as Sinkhole attacks, Spoofed Altered Routing attacks, Flood attacks, and Sybil attacks.

The unique characteristics of wireless networks, such as their presence in open environments, limited computational power of sensors, battery life, and memory capacity, make them distinct from wired networks. Consequently, Intrusion Detection Systems (IDS) designed for wired networks cannot be directly applied to wireless networks. Due to their dynamic infrastructures, wireless networks are more vulnerable to attacks compared to wired networks [4].

## 5. Methodology of intrusion detection

The methodology of an Intrusion Detection System (IDS) revolves around identifying various forms of malicious network traffic and computer usage that may go unnoticed by traditional firewalls. This is crucial for ensuring robust protection against actions that could compromise the availability, integrity, or confidentiality of computer systems [6], There are three main methodologies used by IDSs (as described in Figure I.5):

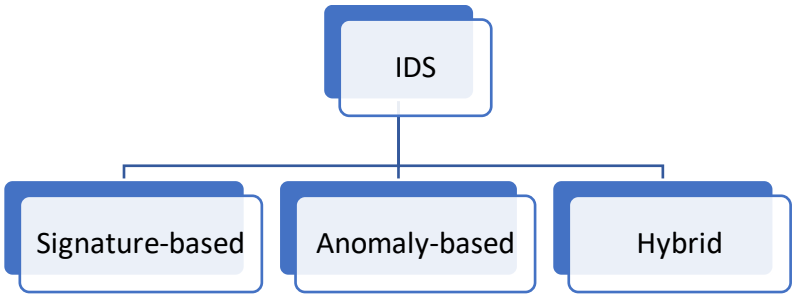



Figure I.5: Classification of IDSs by detection method [6].

## 5.1. Signature-based detection

Signature-based detection is a straightforward method that involves comparing network traffic with a signature database. When a match is found, an alert is generated, otherwise, the traffic continues without intervention [7]. This approach relies on known signatures of potential attacks for detection. However, it can only identify threats that are already known, making it ineffective against unknown threats. Precise signature matching is crucial for detection; even a small variation from a known threat signature can result in the system failing to detect the attack.

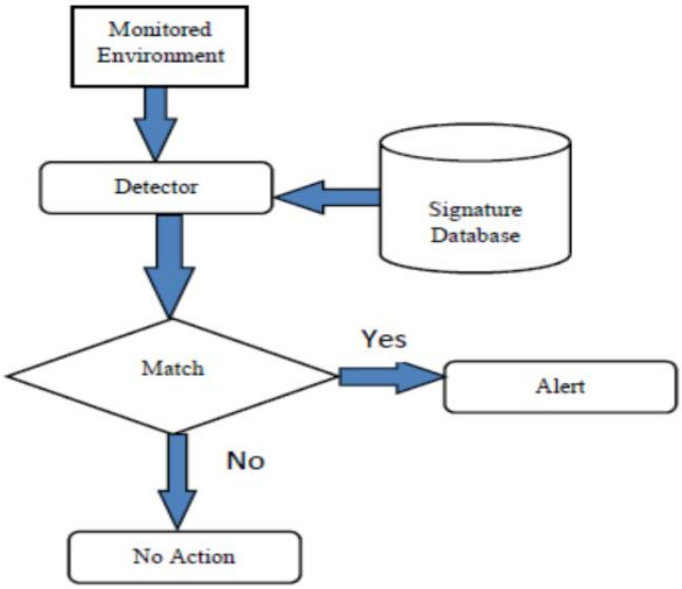



Figure I.6: Signature based methodology architecture [8].

## 5.2. Anomaly-based detection

Anomaly-based detection safeguards against unknown threats by identifying anomalies, which are deviations from expected behavior. When traffic deviates from a set baseline, the IDS system triggers an alert indicating a possible intrusion [2]. Although effective against zero-day attacks, this approach can also produce a large number of false positive alerts resulting from benign activity [9].

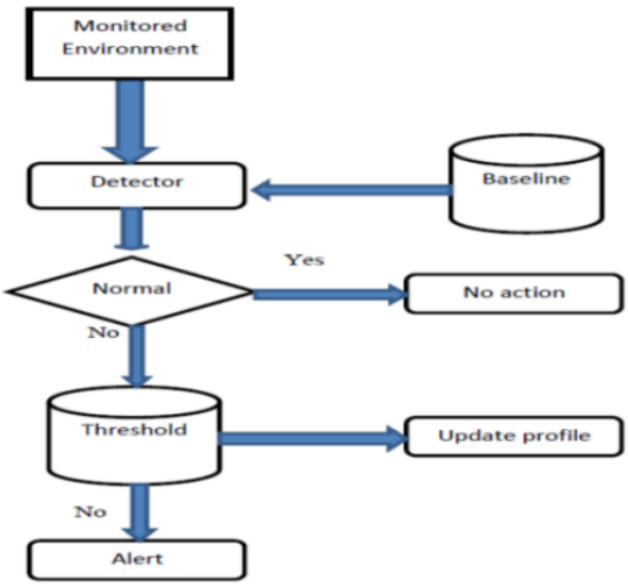



Figure I.7: Anomaly based methodology architecture [10]

## 5.3. Hybrid-Based detection

Hybrid IDS combines the advantages of signature-based and anomaly-based IDS techniques [9], creating a more comprehensive and robust defensive framework [8]. This approach utilizes signature-based detection for quickly and efficiently identifying known attacks, while also employing anomaly-based detection to detect new or unknown attacks that may evade signature-based detection.

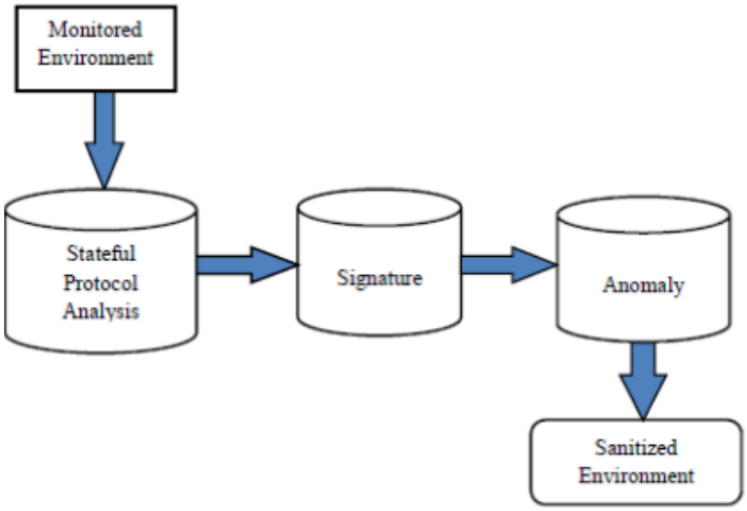



Figure I.8: Hybrid based methodology architecture [8].

# 6. IDS Efficiency

When designing, implementing, or assessing an IDS, it is crucial to consider these measures to ensure a robust and reliable security monitoring system [10].

- Exactitude (Accuracy): in the context of an Intrusion Detection System (IDS), pertains to the accuracy of the system in correctly identifying and classifying security events. It measures how well the IDS can distinguish between normal and malicious activities without generating a high number of false positives or false negatives [10].

- Performance: Performance, in the context of IDS, refers to the system's ability to effectively and efficiently handle a certain volume of network traffic or events. It evaluates how well the IDS scales with increasing load and the number of simultaneous connections while maintaining a responsive and timely detection and response capability [11].

- Fault Tolerance: Fault tolerance measures the resilience of an IDS against failures or faults. It assesses the system's ability to continue functioning and provides security monitoring even in the presence of hardware failures, software errors, or deliberate attacks aimed at disrupting its operation.

- Speed: in the context of IDS, refers to the responsiveness of the system. It measures how quickly the IDS can analyze network traffic, detect security events, and respond to potential threats. A fast IDS is crucial for the timely identification and mitigation of security incidents.

- Completeness: within the context of IDS, refers to the system's ability to detect a wide range of security events and threats. It evaluates how well the IDS covers different types of attacks and vulnerabilities, ensuring that there are minimal blind spots in the security monitoring process [11].

# 7. Limitations and vulnerabilities of IDS

While intrusion detection systems are valuable tools in cyber security, they also face certain challenges, such as:

- False positives and negatives: IDS can produce false alerts, flag legitimate activities as potential threats, or miss actual intrusions, demanding careful tuning and continuous improvement of detection rules.

- Evasion techniques: hackers use sophisticated methods to avoid IDS detection, like encryption and low-and-slow attack patterns, requiring IDS to keep up with evolving threats.

- Limited visibility*:* Network-based IDSs (NIDS) can only see traffic flowing through specific points, missing attacks within encrypted tunnels or switched networks. Host-based IDSs (HIDS) have better visibility but require installation on individual devices, increasing complexity.

- Resource limitations*:* IDSs require processing power and storage to analyze data. They can be overwhelmed by high-volume attacks or struggle to keep up with evolving threats.

- Zero-day exploits: signature-based IDS may miss unknown threats, highlighting the need for anomaly and behavior-based detection methods.

- Configuration errors*:* Improper configuration can render an IDS ineffective or even create security vulnerabilities. Skilled administration is crucial.

- Integration challenges: Integrating IDS with other security systems can be complex, requiring expertise and potentially creating compatibility issues.

- Response and remediation: Efficient response to IDS alerts requires skilled personnel, well-defined incident response processes, and integration with security workflows.

## 8. Advantages of using IDS

Intrusion detection system (IDS) monitoring offers a valuable set of benefits for network security. Here are some key advantages:

- Detecting Malicious Activity: Early detection is crucial in cyber security. An IDS can continuously monitor network traffic and system activity, identifying suspicious patterns or behaviors that might indicate an ongoing attack. This allows security teams to respond quickly and minimize potential damage.

- Improved Network Performance: By identifying bottlenecks, unusual traffic patterns, or resource-hogging activities, an IDS can help pinpoint performance issues within the network. This information is valuable for optimizing network performance and ensuring smooth operation.

- Compliance Adherence: Many organizations are subject to data privacy and security regulations. An IDS can help companies meet these compliance requirements by providing detailed reports on network activity and potential security incidents.

- Valuable Security Insights: An IDS continuously analyzes network data, providing insights into potential vulnerabilities, attack attempts, and overall security posture. This knowledge is essential for proactively strengthening defenses and making informed security decisions.

- Reduced Security Costs: Early detection and prevention of attacks can significantly reduce the cost of security incidents, including data loss, business downtime, and remediation efforts.

- Enhanced Threat Intelligence: By analyzing past attacks and identifying attack patterns, an IDS can contribute to an organization's overall threat intelligence, improving its ability to prepare for future threats.

- Improved Situational Awareness: Security teams gain a real-time view of network activity and potential threats, enabling them to make faster and more informed decisions during security incidents.

# 9. Enhancing Reliability in Intrusion Detection

Enhancing reliability in intrusion detection requires a comprehensive approach. Consider the following strategies to improve the dependability of your IDS:

- Regular Updates: Keep your intrusion detection system current with the latest security signatures and patches. This ensures it can effectively identify and defend against emerging threats.

- Network Segmentation: Divide your network into segments to minimize the impact of a breach and restrict the lateral movement of intruders.

- Machine Learning and AI: Implement machine learning and artificial intelligence algorithms to help your IDS learn and adapt to evolving threats. This can reduce false positives and increase accuracy.

- Continuous Monitoring: Maintain 24/7 monitoring to promptly detect and respond to any suspicious activities. Automated alerts and response mechanisms can aid in this process.

- Anomaly Detection: Use anomaly-based intrusion detection techniques to identify deviations from normal network traffic or system activities.

- User and Entity Behavior Analytics (UEBA): Analyze user and entity behavior patterns to detect insider threats and other anomalies that traditional signature-based IDS may overlook.

- Threat Intelligence Feeds: Integrate threat intelligence feeds into your IDS to stay informed about the latest threats and attack techniques.

- Security Information and Event Management (SIEM): Combine your IDS with an SIEM system to centralize logs and event data, enabling more comprehensive threat analysis and response.

- Staff Training: Ensure your security team is well-trained to effectively manage and respond to security alerts.

- Regular Testing: Conduct periodic assessments of your IDS's performance through penetration testing and simulated attack scenarios to identify any weaknesses.

# 10. Conclusion

In conclusion, Intrusion Detection Systems (IDS) are crucial in modern cybersecurity defenses, providing essential capabilities to protect sensitive data and maintain system integrity. IDS continuously monitor network traffic and system activities to detect and respond to potential threats in real-time, preventing security breaches.

There are two main types of IDS: Network-based (NIDS), which monitors network traffic, and Host-based (HIDS), which monitors individual devices. IDS functionalities include anomaly detection (identifying deviations from normal behavior), signature-based detection (matching known threat patterns), and behavior-based detection (observing unusual user or system actions).

Effective IDS implementation requires careful planning, proper configuration, and regular updates. Integrating IDS with other security measures, like firewalls and SIEM systems, enhances overall security. As cyber threats evolve, IDS remain a vital part of an organization's security strategy, ensuring continuous monitoring, detection, and response to threats.

# Chapter II

# Fire Hawk Optimizer (FHO)

# 1. Introduction

A key component of many scientific, engineering, and industrial applications is optimization, which seeks to solve complex problems by repeatedly investigating and assessing potential solutions. Modern optimization problems are becoming larger and more complicated, which has forced the development of complex algorithms that can navigate large, multidimensional search spaces with ease. Among these, nature-inspired optimization algorithms have garnered significant attention due to their ability to mimic the problem-solving strategies observed in natural systems.

The Fire Hawk Optimizer (FHO) algorithm is a recent addition to the family of nature-inspired optimization techniques. Inspired by the behavior of fire hawks, birds known for their unique hunting strategies involving fire manipulation, FHO presents a novel approach to solving complex optimization problems.

This chapter explores the operational principles and characteristics of the FHO algorithm, and delves into his specific mechanisms, detailing the processes of initialization, exploration, and exploitation that drive its search capabilities.

# 2. Definition

The Fire Hawk Optimizer (FHO) is a novel metaheuristic algorithm inspired by the foraging behavior of whistling kites, black kites, and brown falcons, which are birds that catch prey in nature by setting fires, hence their apt name "Fire Hawks”. Fire hawks are observed to deliberately spread fire by carrying burning sticks to new areas, thereby driving prey into regions where they can be easily captured. This distinctive behavior forms the conceptual foundation of the FHO algorithm, where controlled perturbations and strategic movements emulate the fire hawks' tactics to explore and exploit the search space effectively

In FHO, the optimization process involves setting fires (representing potential solutions) and evaluating the safety of different locations under or outside the fire (representing the quality of solutions). The algorithm updates its solutions based on the positions of prey (representing better solutions) and safe places under/outside the fire (representing global optimum solutions). This approach helps the algorithm avoid getting trapped in local optima and find better solutions.

The FHO algorithm has shown promising performance in numerical investigations, where it was compared against many benchmark functions and other state-of-the-art metaheuristic algorithms. It has also been evaluated on real-world optimization problems and has demonstrated competitive performance. The algorithm is parameter-free, converges quickly, and requires fewer objective function evaluations compared to other algorithms.

# 3. Fire Hawk optimizer (FHO) algorithm

This section includes illustrations of the FHO's inspiration notion, the mathematical model and the flowchart of this metaheuristic algorithm.

## 3.1. Inspiration

Birds like whistling kites, black kites, and brown falcons, known as Fire Hawks, are known to spread fires intentionally. They carry burning sticks in their beaks and talons, dropping them in unburned areas to start small fires. This behavior helps them capture prey, such as rodents and snakes, which become easier targets as they flee in a hurried and nervous manner. (see FigureII.1).



Figure II.1: Photographs of Fire Hawks' behavior around fires [12].

## 3.2. Mathematical model

The FHO metaheuristic algorithm mimics the foraging behavior of fire hawks, considering the process of setting and spreading fires and catching prey [12]. At first, a number of solution candidates (*X)* are determined as the position vectors of the fire hawks and prey. A random initialization process is utilized to identify the initial positions of these vectors in the search space. As in Formula (2,1):

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} X_1^1 & X_1^2 & \cdots & X_1^j & \cdots & X_1^d \\ X_2^1 & X_2^2 & \cdots & X_2^j & \cdots & X_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_i^1 & X_i^2 & \cdots & X_i^j & \cdots & X_i^d \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_N^1 & X_N^2 & \cdots & X_N^j & \cdots & X_N^d \end{bmatrix} \begin{cases} i = 1,2,\dots,N. \\ j = 1,2,\dots,d. \end{cases} \quad (2,1)$$

$$X_i^j(0) = X_{i,min}^j + rand.\left(X_{i,max}^j - X_{i,min}^j\right), \begin{cases} i = 1,2,\dots,N. \\ j = 1,2,\dots,d. \end{cases} \quad (2,2)$$

In the given formula (2,2), $X_i$ represents the $i^{th}$ solution candidate in the search space, where $d$ is the dimension of the considered problem $N$ is the total number of solution candidates in the search space $X_i^j$ is the $j^{th}$ decision variable of the $i^{th}$ solution candidate $X_i^j(0)$ represents the initial position of the solution candidates $X_{i,min}^j$ and $X_{i,max}^j$ are the minimum and maximum bounds of the $j^{th}$ decision variable for the $i^{th}$ solution candidate, and *rand* is a uniformly distributed random number in the range of [0,1].

To determine the locations of the Fire Hawks in the search space, the objective function is evaluated for the solution candidates based on the specific optimization problem. Solution candidates with better objective function values are designated as Fire Hawks, while the remaining candidates are considered prior. The Fire Hawks are responsible for spreading fires around the prey in the search space to facilitate hunting. Initially, the global best solution is regarded as the main fire that the Fire Hawks use to spread fires throughout the search space. This process is illustrated schematically in Figure II.2 and can be expressed mathematically as follows in the formula (2,3), (2,4):

$$PR = \begin{bmatrix} PR_1 \\ PR_2 \\ \vdots \\ PR_k \\ \vdots \\ PR_m \end{bmatrix}, k = 1, 2, \dots, m. \quad (2,3)$$

$$FH = \begin{bmatrix} FH_1 \\ FH_2 \\ \vdots \\ FH_I \\ \vdots \\ FH_n \end{bmatrix}, I = 1, 2, \dots, n. \quad (2,4)$$

where $\boldsymbol{PR}_K$ is the kth prey in the search space regarding the total number of m preys and $\boldsymbol{FH}_1$ is the lth fire hawk considering a total number of n fire hawks in the search space [12].

In the subsequent step of the algorithm, the total distance between the Fire Hawks and the prey is computed to establish the proximity of each bird to its nearest prey, thereby defining their effective territories. The closest prey to the first Fire Hawk with the optimal objective function value is identified, while the territories of the other birds are defined based on the remaining prey. This process is depicted in Figure II.3. The distance $\boldsymbol{D}_k^l$ is calculated using the equation (2,5):

$$D_k^l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \quad \begin{cases} l = 1,2,\dots,n. \\ k = 1,2,\dots,m. \end{cases} \quad (2,5)$$

- $\boldsymbol{D}_k^l$represents the total distance between the lth fire hawk and the kth prey.
- m is the total number of preys in the search space
- n is the total number of fire hawks in the search space.
- $(x_1, x_2)$, $(y_1, y_2)$ represent the coordinates of the Fire Hawks and prey in the search space.

