PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

YAHIA FARES UNIVERSITY OF MEDEA



Faculty of Technology

Master's Thesis

Department of Electrical Engineering

Specialty: Telecommunications Systems

NETWORK INTRUSION DETECTION SYSTEMS USING CONVOLUTIONAL NEURAL NETWORK

Presented by:

Sabrina Amirat

Mohamed Walid Douik

Proposed and directed by:

Dr. Hamza KHEDDAR

Promotion 2023

DEDICATION

To the man and the woman without whom I would have never been where I am today, my parents.

First and foremost, I would like to express my gratitude to Allah for guiding and helping us in accomplishing our work. feedback.

I am incredibly thankful to Dr. Hamza KHEDDAR for his invaluable guidance, advice, and insightful feedback throughout my dissertation journey.

This dissertation is dedicated to my parents, who have been my pillars of strength, my brother Ibrahim, my sister Khadija, and my brother-in-law Mohamed, who have always believed in me.

To my best friend Samah, thank you for your constant support and unwavering encouragement throughout this process. I deeply appreciate everything you have done for me.

To Almaahed International and Elite Plus Academy School, with special recognition to Dr. Ayoub Boukatem and Mr. Abdelfatah, I want to express my sincere gratitude for the outstanding education and life-changing experiences you have provided me. Your unwavering commitment to academic excellence, personalized guidance, and holistic development has significantly shaped my growth and accomplishments.

To my partner Sabrina in this work, thank you for sharing this thesis with me.

- Douik Mohamed Walid

DEDICATION

In heartfelt appreciation and profound gratitude, I extend my sincerest dedications to my beloved parents, whose unwavering support throughout my academic journey has been an invaluable source of strength and encouragement. Your dedication and hard work have taught me the value of perseverance and instilled in me a strong work ethic,

To my dearest sister Iman, your presence has been a constant pillar of support from the first day of university to the very last, sister Nesrine, You were there to hold my hand, making my early days of school a memorable experience, brother Hicham, you were and will always be my source of peace and protection,

Furthermore, we extend our sincere appreciation to Dr. Hamza KHEDDAR, whose guidance, invaluable advice, and insightful feedback have played a pivotal role in shaping our academic growth. Your unwavering commitment to our development has left an indelible mark on our educational journey, and we are forever grateful for your mentorship,

Last but not least, I would like to express my gratitude to my project partner, Walid, who has shared this remarkable endeavor with me. Through collaboration and shared experiences, you have not only contributed to this project but have also created lasting memories that we will cherish.

Thank you.

- AMIRAT Sabrina

ملخص

عملنا يهدف إلى تطوير عملية الكشف عن الهجمات باستخدام أنظمة كشف التسلل. وقد حققنا ذلك من خلال تطوير نموذج قادر على تصنيف مختلف أنواع الهجمات بدقة، ودمج تقنية CNN مع تقنية Auto-Encoder ، وتطبيق نقل التعلم على النموذج المستهدف. وقد أثبت هذا التكامل فعاليته العالية في تحسين أداء نظام الكشف عن التسلل لدينا. ومن خلال الاستفادة من تقنية حققنا،CNN نتائج لافتة للنظر، مع دقة تحقق تبلغ 99.97% وخسارة تحقق منخفضة تبلغ 0.002 . وعلاوة على ذلك، قمنا باستكشاف إمكانية نقل التعلم لزيادة تعزيز قدرات نموذجنا. في البداية، عندما طبقنا تعلم النقل دون ضبط دقيق، كانت دقة التحقق من نموذجنا المستهدف 60.41%، مصحوبة بخسارة تحقق تبلغ 0.6600 . غير أننا لاحظنا، من خلال تطبيق تقنيات الضبط الدقيق، تحسنا كبيرا في الأداء. أظهر النموذج المقترح، بعد خضوعه للتعديل الدقيق، دقة تحقق ملحوظة بلغت 99.85%، مع انخفاض إضافي في خسارة التحقق إلى 0.0068. ومن خلال التدريب المستمر على مجموعات بيانات الهجمات المختلفة، كفلنا أن يصبح نموذجنا أكثر براعة في التعرف على الهجمات المتنوعة وتصنيفها. يسمح أسلوبنا بالكشف الفعال عن الهجمات مع تقليل وقت التدريب المطلوب للنموذج. ونتيجة لذلك، نجحنا في إنشاء نظام قوي وقابل للتكيف لكشف التسلل قادر على تحديد مختلف أنواع الهجمات بدقة عالية.

ومن خلال التدريب المستمر على مجموعات بيانات الهجمات المختلفة، كفلنا أن يصبح نموذجنا أكثر براعة في التعرف على الهجمات المتنوعة وتصنيفها. يسمح أسلوبنا بالكشف الفعال عن الهجمات مع تقليل وقت التدريب المطلوب للنموذج. ونتيجة لذلك، نجحنا في إنشاء نظام قوي وقابل للتكيف لكشف التسلل قادر على تحديد مختلف أنواع الهجمات بدقة عالية.

كلمات مفتاحية :

نظام الكشف عن التسلل , CNN , دقة التحقق ,خسارة التحقق ,نقل التعلم , تقنيات الضبط الدقيق , .Auto-Encoder

ABSTRACT

Our work aimed to develop the detection of attacks using intrusion detection systems (IDSs). We achieved this by developing a model capable of accurately classifying different types of attacks, merging convolutional neural network (CNN) and auto-encoder capabilities, and applying transfer learning (TL) on the target model. This integration proved to be highly effective in enhancing the performance of our IDS. By leveraging the power of CNNs, we achieved remarkable results, with a validation accuracy of 99.97% and an impressively low validation loss of 0.0022.

Furthermore, we explored the potential of TL to further enhance our model's capabilities. Initially, when we applied TL without fine-tuning, the validation accuracy of our target model was 60.41%, accompanied by a validation loss of 0.6600. However, by implementing fine-tuning techniques, we observed a significant improvement in performance. The proposed model, after undergoing fine-tuning, exhibited a remarkable validation accuracy of 99.85%, with a further reduction in validation loss to 0.0068.

Through continuous training on various attack datasets, we ensured that our model becomes increasingly adept at recognizing and classifying diverse attacks. Our approach allows for the efficient detection of attacks while reducing the training time required for the model. As a result, we have succeeded in creating a robust and adaptable intrusion detection system capable of identifying different types of attacks with high precision.

Keywords : Auto-Encoder, CNN, Intrusion detection system, Transfer learning, Pretrained model.

RÉSUMÉ

Notre travail visait à améliore la détection des attaques à l'aide des systèmes de détection des intrusions. Nous y sommes parvenus en élaborant une modèle capable de classer avec précision différents types d'attaques, fusionnant CNN et auto-encodeur et application de Apprentissage par transfert sur le modèle cible. Cette intégration s'est avérée très efficace dans améliorer les performances de notre système de détection des intrusions. En exploitant la puissance des CNN, nous avons obtenu des résultats remarquables avec une précision de validation de 99,97% et une perte de validation impressionnante de 0,0022.

De plus, nous avons exploré le potentiel de Apprentissage par transfert pour améliorer davantage les capacités de notre modèle. Initialement, lorsque nous avons appliqué Apprentissage par transfert sans réglage fin, la précision de validation de notre modèle cible était de 60,41%, accompagné d'une perte de validation de 0,6600. Toutefois, en mettant en œuvre un réglage fin techniques, nous avons observé une amélioration significative de la performance. Le modèle proposé, après soumis à un réglage fin, a montré une précision de validation remarquable de 99,85%, avec une réduction de la perte de validation à 0,0068.

Grâce à une formation continue sur divers ensembles de données d'attaque, nous avons fait en sorte que notre modèle devienne de plus en plus à même de reconnaître et de classer les diverses attaques. Notre approche permet de détection efficace des attaques tout en réduisant le temps d'entraînement requis pour le modèle. En conséquence, nous avons réussi à créer un système de détection des intrusions robuste et adaptable capable de identifier les différents types d'attaques avec une grande précision.

Mots-clés : Auto-Encoder, CNN, Système de détection des intrusions, Apprentissage par transfert, Préformé modèle.

ACKNOWLEDGMENT

First and foremost, we want to express our gratitude to Allah, for the opportunities He has provided for us, who gave us the perseverance and strength to complete this work.

We would like to express our great appreciation to our supervisor Dr. Hamza Kheddar for his countless help, constructive suggestion and guidance in every step of our research.

We also wish to express our deepest gratitude to the members of the committee for spending their precious time on reading our thesis and giving us encouragement and constructive comments..

We would like to thank all the teachers who collaborated in our training since the beginning of our study, we also thank all Technology Faculty members at Yahia Fares Medea University.

Finally, and most importantly, we must thank our dearest family for giving us the support and encouragement that only a family can give.

TABLE OF CONTENTS

Abstract **i**

Acknowledgment **iv**

Table of contents **vii**

List of Figures **ix**

List of tables **x**

Abbreviations list **xi**

I General Introduction **1**

I.1 Introduction 1

II State-of-the-art of machine learning and deep learning **2**

II.1 Introduction 2

II.2 Machine learning 2

II.2.1 Model building techniques 3

II.2.2 Strategy of ML 4

II.2.3 Model post-building techniques 5

II.2.4 Types of ML 5

II.2.4.1 Supervised learning 6

II.2.4.2 Unsupervised learning 9

II.2.4.3 Semi-supervised Learning 11

II.2.4.4 Reinforcement learning 11

II.2.5 Machine learning algorithms 12
II.2.5.1 Decision tree 12
II.2.5.2 Support vector machine (SVM) 13
II.2.5.3 Machine learning applications 14
II.2.5.4 Advantages and disadvantages of ML 14
II.3 Deep learning 15
II.3.1 Working principle 15
II.3.2 Difference between DL and ML 15
II.3.3 DL techniques 16
II.3.3.1 Artificial Neural Network 16
II.3.3.2 Deep neural networks 18
II.3.3.3 Recurrent neural networks 19
II.3.3.4 Long short-term memory 19
II.3.3.5 Convolutional neural networks 20
II.3.4 Advantages and disadvantages of DL 24
II.4 Conclusion 25

III Network intrusion detection systems 26
III.1 Introduction 26
III.2 Cyber Attacks 27
III.3 Intrusion detection systems 28
III.3.1 Intrusion detection in the IoT 29
III.3.2 Intrusion detection for intelligence internet of vehicles 29
III.4 Types of intrusion detection system 30
III.4.1 Network-based IDS 30
III.4.2 Host-based IDS 30
III.4.3 Application protocol-based IDS 31
III.4.4 Protocol-based IDS 31
III.5 IDS methods 32
III.5.1 Signature-based IDSs 32
III.5.2 Anomaly-based IDS 33
III.5.3 Specification based IDS 33
III.5.4 Hybrid IDS 34
III.5.4.1 Integration of misuse and anomaly intrusion detection 34

III.5.4.2 Integration of specification and misuse detection 34
III.6 Firewalls and IDS 34
III.7 IDS and intrusion prevention system 35
III.8 Deploying IDSs 36
III.8.1 Deploying network-based IDSs 36
III.8.2 Deploying host-based IDSs 37
III.8.3 Alarm strategies 37
III.8.4 Response options for IDSs 38
III.8.4.1 Active responses 38
III.8.4.2 Passive responses 38
III.8.5 Conclusion 39

IV implementation of the proposed CNN-based IDS method 40
IV.1 Introduction 40
IV.2 Evaluation metrics 40
IV.3 Trained attacks for the proposed model 41
IV.4 Proposed CNN architecture 45
IV.5 Transfer learning to other model 50
IV.5.1 Trained attacks 51
IV.5.2 Proposed architecture for the target model 53
IV.5.3 Fine-tuning 56
IV.6 Conclusion 60

V General conclusion 61

LIST OF FIGURES

II.1 Distinctions between artificial intelligence (AI) , machine learning (ML) , and deep learning (DL). 3
II.2 ML's approach. 4
II.3 ML techniques. 6
II.4 Different between regression and classification. 7
II.5 Unsupervised learning principle. 10
II.6 Principle of association in unsupervised learning. 10
II.7 RL working principle. 12
II.8 Decision tree principle. 13
II.9 The principle of linear support vector machine (SVM) classification. 14
II.10 Difference between DL and ML. 16
II.11 ANNs' approach. 17
II.12 Deep neural networks basic architecture. 18
II.13 LSTM architecture. 19
II.14 Convolutional neural networks. 20
II.15 Working principle of convolution layer (CL), the mathematical operation of convolution is performed between the input image and a filter of a particular size. 21
II.16 An example of an input matrix after the rectified linear activation unit (ReLU) operation. 21
II.17 Types of pooling layers. 22
II.18 Batch normalization. 23
II.19 Dropout principle. 24

III.1 Nomenclature of cyber-attacks. 28

III.2 IDS on an enterprise network. 29
III.3 Comparison between IDS and intrusion prevention system (IPS). 36

IV.1 An example of SSH brute force attack. 42
IV.2 An example of dos GoldenEye attack. 43
IV.3 Attacks used in our source model. 44
IV.4 Intrusion attack type distribution. 45
IV.5 Our proposed source model. 46
IV.6 The obtained accuracy and validation accuracy of the proposed model. 48
IV.7 Loss and validation loss of the proposed model. 49
IV.8 Confusion matrix of the proposed model. 50
IV.9 The difference between DoS attack and DDoS attack. 52
IV.10 Attacks used in the target model. 53
IV.11 Intrusion attacks distribution for the target model. 53
IV.12 Target model summary before using fine-tuning. 54
IV.13 Accuracy and validation accuracy before using fine-tuning. 55
IV.14 Loss and validation loss before using fine-tuning. 55
IV.15 Target model summary after using fine-tuning. 56
IV.16 Accuracy and validation accuracy of target model after using fine-tuning. 57
IV.17 Loss and validation loss after using fine-tuning. 58
IV.18 Confusion matrix for the proposed target after fine-tuning. 59