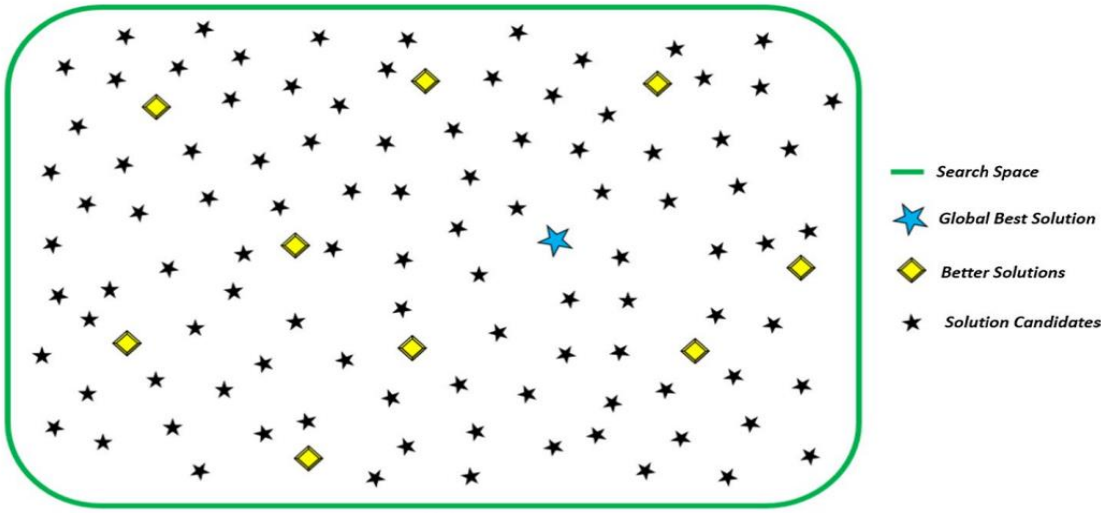



Figure II.2: Schematic presentation of determining fire hawks and prey in the search space [12].

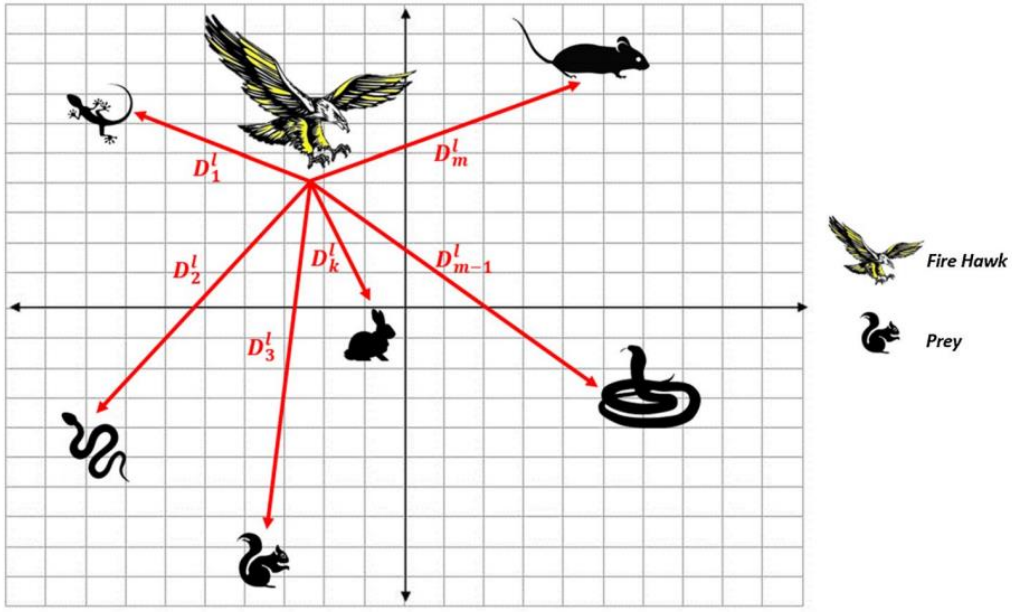



Figure II.3: Schematic presentation for measuring the total distance between the Fire Hawks and the prey [12].

Following the calculation of distances between Fire Hawks and their prey, their territories are established based on the proximity to prey. Each Fire Hawk selects the closest prey within its territory, with stronger Fire Hawks demonstrating more successful hunting compared to weaker ones. Subsequently, the algorithm gathers burning sticks from the main fire and strategically sets fires within their territories to induce prey to flee. This process serves as a method for updating positions within the algorithm's primary search loop [12]. using the equation (2,6):

$$FH_1^{new} = FH_1 + (r_1 \times GB - r_2 \times FH_{Near}), \quad l = 1,2,\dots,n. \tag{2,6}$$

- $\boldsymbol{FH}_1^{new}$ is the new position vector of the lth Fire Hawk $\boldsymbol{FH}_1$
- GB is the global best solution in the search space, considered as the main fire.
- $\boldsymbol{FH}_{Near}$ is one of the other Fire Hawks in the search space
- $r_1$ and $r_2$ are uniformly distributed random numbers in the range of [0, 1] used to determine the movements of Fire Hawks.

In the subsequent stage of the algorithm, the behavior of prey within each Fire Hawk's territory holds significant importance. When a Fire Hawk releases a burning stick, prey may respond by hiding, fleeing, or inadvertently moving closer to the Fire Hawk. These behaviors are integrated into the position updating process through the equation (2,7):

$$PR_q^{new} = PR_q + (r_3 \times FH_1 - r_4 \times SP_1), \quad \begin{cases} l = 1,2,\dots,n. \\ q = 1,2,\dots,r. \end{cases} \tag{2,7}$$

- $\boldsymbol{PR}_q^{new}$ is the new position vector of the $q^{th}$ prey surrounded by the lth Fire Hawk $\boldsymbol{FH}_1$.
- $\boldsymbol{SP}_1$ is a safe place under the lth Fire Hawk territory.
- $r_3$ and $r_4$ are uniformly distributed random numbers in the range of [0, 1] used to determine the movements of prey toward the Fire Hawks and the safe place.

Moreover, the prey has the option to approach the territories of other Fire Hawks, potentially moving closer for an ambush, or it may seek safety outside the Fire Hawk's territory. These behaviors can be taken into account during the position updating process using the equation (2,8):

$$PR_q^{new} = PR_q + (r_5 \times FH_{Alter} - r_6 \times SP), \begin{cases} l = 1,2,\dots,n. \\ q = 1,2,\dots,r. \end{cases} \tag{2,8}$$

- $\boldsymbol{FH}_{Alert}$ represents one of the other Fire Hawks in the search space
- SP denotes a safe place outside the $i^{th}$ Fire Hawk's territory Top of Form
- $r_5$ and $r_6$ are uniformly distributed random numbers

Animals tend to gather in safe areas to protect themselves from hazards. This behavior can be expressed mathematically for both safe places inside the territory (2,9). and outside the territory (2,10) [12]:

$$SP_1 = \frac{\sum_{q=1}^{r} PR_q}{r}, \quad \begin{cases} l = 1,2,\dots,n. \\ q = 1,2,\dots,r. \end{cases} \tag{2,9}$$

$$SP = \frac{\sum_{k=1}^{m} PR_k}{m}, \quad \text{k} = 1,2,\dots,\text{m}. \tag{2,10}$$

## 3.3. FHO algorithm

The FHO algorithm incorporates several key aspects, including handling boundary violations of solution candidates and defining a termination criterion. A mathematical flag is used to control boundary violations of decision variables, ensuring that solutions remain within predefined limits. Additionally, a predetermined number of objective function evaluations or iterations can be used as a termination criterion. The pseudo-code of the FHO algorithm is shown in Figure II.4.

```
procedure Fire Hawk Optimizer (FHO)
    Determine initial positions of solution candidates (Xi) in the search space with N candidates
    Evaluate fitness values for initial solution candidates
    Determine the Global Best (GB) solution as the main fire
    while Iteration < Maximum number of iterations
        Generate n as a random integer number for determining the number of Fire Hawks
        Determine Fire Hawks (FH) and Preys (PR) in the search space
        Calculate the total distance between the Fire Hawks and the preys
        Determine the territory of the Fire Hawks by dispersing the preys
        for l=1: n
            Determine the new position of the Fire Hawks by Eq. 6.
            for q=1:r
                Calculate the safe place under lth Fire Hawk territory by Eq. 9.
                Determine the new position of the preys by Eq. 7.
                Calculate the safe place outside the lth Fire Hawk territory by Eq. 10.
                Determine the new position of the preys by Eq. 8.
            end
        end
        Evaluate fitness values for the newly created Fire Hawks and preys
        Determine the Global Best (GB) solution as the main fire
    end while
    return GB
end procedure
```

Figure II.4: Pseudo-code of FHO [12].

## 3.4. Fire Hawk Optimizer (FHO) Algorithm Flowchart

The flowchart in Figure II. 5 demonstrates the iterative nature of the FHO algorithm, where solution candidates continuously adjust their positions based on the foraging behavior, aiming to improve the population's overall fitness. The algorithm terminates when a certain condition is met, such as reaching a maximum number of iterations, and outputs the best solution found during the optimization process.

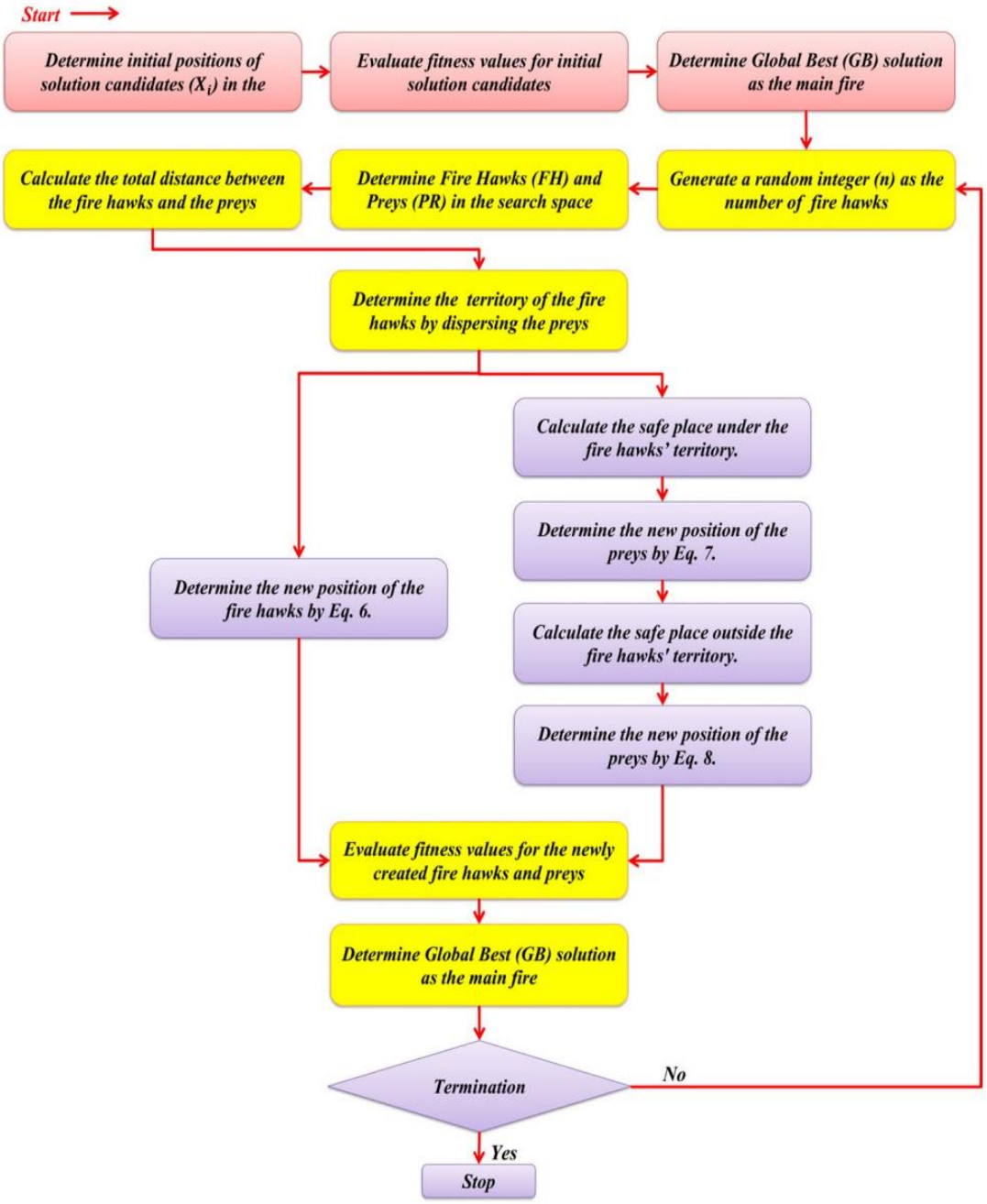



Figure II.5: Flowchart of FHO [12].

# 4. Previous Work on Fire Hawk Optimizer

The Fire Hawk Optimizer (FHO) is a relatively new nature-inspired optimization algorithm that draws its inspiration from the hunting strategies of fire hawks. These birds of prey are known for their ability to detect and exploit fires to flush out prey, leading to the development of an algorithm that mimics this intelligent behavior. Below is a summary of works utilizing FHO across various fields.

## 1. Feature Selection for Intrusion Detection Systems:

Several studies have explored the application of FHO in feature selection for intrusion detection systems (IDS). By selecting the most relevant features from network traffic datasets, these works aim to enhance the detection accuracy and efficiency of IDS. FHO has demonstrated its effectiveness in optimizing feature subsets, leading to improved performance metrics in classification tasks.

## 2. Optimization in Machine Learning Models:

FHO has been employed in optimizing the parameters of various machine learning models. For instance, it has been used to fine-tune hyper parameters in models like Support Vector Machines (SVM) and Neural Networks, resulting in better predictive performance and faster convergence rates.

## 3. Engineering and Industrial Applications:

The algorithm has also found applications in engineering optimization problems. These include optimizing design parameters in mechanical systems, improving energy efficiency in industrial processes, and solving complex scheduling problems. The adaptability and robustness of FHO make it suitable for a wide range of optimization tasks.

## 4. Image Processing and Computer Vision:

In the field of image processing, FHO has been utilized for tasks such as image segmentation and feature extraction. The algorithm's ability to handle large search spaces and find optimal solutions makes it a valuable tool in enhancing image analysis and computer vision applications.

## 5. Bioinformatics and Computational Biology:

Researchers have applied FHO to problems in bioinformatics, such as gene selection and protein structure prediction. By efficiently navigating the complex search spaces associated with biological data, FHO contributes to more accurate and insightful analyzes in these domains.

# 5. Conclusion

In this chapter has provided a comprehensive exploration of the Fire Hawk Optimizer (FHO) algorithm, a novel and powerful addition to the suite of nature-inspired optimization techniques. By emulating the foraging strategies of Fire Hawks, the FHO algorithm has demonstrated exceptional performance across a variety of mathematical test functions, highlighting its efficiency and effectiveness in navigating complex search spaces.

The potential of the FHO algorithm extends beyond mere optimization tasks; its application in various domains can lead to significant improvements in solution quality and computational efficiency. As we transition to the next chapters, our focus will shift towards utilizing the FHO algorithm for feature selection in intrusion detection systems. Feature selection is a critical component in enhancing the performance of such systems, and the superior capabilities of the FHO algorithm make it an ideal candidate for this task.

By applying the FHO algorithm to the feature selection problem, we aim to identify the most relevant features that contribute to accurate and efficient intrusion detection. This endeavor will not only showcase the practical applicability of the FHO algorithm but also its potential to make a substantial impact in the field of cybersecurity.

# Chapter III

Deep Learning (DL)

# 1. Introduction

Artificial intelligence (AI) has been transformed by deep learning, a subset of machine learning, which allows machines to learn and make detection and prediction of large and complex data such as videos, images, and texts, at a degree of efficiency and accuracy that was before unattainable. Rooted in the structure and function of the human brain, deep learning leverages complex neural networks to model and solve intricate problems across various domains. The rapid advancements in computational power, coupled with the availability of large datasets, have fueled the growth of deep learning, making it a cornerstone of modern AI research and applications.

This chapter aims to provide a comprehensive overview of deep learning, tracing its evolution from its inception to its current state-of-the-art applications. We will explore the fundamental concepts and principles that underpin deep learning, including the architecture and functioning of neural networks. Key components such as neurons, activation functions, optimization algorithms and diverse architectures that have emerged, will be discussed in detail to lay a solid foundation for understanding how deep learning models operate.

# 2. Definition

Deep learning is a subfield of machine learning that utilizes artificial neural networks with multiple layers to process and learn from complex data representations. These deep neural networks are capable of autonomously extracting features from raw data, eliminating the need for manual feature engineering. Through a process called backpropagation, the models adjust their internal parameters iteratively to minimize prediction errors and improve their performance over time. Deep learning has achieved remarkable success in tasks such as image recognition, speech recognition, natural language processing, and more, thanks to its ability to handle large datasets, capture intricate patterns, and generalize well to unseen examples [13].

# 3. Deep Learning Classification

As discussed earlier, deep learning encompasses a broad range of machine learning techniques and architectures, characterized by the use of multiple layers of hierarchical non-linear information processing. Based on their intended use, such as synthesis/generation or recognition/classification, most work in this area can be categorized into three major classes [14]:

## 3.1. Deep Networks for Unsupervised or Generative Learning

These networks aim to capture high-order correlations in the observed or visible data for pattern analysis or synthesis when no information about target class labels is available. This category, known as unsupervised feature or representation learning, seeks to model the joint statistical distributions of the visible data and their associated classes (if available) as part of the visible data. In cases where generative models are used, applying Bayes' rule can transform these generative networks into discriminative ones for learning purposes.

## 3.2. Deep Networks for Supervised Learning

These networks are designed to provide discriminative power for pattern classification by modeling the posterior distributions of classes conditioned on the visible data. Target label data are always available, either directly or indirectly, for supervised learning. These networks are also referred to as discriminative deep networks.

## 3.3. Hybrid Deep Networks

Hybrid deep networks combine different types of neural network architectures or components to leverage their respective strengths. By integrating deep learning architectures with other models, such as traditional machine learning algorithms or rule-based systems, hybrid networks can achieve enhanced performance across various tasks. These networks are utilized in areas that require diverse information processing, such as natural language understanding, computer vision, and reinforcement learning.

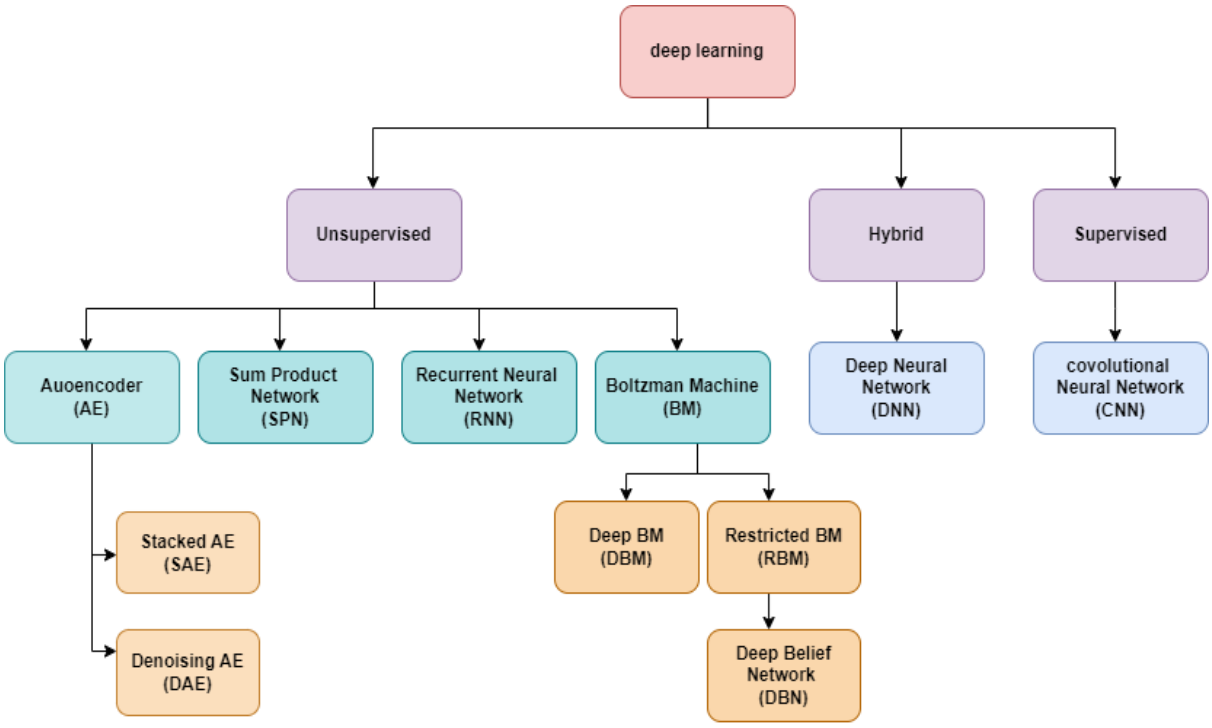



Figure III.1: Classification of deep learning techniques [14].

# 4. Some methods of deep learning

Deep learning algorithms work with almost any kind of data and require large amounts of computing power and information to solve complicated issues [15]. Now let us deep dive into some deep learning algorithms.

## 4.1. Artificial neural network

Artificial neural networks (ANN) are computational models inspired by the structure andfunctioning of the human brain. They are capable of learning from data to perform various tasks, such as classification, regression, and pattern recognition. ANN consist of interconnected nodes, called neurons, which are organized in layers.

Each neuron receives input signals, processes them using an activation function, and passes the output to the next layer of neurons through weighted connections. These weights are adjusted during the learning process to improve the network's performance. ANNs are used in a wide range of applications, including detect malicious patterns in network traffic [16].

### 4.1.1. Formal Neuron (Perceptron)

An artificial neuron, often called a node or perceptron, is a mathematical function used to transform raw data into interpretable information. It receives inputs from preceding nodes or external sources and produces an output. Each input is assigned a weight, $w_i$ which indicates its relative significance compared to other inputs. The node then applies a function f to the weighted sum of its inputs as shown in the following formula [17]:

$$f(output) = f(x) + f(\sum_{i=1}^{n} w_i\, x_i + z\ biais \tag{3.1}$$

The figure presented below provides an example of an artificial neuron that receives numerical inputs ($x1, x2, \ldots, xn$) with corresponding weights ($w1, w2, \ldots, wn$). The bias term isutilized to fine-tune the output.

The calculated output (output) can be determined using equation (3.1).

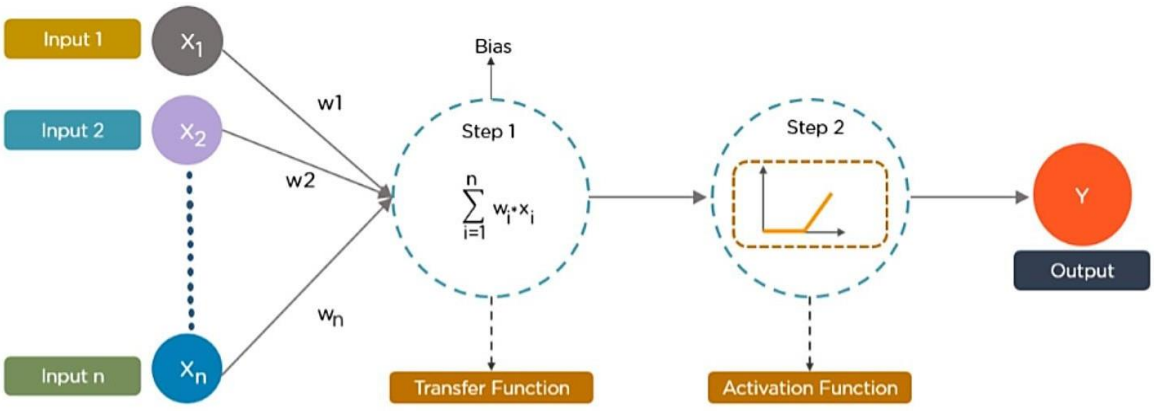



Figure III.2: Example of a simple perceptron [17].

### 4.1.2. The different layers of a neural network

The components of a deep neural network are the following:

- **Input layer***:* An artificial neural network has several nodes that input data into it. These nodes make up the input layer of the system [18].

- **Hidden layers:** The input layer processes and passes the data to layers further in the neural network. These hidden layers process information at different levels, adapting theirbehavior as they receive new information. Deep learning networks can have a lot of hidden layers that they can use to analyze a problem from several different angles.

  For example, if you were given an image of an unknown animal that you had to classify, you would compare it with animals you already know. For example, you would look at the shape of its eyes and ears, its size, the number of legs, and its fur pattern. You would try to identify patterns, such as the following:

  ✓ The animal has hooves, so it could be a cow or deer.
  ✓ The animal has cat eyes, so it could be some type of wild cat.

  The hidden layers in deep neural networks work in the same way. If a deep learning algorithm is trying to classify an animal image, each of its hidden layers processes a different feature of the animal and tries to accurately categorize it [18].

- **Output layer*:*** The output layer consists of the nodes that output the data. Deep learning models that output "yes" or "no" answers have only two nodes in the output layer. On theother hand, those that output a wider range of answers have more nodes [18].

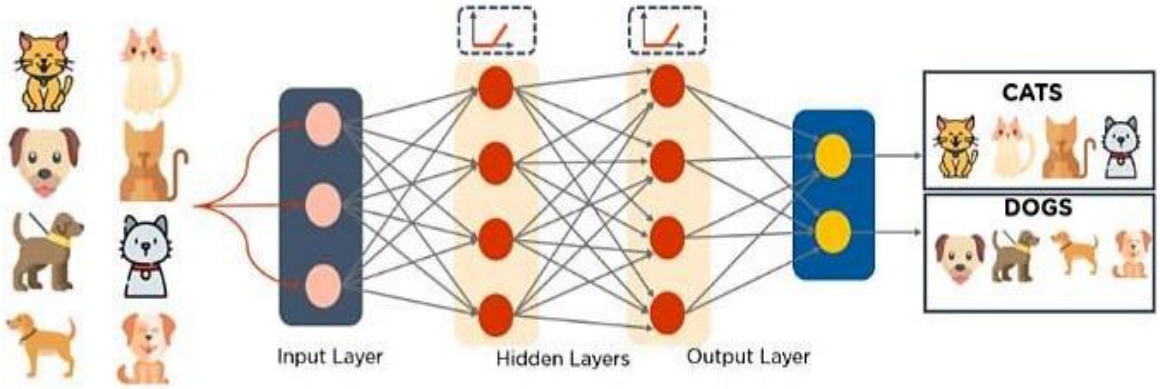



Figure III.3: Example of fully connected network with four layers(Input,

Hidden and Output) [18].

### 4.1.3. Activation function

An activation function serves to introduce non-linearity into a neuron's output, mimicking biological processes. Neurons in each layer apply this function [23]. The following are the most commonly used functions.

- **The SoftMax function**: is a mathematical operation that transforms a real-valued vector into a probability vector. It is frequently employed in the final layer of a model to convert a score into a probability distribution, as determined by the following formula:

$$softmax(x_i) = \frac{e^{xi}}{\sum_{j=1}^{n} e^{xj}} \quad (3.2)$$

- **The ReLu function**: The ReLU function stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero, as given by the following formula:

$$f(x) = Max(x, 0) \quad (3.3)$$

- **The Sigmoid function**: is a mathematical function that maps input values to a rangebetween 0 and 1, according to the following formula:

$$f(x) = \frac{1}{1+e^{-x}} \quad (3.4)$$

- **The Tanh function**: is a mathematical operation representing the hyperbolic tangentfunction. It's a shifted version of the sigmoid function, producing output values between -1 and 1 according to the formula:

$$f(x) = \frac{e^{x} + e^{-x}}{e^{x} + e^{-x}} \quad (3.5)$$

### 4.1.4. The learning rate

The learning rate is a hyper parameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.

The learning rate may be the most important hyper parameter when configuring your neural network. Therefore, it is vital to know how to investigate the effects of the learning rate on model performance and to build an intuition about the dynamics of the learning rate on model behavior [17].

### 4.1.5. Loss function

A loss function quantifies how well a model is performing by measuring the difference between the predicted values and the actual values in the training data. The goal of optimization is to minimize this difference, or loss, which leads to a better-performing model. Different types of models and tasks may require different loss functions, and choosing the right one is crucial for training an effective model, such as cross-entropy loss for classification tasks [17].

- **The categorical cross-entropy function:** is a loss function used in multi-class classification tasks. It measures the difference between two probability distributions [24]. It is computed using the following formula:

$$L_{CE} = \sum_{i=0}^{c} y_i \, Log(s_i) \tag{3.6}$$

Where $y_i$ is the predicted value, $S_i$ is the result of the SoftMax function for the $i^{th}$ class and c is the number of classes.

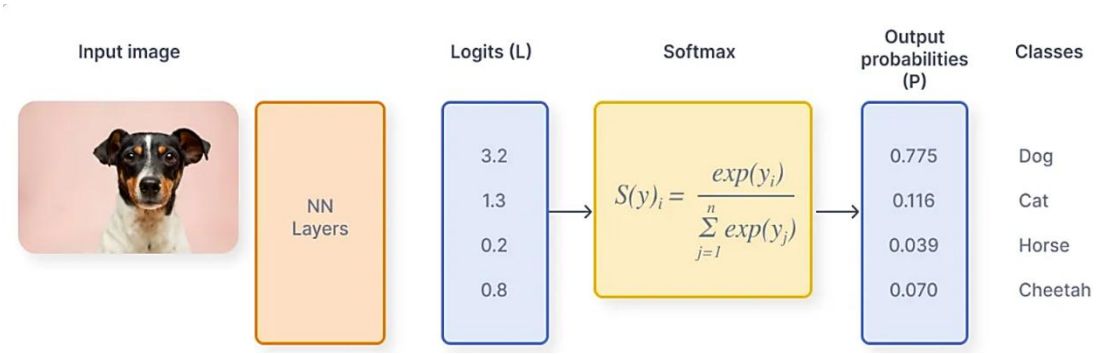



Figure III.4: Image classification using cross-entropy loss [17].

### 4.1.6. Batch size

The batch size is the number of training examples used in each iteration of the training process. Larger batch sizes can lead to faster training times and more stable convergence, but they can also reduce the model's ability to generalize to new data. Smaller batch sizes, on the other hand, can sometimes improve generalization but may require more iterations to converge and can be less computationally efficient. The choice of batch size often depends on the specific dataset and model being used [25].

### 4.1.7. Propagation

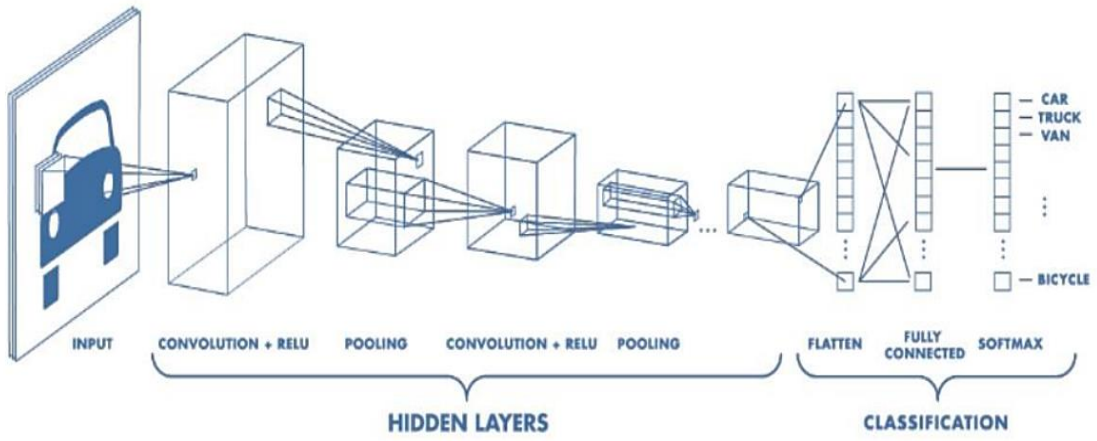
This is a stage in the neural network learning process consisting of two steps:

- **Forward propagation:** also known as the feed forward phase, is the process of computing and storing intermediate variables (including outputs) of a neural network in order from the input layer to the output layer [26].

- **Backward propagation:** also known as back propagation, is a step in the neural network learning process that involves calculating the gradient of the loss function with respect to the network's weights, using the chain rule of calculus. This gradient is then used to update the weights of the network through an optimization algorithm, such as gradient descent [26].

## 4.2. Convolutional Neural Networks (CNNs)

A Convolutional Neural Network, also known as a CNN, is a well-known and widely used type of neural network. It utilizes a specialized mathematical operation called convolution to analyze input data. This operation is preferred over matrix multiplication due to its shift-invariant benefits and low computational costs, making it highly efficient in processing pixel data. As a result, CNNs are commonly utilized in applications related to images [19,20]. or videos [21,22].




FigureIII.5 The structure of a classic CNN model [19].

Figure III.5 depicts the arrangement of a CNN that features three unique layers: convolutional, pooling, and fully-connected. The purpose of each layer is distinct and crucial for the successful operation of the network.

### a. Convolution layers

First of all, the fundamental layer in a CNN is the convolutional layer, responsible for convolving the input data to generate feature maps. The process involves transforming an input tensor, typically an image, of size $N \times C\text{in} \times H\text{in} \times W\text{in}$ to an output feature map of size $N \times C\text{out} \times H\text{out} \times W\text{out}$ via a convolution operation. The conversion equation for this process can be expressed as:

$$out(N^i, C_{out}^j) = bias(C_{out}^j) + \sum_{k=0}^{c_{in}-1} weight(C_{out}^j, k) \star input(N^i, k) \qquad (3.7)$$

The 2D convolution operation is represented by the symbol ⋆, where $N$ denotes the batch size, $C$ represents the number of channels, and $W$and $H$ respectively denote the widths and heights of the vectors. The convolution process can be better understood through a visualization, which is provided in Figure III.6

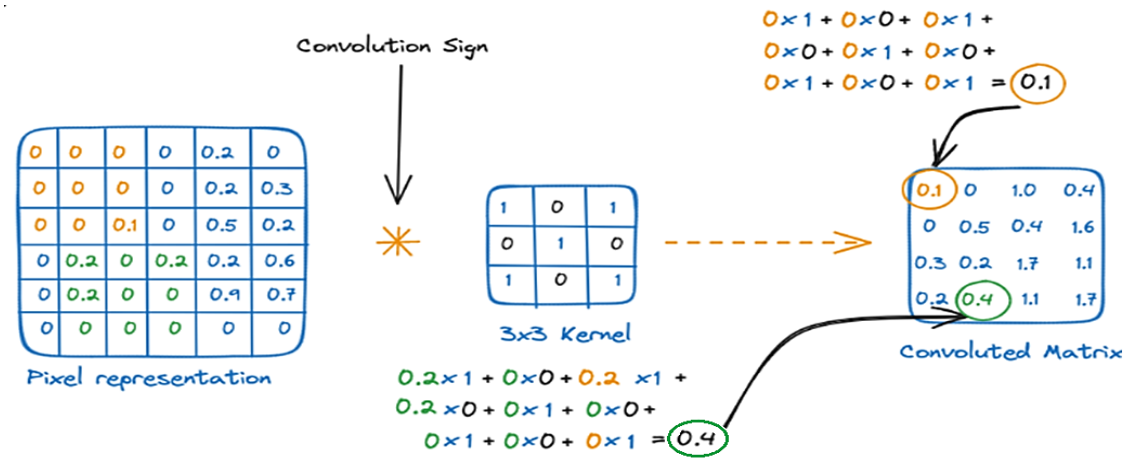



Figure III.6 Application of the convolution process [20].

**b. Pooling layer**

Secondly to compress input data, pooling layers are frequently utilized in conjunction with convolutional layers by combining a group of neurons into a single neuron. Max pooling and average pooling are the two most commonly used methods for integrating neurons in pooling layers. Max pooling selects the highest value from the cluster of neurons, while average pooling computes the average value of the neurons. To better understand the process, please refer to Figure III.7 below.

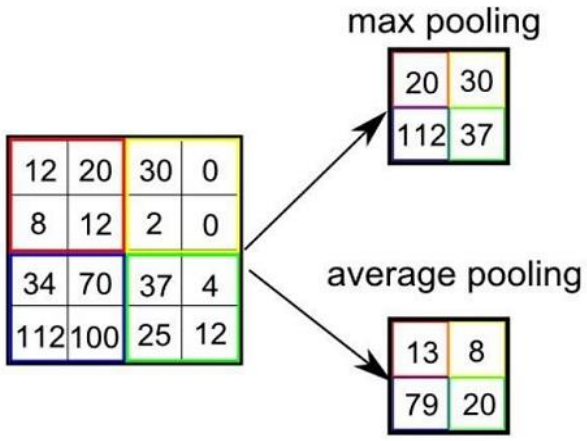



Figure III.7: The process of different pooling layer [19].

### c. Fully connected layers

Lastly, the fully connected layers are essentially similar to the classic multilayer perceptron (MLP) [21]. This layer type consists of neurons in different layers that are all interconnected. The computation within a fully connected layer can be formulated as follows (13):

$$y = \phi\,(w_x + b) \tag{3.8}$$

Here, ϕ denotes the activation function, x is the input vector, y is the output vector, b is the bias, and W is the weight vector. In CNNs, fully connected layers are typically used to project flattened feature vectors from feature space into the number of possible categories for recognition and classification.