LIST OF TABLES

IV.1 Evaluation metrics. 41
IV.2 Confusion matrix and classification results for the model test. 50
IV.3 Confusion matrix and classification results for the model test. 59

ABBREVIATIONS LIST

AI	artificial intelligence
AIDS	anomaly-based intrusion detection system
ANN	artificial neural network
APIDS	application protocol-based intrusion detection system
BN	batch normalization
CARUM	center for atmospheric research at the University of Massachusetts
CL	convolution layer
CNN	convolutional neural network
DDoS	distributed denial of service
DL	deep learning
DNN	deep neural network
DoS	denial of service
FAR	false alarm rate
FCL	fully connected layer
FN	false negative
FP	false positive
FTP	file transfer protocol
GD	gradient descent
HIDS	host intrusion detection system

HTTP hypertext transfer protocol

HTTPS hypertext transfer protocol Secure

IDS intrusion detection system

IoT internet of things

IoV internet of vehicles

IPS intrusion prevention system

LR logistic regression

LSTM long short-term memory

ML machine learning

MLP multilayer perceptron

NIDS network intrusion detection system

PIDS protocol-based intrusion detection system

ReLU rectified linear activation unit

RL reinforcement learning

RNN recurrent neural networks

SIDS signature-based intrusion detection systems

SIEM security information and event management

SL supervised learning

SPAN switched Port analyzer

SSH secure shell protocol

SSL semi-supervised learning

SVM support vector machine

TAP test access point

TL transfer learning

TN true negative

TP true positive

UDP user datagram protocol

UL unsupervised learning

CHAPTER I

GENERAL INTRODUCTION

I.1 Introduction

In today's interconnected world, networks play a crucial role in facilitating communication, sharing information, and enabling various online services. However, this increased reliance on networks has also brought about significant security challenges. One of the most pressing concerns is the prevalence of attacks on these networks, which can lead to data breaches, unauthorized access, and disruption of services. Modern attacks often leverage encrypted communication channels, rendering traditional firewall techniques ineffective in detecting malicious activities concealed within encrypted traffic. Additionally, firewalls primarily operate at the network, transport, and application layers, which limits their ability to detect attacks at higher layers of the network stack. This deficiency becomes particularly problematic when attacks originate from within the network perimeter, as firewalls are unable to discern potentially harmful activities from legitimate ones, they are no longer sufficient in combating the sophisticated and evolving nature of attacks. In this context, we will discuss the potential of IDS as a more comprehensive solution and that it offers a more proactive and comprehensive approach to network security. They complement firewalls by providing an additional layer of defense, detecting and alerting administrators about potential threats that may bypass traditional security measures. By continuously monitoring network activity, IDS can provide valuable insights into the nature and extent of attacks, enabling swift response and mitigation efforts. This thesis is structured as fellows, on chapter 2, we scrutinized the fundamental ideas behind machine learning (ML) and deep learning(DL), in chapter 3, we immersed ourselves in the study of IDS and its techniques. Furthermore, in Chapter 4, we will outline our project's aim to develop a model capable of accurately classifying distinct types of attacks, utilizing CNNs and autoencoders, along with the power of TL. This project holds the potential to significantly improve network security by enabling proactive identification and mitigation of threats, ultimately safeguarding the integrity and confidentiality of networked systems.

CHAPTER II

STATE-OF-THE-ART OF MACHINE LEARNING AND DEEP LEARNING

II.1 Introduction

The term artificial intelligence (AI) refers to the construction of computer programs that can perform tasks that are currently done more effectively by humans. The field of AI focuses on the creation of intelligent machines that can perform tasks that typically require human intelligence [1]. AI is an interdisciplinary discipline that encompasses various methodologies and techniques, and it can be described as a computer system that can carry out tasks that involve high-level mental processes like perceptual learning, memory organization, and critical reasoning. AI has two key components: the "artificial" side, which involves the use of computers or complex electrical processes, and the "intelligent" side, which is concerned with imitating human behavior [2]. Figure II.1 illustrates the distinctions between AI , ML , and DL.

Overall, AI is a rapidly evolving field with numerous applications, including natural language processing, image recognition, and decision-making. Its potential impact on society is vast, and it continues to attract significant research attention and investment.

II.2 Machine learning

ML involves the use of data examples or prior knowledge to program computers to optimize a performance criterion. A model is defined with a few parameters, and a software program is run to optimize these parameters gradually through training data or prior knowledge while learning. ML models can be either descriptive, to learn from data, or predictive, to assist in making future decisions. In order to draw conclusions from a sample, ML utilizes the science of statistics to develop mathematical models. Computer science plays two distinct roles in ML. First, the vast amount of data used for learning must be stored and processed to create efficient algorithms

for optimizing the model. Second, once the model has been learned, the representation and algorithmic solution for inference must also be effective. In some cases, the effectiveness of the learning or inference algorithm, specifically its space and time complexity, maybe just as crucial as the precision of its prediction [3].

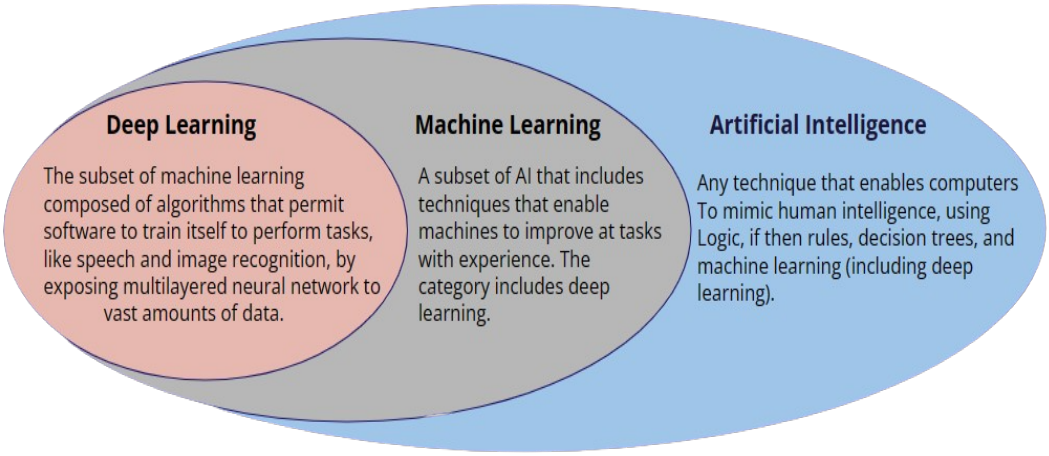


Figure II.1: Distinctions between AI , ML , and DL.

II.2.1 Model building techniques

ML models are often referred to as "black boxes" due to their incomprehensibility, which can hinder their application in critical domains such as self-driving vehicles and financial investing. To overcome this limitation, model developers can now use visual analytics approaches to generate transparent and interpretable models by exploring ways to visualize the internal workings of ML models.

To create effective models, model developers require a strong understanding of the models and must identify issues with the training process when the training fails or the model underperforms. As considerable effort is invested in enhancing model performance throughout the model construction process, there is a need for better model steering.

To address these demands, researchers have developed various visual analytics techniques, to enhance model understanding, diagnosis, and steering ,The goal of model understanding approaches is to graphically describe how a model works, including how parameter changes affect the model and why a certain output results from a particular input. By interactive study of the training process, model diagnosis approaches aim to identify flaws in model training. The main goal of model steering techniques is to actively enhance model performance [4]. Data processing and feature engineering are two important steps that must be taken before creating a model. They are essential because, according to actual experience, ML models perform worse when given low-quality data and features [5] [6]. Missing values, outliers, and noise in instances and their labels are examples of poor data quality. Issues with feature quality include redundant features, unnec-

essary features, and more. While dealing with these problems manually takes time, automation techniques might not work well. As a result, a number of visual analytics strategies have been created to lessen the work required from specialists while also enhancing the effectiveness of automatic methods for providing high-quality data and features [7].

II.2.2 Strategy of ML

With the use of ML, a system may create prediction models from prior data and make predictions in response to new input. The precision of these predictions relies on the amount of data utilized; improved predictions are produced by using more data. We can feed data into generic algorithms and allow ML build a model based on that data to provide predictions instead of manually coding a solution to a complicated problem. ML has changed our approach to problem-solving. The block diagram in Figure II.2 illustrates the operation of a ML algorithm [8].

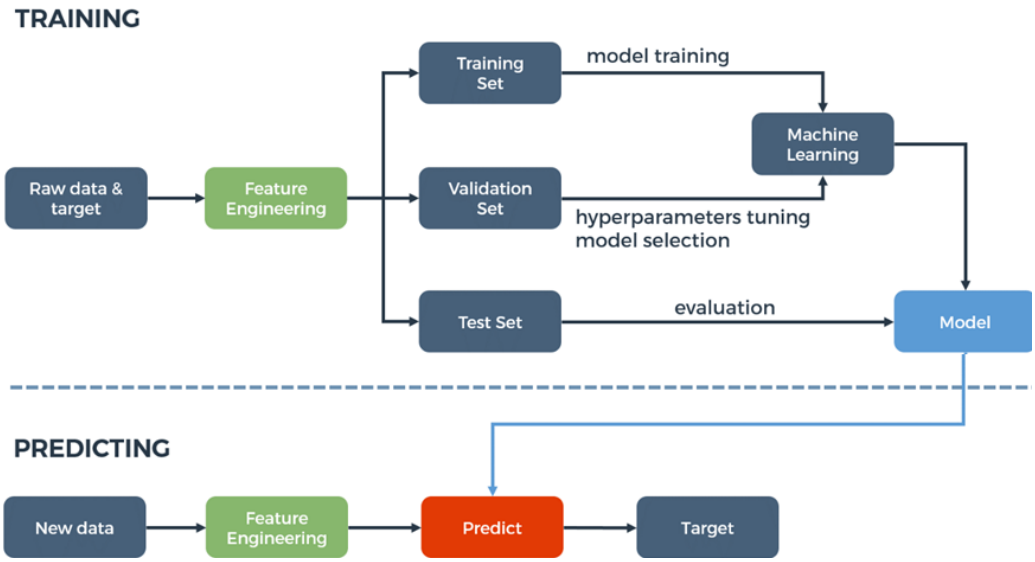


Figure II.2: ML's approach.

- Feature engineering: The pre-processing stage of ML, known as feature engineering, involves the extraction of features from raw data. The accuracy of the model for unknown data is increased since it aids in better illustrating an underlying issue to prediction models. The feature engineering process chooses the most practical predictor variables for the model, which is made up of predictor variables, an outcome variable [8].

- Model Training: In the data science development lifecycle, the model training phase is where practitioners attempt to match the ideal weights and bias to a ML algorithm in order to minimize a loss function throughout the prediction range [9].

- Hyperparameters: In ML, hyperparameters are parameters that the user directly defines to regulate the learning process and enhance the model's learning process, and their values are established prior to the model learning process, such as Batch Size, Train-test split

ratio, Number of Epochs, Branches in Decision Tree, Learning rate for training a neural network..etc [8].

- Evaluation: The process of evaluating a model involves using certain metrics to assess its performance. There are many metrics like Accuracy, Precision, Recall, F1 score, Area under Curve, Confusion Matrix, and Mean Square Error.. etc [10].
- Prediction: The output of an algorithm that has been trained on previous data and then applied to new data to estimate the likelihood of a specific result is referred to as a prediction [11].

II.2.3 Model post-building techniques

To establish reliability in the outcomes produced by a ML model, it is important to help users, such as domain experts, understand the model's output in an intuitive manner. To accomplish this, various visual analytics techniques have been developed to analyze the model's output. These techniques are typically geared towards model consumers rather than model developers, in contrast to methods used for understanding the model during its construction. Therefore, these techniques focus on presenting and exploring the model's output in an easily understandable manner, rather than revealing its internal workings

We classify the corresponding techniques in accordance with the types of data examined because these methods are typically data-driven. In visual design, the temporal property of data is crucial. Hence, we divide approaches into two categories: those that comprehend static data examination outputs and those that comprehend dynamic data examination outcomes. In most cases, a visual analytics system for comprehending the outcomes of static data analysis scrutinizes all model output as a huge collection and Examines the static structure. In the case of dynamic data, the system concentrates on revealing the data's evolution through time, which is determined by the analysis model, in addition to helping users understand the analysis results at each time point [4].

II.2.4 Types of ML

Because there are numerous types of ML systems, it is beneficial to classify them according to specific fundamental criteria. It can be categorized based on [12]:

- Whether training/labels are monitored by people or not.
- Whether the predicted model is created by identifying patterns in training data or by comparing new data points with previous ones, which is known as instance-based learning versus model-based learning.

Based on the aforementioned criteria, several types of ML emerges and they are detailed in the upcoming subsections as illustrated in Figure II.3.