## 4.3. Recurrent Neural Networks (RNN)

Recurrent neural networks (RNN) stand as stalwarts in the realm of deep learning, esteemed for their adeptness in handling sequential data. Manifesting a cyclical essence, an RNN orchestrates a symphony of operations across different time steps. With each iteration, it gracefully melds input signals with its internal state, perpetually refining itself in a dance of adaptation. This intrinsic architecture fosters an environment where sequential data finds sanctuary, nestling into the recesses of memory for subsequent recall and utilization [27].

However, RNNs have been demonstrated to be inadequate in maintaining information from a distant past, making them susceptible to long-term dependency issues [28]. As a result, researchers often employ a variant of RNNs known as long short-term memory (LSTM) in their investigations.

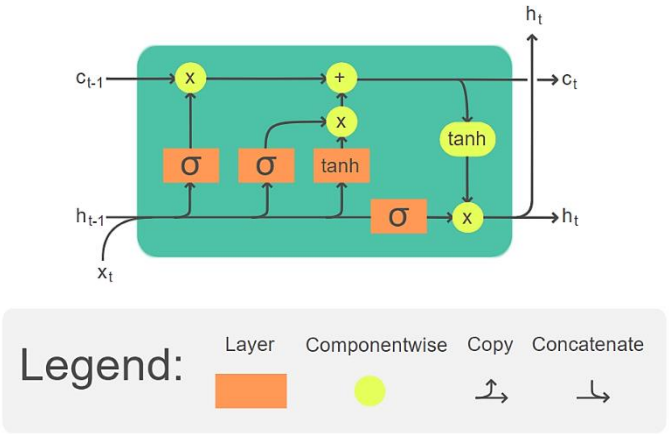



Figure III.8: The structure of a LSTM unit [28].

### 4.3.1. Long Short-Term Memory Networks (LSTMs)

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture designed to handle and model sequences of data, particularly those with long-range dependencies. Unlike traditional RNNs, which struggle with retaining information over long sequences due to issues like vanishing and exploding gradients, LSTMs are specifically designed to address these challenges [29]. Key features of LSTM include:

**1. Memory Cells:** LSTMs have memory cells that can maintain information over long periods, which is essential for learning dependencies in time-series data.

**2. Gates:** LSTM units contain three types of gates that regulate the flow of information:

- Input Gate: Controls how much of the new input should be added to the memory cell.
- Forget Gate: Decides how much of the past information should be forgotten.
- Output Gate: Determines how much of the information in the memory cell should be outputted.

**3. Self-Connections:** The memory cells have self-connections that help preserve the gradient, making it easier to train the network on sequences with long-term dependencies.

These features make LSTMs highly effective for various applications, including natural language processing, speech recognition, time-series forecasting, and any other tasks.

## 4.4. Autoencoders

The AE, or autoencoder, is named for its ability to internally represent data in a lower-dimensional space compared to the input data. A crucial feature of the autoencoder is that its hidden layer has a lower dimensionality than both the input and output layers. This property enables the autoencoder to perform dimensionality reduction on the data, similar to the fundamental principles of principal component analysis (PCA) [30].

Like other deep learning models, the learning method of an autoencoder utilizes a function represented by $f$. The input to this function is the original data $q_i$, and the output is $f(q_i)$ for $i = 1,\ldots, I$. The neural network in an autoencoder is trained using a loss function, which quantifies the difference between the input data and the reconstructed output. The objective is to minimize this loss function during training.

$$Loss = \sum_{i=1}^{I} \| q_i - f(q_i) \|_2 \tag{3.9}$$

Deep Autoencoder architecture consists of an encoder and a decoder. The encoder compresses the input data into a lower-dimensional latent representation, while the decoder reconstructs the original input data from this representation. Together, they enable tasks such as dimensionality reduction, data compression, denoising, and data generation [31,32].

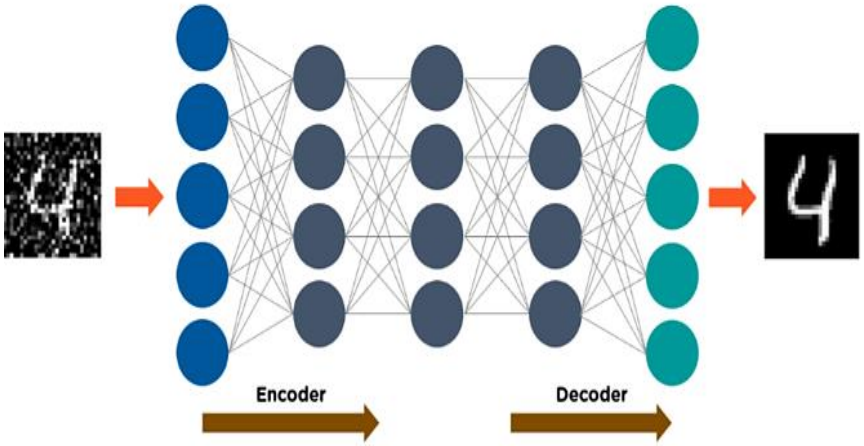



Figure III.9: Deep Autoencoder Architecture [31].

## 4.5. Deep Belief Networks (DBNs)

Deep Belief Networks (DBNs) are generative models composed of multiple layers of stochastic latent variables. These latent variables typically take binary values and are commonly referred to as hidden units. DBNs can be conceptualized as a series of Boltzmann Machines stacked on top of each other, with inter-layer connections. Each Restricted Boltzmann Machine (RBM) layer interacts with both the preceding and succeeding layers. DBNs find applications in tasks such as image recognition, video recognition, and analyzing motion-capture data [33].

DBNs are trained using greedy learning algorithms, employing a layer-by-layer approach to learn the top-down generative weights. The learning process involves running Gibbs sampling on the top two hidden layers of the DBNs. This step entails drawing a sample from the Restricted Boltzmann Machine (RBM) defined by these top two hidden layers [34]. Subsequently, DBNs draw a sample from the visible units by performing ancestral sampling through the remaining layers of the model. Through this training process, DBNs learn that the values of latent variables in each layer can be inferred by a single bottom-up pass.

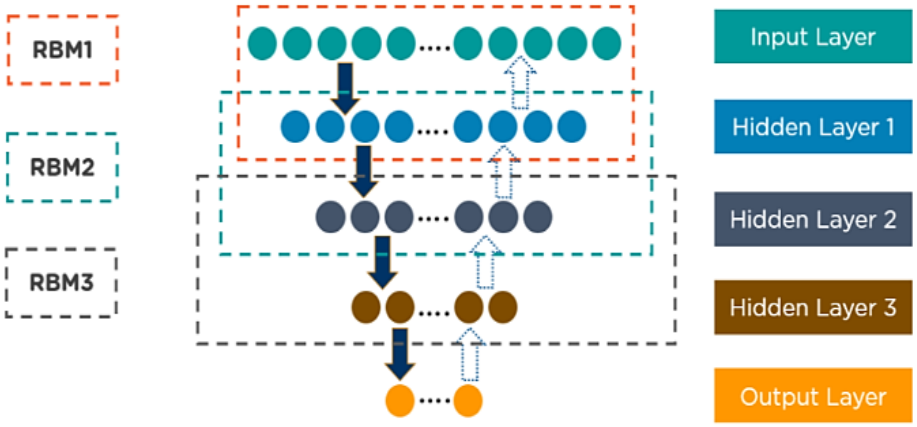



Figure III.10: The architecture of a DBN (Deep Belief Network) [34].

## 5. Evaluation metrics of models

Evaluation metrics are essential for assessing the performance of machine learning models. They help us understand how well a model is performing, compare different models, and improve their performance.

But before diving into various evaluation metrics, it's important to understand some basic terms that are commonly used in assessing the performance of classification models. These terms are True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) [35]:

- True Positives (TP): The number of instances that are actually positive and are predicted correctly as positive by the model.

- True Negatives (TN): The number of instances that are actually negative and are predicted correctly as negative by the model.

- False Positives (FP): The number of instances that are actually negative but are predicted incorrectly as positive by the model.

- False Negatives (FN): The number of instances that are actually positive but are predicted incorrectly as negative by the model.

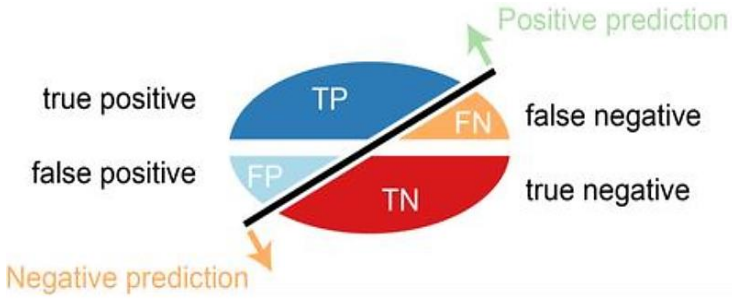



Figure III.11: Elliptical representation of four binary results of the test set classification [35].

From the most common evaluation metrics, we can identify:

- **Accuracy**: is calculated as the sum of two accurate predictions (TP + TN) divided by the total number of data sets (P + N). The best accuracy is 1.0, and the worst is 0.00.

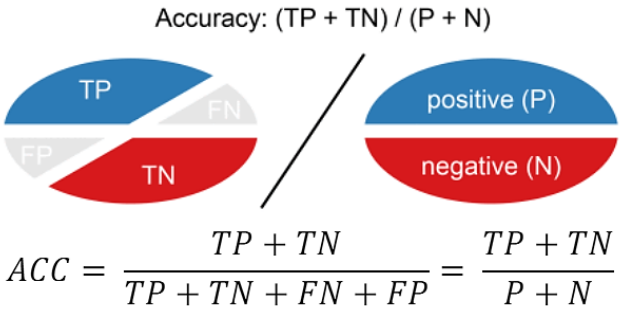



Figure III.12: Two ellipses show how accuracy is calculated [35].

- **Recall (Sensitivity):** is calculated as the number of accurate positive predictions (TP) divided by the total number of positive (P). Also called Sensitivity or Recall (REC). The best TP Rate is 1.0 and the worst 0.0.

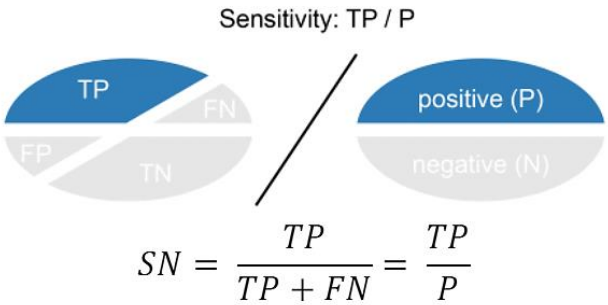



Figure III.13: Two ellipses show how the sensitivity is calculated [35].

- **Precision**:is calculated as the number of correct positive predictions (TP), divided by the total number of positive predictions (TP + FP). The best accuracy is 1.0 and the worst 0.0.

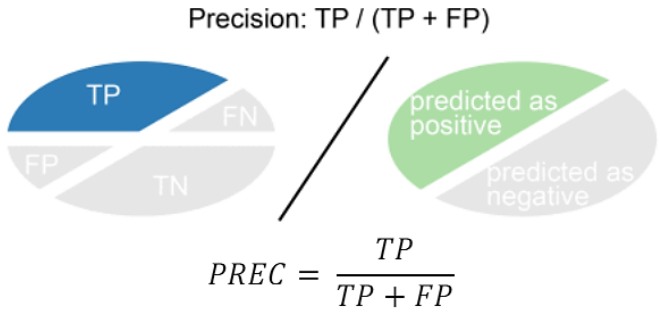



Figure III.14: Two ellipses show how precision is calculated [35].

- **F-Measure:** or F-score is a measure of the accuracy of the test. It is calculated, based on precision and reminders, by the formula:

$$F\ score = \frac{2 \times precision \times recall}{precision \times recall} \tag{3.10}$$

- **Confusion matrix**: Represents the correctly classified *TP* values, *FP* values in the relevant class while it should be in another class, and *FN* values in another class while it should be in the relevant class and the correctly classified *TN* values in the other class. The most frequently used performance metrics for classification according to these values are accuracy (*ACC*), precision (*P*), sensitivity, and *F*-score values [35].

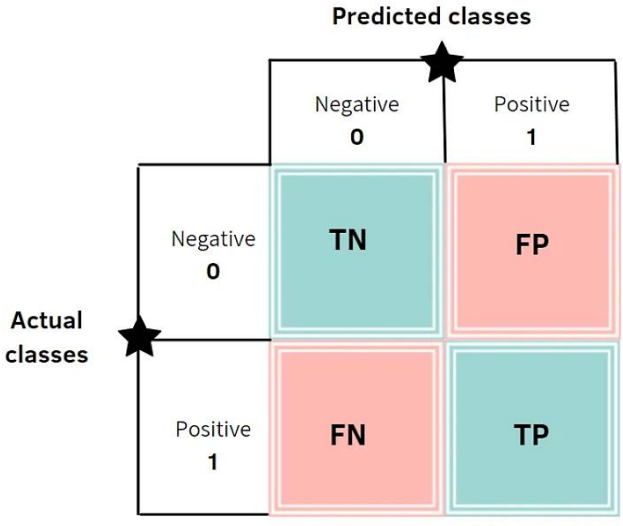



Figure III.15 : confusion matrix Table structure [35].

# 6. Conclusion

In this chapter, we have presented a comprehensive overview of the current state-of-the-art in deep learning. We delved into the foundational principles and structures of neural networks, exploring a diverse range of architectures that have emerged. These include Artificial Neural Networks (ANN), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), Autoencoders, and Deep Belief Networks (DBNs).

In summary, deep learning represents a revolutionary advancement in artificial intelligence, offering a suite of powerful tools for solving complex problems and pushing the boundaries of what machines can achieve. As we move forward, the next chapter will build on this foundation, applying deep learning methodologies to the design of an intelligent intrusion detection system. This application will illustrate the practical implementation of deep learning techniques in enhancing security measures, demonstrating the real-world impact and potential of this cutting-edge technology.

# Chapter IV

Conception of the proposed approach

# 1. Introduction

In this chapter, we will present in detail of the design of our proposed method for dimensionality reduction and classification of a large volumetric data taken from the network flow, for the purpose of intrusion detection.

In the first step, we pre-processed the data, which aimed to improve data quality, remove inconsistencies, and handling missing values.

For the second step consists of doing a dimension reduction of the database which contains a large number of attributes, using Wrapper methods in conjunction with Fire Hawk Optimizer (FHO).

Finally, we resorted to a classification approach based on deep analysis. We used CNN (Convolutional Neural Network) architecture, which is a type of deep learning algorithm.

# 2. General architecture of the proposed approach

The following figure (Figure IV.1) shows the general architecture of the proposed solution. This architecture consists of three parts:

The first part represents the pre-processing phase of the dataset CSE-CIC-IDS-2018. The second part is the feature selection process which consists of determining the best subset of features in terms of attack detection accuracy. This phase is based on the use of the Fire Hanks Optimizer Algorithm as a metaheuristic, which guides the search in the space of possible solutions. The selected subsets are then evaluated using a Random Forest model to identify the best feature subset. The last phase represents the validation phase, which takes the best subset of attributes from the selection phase to validate the CNN model obtained using the test dataset and gives the values of evaluation metrics such accuracy, precision, recall and F1-score.

# 3. Pre-processing of dataset

## 3.1. Data cleaning

The preprocessing workflow starts with data cleaning, which usually includes deleting rows containing invalid or missing data, and deleting one-valued columns (e.g. columns where all values are zero) [36], and some features have little effect on interpreting the behavior of data and traffic, whether it is normal or not. Therefore, these features such as the timestamp feature and internet protocol (IP) addresses that do not help train the neuron to detect errors and intrusions are deleted.

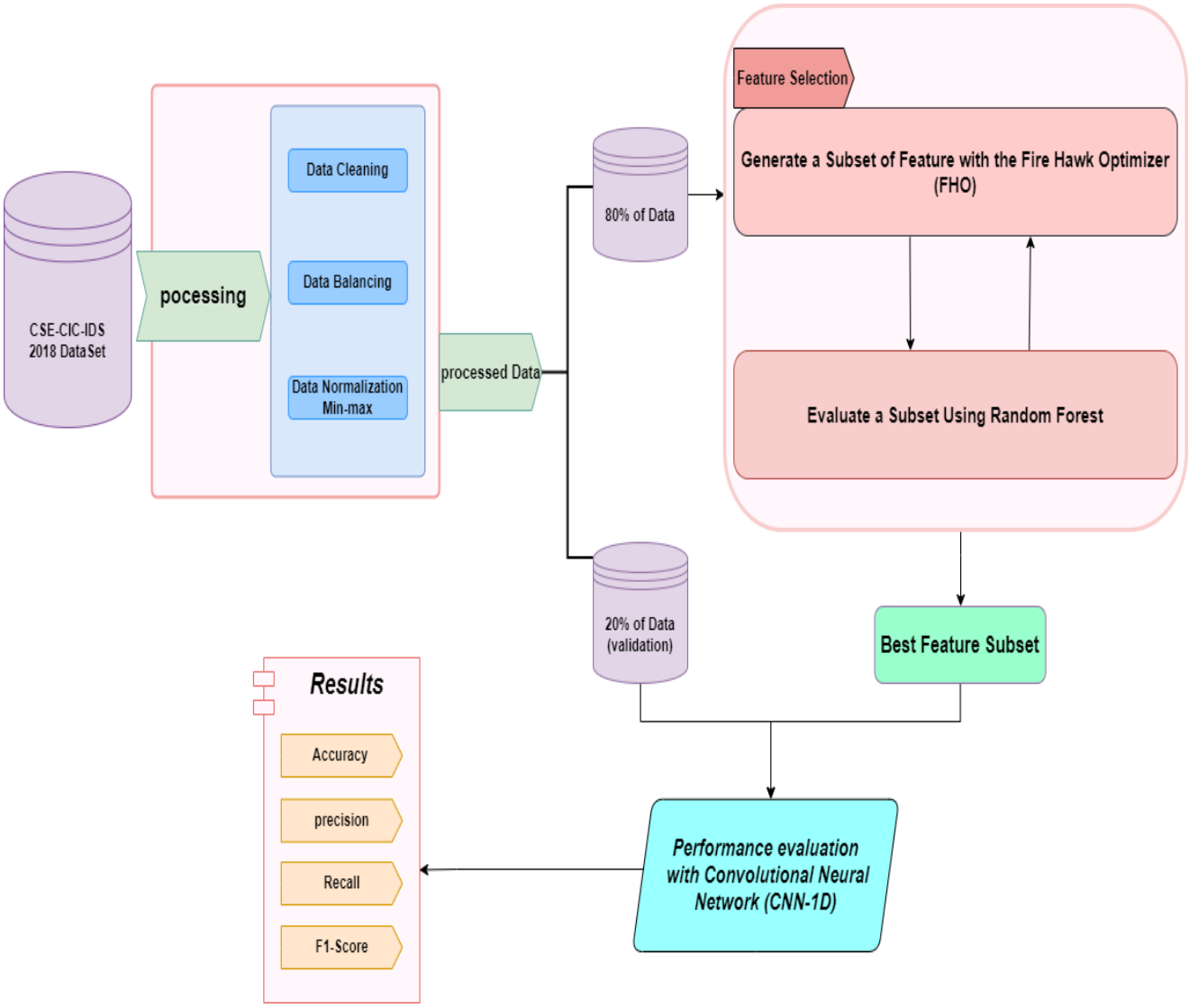



Figure IV.1: Diagram of the proposed approach for intrusion detection.

## 3.2. Data Balancing

To address class imbalance, we employed the under-sampling method on the database. This technique randomly removes rows to create a more balanced distribution between attack and non-attack instances. By reducing the over representation of certain classes, we aim to achieve a more equitable distribution across all classes.

## 3.3. Data Normalization

At this stage, data normalization involves converting numeric columns to a common scale. In many machine learning scenarios, certain feature values may significantly outweigh others, potentially skewing the learning process towards those with higher values. However, the magnitude of these values doesn't necessarily correlate with their importance in predicting the model output.

Data normalization addresses this by transforming multiple scaled data to a uniform scale. After normalization, all variables exert similar scale-related effects on the model, thus enhancing the stability and performance of the learning algorithm.

One widely used normalization technique is min-max scaling, which operates by rescaling a feature to a predefined range, typically [0, 1]. This process is achieved by subtracting the minimum value of the feature (denoted as $x_{min}$) from each data point (x) and then dividing the result by the range of the feature [36].

$$X_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4.1)$$

- $X_{new}$: represents the scaled value of the feature.
- x is the original value of the feature.
- $x_{min}$: is the minimum value of the feature.
- $x_{max}$: is the maximum value of the feature.

# 4. Feature selection

Feature selection involves selecting the most relevant features from the input dataset while discarding irrelevant ones [37]. There are three primary approaches to feature selection:

- *Filter methods:* These approaches filter the dataset to a subset containing only relevant features by ordering the features from best to worst. The ranking of features depends on the intrinsic properties of the data, for example, variance, consistency, distance, information, correlation, etc.

- *Wrapper methods:* Wrapper methods use a machine learning model to assess the performance of features. Based on this evaluation, features are either retained or eliminated to enhance the model's accuracy. Although more accurate than filtering, this method is also more complex.

- *Embedded methods:* Embedded methods assess the importance of each feature during the various training iterations of the machine learning model.

Feature selection, which is a data mining task aimed at reducing dimensionality within a dataset by selecting the most relevant features, can be formulated as a combinatorial optimization problem. In fact, when applied to intrusion detection, this problem would have the detection rate to be maximized as the objective function and the solution space would consist of all possible subsets of the initial set of features. Consequently, metaheuristics can be used to solve this problem.

In this work, use a wrapper method based on Fire Hawk Optimizer (FHO) for selecting the best subset of features. The wrapper method evaluates subsets of features by a classification algorithm based on deep learning, while FHO improves the selection process by iteratively updating subsets of features represented by fire hawks, according to the following steps:

- *Initializing Fire Hawks Population:* Start by initializing a population of fire hawks, each representing a potential feature subset. These subsets are encoded as binary vectors, where 1 represents inclusion and 0 represents exclusion of feature.

| 0 | 1 | 0 | ……. | 1 |
|---|---|---|---|---|
| 1 | | | | N |

- *Fitness Function Evaluation:* each feature subset is evaluated using the following fitness function:

$$fitness = accuracy \times \alpha + (1 - \alpha) \times \left(1 - \frac{Number_of_selected_features}{Tota_number_of_features}\right) \quad (4.2)$$

  - *accuracy:* is the performance metric of the deep learning model trained on the subset.
  - $\alpha$*:* is a weight parameter that balances the importance between accuracy and the ratio of selected features.

The value of fitness function depends on the performance of the deep learning model trained on the subset. This evaluation is conducted using a validation dataset. Higher values of the fitness function indicate better feature subsets.

- *Exploration and Exploitation:* FHO uses exploration and exploitation strategies to navigate the feature space. It simulates the behaviors of fire hawks, such as diving to explore new areas and cooperative hunting to exploit promising feature combinations.

- *Location Update:* We update the locations (distinct subsets) of fire hawks frequently based on their fitness values and the locations of other fire hawks. This process involves adjusting binary vectors representing subsets of features to effectively explore and exploit the feature space. Iterations continue until a stopping criterion is met, such that the maximum number of iterations is reached.

- *Binarization:* Before proceeding to the fitness function evaluation, each location vector representing a feature subset undergoes binarization using the transfer function S4. The transfer function S4 ensures smooth transitions between binary values, facilitating exploration and exploitation strategies in subsequent steps. The transfer function S4 is defined by the formula (4.3):

$$S_4 = \frac{1}{1+e^{-\beta(x-\alpha)}} \quad (4.3)$$

  - $x$ is the input value (binary value representing inclusion or exclusion of a feature).
  - $\alpha$ and β are parameters that control the shape of the sigmoid curve.

- *Selection of Optimal Subset:* In the end, The Fire hawk with the best fitness value (i.e., the lowest error rate on the validation set) corresponds to the optimal feature subset. This subset is selected as the final solution for feature selection phase.

# 5. Data classification

After the reduction, in a remarkable way, of the number of attributes (features) of the original database, it is now time to predict the classes of the new instances. In this study, two types of classification algorithms are applied for prediction and evaluation of best subset performance, with are: the random forest classifier and the One-dimensional Convolutional Neural Networks (1D CNN).

## 5.1. Random Forest (RF) Classifier

A Random Forest classifier is an ensemble learning method used for classification tasks, which operates by constructing multiple decision trees during training and outputting the mode of the classes (classification) of the individual trees. The idea behind this method is to combine the predictions of several base estimators to improve generalizability and robustness over a single estimator.

Random forest is not only a powerful prediction model but also an effective tool for feature selection in machine learning. During the training process, it assesses the importance of each feature by measuring its contribution to reducing impurity or inaccuracy in the model. Features that significantly influence decision-making across multiple trees receive higher importance scores. By analyzing these scores, data scientists can identify the most impactful features in their dataset. This inherent feature ranking capability streamlines the selection process, enabling practitioners to focus on the factors that will most significantly affect their analysis. Consequently, the feature selection ability of random forest enhances model efficiency, minimizes over fitting, and improves the overall interpretability of machine learning systems [38].

## 5.2. One-dimensional Convolutional Neural Networks (1D CNN) method

1D Convolutional Neural Networks (1D CNNs) are a class of neural networks that can be used for processing sequential data, such as time series or sequences of text. They are especially useful for tasks where the input data has a sequential structure, such as natural language processing, signal processing, or anomaly detection [39].

In a 1D CNN, the input and output data are both one-dimensional arrays, with the input typically representing a sequence of values or features. The network applies one-dimensional convolutional filters over the input data, which allows it to learn patterns and relationships within the sequence. This makes 1D CNNs well-suited for tasks like anomaly detection in network flow data, where the input is a sequence of network flow. 1D CNNs have proven to be effective in a variety of applications involving sequential data, thanks to their ability to capture local patterns and dependencies within the data [39].

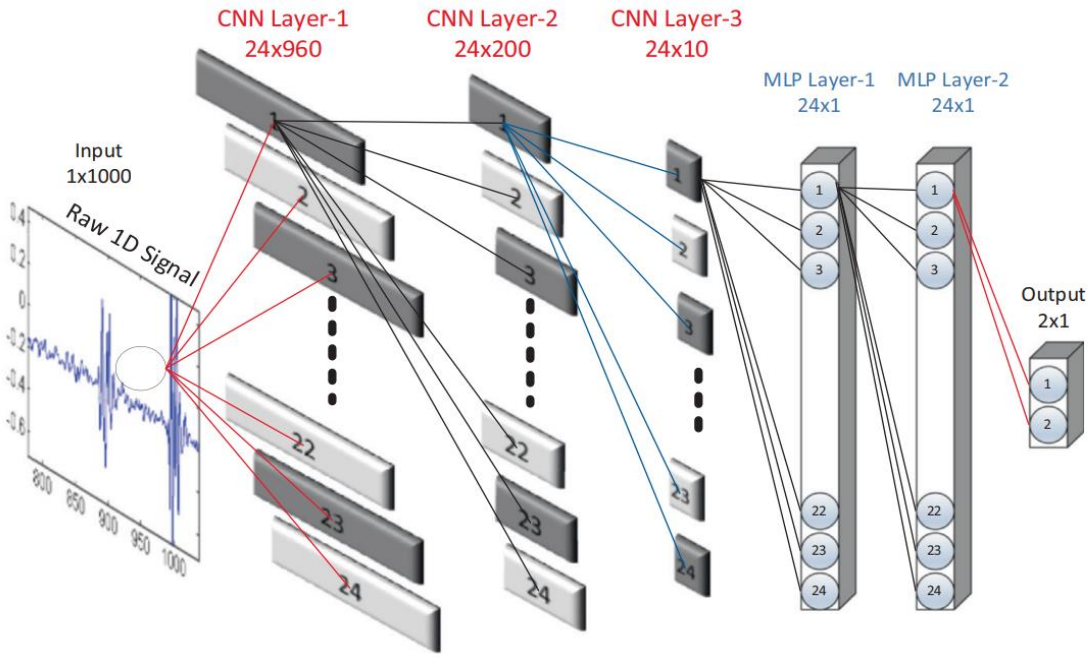



Figure IV.2: A sample 1D CNN configuration with 3 CNN layers and 2 MLP layers [39].

## 5.3. The Architecture of 1D CNN model

The typical structure of a 1D Convolutional Neural Network (1D CNN) is composed of three important layers: 1D convolutional layers, pooling layers, and fully connected layers. In addition to these layers, there are two more important components: the dropout layer and the activation function [40] (see the figure IV.3).

### 5.3.1. *One-dimensional convolutional layer*

This layer detects features in a vector. The raw one-dimensional input vector x[n], where n = 0, 1, . . ., N-1 is given as input to the first layer of the CNN architecture. The layer utilizes the following parameters:

1. Filters or Kernels: These are used to produce feature maps by performing convolutions with the input data. The number and size of the kernels are crucial for capturing relevant features from the input data [40].

2. Activation function: This function helps the model learn and approximate complex relationships between the variables of the network. Common activation functions include ReLU, Softmax, and sigmoid functions [40].

3. Stride: The stride value defines how the kernel moves across the input data. The most common value is 1, meaning the kernel moves over one column of the input data at each iteration.

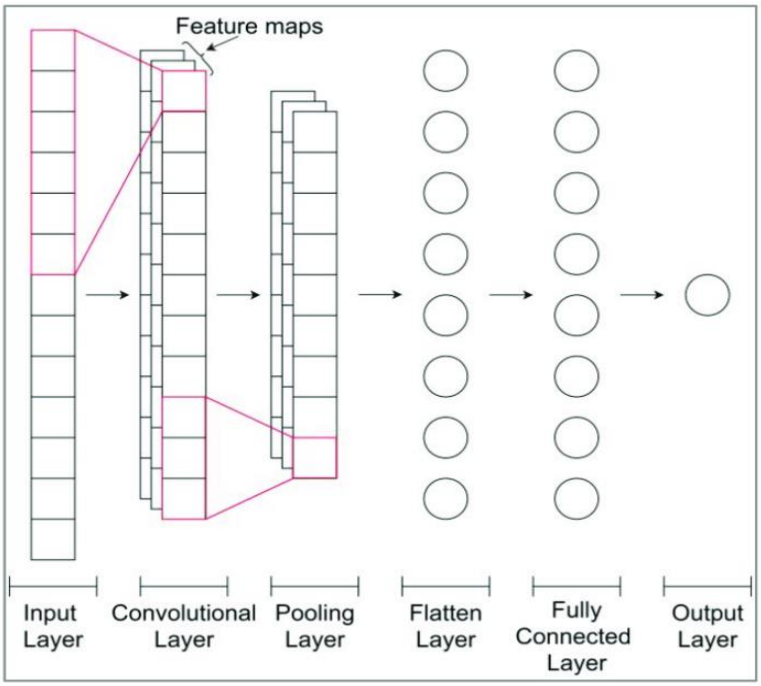



Figure IV.3: One-dimensional convolutional neural network (1D CNN) architecture [40].

### 5.3.2. *Pooling Layer*

This layer is often placed after the convolutional layer to reduce the size of the convolved feature map and computational costs. Common pooling operations include max pooling, average pooling, and sum pooling [40].

### 5.3.3. *Flatten layer and dropout layer*

The flatten layer transforms the input data into a one-dimensional vector to be fed into the fully connected (dense) layer. A dropout parameter is added after the flatten layer to prevent overfitting. During training, the dropout layer randomly drops some neurons from the neural network, reducing the model size.

### 5.3.4. *Dense fully connected layer*

The flattened output is given as input to the dense fully connected layer, which produces the final output. The activation function is also an important parameter of this layer [40].

## 5.4. Model Architecture of 1D CNN for IDS

The architecture of a Convolutional Neural Network (CNN) tailored for classification purposes, extensively employed within our project for tasks such as pattern recognition and feature processing, serves as a pivotal component in our pursuit of an effective and robust Intrusion Detection System (IDS).

This meticulously designed CNN framework, comprising four Conv1D layers organized into 2 pairs, each integrating two Conv1D layers, embodies our commitment to extracting intricate features from input data iteratively. With the initial pair embedding 64 filters and the subsequent pair hosting 128 filters, the CNN adeptly maneuvers through input data to discern spatial patterns across multiple abstraction levels. Following convolutional operations, Rectified Linear Unit (ReLU) activation functions introduce non-linearity, enabling the model to capture complex relationships within the data.

Additionally, Max-Pooling layers interspersed after each Conv1D pair optimize computational efficiency and mitigate overfitting risks by reducing spatial dimensions of feature maps. Meticulously inserted Dropout layers post-max-pooling strategically withhold a fraction of neurons during training to cultivate model robustness. After intricate feature extraction, a Flatten layer seamlessly transforms multidimensional feature maps into a one-dimensional vector, facilitating integration into fully connected dense layers. one dense layer, housing 128 units, decode feature-rich representations derived from convolutional layers, capturing intricate patterns and relationships within the data. Culminating this architectural marvel is the output layer, meticulously employing SoftMax activation to yield a probability distribution over classes, encapsulating the probability of each class given the input data.

This CNN architecture, meticulously crafted and optimized, stands as a robust and versatile framework, driving advancements in intrusion detection capabilities within our project.

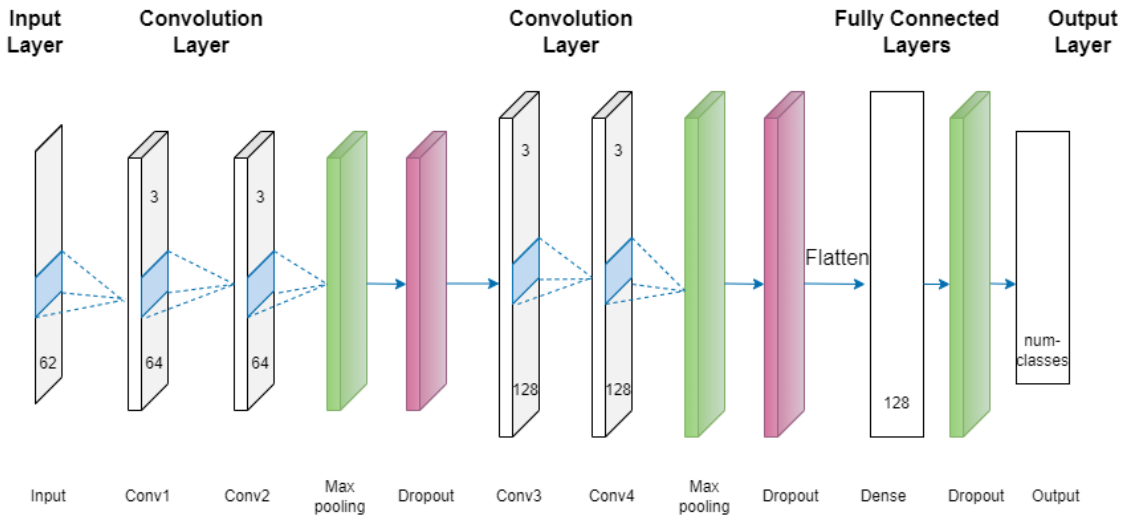



Figure IV.4: Model Architecture of 1D CNN for IDS.

# 6. Conclusion

This chapter detailed our method for reducing dataset dimensionality in an intrusion detection system to enhance classification performance. We presented the system architecture and discussed the integration of the Feature Highlighting Optimization (FHO) technique for feature selection. Each phase of our method was explained, from preprocessing raw data to feature extraction and the FHO technique, which selects significant features without losing critical information.

We described the implementation and optimization of deep learning models using the reduced feature set and explored various architectures to identify the most effective model. The integration of the selected model into the overall system was discussed, emphasizing seamless interaction for real-time intrusion detection.

Performance evaluation using metrics such as accuracy, precision, recall, and F1-score showed significant improvement, validating the FHO technique's effectiveness in enhancing deep learning models for intrusion detection. In conclusion, our method achieved a more efficient and accurate detection system, paving the way for further advancements in the field.

# Chapter V

## Tests and results

# 1. Introduction

We have now reached the final stage of our work. In this phase, we will implement the different phases of our architecture, integrating all the necessary modules and libraries to ensure that our model works properly. Finally, we will present the results obtained by applying the method described previously.

# 2. Development environment

To implement our solution, we require a set of hardware and software tools. This section presents an overview of the tools that been used.

## 2.1. Hardware specifications

In order to implement our solution, we used:

- A desktop computer equipped with an Intel i5-11320K processor, operating at 2,50 GHz, 8 GB of RAM, a 512 GB NVMe hard drive, and running the Windows 11 Pro operating system was utilized for pre-processing and model training purposes.
- Google Colaboratory: is a free and open-source platform introduced by Google, specifically designed for developers of data science applications, particularly machine learning applications.
- Kaggle: is a popular online community and platform for data scientists and machine learning practitioners. It provides various resources and tools to help users learn, build, and showcase their data science skills.

## 2.2. Software specifications and libraries

### 2.2.1. Software

**Python:** is a high-level, interpreted, multi-paradigm, and cross-platform programming language. It is used for web development, AI, machine learning, operating systems, mobile application development, video games, and many other purposes [41]. And the version of Python that we worked is Python 3.10.12.

**Visual Studio Code:** also known as VS Code, is a source code editor developed by Microsoft for Windows, Linux, and MacOS. It includes various features such as debugging support, syntax highlighting, and intelligent code completion. [42].

**Jupyter:** is a popular open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It supports various programming languages, including Python, R, and Julia, making it an ideal tool for data cleaning

and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and more. Jupyter notebooks are widely used in academia, industry, and research for interactive computing and data analysis.

### 2.2.2. Libraries

**NumPy:** Founded in 2005, NumPy is an open-source project built on the Numerical and NumArray libraries. It facilitates numerical computation in Python, offering support for large, multidimensional arrays and matrices along with a wide array of high-level mathematical functions. NumPy is fundamental for scientific computing in Python [43]. Our study was conducted with version NumPy 1.25.2.

**Pandas:** is defined as a fast, easy open-source software library that is used for data analysis in Python programming language. It is rapidly used in different projects like software development, machine learning, computer vision, natural language processing [44]. Our study was conducted with version Pandas 2.0.3.