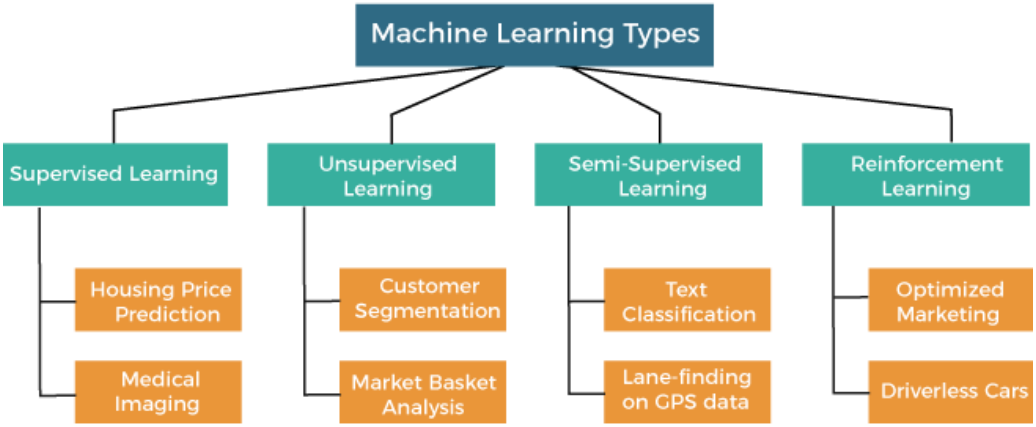


Figure II.3: ML techniques.

II.2.4.1 Supervised learning

The supervised learning (SL) is a type of ML that utilizes labeled data to train the algorithm. Although precise data labeling is required, it is highly effective when used in the appropriate circumstances. In this type of learning, the algorithm typically requires a substantial training dataset to develop a robust understanding of the problem, the proposed solution, and the relevant data points. The properties of the training dataset are similar to those of the final (test) dataset, and it provides the algorithm with the labeled inputs required to solve the problem. The algorithm looks for correlations between the provided parameters to establish a cause-and-effect relationship between the variables in the dataset. After training, the algorithm understands the operation of the data and the connection between input and output. The final dataset is then used to test the solution, and the algorithm continues to learn from it in the same way it learned from the training dataset, improving over time by discovering new correlations and patterns [13]. The two main steps of SL are as follows [1]:

1) Training: the learning stage in which a classification model is created by looking at the available data, also known as the training dataset.

2) Testing: The model created during the training phase is applied to categorize newly discovered instances.

In general, a machine can learn a relation $f : x \rightarrow \mathrm{y}$ that links x to y by studying millions of examples of associations $x \rightarrow y$. The goal of SL is to create a model (classifier) that can accurately categorize new examples [2].

There are two applications of SL, which are *classification* and *regression*. In ML, regression techniques focus on ***approximating*** the mapping function f between input variables x and output variables that are numerical or continuous y, whereas classification algorithms aim to ***estimate*** the mapping function f between input variables x and output variables that are categorical or discrete y [14]. In other words, the output of classification is presented in categories, whereas regression makes decisions based on prediction and considers the values of desired outcomes in

the training set [15]. Figure II.4 depicts the difference between the two mentioned approaches.

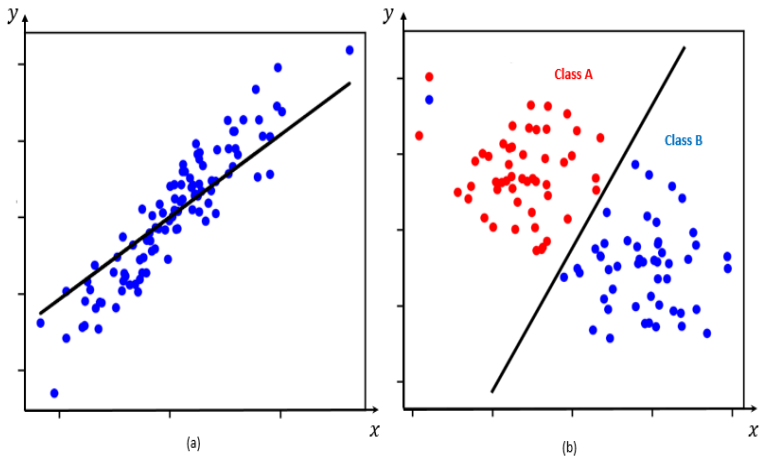


Figure II.4: Different between regression and classification.

- **Regression:** There are various types of regression, each with its own significance in specific contexts (Figure II.4 (a)). In this context, we will discuss the important forms of regression, which include [15]:

  - **Linear regression:** Aims to predict one value based on other values, where having more input data can increase the model's accuracy and realism [16]. This is typically achieved through the formula:

$$y = w_1.x + w_0 \tag{II.1}$$

    where w_0 and w_1 are values to be learned, x is the input, and y is the output. To evaluate the accuracy of a ML model, the cost function is utilized as a parameter to estimate how well the model will perform in real-world scenarios [17]. The cost function calculates the performance of a ML model given a dataset referred to as the cost function. It quantifies the difference between expected and predicted values and represents this difference as a single real number. The construction of the cost function can vary depending on the problem being addressed. Its primary objective is either to *minimize* the returned value, known as cost, loss, or error, by finding the optimal values for model parameters or to *maximize* the returned value, known as a reward, by finding the values of model parameters that generate the largest number. For example, for the algorithms that use gradient descent (GD) to optimize model parameters, the cost function must be differentiable [18].

$$J(\theta_0, \theta_1) = \frac{1}{2.m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \tag{II.2}$$

$$h_\theta(x) = \theta_0 + \theta_1.x \tag{II.3}$$

Where, θ_0, θ_1 are parameters, $h_\theta(x)$ is hypothesis.

The GD is an optimization algorithm that is used for optimizing the cost function or error in the model. It enables the models to take the gradient or direction to reduce the errors by reaching to least possible error [19].

$$\theta_j = \theta_{j,k} - \alpha.\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_j} \qquad j = 1 \quad \text{and} \quad j = 0 \tag{II.4}$$

α: A hyperparameter, also called the learning rate, allows the fine-tuning of the descent process. In particular, with an appropriate choice of it, we can escape the convergence to a local minimum and descend towards a global minimum instead.

– **Polynomial regression:** Is a ML technique that models the relationship between dependent and independent variables through the use of *polynomial* equations with a degree of h and a set of coefficients b. This method allows for the capture of curvilinear interactions between the variables, and the resulting model takes the form of a polynomial equation. The polynomial regression model can be expressed as [20]:

$$y = b_0 + b_1.X + b_2.X^2 + \cdots + b_hX^h \tag{II.5}$$

- **Classification:** Is the process of finding a function that can classify a dataset based on multiple parameters. A computer program is trained on a training dataset and then, using that training, it separates the data into various groups. The objective of the classification algorithm is to identify the mapping function that transforms the input x into a discrete output y. An excellent example of a classification problem is email spam detection, where the model distinguishes between spam and legitimate emails based on various factors using training data from millions of emails. If an incoming email is identified as spam, it will be placed in the spam folder. The following figure (Figure II.4 (b)) can help to better understand classification methods. The figure displays two classes, *class A* and *class B*, which share similarities with each other and differ from other classes [21].

  There are two categories of classification: (i) Binary classifier, which is used when there are only two possible solutions to a classification issue, such as Yes or No, Male or Female, Spam or Not Spam, etc. (ii) Multi-class classifier, which is known as multi-class if there are more than two possible outcomes, for example, classifications of types of crops, music, etc.

- **Logistic regression :** The logistic regression (LR) is a SL classification technique that can be used to predict the probability of a target variable. This approach is particularly well-suited for situations where the dependent variable is dichotomous, meaning there are only two possible classes. In LR, the dependent variable is represented as a binary variable, with data recorded as either 1 (representing success/yes) or 0 (representing failure/no). This allows the model to estimate the probability of the outcome based on the values of the predictor variables. LR is one of the most basic ML algorithms and can be applied to a variety of classification problems, including spam detection, diabetes prediction, cancer detection, and many others [22].

However, it is also possible to use LR for predicting two additional types of target variables, leading to three different types of LR :

- **Binary or Binomial:** In this type of LR, the dependent variable only has two categories or types, such as win or loss, yes or no, or success or failure.
- **Multinomial:** This type of LR allows for three or more unordered categories or types for the dependent variable, without any quantitative significance. For example, the dependent variable may have categories like "Type A," "Type B," or "Type C."
- **Ordinal:** In this type of LR, the dependent variable can have three or more ordered categories or types that are statistically significant. For instance, the variable may have categories like "bad" or "good," "very good," or "excellent," with scores of 0, 1, 2, or 3 [23].

- **Advantages and disadvantages of SL:** Using SL, the model is capable of predicting outcomes based on past experiences. This method provides a high degree of certainty when it comes to object classification. In real-world scenarios such as spam filtering and fraud detection, SL models are employed to solve these problems. On the other hand, its models may not be suitable for challenging tasks. If the test data and the training dataset are dissimilar, SL will be unable to provide accurate predictions. The training process may be time-consuming. Adequate information about object classes is necessary for SL to be effective [24].

II.2.4.2 Unsupervised learning

As its name indicates, is a ML approach where models are not supervised by training datasets. Instead of relying on pre-determined output data, models utilize the given data to reveal concealed patterns and insights, close to how the human brain learns new information. unsupervised learning (UL) cannot directly address classification or regression problems because it lacks matching output data. Its goal is to expose the fundamental structure of a dataset, group the data into categories based on similarities (clustering), and present the dataset in a compact form [12].

For instance, consider a scenario where an UL algorithm is given an input dataset consisting of photos of various dog and cat breeds. The algorithm is not trained on this particular dataset and thus has no prior knowledge of its characteristics. The UL algorithm's task is to let the visual features of the pictures speak for themselves. It will use clustering techniques to group the collection of pictures based on their visual similarities.

- **Principle of UL:** In UL, unlabeled input data is utilized, which means that there is no information regarding its category or any associated outputs. The ML model is then trained using this unlabeled input data. Firstly, the raw data is examined to uncover any hidden patterns before implementing suitable algorithms, such as k-means clustering, decision trees, and so on. After applying the appropriate method, the algorithm partitions the data items into groups based on the similarities and dissimilarities between the objects. The below illustration (Figure II.5) depicts the functioning of UL.

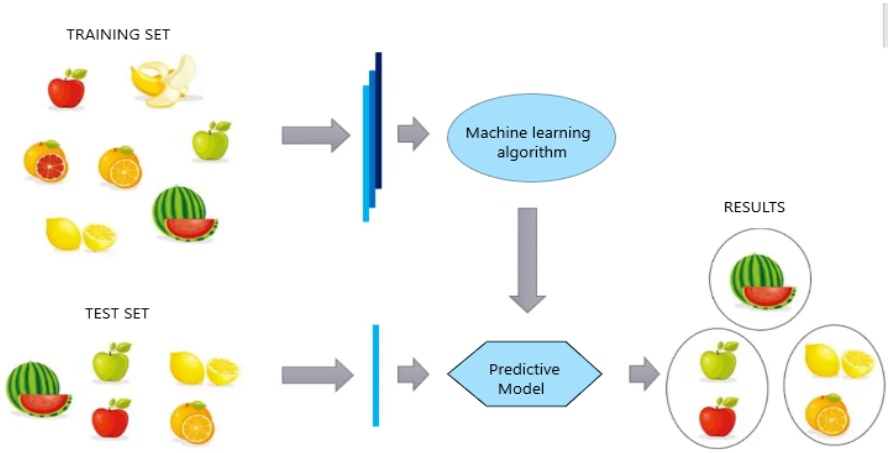


Figure II.5: Unsupervised learning principle.

- **Types of UL:** The UL algorithm may be divided into two different categories of issues :

  1) **Clustering:** Is a way of organizing items into clusters so that the objects that have the most similarities stay in one group and share little to none with the objects in another group. The data items are classified based on the existence or lack of similarities discovered by cluster analysis (Figure II.5).

  2) **Association:** Is used to uncover links between variables in a sizable database using UL techniques. It establishes the group of items that co-occur in the collection. Marketing strategy is more successful because of the association rule. Customers who buy X (let's say, bread) also frequently buy Y (let's say, butter or Rice). Market basket analysis is an illustration of an association rule in action (FigureII.6) [25].

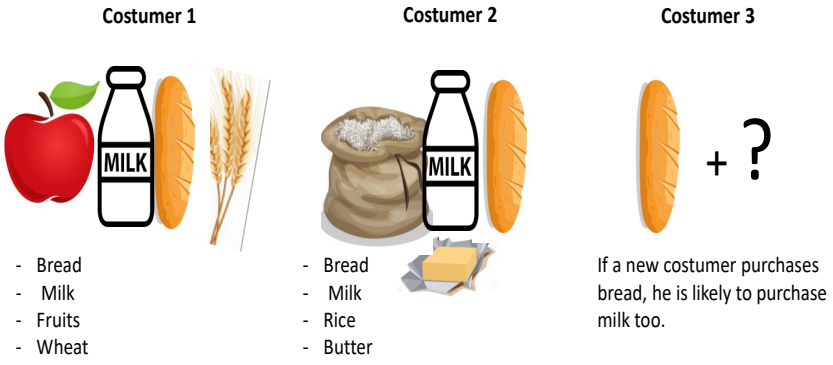


Figure II.6: Principle of association in unsupervised learning.

3) K**-Means :** It is a technique for UL that addresses clustering issues. Data sets are divided into a certain number of clusters (let's call that number K) in such a way that each cluster's data points are homogeneous and distinct from those in the other clusters. The following steps are to form clusters with K-means :

   1) For each cluster, the K-means algorithm selects K centroids, or points.

2) With the nearest centroids, or K clusters, each data point forms a cluster.

3) From the existing cluster members, it now generates new centroids.

4) Each data point's closest distance is determined by using these new centroids. So this process is repeated until the centroids do not change [26].

- **Advantages and disadvantages of UL:** The UL is utilized for more intricate (complicated) tasks in comparison to SL since it lacks labeled input data, Additionally, UL is advantageous because obtaining unlabeled data is simpler and cost-less than obtaining labeled data, On the other hand, Due to the lack of comparable output, ULis inherently more challenging than SL. Furthermore, as the input data is not labeled and the algorithms do not know the precise output in advance, the outcome of the UL method may be less accurate.

II.2.4.3 Semi-supervised Learning

With this kind of learning, both classified and unclassified data are provided. An acceptable model for the categorization of data is created using this mix of labeled and unlabeled data. Unlabeled data is typically abundant and labeled data is typically sparse (as was previously described in the explanation of UL). The goal of semi-supervised learning (SSL) classification is to develop a model that is more accurate at classifying future test data than the model developed just from labeled data. SSL is a method comparable to how we learn. The provision of a child with: (i) Data produced by the environment that is not labeled. A child's immediate environment is packed with unlabeled information at first. (ii) Data with labels provided by the supervisor. As an example, a father can point to an object and say its name to teach his children what the label (name) on it is [27].

Computer programs that recognize speech can be improved through SSL. Moreover, algorithms that are SSL can gain knowledge from both labeled and unlabeled data. Additionally, algorithms for SSL learning can combine overfitting tendencies. In contrast, for difficult problems, SSL learning is inappropriate. Besides, for color-coded pictures or document categorization, SSL learning does not perform well.

II.2.4.4 Reinforcement learning

The process of learning through reinforcement involves an agent (actor) that can observe its surroundings, select and execute actions, and receive rewards or punishments, denoted as negative rewards. The agent's objective is to identify the best policy, or optimal strategy, for maximizing rewards over time. The policy determines the action the agent should take in a given situation [12]. Figure II.7 illustrates the principle of reinforcement learning (RL).

Overall, RL is different from supervised, unsupervised, and SSL in that it involves an agent learning to make decisions based on feedback from the environment, rather than learning from labeled or unlabeled data. For instance, many robots use RL to teach themselves how to walk [12].

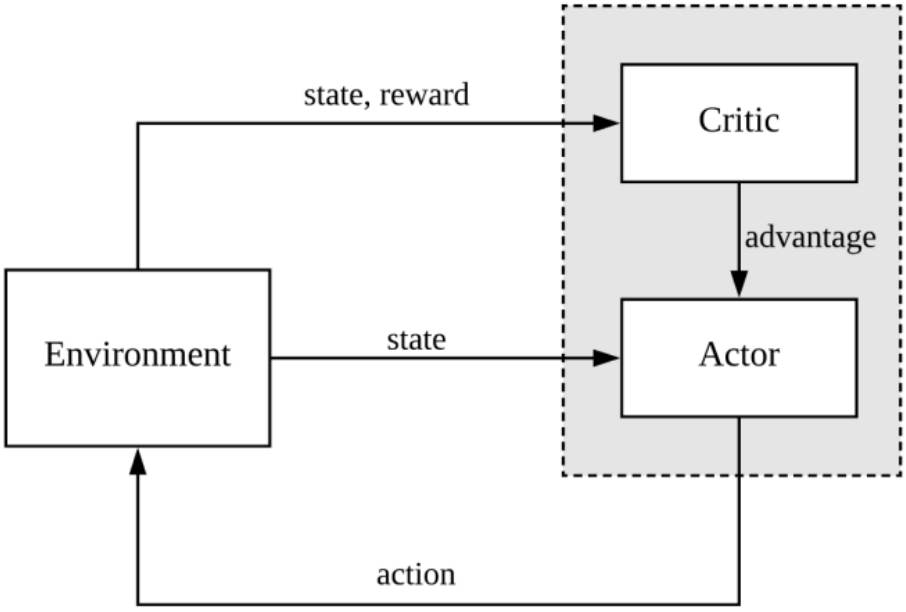


Figure II.7: RL working principle.

II.2.5 Machine learning algorithms

ML uses a variety of techniques to solve data issues, data scientists emphasize that there isn't a one-size-fits-all algorithm that is suitable for every situation. The selection of the appropriate algorithm is influenced by various factors, such as the specific problem being addressed, the number of variables, the optimal model for the task, and other relevant considerations. The following is a concise overview of some of the most prevalent ML algorithms [28].

II.2.5.1 Decision tree

Decision trees are a type of supervised ML models that are suitable for classification tasks instead of regression. These models can take a vector of attributes as input and produce a decision as output. The input and output may be either continuous or discrete.

The decision tree in Figure II.8 consists of nodes, branches, and leaves. Each internal node in the tree corresponds to a test of an attribute's value, and the branches that emerge from the node represent the potential values of the attribute. By evaluating the results of these tests, the decision tree makes decisions. The class of the target variable is determined by the leaf node in which the observation ends up after the testing sequence [29].

To apply decision trees to classification problems, two main steps are required [2]:

The first step: is to construct a decision tree by utilizing a learning algorithm to analyze the training data and determine the optimal sequence of attribute tests that partition the data into distinct classes.

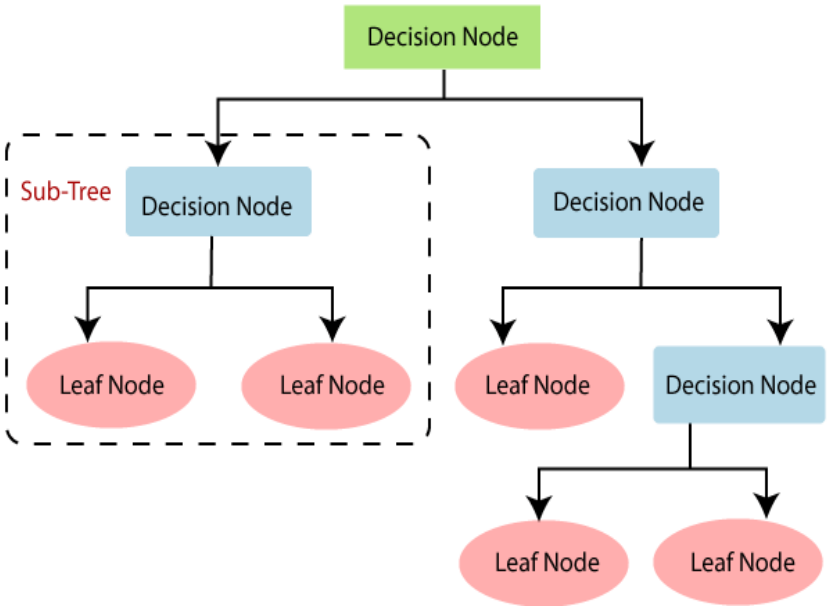


Figure II.8: Decision tree principle.

The second step: is to use the decision tree to classify new instances. This involves traversing the tree based on the values of the attributes of the instance, and eventually arriving at a leaf node that corresponds to a specific class. This process is referred to as inference or classification.

II.2.5.2 Support vector machine (SVM)

One of the widely used SL algorithms is SVM, which is primarily used for solving classification problems in ML. Although SVM can be used for regression problems as well, its main application is in classification tasks [30].

The primary goal of the SVM algorithm is to find the optimal decision boundary or hyperplane that can effectively divide an n-dimensional space into different classes, enabling us to efficiently classify new data points in the future. The hyperplane is determined by identifying the extreme points and vectors that aid in its creation. The SVM approach is based on support vectors, which represent the extreme instances that are closest to the decision boundary [31]. The Figure II.9 illustrates how a hyperplane is used as a decision boundary to classify two distinct groups [32]:

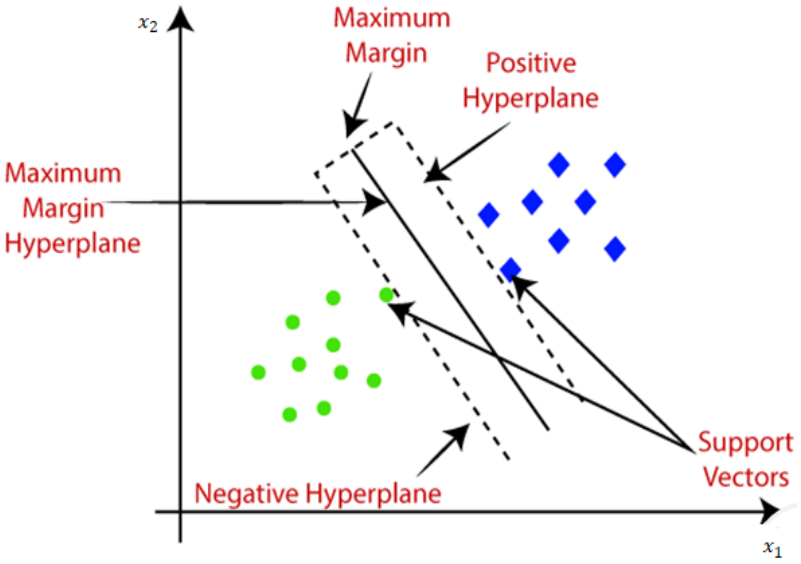


Figure II.9: The principle of linear SVM classification.

II.2.5.3 Machine learning applications

Due to its involvement in various fields, ML has become an essential part of our everyday lives. it allows to Distinguish and identify objects (faces, images, letters, words, sounds and music, etc). Moreover, it assists in the search for treatments for fatal illnesses and Creates suggestions and judgments using models of the human brain's neural network. Also ML helps in the construction of robots that can carry out a variety of sensitive or difficult tasks in challenging situations for humans.

II.2.5.4 Advantages and disadvantages of ML

ML Easily recognizes trends and patterns, That is to say it's able to look at vast amounts of data and identify certain trends and patterns that people would miss. Besides, ML eliminates the requirement for human intervention at each level of the project. As a result of granting computers the capacity to learn, they are now also empowered to develop better algorithms and make predictions. Furthermore, processing of multidimensional and heterogeneous data, to be more precise ML methods enable the processing of multidimensional and heterogeneous data in dynamic or unpredictable situations. Despite this, ML requires adequate time for algorithms to learn and develop to the point where they can accomplish their objective with a high degree of accuracy and relevance. In addition, accurately interpreting the results that the algorithms produce is another significant problem. The algorithms you choose should be carefully selected to match your needs. Besides ML is autonomous, errors happen frequently [2].

II.3 Deep learning

The field of DL is a branch of ML and AI that emulates human learning of specific subjects, such as features. DL is of great importance in data science, which encompasses statistics and predictive modeling. Data scientists are responsible for handling enormous amounts of data, and DL makes this process faster and more efficient. At its most fundamental level, DL can be considered a means of automating predictive analytics. Unlike traditional linear ML algorithms, DL algorithms are arranged in a hierarchy of increasing complexity and abstraction [33].

II.3.1 Working principle

Each algorithm within the hierarchy executes a nonlinear transformation on its input and generates a statistical model based on its learning. This process iterates until the output achieves a satisfactory level of precision. The term "deep" refers to the multiple layers of processing that data must undergo. One of the advantages of DL is that it autonomously generates the feature set without the need for human intervention and without any supervision. In addition, UL using DL is typically more precise, as well as faster.

The computer program is often provided with training data, such as a collection of photographs that have been tagged with meta-tags indicating whether they contain dogs or not. Based on the information gathered from the training data, the program builds a feature set for the dogs and a predictive model. With each iteration, the predictive model becomes increasingly sophisticated and precise, improving its complexity and accuracy. This enables the program to learn and adapt to new information and perform more advanced tasks. The training data serves as a basis for the computer to generate predictions and identify patterns, allowing it to recognize dogs and distinguish them from other animals in new, untagged images [34].

deep neural networks (DNNs) are commonly used in DL models, as many of these methods utilize neural network topologies. The term "deep" is used to denote the number of hidden layers in the network. While traditional neural networks usually have only two or three hidden layers, deep networks can have up to 150 layers [35].

Because of their ability to process large amounts of data and identify complex patterns, DNNs have become increasingly popular in various fields, including computer vision, natural language processing, and speech recognition. These networks are used for tasks such as image classification, object detection, and language translation, among others.

II.3.2 Difference between DL and ML

As previously mentioned, DL is a subset of ML that is utilized for extracting information from large datasets stored in computers. However, DL is often prone to negative overtraining effects, which can be remedied by human intervention to correct mistakes made by computers. Figure II.10 illustrates the main differences between ML and DL classification techniques.

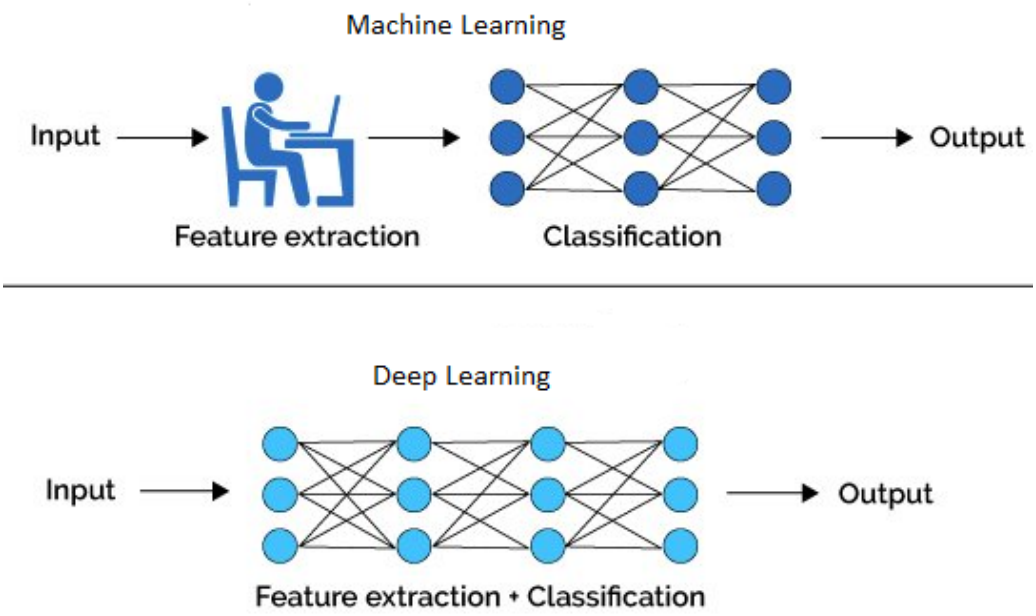


Figure II.10: Difference between DL and ML.