**Keras:** it is an open-source library written in Python. It offers consistent and simple APIs for interacting with neural network algorithms [45]. For the version that we used it Keras 2.15.0.

**TensorFlow:** is a free and open-source library developed by Google for machine learning and artificial intelligence. It can be used for a variety of tasks, but it focuses specifically on training and inference of deep neural networks [46]. For the version that we used it TensorFlow 2.15.0.

**scikit-learn:** often abbreviated as sklearn, is a popular machine learning library for Python. It provides simple and efficient tools for data mining and data analysis, built on NumPy, SciPy, and matplotlib. scikit-learn includes a wide range of machine learning algorithms for classification, regression, clustering, dimensionality reduction, and more. It also provides tools for model selection, evaluation, and data preprocessing [47]. For the version that we used it 1.2.2.

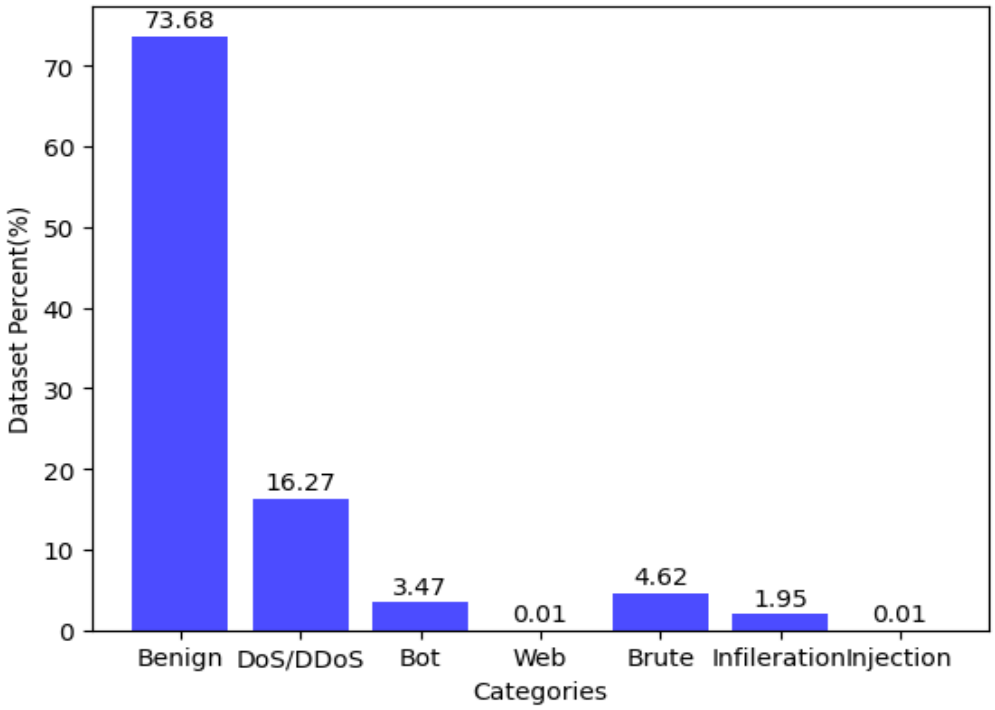
The following Figure V.1 shows the logos of the specifications cited above:




Figure V.1: Logos of software specifications and libraries used.

# 3. Description of the database used

The CSE-CIC-IDS2018 dataset is a comprehensive dataset for network intrusion detection research, created collaboratively by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). This dataset is one of the most recent IDS datasets, it simulates real-world network traffic, including benign and malicious activities, over a span of ten days. It encompasses a variety of attack scenarios such as brute force, botnets, denial of service (DoS), distributed denial of service (DDoS), infiltration, and web attacks. Figure V.2 depicts the proportion and distribution of each traffic category.




FigureV.2: Frequencies for intrusion categories [48].

The dataset includes detailed features extracted from network traffic, such as flow duration, protocol type, source and destination IPs, and payload sizes, making it valuable for developing and evaluating intrusion detection systems using machine learning and deep learning techniques. Since the dataset has 80 features, the following table (Table V.1) display only a subset of them, where the first column contains the feature's name and the second column contains a brief explanation [48,49].

The attacking infrastructure consists of 50 machines, whereas the infrastructure of the victim firm consists of 420 machines and 30 servers across five departments (see Figure.V.3). The dataset contains each computer's network traffic and system logs, as well as eighty characteristics collected from the recorded network traffic.

| Feature name | Feature short description |
|---|---|
| Dst Port | Destination port of connection |
| Protocol | Protocol used during connection |
| Timestamp | Time that connection occurred |
| Flow duration | Duration that connection occurred |
| Tot Fwd Pkts | Total number of forward packets |
| Tot Bwd Pkts | Total number of backward packets |
| TotLen Fwd Pkts | Total length of forward packets |
| Fwd Pkt Len Max | Maximum length of forward packets |
| Bwd Pkt Len Mean | Mean size of packet in backward direction |
| Flow IAT Std | Standard deviation time between two packets sent in the forward direction |
| Fwd Seg Size Min | Minimum segment size observed in the forward direction |
| … | … |
| Active mean | Mean time a flow was active before becoming idle |
| Idle Std | Standard deviation time a flow was idle before becoming active |
| Idle Min | Minimum time a flow was idle before becoming active |
| Label | Describes if file is Attack or Benign |

Table V.1: Description of a part of features in CSE-CIC-IDS2018 dataset [49].

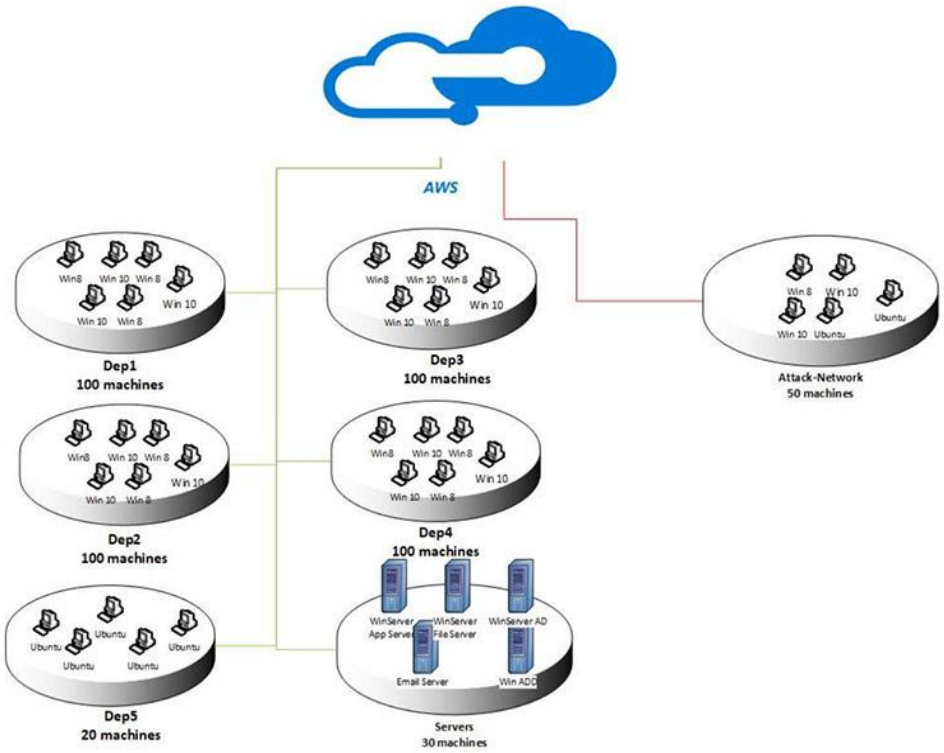



Figure.V.3: Network Topology [50].

For our study, we selected data from 4 days. We have obtained these results after pre-processing.

| Days | Classes | Before Pre-processing | After Pre-processing |
|---|---|---|---|
| **Day 15-02-2018** | **3 classes** | **Number of rows:** 1048575<br>**Number of columns (Features):** 80<br><br>***Benign*:** 996077<br>***DoS attacks GoldenEye*:** 41508<br>***DoS attacks-Slowloris***: 10990 | **Number of rows:** 32970<br>**Number of columns:** 62<br><br>***Benign:*** 10990<br>***DoS attacks-Goldeneye:*** 10990<br>***DoS attacks-Slowloris*:** 10990 |
| **Day 21-02-2018** | **3 classes** | **Number of rows:** 1048575<br>**Number of columns:** 80<br><br>***Benign*:** 360833<br>***DDOS attack-HOIC*:** 686012<br>***DDOS attack-LOIC-UDP*:** 1730 | **Number of rows:** 5190<br>**Number of columns:** 54<br><br>***Benign*:** 1730<br>***DDOS attack-HOIC*:** 1730<br>***DDOS attack-LOIC-UDP*:** 1730 |
| **Day 22-02-2018** | **4 classes** | **Number of rows:** 1048575<br>**Number of columns (Features):** 80<br><br>***Benign:*** 1048213<br>***Brute Force -Web*:** 249<br>***Brute Force -XSS*:** 79<br>***SQL Injection*:** 34 | **Number of rows (Samples): 1**36<br>**Number of columns:** 59<br><br>***Benign* :**34<br>***Brute Force -Web*:** 34<br>***Brute Force -XSS*:** 34<br>***SQL Injection*:** 34 |
| **Day 23-02-2018** | **4 classes** | **Number of rows:** 1048575<br>**Number of columns:** 80<br><br>***Benign:*** 1048009<br>***Brute Force -Web*:** 362<br>***Brute Force -XSS*:** 151<br>***SQL Injection*:** 53 | **Number of rows (Samples): 212**<br>**Number of columns: 59**<br><br>***Benign:*** 53<br>***Brute Force -Web*:** 53<br>***Brute Force -XSS*:** 53<br>***SQL Injection*:** 53 |

Table V.2: Physical characteristics of the CSE-CIC-IDS-2018 database.

# 4. Dimensionality Reduction

As we already mentioned in the previous chapter, we chose Fire Hawk Optimizer (FHO) as dimensionality reduction module.

In our implementation, we have set the population size to 20 and the number of iterations to 10. These parameters govern the exploration and exploitation capabilities of the FHO algorithm, determining how many candidate solutions are considered in each iteration and how many iterations are executed to refine the feature selection process. By fixing these parameters, we aim to strike a balance between computational efficiency and the quality of feature selection outcomes.

## 4.1. Results of dimensionality reduction

The following table presents the results obtained by applying the aforementioned steps to the four days selected for our study.

| File | Before Applying FHO | After Applying FHO | |
|---|---|---|---|
| | ***Number of Features*** | ***Number of Features*** | ***Fitness*** |
| **Day 15-02-2018** | 62 | 4 | 99.90 |
| **Day 21-02-2018** | 54 | 3 | 99.90 |
| **Day 22-02-2018** | 59 | 5 | 97.33 |
| **Day 23-02-2018** | 59 | 7 | 91.50 |

Table V.3: Results of dimensionality reduction.

## 4.2. Analysis of the results of dimensionality reduction

As we can see, the algorithm proposed by FHO successfully achieved its goal of reducing the dimensions of our database while maintaining high fitness. The following figures show the graphs of the convergence of the fitness function (to maximize) and the number of features selected (to minimize) for different 4 files in the dataset, for a number of iterations equal to 10.

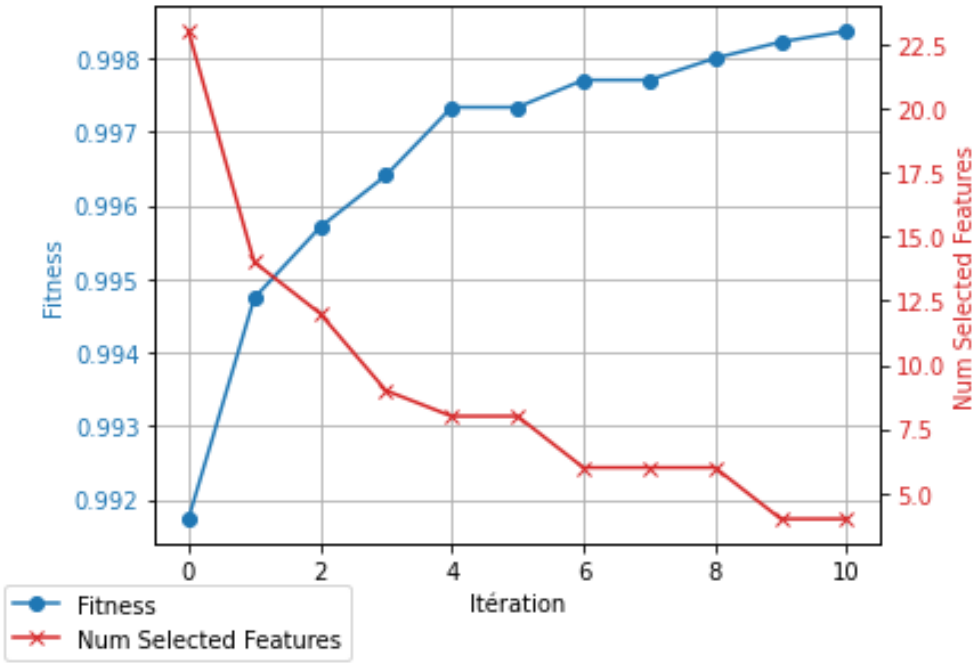



Figure V.4: Graph of Fitness and the Number of Selected Features on 15-02-2018.

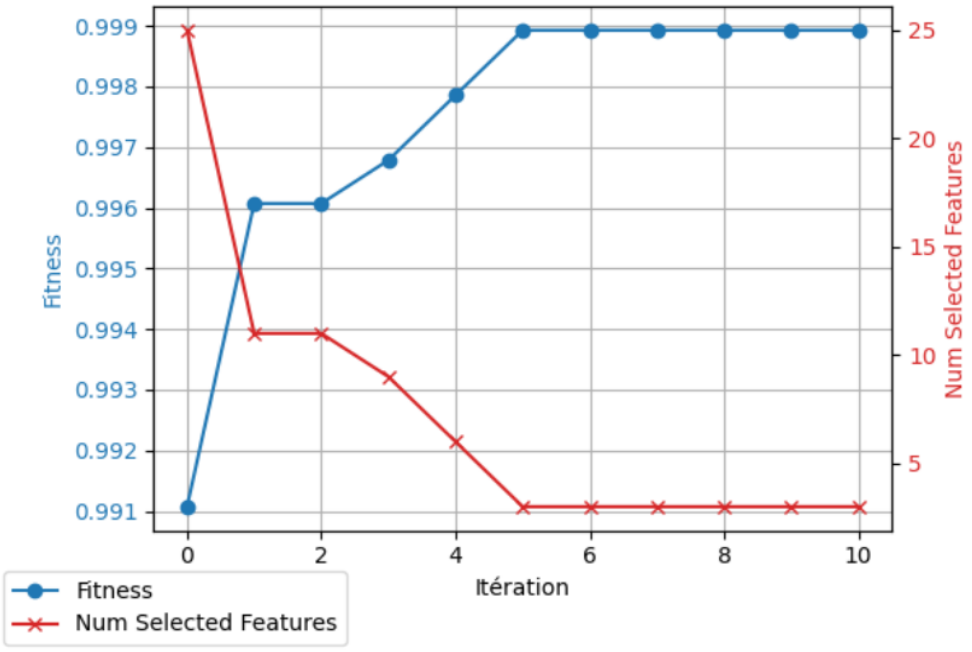



Figure V.5: Graph of Fitness and the Number of Selected Features on 21-02-2018.

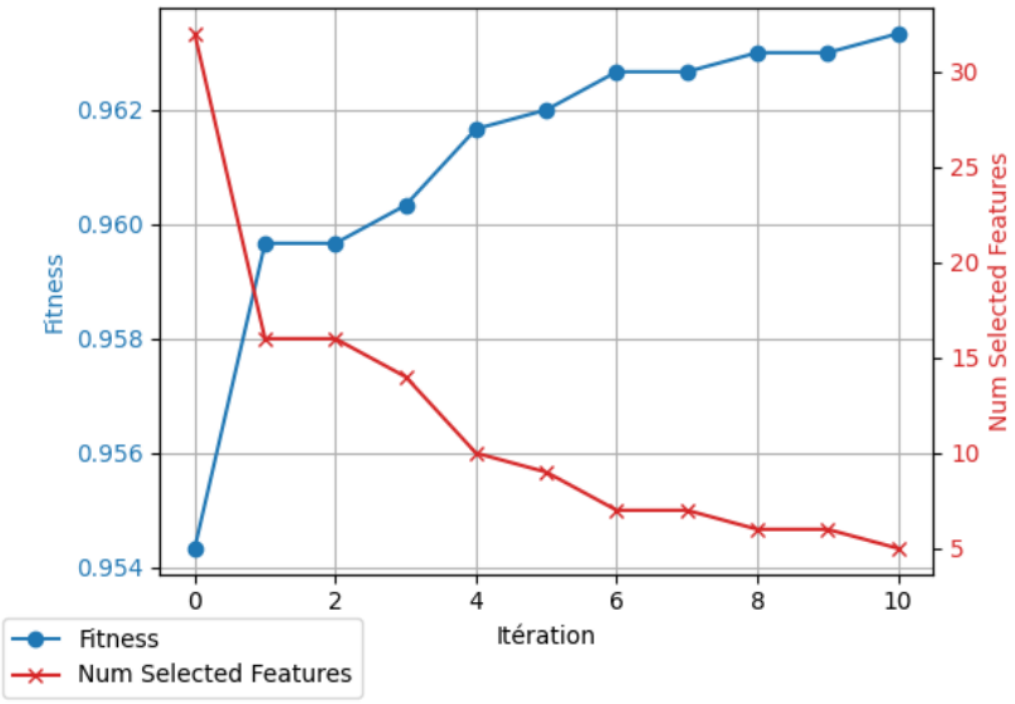



Figure V.6: Graph of Fitness and the Number of Selected Features on 22-02-2018.

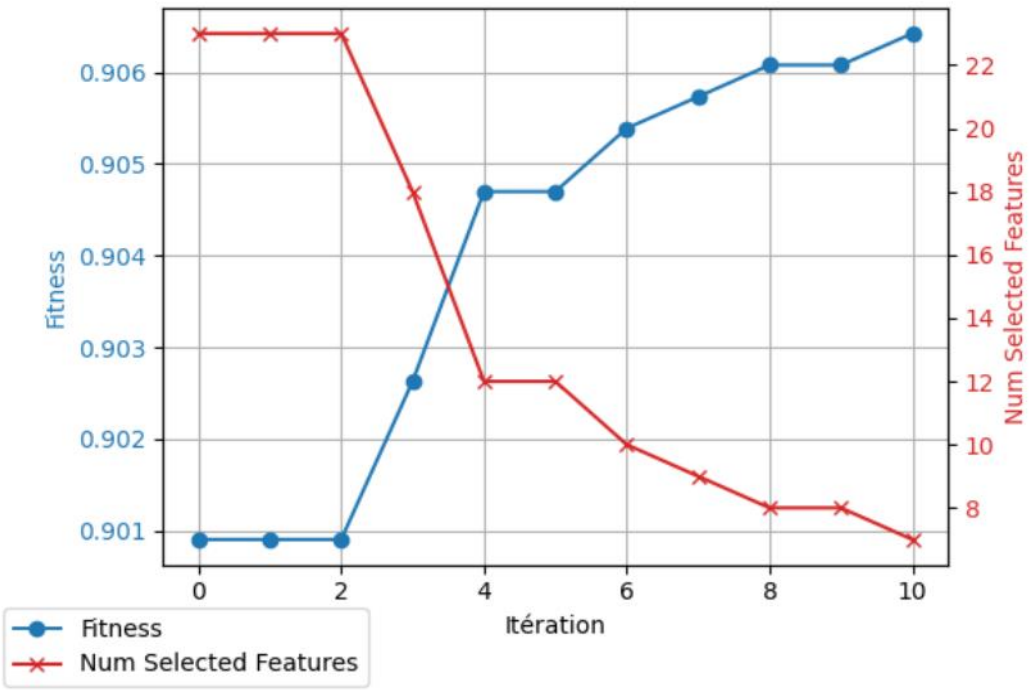



Figure V.7: Graph of Fitness and the Number of Selected Features on 23-02-2018.