In contrast to DL, ML involves the manual identification of features and the provision of training data to computers, allowing them to make optimal decisions. According to [36], the following are some of the differences between DL and ML. Both are branches of extracting information from big data arrays stored in computers. However, DL requires more time and more complex mathematical operations than ML. DL also requires a large amount of data, while ML only needs a small amount of data initially, which later becomes unnecessary. While ML works with structured data, DL deals with unstructured data, and the output of DL includes information, pictures, audio, video, and even text, whereas the output of ML is only a quantity of information. Unlike ML, DL requires powerful hardware to process data efficiently. Furthermore, DL independently generates new features, while ML requires users to precisely identify the features. Figure II.10 illustrates the main differences between ML and DL classification techniques.

II.3.3 DL techniques

DL has a wide variety of techniques that have been used to address issues. In the following subsections, we detail some of these strategies.

II.3.3.1 Artificial Neural Network

artificial neural networks (ANNs) are used in AI to simulate the network of neurons that make up the human brain, giving computers the ability to comprehend information and make decisions in a way similar to that of a person. Computers are programmed to function exactly like a network of linked brain cells to create an ANN. Understanding the components of a neural network is necessary to comprehend the idea of the architecture of an ANN. A vast number of artificial

neurons, also known as units, are placed in a hierarchy of layers to form what is known as a neural network

An ANN is composed primarily of three layers (Figure II.11), namely [37]:

Input Layer: This layer is responsible for receiving inputs from various programming-provided formats.

Hidden Layer: Placed between the input and output layers, this layer carries out the required calculations to uncover hidden patterns and characteristics.

Output Layer: After the input has been transformed, the hidden layer generates a range of outputs that are subsequently transmitted through this layer.

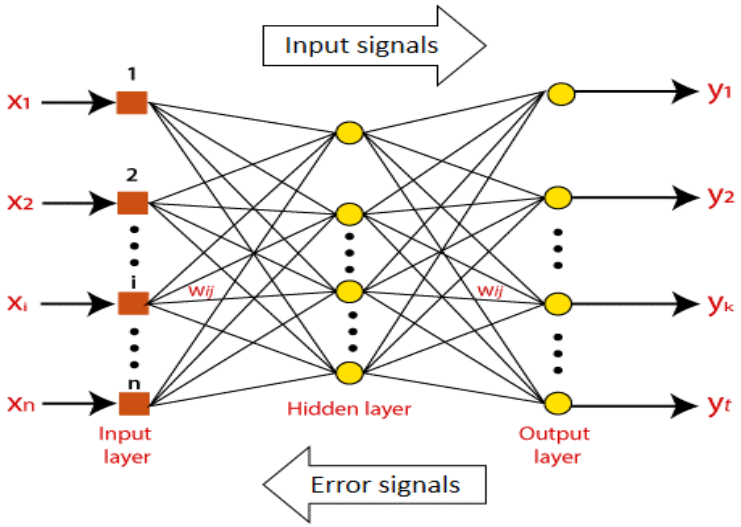


Figure II.11: ANNs' approach.

ANN could be classifier to many categories, including:

- **Feedback ANN:** In this type of ANN, known as feedback neural network, the output is fed back into the network to obtain the most internally-generated outcomes. According to the center for atmospheric research at the University of Massachusetts (CARUM), feedback networks are highly effective in solving optimization problems as they enable information to be fed back into the network. By using feedback ANNs, the internal system error can be corrected [37].

- **Feed-Forward ANN:** is a neural network type that includes one or more layers of neurons, as well as input and output layers. The network's overall strength is determined by the combined behavior of its connected neurons, and the output is determined by evaluating the network's input and output in conjunction. The key advantage of this network is that it can learn to recognize and classify input patterns [38].

- **Multilayer Perception:** is comprised of three or more layers and is employed to classify data that cannot be linearly separated. This network is fully connected, with each node in a given layer linked to every other node in the layer above. multilayer perceptron (MLP) utilizes a

nonlinear activation function, and its input and output layers are connected in a directed graph. As a DL technique, it utilizes backpropagation to train the network. It is widely used in various applications such as speech recognition and machine translation [38].

II.3.3.2 Deep neural networks

An ANN is referred to as a DNN when it contains more than one hidden layer in its design. With the use of mathematical modeling, these networks handle complicated data. Feed-forward neural networks are sometimes referred to as DNNs. In these networks, data flows only in a forward direction and never backward, which prevents the same node from being visited twice. Such networks can categorize large volumes of data, often in the millions, but require substantial amounts of data to be trained effectively.

DNNs typically consists of multiple hidden layers, as well as input and output layers. These networks are designed to handle complex, nonlinear relationships among input data, even when that data is unstructured and unlabeled. DNNs are structured hierarchically, with each layer of neurons building upon the one before it, much like the neural architecture of the human brain [39].

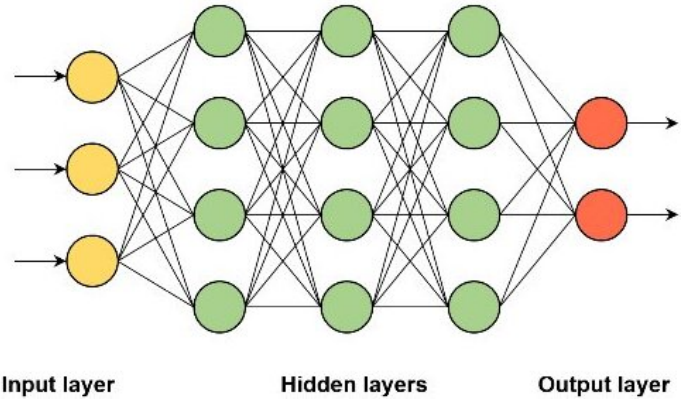


Figure II.12: Deep neural networks basic architecture.

Neurons in a neural network receive signals from other neurons, and these signals are relayed to other neurons based on the information they contain. The signal is transmitted to the next neuron only if its value exceeds a certain threshold; otherwise, it is ignored. Data is thus passed from one layer of neurons to the next, producing output at each stage until it reaches the output layer, which makes a yes-or-no prediction based on the probability.

In other words, each layer of neurons contains multiple neurons, and each neuron performs an activation function that determines its activity level. To pass the signal to the next neuron in the layer, the neurons act as gateways. The weights of the input influence the output of the following neuron and, ultimately, the output of the final output layer. Although the weights are initially assigned at random, they are iteratively trained to tune them and ensure that the network produces the intended prediction [39].

II.3.3.3 Recurrent neural networks

An recurrent neural networks (RNN) is a type of neural network that has cyclic connections in its design, which allows it to understand the temporal dynamics of sequential input. An RNN 's hidden layer is made up of many nodes [40]. Using its current input x_t and the previously concealed state h_{t-1}, each node has a function that generates the presently hidden state ht and output y_t. Mathematically, h_t and y_t can be expressed as follow:

$$h_t = F(W_h h_{t-1} + U_h.x_t + b_h) \tag{II.6}$$

$$y_t = F(W_y h_t + b_y). \tag{II.7}$$

Where, W_h is the weight for the hidden-to-hidden recurrent connection. U_h is the input-to-hidden connection. W_y is hidden in the output connection. b_h bias terms for the hidden. b_y are the output states. F is the activation function for each node.

II.3.3.4 Long short-term memory

The long short-term memory (LSTM) architecture is used to train large neural network structures. LSTM is significantly faster at learning and produces a higher number of successful runs than RNN. Moreover, LSTM is capable of solving complex artificial long-time lag tasks that previous recurrent network methods failed to address.Figure II.13 illustrates LSTM architecture [41].

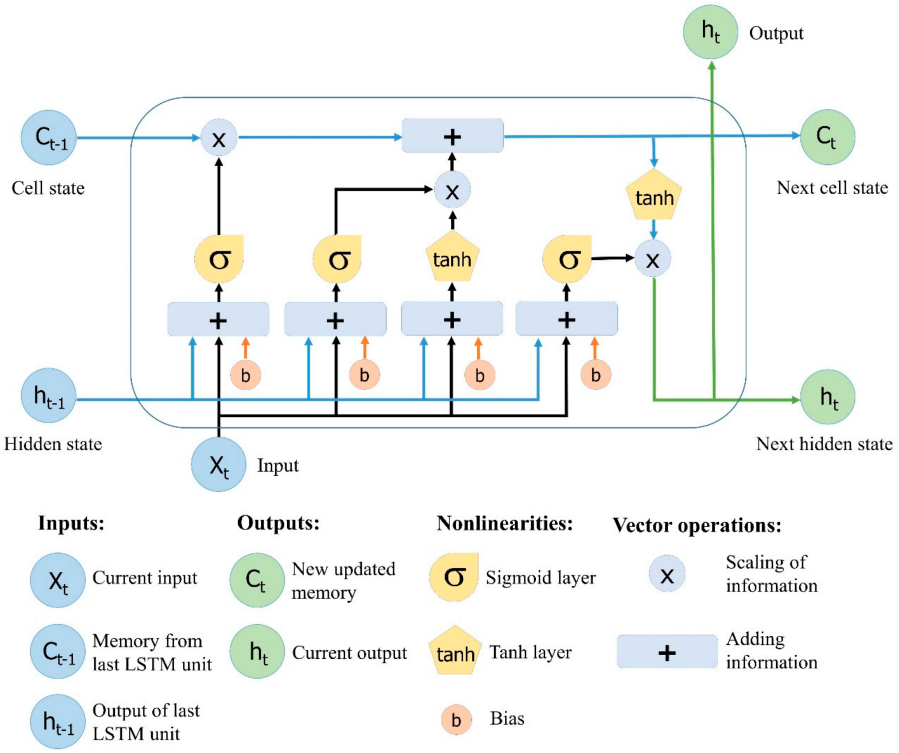


Figure II.13: LSTM architecture.

II.3.3.5 Convolutional neural networks

The most effective models for categorizing photos nowadays are CNNs. It has two distinct parts. A matrix of pixels representing an image is provided as input, for a gray-scale picture, which has two dimensions. For colorful images, the three primary colors, red, green, and blue, are represented by a third dimension with a depth of three. The convolutional layers of a CNN are its second part. It operates as an image feature extractor. A succession of filters or convolution kernels is applied to an image creating new images called feature maps. Some intermediary filters use a max pooling operation to lower the image's resolution (down-sampling). Finally, the feature maps are flattened and concatenated into a feature vector, called CNN code [14]. Figure II.14 illustrates an example of CNN architecture.

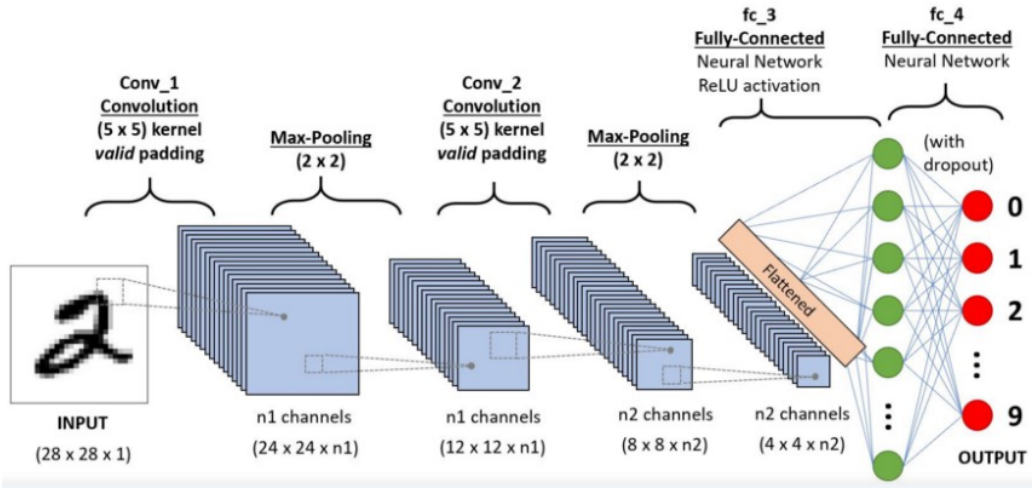


Figure II.14: Convolutional neural networks.

A variety of independent processing layers make up the CNN architecture. The description of each function of these layers will e discussed in the next subsections.

- **Convolution layers:** The CL is a fundamental element of a CNN, consisting of multiple filters. These filters can be visualized as small windows that move across the input image, computing the dot product between their values and the pixel values of the input image. A two-dimensional activation map that shows the responses of the filter at each place is the result of this action. The depth of the filters needs to be equivalent to that of the input image, but their width and height are adjustable hyper-parameters. It's also important to consider other hyper-parameters as they affect the dimensions of the resulting image, which are:

  a. **The number of filters:** corresponds to their ability to detect specific features within the input. In the presence of variously angled edges or clusters of color, different neurons within the depth dimension may activate. As a result, the depth of the output feature map will also be affected.

  b. **Stride:** parameter dictates the number of pixels the filter will move while scanning the image. If the stride is set to two or more, the filter will advance by that many pixels at

a time, leading to an output that is smaller than the input.

c. **Zero-padding:** padding the input image's border with zeroes (0s) is denoted as zero-padding. It is primarily employed to maintain the input size, resulting in the output size being preserved.

Figure II.15 shows the working principle of CL.

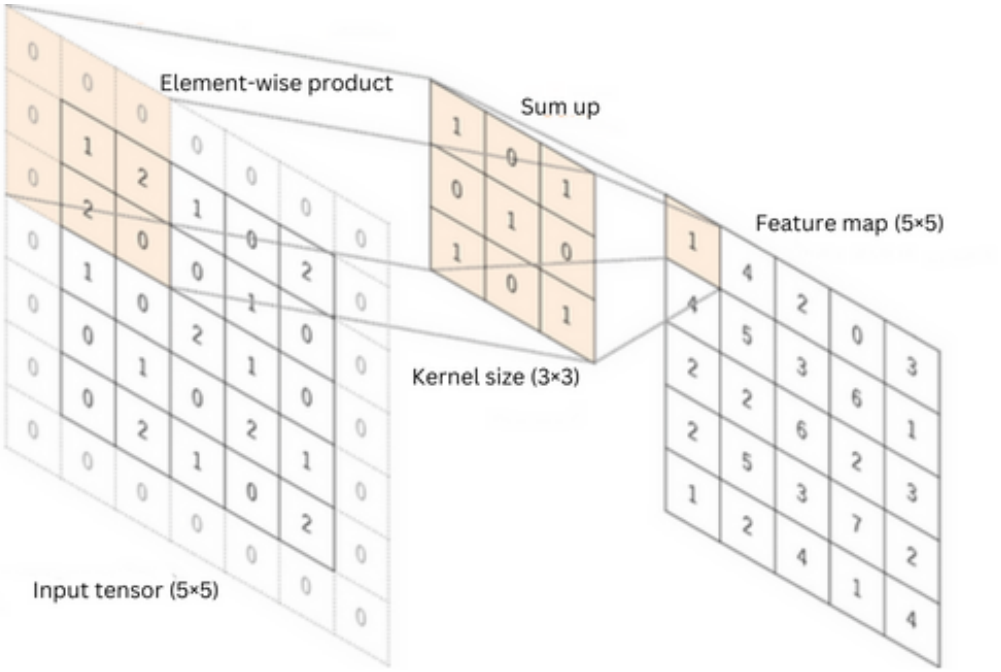


Figure II.15: Working principle of CL, the mathematical operation of convolution is performed between the input image and a filter of a particular size.

- **Rectified linear activation unit:** The ReLU is widely considered a significant breakthrough in the field, representing a turning point in developing neural networks. Despite its simplicity, it outperforms previous activation functions, such as sigmoid or tanh. Mathematically, it is expressed as: $f(x) = max(0, x)$. Both the ReLU function and its derivative are monotonic. If the function receives any negative input, it returns 0, but if it receives any positive value x, it returns that value. As a result, it generates an output ranging from 0 to infinity [42]. Figure II.16 illustrates an example of an input matrix after the ReLU operation.

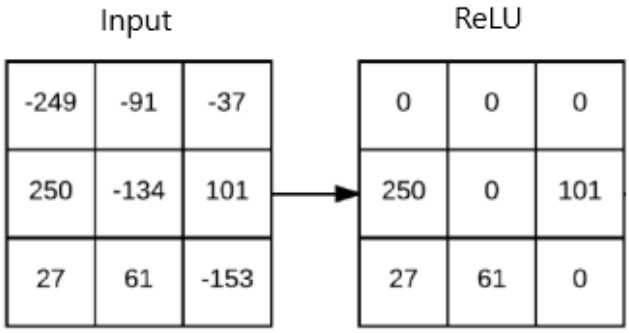


Figure II.16: An example of an input matrix after the ReLU operation.

- **Pooling layers:** The pooling layer aims to reduce the size of feature maps, which results in fewer parameters to learn and reduces the computation required by the network. It is considered one of the most effective methods for down-sampling feature maps. Pooling also helps achieve approximate invariance to small input translations in all scenarios, where most pooled outputs remain unchanged when the input is translated slightly. This is known as translation invariance.

  The most commonly used type of pooling is max-pooling. For instance, applying a max-pooling layer (Figure II.17 (a)) with a filter size of 2×2 and a stride of 2 to an input feature map eliminates 75% of activation, as only the maximum value of the 4 input numbers is retained. This allows only the most significant or sharp features to pass through the rest of the network. Average pooling (Figure II.17 (b)) is another common type of pooling layer; this type uses the average value rather than the maximum value between the input feature map values.

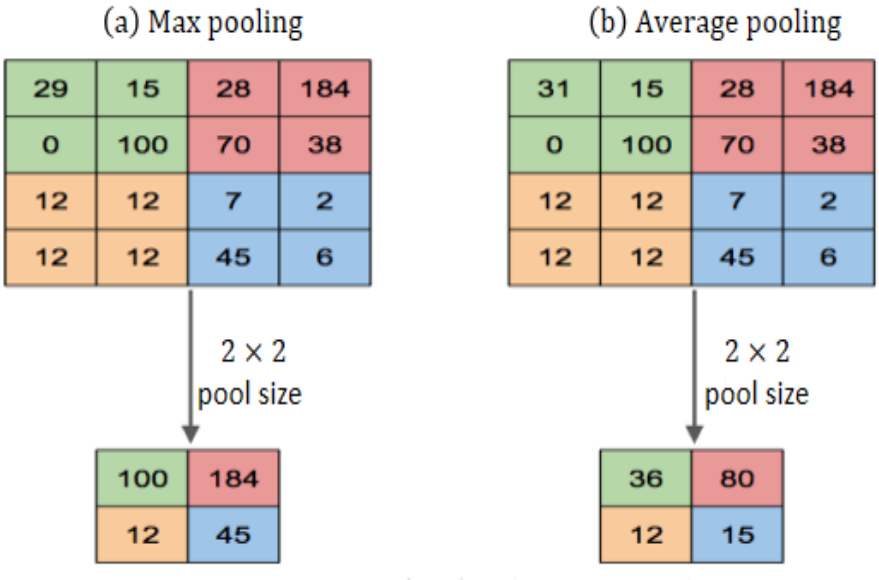


Figure II.17: Types of pooling layers.

- **Batch Normalization:** In statistics, normalization involves limiting a dataset or values to a range of zero (0) and one (1). Alternatively, it may refer to adjusting the mean of the data distribution to zero and the standard deviation to one, as described by [43]. The most prevalent form of normalization is batch normalization (BN), which normalizes the activations of an input volume before transmitting it to the subsequent layer in the network. Typically, a BN layer is added after an activation layer. By incorporating BN, the number of epochs required to train a neural network is significantly reduced. Furthermore, it helps to "stabilize" training, allowing for a broader range of learning rates and regularization strengths, as reported in [44]. Figure II.18 illustrates the principle of BN.

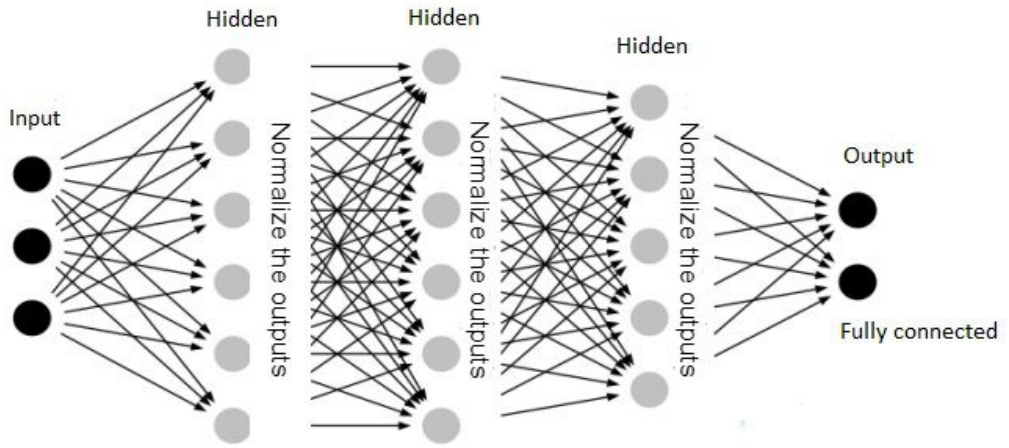


Figure II.18: Batch normalization.

- **Fully connected layer:** The final layer of the CNN is fully connected layer (FCL), sometimes known as the hidden layer. After multiple convolutional and max-pooling layers, FCL provides high-level reasoning in the neural network. Every neuron in a FCL has connections to all activations in the previous layer, and it takes inputs from the one-dimensional flatten layer (1D Layer). The FCL depicted in Figure II.18 represent the last layer of the neural network. Several of these layers can be incorporated into our categorization model depending on the level of depth we desire. It's crucial to keep in mind that the training dataset is solely responsible for this. The last hidden layer's output is sent to the softmax or sigmoid function to establish the probability distribution across all classes [45].

- **Regularization layer (Dropout):** A challenge that can arise with large neural networks trained on a small dataset is overfitting. As a result, the model may end up learning statistical noise from the training data rather than the significant features of the data. Consequently, the model may not generalize well when tested on new data.

  To mitigate this issue, the most commonly used regularization layer with CNN is the Dropout layer. This layer can help address overfitting by randomly setting its input units to 0 during training at a rate specified by the probability hyperparameter (referred to as rate). It drops out the layer's outputs at each step during training. It is important to note that the Dropout layer only operates during training, and no values are removed during inference [46]. Figure II.19 illustrates an example of Dropout.

- **Activation function:** Non-linear functions are what they're called. A neural network must learn, represent, and analyze any data as well as any arbitrary complicated function that maps inputs to outputs. As a result, we can generate non-linear mappings from inputs to outputs by introducing non-linearity to the network using non-linear activation functions. There are several activation functions, such as the sigmoid function that maps any input value to a value between 0 and 1, which is commonly used in the output layer of binary classification problems, the softmax function that is commonly used in the output layer

of multi-class classification problems rectified Linear unit, Hyperbolic tangent (tanh) function...etc. An important property of an activation function is that it must be differentiable so that we can use a back propagation optimization strategy to compute errors or losses with respect to weights and then optimize weights using Gradient Descend or any other optimization technique to reduce errors [47].

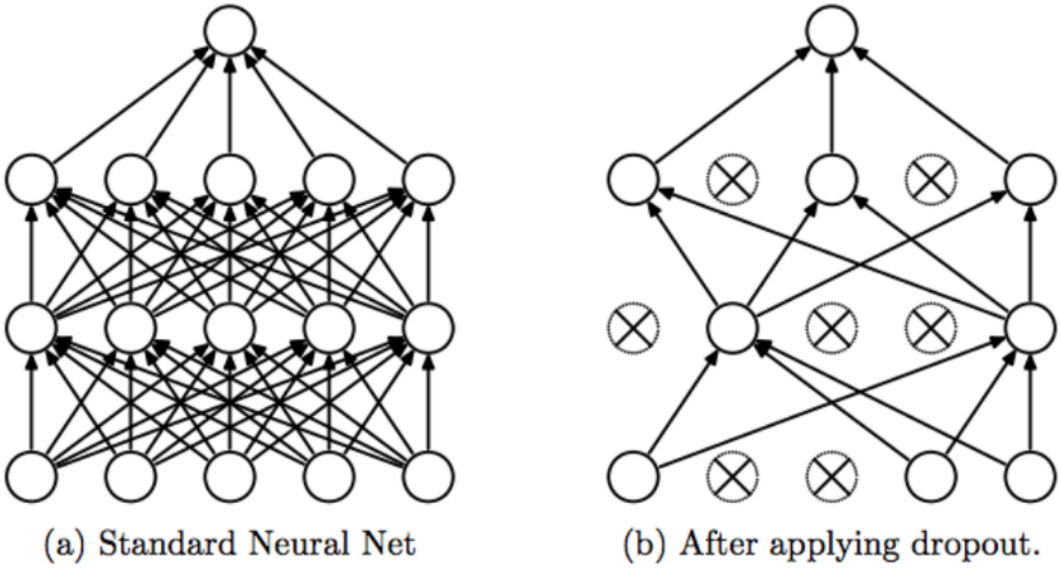


Figure II.19: Dropout principle.

II.3.4 Advantages and disadvantages of DL

DL has several advantages that make it a popular choice for ML tasks. Firstly, DL algorithms can automatically learn features from the data, eliminating the need for manual feature engineering.

This is particularly useful for tasks like picture recognition where defining features can be difficult. Secondly, DL algorithms can handle large and complex datasets that traditional ML techniques struggle with. DL algorithms have also shown improved performance on a wide range of tasks, including speech and picture identification, natural language processing, and computer vision. Additionally, DL algorithms like RNN and LSTM networks are well suited to handle sequential data like time series, voice, and text, due to their ability to maintain context and memory throughout time. However, there are also some drawbacks to DL, such as the issue of overfitting, which is a common problem with large neural networks. The quality of the data used to train DL algorithms is also crucial, and biased or noisy data can significantly affect model performance.

Finally, DL models are often referred to as black-box models because it can be difficult to understand how the model makes predictions and identify the factors that influence those predictions.

II.4 Conclusion

In this chapter, we discussed the similarities and differences between ML and DL. Both fields involve the use of algorithms to enable machines to learn from data and make predictions or decisions. ML is a broader field that includes a range of techniques for building predictive models, while DL is a subset of ML that involves the use of neural networks with many layers to learn complex representations of data. Both ML and DL require large amounts of data and computational resources to train models effectively. However, they face challenges related to interpretability and the potential for bias in decision-making. Despite these challenges, ML and DL have revolutionized the way we approach problems in a wide range of domains, including finance, healthcare, and marketing. They are likely to continue to play a key role in the development of intelligent systems in the future. The next chapter on IDS will cover the topic of detecting unauthorized access to computer networks and systems. It will discuss the various types of IDS, including network-based and host-based systems, and the techniques they use to detect potential threats. The chapter will also cover the challenges associated with intrusion detection, such as false positives and false negatives. Additionally, the chapter will examine the role of IPS in mitigating the effects of successful intrusions, and the importance of a layered approach to network security.