# 5. Classification

For the classification task, we initially employed a Random Forest algorithm to establish a baseline performance. Following this, we transitioned to a more sophisticated approach by utilizing a Convolutional Neural Network (CNN). Extensive testing phase was conducted to determine the optimal number of layers for the CNN architecture. Our CNN model was trained for 50 epochs with a batch size of 64 instances. Additionally, we adopted 1D convolution with a kernel size of 3 and 128 output filters, determined through rigorous evaluation tests. Size of 2 was applied for the max-pooling layer. The activation function chosen for both the convolutional and fully connected layers is ReLU, as well as for the final layer, was softmax. The number of classes in the final layer, denoted as Nb_Classes, varied on the dataset used. This hybrid approach allowed us to comprehensively evaluate the performance of both Random Forest and CNN-1D models.

## 5.1. Evaluation parameters of the CNN network

We have chosen a CNN network as our classification model to classify the vectors derived from the dimensionality reduction process conducted by the Fire Hawk Optimizer (FHO) algorithm. Our primary evaluation metric for assessing the model's performance is accuracy, aiming to maximize precision in class predictions. For the loss function, we have selected categorical_crossentropy, which aligns with our multi-class classification objective. In terms of optimization, we have configured the optimizer with a learning rate of 0.01 and a momentum of 0.9.

## 5.2. Classification results

The proposed CNN network enabled us to classify instances from the database with very satisfactory accuracy. The following Table V.4 provides the results obtained by applying the previous steps to the various files in the database.

| **File** | **Number of rows** | **Training** | | **Validation** | |
|---|---|---|---|---|---|
| | | *Accuracy %* | *Loss* | *Accuracy %* | *Loss* |
| Day 02-15-2018 | 32970 | 96.7 | 3.2 | 96.3 | 3.1 |
| Day 02-21-2018 | 5190 | 99.3 | 1.9 | 99.6 | 1.8 |
| Day 02-22-2018 | 136 | 63.6 | 10.3 | 83.4 | 10.2 |
| Day 02-23-2018 | 212 | 78.4 | 10 | 70.7 | 10.2 |

Table V.4: Classification results of the four files used.

## 5.3. Analysis of classification results

The following figures show the graphs of the loss function as well as the accuracy obtained for the different four files used from CSE-CIC-IDS2018 database, for a number of epochs vary depending on the dataset (equal to 15, 20 or 50) and this with both datasets: training and validation.

### 5.3.1. *Dataset of day 02-15-2018*

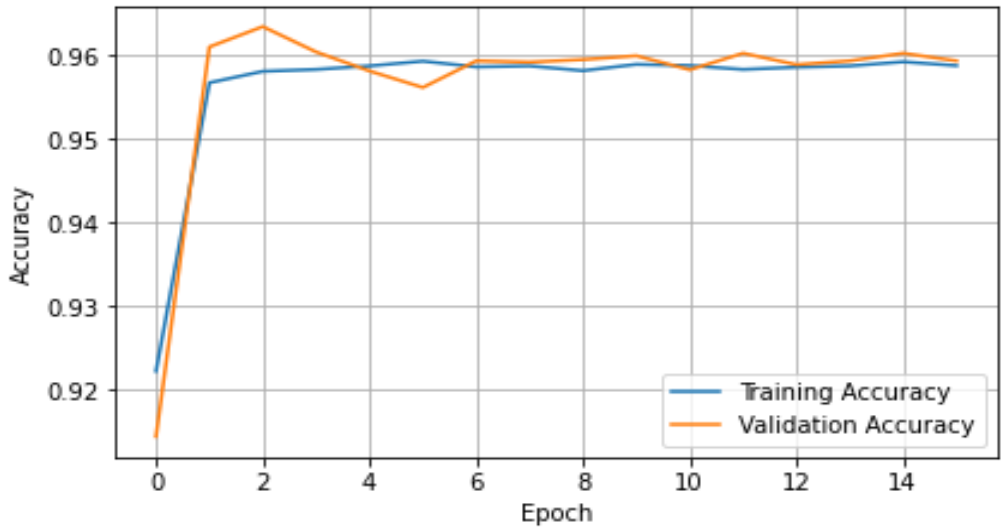



Figure V.8: Training and Validation Accuracy of CNN 1D Model on 02-15-2018 dataset.

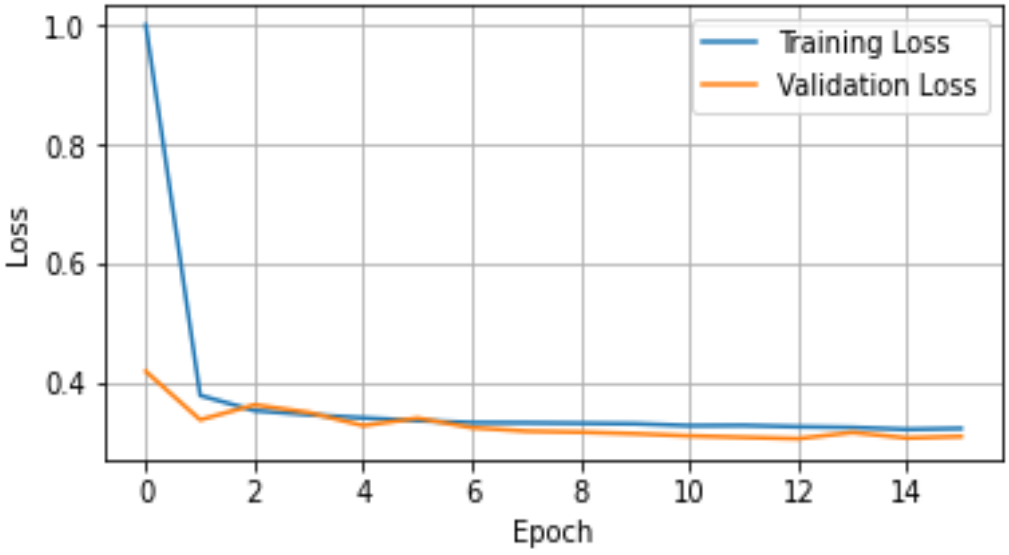



Figure V.9: Training and Validation Loss of CNN 1D Model on 02-15-2018 dataset.

Overall Performance:

This following confusion matrix visualizes the performance of a multi-class classification model, showing the number of correct and incorrect predictions for each class. The matrix has three classes (0, 1, and 2), and each cell contains the count of instances where the actual class (true label) matches or differs from the predicted class.

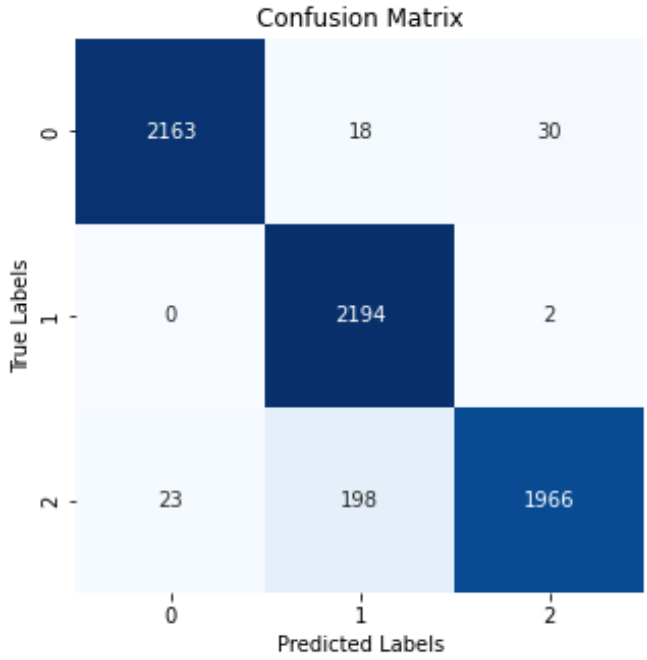



Figure V.10: Confusion Matrix of 02-15-2018 dataset .

## 5.3.2. *Dataset of day 02-21-2018*

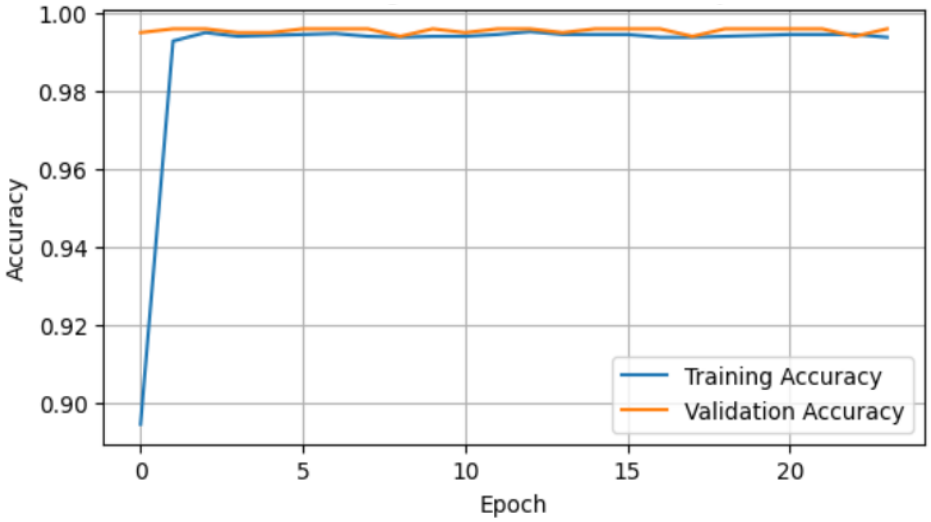



Figure V.11: Training and Validation Accuracy of CNN 1D Model on 02-21-2018 dataset.

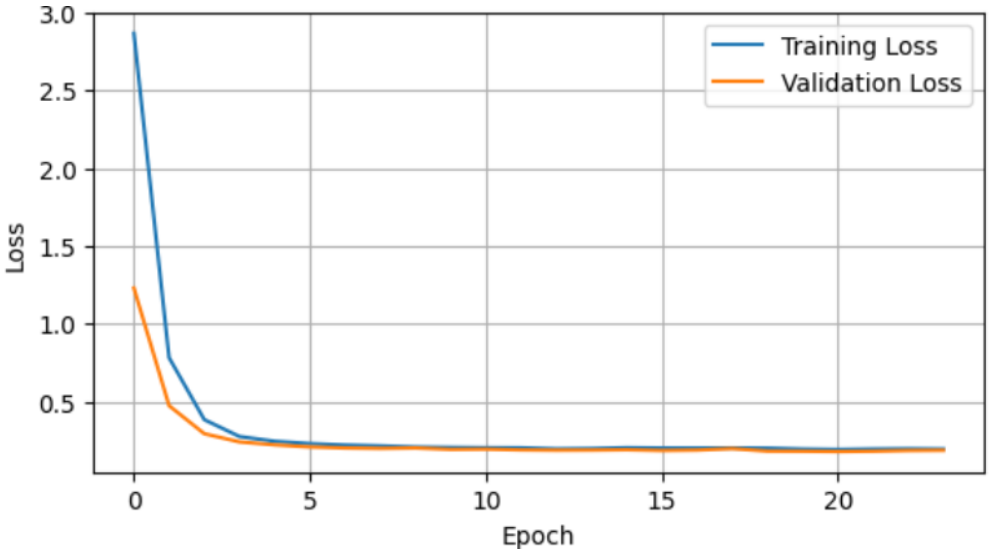



Figure V.12: Training and Validation Loss of CNN-1D Model on 02-21-2018 dataset.

Overall Performance

The model has perfect accuracy for classes 0 and 1. There are a few misclassifications for class 2, where 4 instances were incorrectly predicted as class 1. The overall performance of the model is excellent, with the vast majority of instances correctly classified.

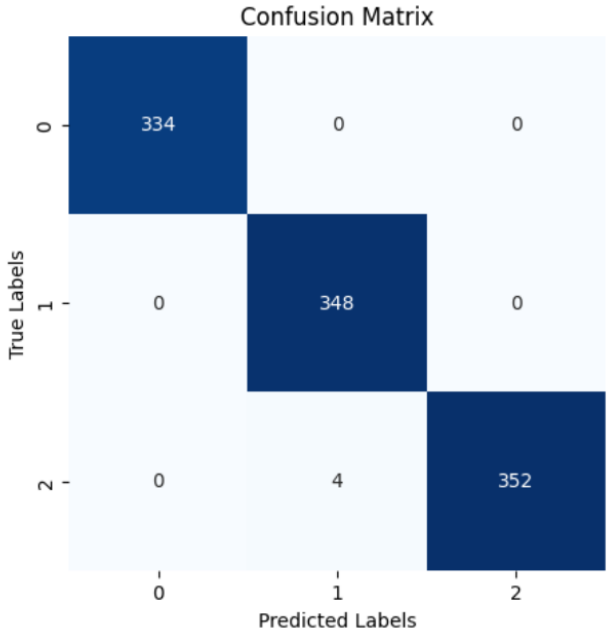



Figure V.13: Confusion Matrix of 02-21-2018 dataset.

### 5.3.3. *Dataset of day 02-22-2018*

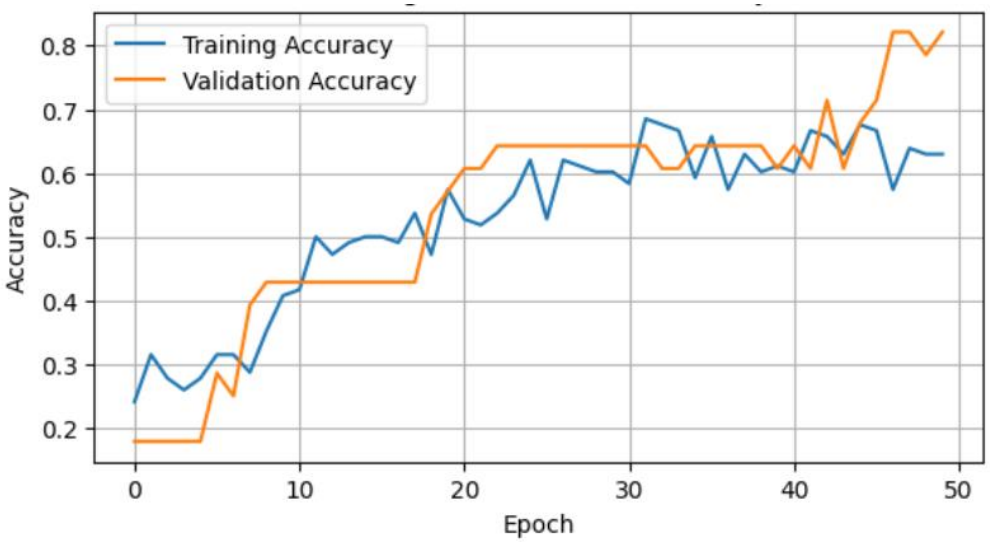



Figure V.14: Training and Validation Accuracy of CNN-1D Model on 02-22-2018 dataset.

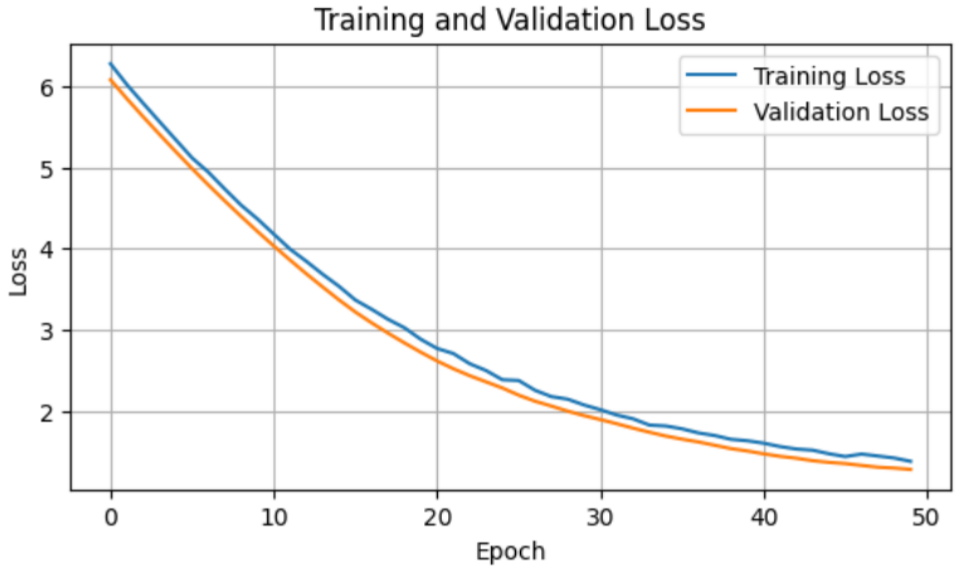



Figure V.15: Training and Validation Loss of CNN-1D Model on 02-22-2018 dataset.

Overall Performance

The model shows a strong performance with high correct prediction counts for each class. There are a few misclassifications:

- Class 0 is misclassified as class 3 once.
- Class 1 is misclassified as class 2 three times.
- Class 3 is misclassified as class 1 once.

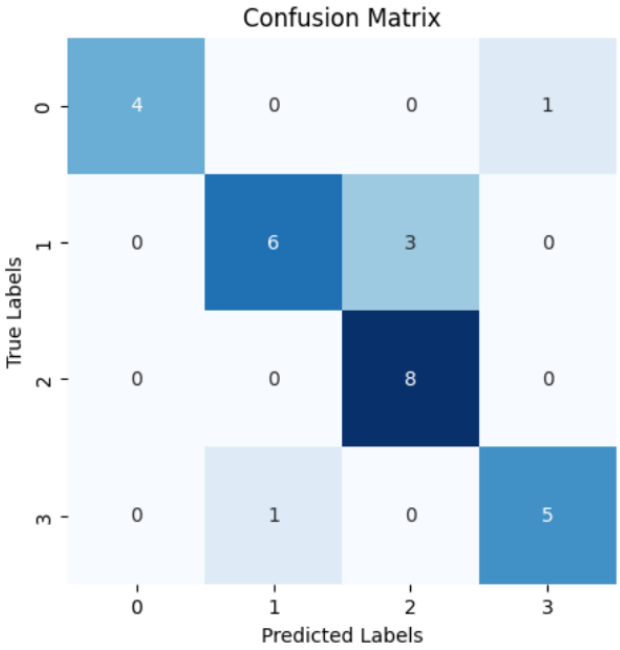



Figure V.16: Confusion Matrix of 02-22-2018 dataset.