Finally, in our proposed approach for detecting attacks, we utilized DL to enhance the effectiveness and accuracy of the IDS.

CHAPTER III

NETWORK INTRUSION DETECTION SYSTEMS

III.1 Introduction

The size, use, and complexity of computer networks have all grown significantly in the last decade. Along with the rapid growth of the latest technologies like internet of things (IoT), cloud computing, and innovative network structures, new types of devices have also started to appear. Due to this, their security concerns have also significantly increased. Malware, a broad category of various kinds of computer programs that are developed by attackers with harmful intent, is one of the most prevalent cyber-security threats and attacks. A data breach occurs when a person has unauthorized access to sensitive information and private data, such as credit card numbers, etc. Social engineering is a strategy that attackers employ to manipulate users, scam them, and obtain sensitive information. Furthermore, specific types of cyber attacks contribute to these security concerns. These include:

- Zero-day exploits: These attacks target vulnerabilities in software that are unknown to the software developers or vendors, leaving systems unprotected until a patch is released.
- Rogue software: This refers to malicious software that disguises itself as legitimate applications, often tricking users into installing and running them.
- Malvertising: This involves the distribution of malicious software through online advertising, exploiting vulnerabilities in web browsers and plugins.
- Drive-by downloads: These attacks occur when malware is automatically downloaded onto a user's device without their knowledge or consent, typically through visiting a compromised website.
- DNS tunneling: Attackers use this technique to bypass network security measures by encapsulating malicious traffic within DNS requests and responses.

- Denial of service (DoS): In a DoS attack, an attacker overwhelms a target system or network with a flood of traffic, rendering it unavailable to legitimate users.
- SQL injection: This type of attack targets web applications by manipulating database queries, potentially gaining unauthorized access to sensitive data.
- Man-in-the-middle: This attack intercepts communication between two parties, allowing the attacker to eavesdrop, modify, or impersonate the communication.
- Password attacks: These attacks involve attempting to gain unauthorized access to systems or accounts by systematically guessing passwords or exploiting weak authentication mechanisms.
- Phishing: these attacks employ deceptive tactics, often through emails or websites, to trick users into revealing sensitive information such as usernames, passwords, or credit card details.
- Malware: This refers to a wide range of malicious software, including viruses, worms, Trojans, and ransomware, designed to disrupt or damage computer systems and steal sensitive information.

For the aforementioned reasons, there is an urgent need to develop efficient, effective, and dependable strategies for defensive capabilities against these security assaults given their increasing intensity and complexity. IDS have the potential to be a key component of the entire defensive architecture that can offer effective deterrence and defense against a variety of threats [48].

III.2 Cyber Attacks

A cyber-attack is any criminal behavior that attacks electronic information systems, networks, or infrastructure. Information can be stolen, changed, or destroyed as its primary goal. In the current cyber-attack scenario, attack vectors that take advantage of a lack of readiness and (system as well as human) preparedness to access sensitive data or compromise systems are frequent. Attack vectors are also designed to take advantage of human weaknesses. A cyber-attack can be thought of as an instance in which several vulnerabilities are used to take advantage of a system or network vulnerability. It might be difficult to become familiar with emerging technology, security trends, and threat intelligence. Even while the target might or might not be aware of all forms of cyber-attacks, it may have established a mechanism to cope with some of them. Depending on the risk analysis, an inherent risk or a residual risk may be the root cause of a cyber-attack. Figure III.1 depicts some of the nomenclatures of cyber-attacks [49].

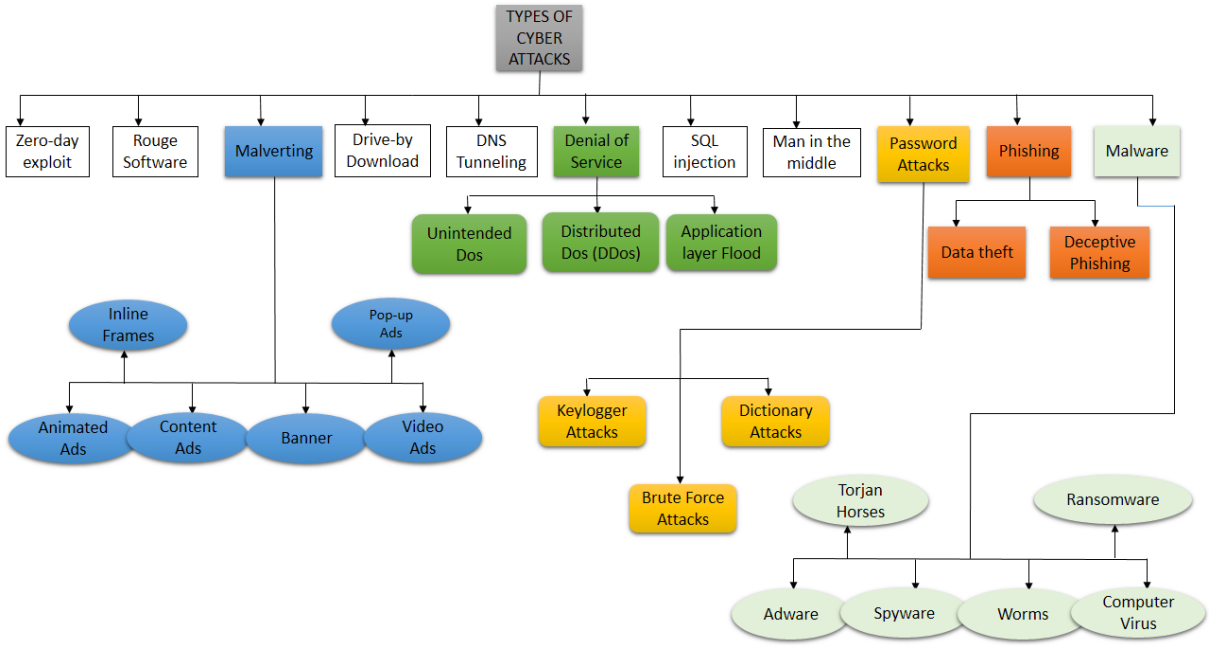


Figure III.1: Nomenclature of cyber-attacks.

III.3 Intrusion detection systems

A system called an IDS monitors network traffic for suspicious behavior and sends out alerts when it is found. It is software that scans a system or network for malicious activities or policy violations. Any illegal activity or violation is often captured either centrally using a security information and event management (SIEM) system or reported to an administrator. A SIEM system integrates outputs from several sources and uses alarm filtering methods to distinguish between legitimate and erroneous alarms.

While monitoring networks for potentially harmful behavior, IDS are also prone to raising false alarms. Consequently, enterprises must adjust their IDS products after the first installation. It entails correctly configuring IDS to distinguish between normal network traffic and malicious activities [50].

An IDS only needs to detect risks, thus it is placed out-of-band on the network infrastructure, as a result, it does not participate in actual real-time information exchange between the sender and receiver. Instead, IDS solutions frequently use a test access point (TAP) or switched Port analyzer (SPAN) port to examine a copy of the inline traffic stream, thus ensuring that IDS does not impact inline network performance.

The IDS is a listen-only device as well, as previously stated. The IDS keeps track of traffic and informs the administrator of its findings, but it is unable to immediately stop an identified exploit from seizing control of the system. Once they have gained access to the network, attackers can swiftly exploit weaknesses, making the IDS an ineffective deployment for prevention devices [51]. One such IDS can be set on each host in the computer network, at a centralized location, or it can

be distributed evenly over the whole network, Figure III.2 illustrates the incorporation of an IDS within an enterprise network framework [48].

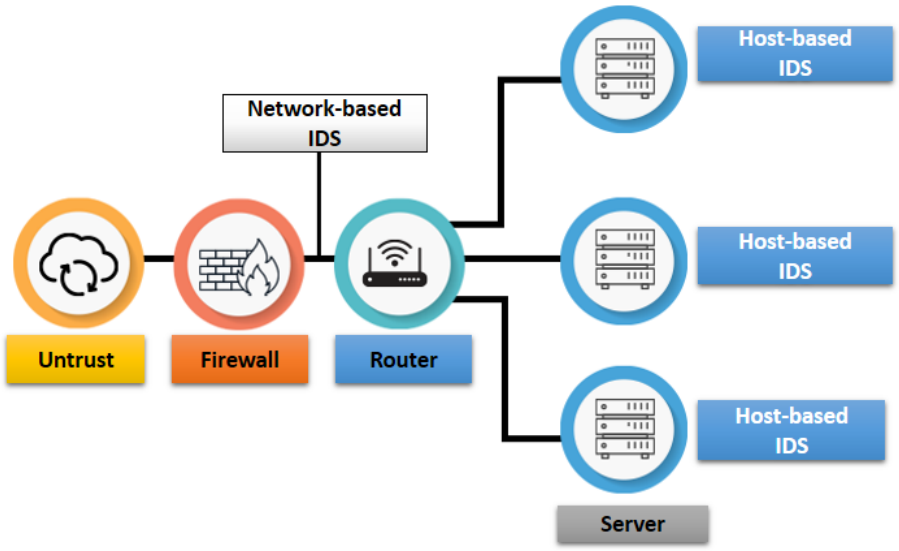


Figure III.2: IDS on an enterprise network.

III.3.1 Intrusion detection in the IoT

The IoT is a network of interconnected devices that allows for seamless information transfer between physical objects. IoT devices have access to the most private information and are predicted to outnumber mobile devices in terms of prevalence, as a result, attack probabilities will increase. Creating IoT IDS is imperative to safeguard the communication enabled by IoT technologies as security forms an integral part of most IoT applications. Numerous studies have utilized different datasets, ML, and DL techniques to validate the development of IoT IDS. Nevertheless, the most suitable dataset, ML method, or DL strategy for creating a successful IoT IDS remains uncertain [52].

III.3.2 Intrusion detection for intelligence internet of vehicles

With the IoT rapid evolution, it has been expanded to include automobiles in order to structure the intelligent transportation system (ITS), or the intelligent internet of vehicles (IoV). For example, the global positioning system (GPS) and the advanced driving assistance system are just two examples of the kind of applications that the intelligent IoV may offer for vehicles. It is one of the most important methods for utilizing ITS to create smart cities. In order to give users more bandwidth, several networking technologies have recently been used in IoVs, including the 5G-enabled communication network. Private user information is then transferred acrossIoVs in greater quantities. The IoVs is also an open network for all users, even though the user himself or herself can be a potential attacker. The IoV is frequently attacked by different malware due to its openness. The distributed denial of service assault, also known as a brute-force attack, is

a major threat to automobiles at the moment since it overwhelms their cache and computational power. Thus, IDS have gained increasing amounts of attention in IoVs in order to safeguard users' security and privacy [53].

III.4 Types of intrusion detection system

There are many different types of IDS, ranging from antivirus programs to hierarchical systems that monitor the traffic of a whole backbone network. Network intrusion detection system (NIDS) and host intrusion detection system (HIDS) intrusion detection systems are the two classes that are used most often [54].

III.4.1 Network-based IDS

NIDS are the most used in the industry. This acpIDS tracks and examine network traffic to find assaults. One NIDS can monitor the network traffic impacting numerous hosts linked to the subnetwork by listening on a network segment or switch, to protect those hosts. The majority of the time, NIDS are made up of a collection of dedicated hosts or sensors that are positioned throughout the network. These devices keep track of network traffic, analyze it locally, and alert a central control panel to assaults. To make it more challenging for an attacker to locate and detect sensors, many of them are built to operate in hidden mode [55].

NIDS offers several advantages in enhancing network security. Firstly, a few strategically placed NIDS can monitor a large network without significantly impacting its operation. NIDS are also typically passive devices that listen on the network cables and nodes and are easy to add to existing networks. Additionally, NIDS can be made extremely safe and invisible to attackers [56]. However, there are also some disadvantages to NIDS. For example, they may struggle to examine every packet on a large or active network, which potentially missing attacks during heavy traffic [57]. NIDS also have limitations in modern switch-based networks, as they may only monitor one host, and may not be able to defend against assaults that fragment packets, which possibly cause instability or crashing [57]. Although NIDS can offer a crucial network security layer, their limitations and implementation must be thoughtfully considered to guarantee their effectiveness [55].

III.4.2 Host-based IDS

HIDS are network applications that run on separate hosts or sensors. Just the incoming and outgoing packets from the device are monitored by a HIDS, which notifies the administrator of any unusual or malicious activity. It compares the current snapshot of the system files with the previous snapshot. An alert is given to the administrator to look into if the analytical system files were altered or deleted. Mission-critical equipment, which are not anticipated to modify their layout, are an example of HIDS utilization [58].

HIDS offer several advantages, such as the ability to detect attacks that NIDS cannot, as they can monitor host-specific events. HIDS can operate in encrypted environments, and network switches do not affect their ability to monitor events [59]. However, there are also some disadvantages to using HIDS. Configuring and managing HIDS can be challenging, as each host being monitored has its own information to be configured and monitored. HIDSs can only observe network packets received by their host, making them unsuitable for detecting network scans or surveillance targeting the entire network [55]. Certain denial-of-service attacks can deactivate HIDS, and they can impact system performance by using computer resources for monitoring. Therefore, it's important to consider the advantages and disadvantages of HIDS when deciding whether to use them and to develop a comprehensive security strategy to address theris potential risks.

III.4.3 Application protocol-based IDS

Application protocol-based intrusion detection system (APIDS) are a unique subset of HIDSs that examine the happenings inside a software program. The application's transaction log files are the most typical information sources used by APIDSs.