### 5.3.4. *Dataset of day 02-23-2018*

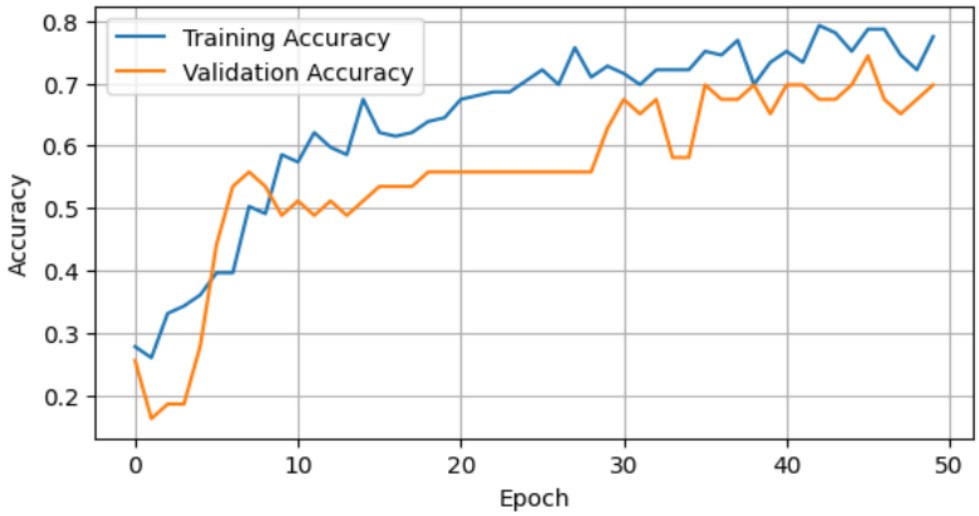



Figure V.17: Training and Validation Accuracy of CNN-1D Model on 02-23-2018 dataset.

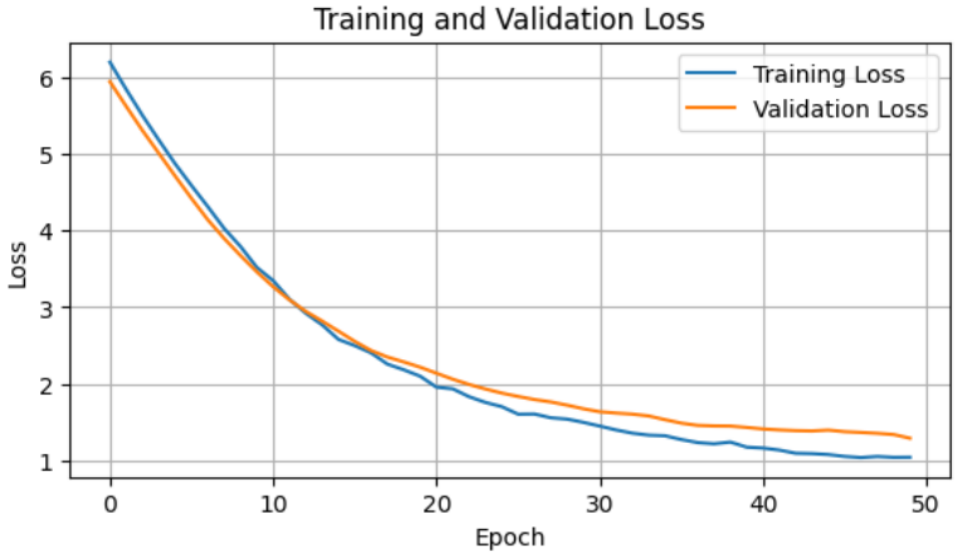



Figure V.18: Training and Validation Loss of CNN-1D Model on 02-23-2018 dataset.

Overall Performance

The model generally performs well with a majority of correct predictions for each class. Misclassifications occur, especially between class 2 and class 3:

- Class 0 is misclassified as class 1 and class 2.
- Class 1 is misclassified as class 2.
- Class 2 is misclassified as class 1 and class 3.
- Class 3 is often misclassified as class 2.

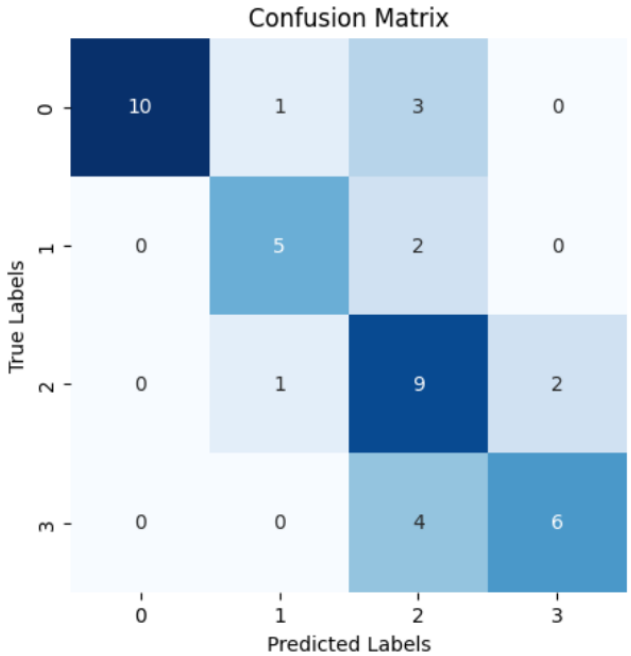



Figure V.19: Confusion Matrix of 02-23-2018 dataset.

# 6. Comparison of the CNN model with the Random Forest model

To evaluate the CNN model, we compared our results with those obtained from a research study using the Random Forest model, without feature selection and with feature selection. The table V.5 give a comparison of the results obtained by the two classification algorithms before applying feature selection and after using feature selection, on the four datasets.

| File | Metric | CNN without feature selection | Random Forest without feature selection | CNN with feature selection | Random Forest with feature selection |
|---|---|---|---|---|---|
| **Day 02-15-2018** | **Accuracy %** | 100 | 100 | 97 | 100 |
| | **Recall %** | 100 | 100 | 100 | 100 |
| | **Precision %** | 100 | 100 | 99 | 100 |
| | **F1 score %** | 100 | 100 | 98 | 100 |
| **Day 02-21-2018** | **Accuracy %** | 100 | 100 | 100 | 100 |
| | **Recall %** | 100 | 100 | 100 | 100 |
| | **Precision %** | 100 | 100 | 100 | 100 |
| | **F1 score %** | 100 | 100 | 100 | 100 |
| **Day 02-22-2018** | **Accuracy %** | 60 | 81 | 83 | 96 |
| | **Recall %** | 100 | 83 | 100 | 100 |
| | **Precision %** | 100 | 71 | 100 | 86 |
| | **F1 score %** | 89 | 77 | 89 | 92 |
| **Day 02-23-2018** | **Accuracy %** | 60 | 84 | 71 | 81 |
| | **Recall %** | 100 | 84 | 75 | 80 |
| | **Precision %** | 75 | 89 | 100 | 89 |
| | **F1 score %** | 80 | 84 | 83 | 84 |

Table V.5. Comparison of the CNN model with the Random Forest model.

By comparing the results, we observed that Both CNN and Random Forest models perform perfectly (scores across all metrics = 100%) on Days 02-15-2018 and 02-21-2018 when we do the classification without feature selection. On Days 02-22-2018 and 02-23-2018, the performance of both models drops, with CNN experiencing a more significant decrease in accuracy and F1 score compared to Random Forest (for CNN model the accuracy drops significantly to 60%, while recall and precision remain high, leading to an F1 score of 89% in 02-22-2018 day, and a noticeable drop in precision (75%) and F1-score (80%) in the 02-23-2018 day).

In the case of the use of feature selection, the both classifiers gave high scores for the four metrics with a small difference in favor of the random forest model, especially in accuracy and F1-score metrics for days 02-15-2018 and 02-21-2018. On the datasets of the days 02-22-2018 and 02-23-2018, the accuracy of CNN model is improved form 60% (in the classification without feature selection) to 83% and 71% respectively, and F1 score is improved from 80% to 83% on the dataset of day 02-23-2018. Random forest model gives significant improvement in accuracy (from 81% to 96%) and slightly drop in accuracy (from 84% to 81%).

These results illustrate how feature selection using the fire hanks optimizer algorithm can positively impact model performance by enhancing accuracy, recall, precision, and F1 score, ultimately leading to more robust and effective classification models. By focusing on relevant features, the model can make more accurate and reliable predictions.

# 7. Conclusion

This final chapter meticulously outlines the tools and methodologies employed to conduct the experimental work, offering a comprehensive account of the procedures followed and the technologies utilized. The primary focus was on evaluating the effectiveness and accuracy of the predictive models developed during the study.

The results obtained from these experiments were thoroughly analyzed and compared to assess their performance. Each model's prediction accuracy was scrutinized using various statistical and graphical methods, ensuring a robust comparison. The analysis highlighted the strengths and weaknesses of each model, providing insights into their suitability for different types of predictive tasks.

# General Conclusion

In this thesis, we have presented the development and evaluation of an advanced intrusion detection system (IDS) leveraging the Fire Hawk Optimizer (FHO) algorithm in conjunction with deep learning technique. The primary goal was to enhance the reliability andeffectiveness of intrusion detection systems by integrating the optimization capabilities of FHO with the robust pattern recognition abilities of deep learning models. This work has made significant progress in addressing cyber threats and enhancing network security, contributing to the development of effective system for detecting new and complex attacks.

One of the key achievements of this work is the development and implementation of an enhanced Intrusion Detection System (IDS) that integrates a one-dimensional enhanced convolutional neural network (CNN-1D) with the Fire Hawk Optimizer (FHO) algorithm for feature selection. This approach addresses the limitations imposed by traditional IDS methodsby leveraging the powerful capabilities of deep learning and advanced feature selection techniques. The CNN-1D works to significantly improve the accuracy of detecting abnormal network traffic by automatically extracting higher-level features from the raw data. The FHO algorithm demonstrated significant effectiveness in reducing the dimensionality of the dataset while retaining critical features essential for accurate detection. This document demonstrates how the combination of these advanced techniques can lead to more efficient and effective intrusion detection, providing a strong and adaptable solution to network security challenges.

We used the CSE-CIC-IDS2018 dataset, which is a comprehensive and widely recognized dataset used to evaluate the performance of Intrusion Detection Systems (IDS). It includes a wide range of network traffic patterns, including normal and malicious activities, making it an ideal source for training and testing deep learning models.

Extensive experiments were conducted using this dataset to evaluate the proposed system. Results indicated that the combination of FHO and deep learning achieved high accuracy, recall, precision, and F1 scores. Specifically, the IDS maintained robust performance across multiple attack types, validating the efficacy of the proposed approach.

Looking to the future, there are several perspectives that should be considered for further research:

• Testing the proposed method on the entire data of all 10 days of the dataset and also onother benchmark datasets.

• Development of Advanced parallel implementation of the proposed feature selection algorithm for reducing the execution time and enhancing the efficacy of IDS.

In summary, this document demonstrates that the integration of the Fire Hawk Optimizer algorithm with deep learning techniques represents a significant advancement in the field of intrusion detection systems.

# Bibliographic references

[1] S. H. Abbas, W. A. Khuder Naser, and A. A. Kadhim; Subject review: Intrusion Detection System (IDS) and Intrusion Prevention System (IPS); Global Journal of Engineering and Technology Advances, vol. 14, no. 2, pp. 155–158, 2023.

[2] S. Bezborodov; Intrusion Detection Systems and Intrusion Prevention System with Snort provided by Security Onion; Bachelors Thesis, Information Technology, 2016.

[3] A. Lazarevic, V. Kumar,and J.Srivastava,Intrusion Detection: A Survey;in Managing Cyber Threats, Springer-Verlag, 2005, pp. 19–78.

[4] S. M. Othman, T. Nabeel, A. T. Zahary, N.T.Alsohybe,and F. Mutaher Ba-Alwi,Survey on Intrusion Detection System Types,2018.

[5] M. Labonne, Anomaly-based network intrusion detection using machine learning, n. d.

[6] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, Survey of intrusion detection systems: techniques, datasets and challenges,Cybersecurity, vol. 2, no.1,2019.

[7] U. H. Rao and U. Nayak, Writing Together. The InfoSec Handbook an Introduction to Information Security, Heinz Weinheimer, 2014.

[8] D. Mudzingwa and R. Agrawal, A study of methodologies used in intrusion detection and prevention systems (IDPS), in Conf. Proc. IEEE SOUTHEASTCON, 2012.

[9] Z. Halim et al., An effective genetic algorithm-based feature selection method for intrusion detection systems, Computers and Security, vol. 110, 2021.

[10] V. Priyalakshmi and R.Devi, Evaluation of Efficient Classification Algorithm for Intrusion Detection System,International Journal of Advanced Research in Science, Communication and Technology, pp. 39–45, 2022.

[11] I. H. Sarker, Y. B. Abushark, F. Alsolami, and A. I. Khan, IntruDTree: A machine learning based cybersecurity intrusion detection model, Symmetry, vol. 12, no. 5, 2020.

[12] M. Azizi, S. Talatahari, and A. H. Gando mi Fire Hawk Optimizer: a novel metaheuristic Algorithm, Artificial Intelligence Review, vol. 56, no. 1, pp. 287– 363,2023.

[13] B. I. Farhan and A. D. Jasim, Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset, Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 2, pp. 1165–1172, 2022.

[14] L. Deng and D. Yu, Deep learning: Methods and applications, Foundations and Trends in Signal Processing, vol. 7, no. 3-4, pp. 197–387, 2013.

[15] Intelligence artificielle, Larousse. [Online]. Available : https://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257.

[16] Neural networks | A beginner 39; s guide, Geeks for Geeks. [Online]. Available: https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/.

[17] K. Aderghal, Classification of multimodal MRI images using deep learning: Application to the diagnosis of Alzheimer 39; s disease, 2021.

[18] What is a Neural Network, AWS. [Online]. Available: https://aws.amazon.com/what-is/neural-network/?nc1=h_ls.

[19] S. Khan, M. Sajjad, T. Hussain, A. Ullah, and A. S. Imran, A review on traditional learning and deep learning models for WBCs classification in blood smear images, IEEE Access, vol. 9, pp. 10657–10673, 2021.

[20] M. M. Adnan et al., Automatic image annotation based on deep learning models: Asystematic review and future challenges, IEEE Access, vol. 9, pp. 50253–50264,2021.

[21] A.Wu et al.,Convolutional reconstruction-to-sequence for video captioning, IEEE Trans. Circuits Syst. Video Technol., vol. 30, no. 11, pp. 4299–4308, 2020.

[22] K. Khurana and U. Deshpande, Video question-answering techniques, benchmark datasets and evaluation metrics leveraging video captioning: A comprehensive survey, IEEE Access, vol. 9, pp. 43799–43823, 2021.

[23] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann, 2011.

[24] Categorical cross entropy loss function, Peltarion Platform. [Online]. Available: https://peltarion.com/knowledge-center /documentation/modeling-view / build -an -ai -model/loss-functions/categorical-crossentropy.

[25] N. S. Keskar et al., On large-batch training for deep learning: Generalization gap and sharp minima,2016.

[26] Forward propagation, backward propagation, and computational graphs, Dive into Deep Learning 0.17.5 documentation. [Online]. Available: https://d2l.ai/chapter_multilayer-perceptrons/backprop.html.

[27] Recurrent neural network, Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network.

[28] P. T. S and P. Shrinivas Acharya, evaluating neural networks using Bi-Directional LSTM for network IDS (intrusion detection systems) in cyber security, Global Transitions Proceedings, vol. 2, no. 2, pp. 448–454, 2021.

[29] A. Boukhalfa et al., LSTM deep learning method for network intrusion detection system, International Journal of Electrical and Computer Engineering, vol.10, no.3, pp.3315–3322, 2020.

[30] G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science, vol. 313, no. 5786, pp. 504–507, 2006.

[31] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798-1828, 2013.

[32] D. H. Ballard, Modular learning in neural networks, in Proc.8th Annu. Conf. Cognitive Science Society, 1987, pp. 101-108.

[33] A. Biswal, Top 10 Deep Learning Algorithms You Should Know in 2024, Simplilearn.com, Feb. 15, 2024. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithms.

[34] N. S. Chauhan, Introduction to Restricted Boltzmann Machines (RBMs), The AI Dream.com, Nov. 28, 2020. [Online]. Available: https://www.theaidream.com/post/introduction-to-restricted-boltzmann-machines-rbms.

[35] Ž. Vujović, Classification Model Evaluation Metrics, International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, pp. 599–606, 2021.

[36] L. Göcs and Z. C. Johanyák, Identifying Relevant Features of CSE-CIC-IDS2018 Dataset for the Development of an Intrusion Detection System, 2023.

[37] What is Dimensionality Reduction? Overview, and Popular Techniques, Simplilearn.com, Nov. 7, 2023. [Online]. Available: https://www.simplilearn.com/what- is-dimensionality-reduction-article.

[38] A. Assiri, Anomaly classification using genetic algorithm-based random forest model for network attack detection, Computers, Materials and Continua, vol. 66, no.1, pp. 767–778, 2021. [Online]. Available: https://doi.org/10.32604/cmc.2020.013813.

[39] S. Kiranyaz et al., 1D convolutional neural networks and applications: A survey, Systems Signal Processing, vol. 151, 2021.

[40] S. Guessoum et al., The Short-Term Prediction of Length of Day Using 1D, 2022.

[41] Welcome to Python.org, Python. [Online]. Avalable: https://www.python.org/.

[42] Visual Studio Code - Code Editing. Redefined, Visual Studio Code. [Online]. Available: https://code.visualstudio.com/.

[43] NumPy, [Online]. Available: https://numpy.org/.

[44] Pandas, [Online]. Available: https://pandas.pydata.org/.

[45] Keras: The Python Deep Learning API, Online]. Available: https://keras.io/.

[46] TensorFlow, [Online]. Available: https://www.tensorflow.org/.

[47] Scikit-learn, [Online]. Available: https://scikit-learn.org/stable/.

[48] S. Songma, T Sathuphan, and T. Pamutha, Optimizing Intrusion Detection Systems in Three Phases on the CSE-CIC-IDS-2018 Dataset, Computers, vol. 12, no. 12, 2023. [Online]. Available: https://doi.org/10.3390/computers12120245.

[49] K. Ren, Y. Zeng, Z. Cao et al., ID-RDRL: A deep reinforcement learning-based feature selection intrusion detection model, Sci. Rep., vol. 12, 15370, 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-19366-3.

[50] CSE-CIC-IDS2018, A collaborative project between the Communications Security Establishment (CSE); the Canadian Institute for Cybersecurity (CIC), [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html.