APIDSs can detect suspicious behavior as a result of authorized users exceeding their authorization since they have the application interface capability and extensive domain or application-specific built into the analysis engine. This is due to the fact that these issues are more likely to arise in user, data, and application interactions [55].

APIDS presents various advantages, including the ability to observe user-application interactions and trace unlawful activities to specific users. They can also operate in encrypted environments by interacting with the application at transaction endpoints where information is presented in an unencrypted form [60]. On the other hand, APIDS has some disadvantages as well. The application logs they rely on for monitoring are often less securely protected than the operating system audit trails used by HIDS, making them more vulnerable to attacks. Additionally, APIDS may not be able to identify a trojan horse or other similar software tampering threats since they monitor events at the user level of abstraction. Therefore, it is recommended to use APIDSs in conjunction with HIDS and/or NIDS for comprehensive security coverage.

III.4.4 Protocol-based IDS

A system or agent that consistently resides at the front end of a server, controlling and interpreting the protocol between a user/device and the server, makes up a protocol-based intrusion detection system (PIDS). By constantly monitoring the protocol stream hypertext transfer protocol Secure (HTTPS) and accepting the associated hypertext transfer protocol (HTTP) protocol, it tries to protect the web server. As HTTPS is not enough secured and does not immediately enter the web presentation layer, the system would need to be located within this interface in order to use HTTPS [61].

PIDS has several advantages that make them effective in detecting known protocol vulnerabil-

ities and exploits. They can operate independently of applications, identify anomalies in network traffic, and detect attacks that may bypass application-level security measures. However, PIDS may not be effective against zero-day attacks, and attackers can use obfuscation techniques to evade detection. False positives can also occur, which can disrupt network operations. Additionally, PIDS is limited to detecting attacks based on protocol specifications and cannot detect attacks that bypass the protocol entirely. Therefore, a comprehensive security strategy should incorporate multiple layers of defense, including network-based and host-based IDSs, and other security measures to provide a robust security posture. Despite its limitations, PIDS can still be an effective tool in detecting known protocol-based attacks and can complement other security measures [62].

III.5 IDS methods

An IDS is made to analyze network traffic after data collection and compare traffic patterns to well-known attacks. This technique is often known as pattern correlation. An IDS will alert specific technicians or IT administrators when it detects suspicious or harmful behavior. IDS alerts give you the ability to quickly start troubleshooting, identify the causes of problems, or find and stop harmful agents in their tracks. The two main intrusion detection methods used by IDS are anomaly-based intrusion detection and signature-based intrusion detection [63].

III.5.1 Signature-based IDSs

In order to identify a known attack, signature-based intrusion detection systems (SIDS), sometimes referred to as knowledge-based detection or misuse detection, uses pattern-matching methods. Matching techniques are employed in SIDS to identify a previous attack. In other words, an alarm signal is triggered when an intrusion signature matches an intrusion signature that already exists in the signature database. In SIDS, the host's logs are examined to look for sets of instructions or behaviors that have been recognized as malware . The fundamental concept is to create a database of intrusion signatures, compare the current set of actions to the signatures already in use, and raise the alarm in the event of a match. Typically, SIDS provides great detection accuracy for known intrusions. However, until the new attack's signature is extracted and saved, SIDS cannot identify zero-day attacks because there isn't a matching signature in the database until the new attack's signature is extracted and stored.

Because there is no signature for such attacks, SIDS approaches have proven steadily less successful. Other factors like polymorphic malware variations and an increase in targeted attacks all contribute to weakening the viability of this traditional model. Using anomaly-based intrusion detection system (AIDS) techniques could be a solution to this issue. Instead of profiling what is incorrect, as explained in the section below, AIDS works by differentiating between acceptable and undesirable behavior [52].

III.5.2 Anomaly-based IDS

Nevertheless, AIDSs can notify you of unknown suspicious behavior. An AIDSs educates the detection system to recognize a normalized baseline rather than looking for known threats. AIDSs simply identifies any out-of-the-ordinary behavior to raise alerts by comparing all network activity to a baseline that defines how the system normally operates [64].

An AIDSs will raise a potential flag for concern if it detects anything that does not match the current normalized baseline, such as a user trying to log in outside of normal business hours, new devices being added to a network without authorization, or a flood of new IP addresses trying to connect to a network. The drawback of this is that many benign behaviors will be tagged for being abnormal despite not being malicious. Because anomaly-based intrusion detection is more likely to produce false positives, it may take more time and resources to thoroughly look into all of the potential threat alarms. Yet, this potential drawback is precisely what enables anomaly-based intrusion detection to identify zero-day vulnerabilities that signature-based detection is unable to detect [65]. A list of known, existing risks is the only one that signature-based detection can find. On the other hand, it also has a quick processing time and better accuracy for attacks that are already known. These two detection techniques each have benefits and drawbacks that, in most cases, work best when used in combination.

The four primary categories of AIDSs methods are DL,RL,UL, and SL. While learning under supervision, all input and output variables are gathered, examined, and an algorithm is used to learn normal user behavior from the input to the output. The goal is to sufficiently approximate the mapping function so that whenever a new input record is gathered that predicts the output variables for that record. Whereas RL techniques allow an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences, UL tries to identify the requested actions from existing system data such as protocol specifications and network traffic instances where you only have input data and no corresponding output variables. Gaining a suitable action model that would maximize the agent's overall cumulative reward is the goal of RL [52].

III.5.3 Specification based IDS

Specifications are used to identify expected behaviors of programs and systems, and observable actions that differ from the specifications are detected as intrusions. Derivation from normalcy is another criterion utilized by specification-based intrusion detection to identify intrusions. Unlike anomaly detection, which specifies normality via the observation of users' previous normal actions, specification-based IDSs specifies normality through the use of manually created rules that define the intended behaviors of programs and systems.

In order to detect new intrusions with fewer false alarms, a specification-based intrusion detection approach is suggested as a solution for signature and anomaly detection systems. If the description of the intrusions is accurate, misuse detection systems can identify known intrusions with zero false alarms. However, technologies for detecting misuse are unable to find new in-

vasions. While false alarm rates (FARs) are a problem for anomaly detection systems, they can indeed discover new attacks. No previous knowledge of intrusions is required in specification-based techniques for intrusion detection; intrusions are instead discovered by thoroughly analyzing deviations from specifications. As a result, it may identify both well-known and undiscovered invasions. Furthermore, as the specifications are made to define the legitimate or intended behaviors of programs or systems, changes in the environment should not affect them since a program's legitimate behaviors remain the same even when its users change their behavior patterns. Or, to put it another way, if the specifications are appropriately created, the illegal behaviors that diverge from them are always illegal [66].

III.5.4 Hybrid IDS

Different strengths and weaknesses are related to each of the three methods of intrusion detection. In order to combine their strengths and minimize their weaknesses, significant research has been invested in integrating distinct IDS [66], which are described as follows:

III.5.4.1 Integration of misuse and anomaly intrusion detection

Anomaly-based detection methods have the significant flaw of being unable to identify new types of intrusions. The specified profile set by the network administrator is the only thing it can detect as well. Although while misuse detection approach gets beyond these issues, it still can not find anything that is not covered by the system's established rules, and it also requires frequent system rule updates. The need to create an IDS that is both far more effective than the current system and solves both of its weaknesses. The above challenge would have a hybrid intrusion detection solution. A hybrid IDS may be described as a system that combines both anomaly detection and misuse detection techniques [67].

III.5.4.2 Integration of specification and misuse detection

As mentioned before, attack behavior that deviates from system or program specifications can be detected using specification-based intrusion detection. However, by using system or program specifications, some threats cannot be caught. For example, taking system resources during a race condition attack won't affect the behaviors of the privileged program being watched, therefore, a race condition attack cannot be detected based on a program's specifications. As a result, misuse detection is seen as a potential technique that can make up for the specification-based approach's flaws and identify intrusive activities that intrusion signatures detect. [66].

III.6 Firewalls and IDS

A firewall is a network device that manages packet traversal over a protected network's borders in accordance with a given security policy. A firewall security policy is a set of logical rules that

specify how network packets should be handled in response to certain filtering criteria. A rule is made up of an action field, and a series of filtering fields (also known as network fields), such as protocol type, source and destination IP addresses, and ports. A rule's filtering fields indicate potential values for the corresponding fields in real network traffic that corresponds to the rule. A single value or a range of values might be present in each network field. Filtering may either accept a packet and allow it to pass through a secure network, or it can refuse a packet and prevent it [68].

Despite the fact that they both deal with network security, firewalls and IDS differ in that a firewall searches outward for intrusions in order to prevent them from happening. Firewalls impose access restrictions between networks to prevent infiltration and do not alert users to an internal network assault. However, An IDS analyses a suspected intrusion after it has occurred and generates an alert [69].

III.7 IDS and intrusion prevention system

The same as an IDS (Figure III.3 (a)), an IPS is a hardware component or software for network security that continually scans network activity for threats (Figure III.3 (b)). IDS detects risks, however; IPS goes a step further by taking the proper action automatically, including actions like reporting, blocking traffic from a certain source, deleting packets, or reconnecting. A "honeypot" is a decoy that contains fake data that some IPS programs can be set up with to mislead attackers away from their intended targets that contain real data. The differences between the two systems can simply be summarized as follows:

a. Response: An IDS will stop at the detection stage, leaving your department and you free to choose what to do. An IPS will either try to contain the threat or stop unauthorized users from accessing your network further.

b. Protection: An IPS operates automatically due to the differences mentioned above, giving an attacker minimal time to continue compromising an organization.

c. Impact: False positives could have a bad influence on your industry because of that automation. In the interest of safety and security, an IPS may shut down your network or block traffic to and from a specific device, even if the threat is not needed for such harsh action (or the alert was a false positive).

Both solutions are quite beneficial, and many vendors provide an IDS and IPS, or IDPS, as a solution that combines the advantages of both systems, so organizations shouldn't necessarily think about picking one over the other.

Before selecting the appropriate IDS and/or IPS solution, security leaders should be aware of the requirements of their business and have a list of the data that needs to be monitored. They should also assess their own security department to decide whether they prefer a hybrid strategy, an automated solution, or an agency that can respond appropriately. IPS and IDS are typically put

behind the firewall in the security stack to catch any suspicious activity that manages to get past. The security team can see incoming traffic and any potential security gaps thanks to the IDS's proximity to the internal network and the user. the difference between IPS and IDS architecture is clearly illustrated in Figure III.3 [70].

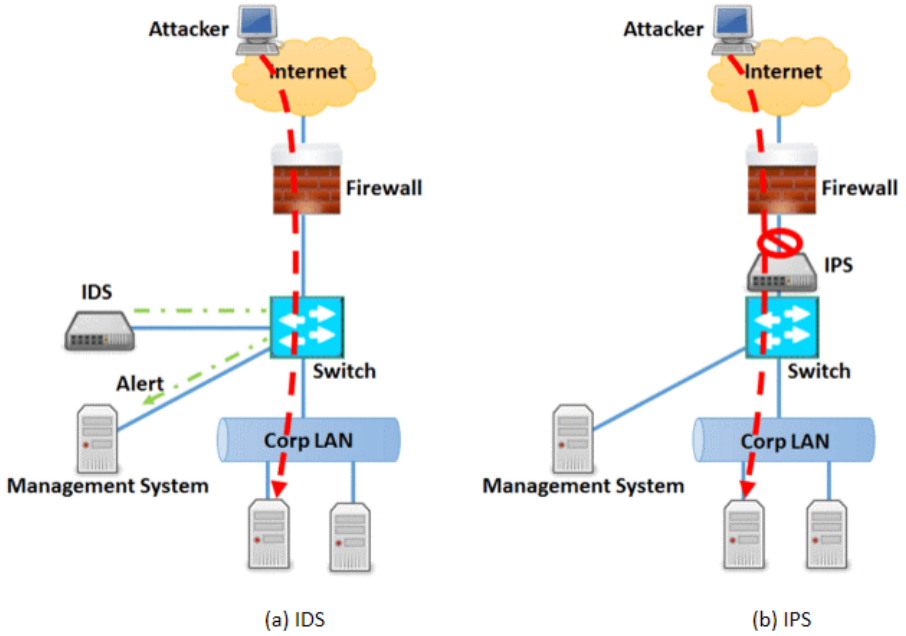


Figure III.3: Comparison between IDS and IPS.

III.8 Deploying IDSs

The computer network security infrastructure of any major firm must include intrusion detection equipment. An effective IDS deployment, needs extensive planning, preparation, prototyping, testing, and specialist training due to the limitations of today's intrusion detection technologies and the low level of security controls of many system administrators [55].

III.8.1 Deploying network-based IDSs

Deploying NIDS is a critical step in enhancing the security of a network. The first step is to determine the network's topology and the locations of critical assets. Once the network's layout is known, IDSs sensors should be placed strategically to maximize their effectiveness. These sensors should be placed at key points in the network, such as on the network's perimeter, behind firewalls, and in critical internal network segments [71].

The next step is to configure the IDSs to analyze network traffic and identify potential security breaches. This involves setting up rules and alerts that will trigger when the IDSs detects suspicious activity. These rules should be specific to the network's needs and should be updated

regularly to ensure that the IDSs remains effective against new and emerging threats.

To ensure that the IDSs is functioning correctly, it is important to test the system regularly. This can be done through manual testing or automated testing tools. It is also essential to monitor the IDSs logs to identify any false positives or false negatives that may occur.

Finally, it is important to ensure that the IDSs is integrated with other security measures, such as firewalls, antivirus software, and security information and event management (SIEM) systems. This integration will ensure that security events are identified quickly and can be responded to effectively.

Overall, deploying NIDS requires careful planning and configuration. However, once properly deployed, IDSs provide an essential layer of security to help protect networks from potential threats [71].

III.8.2 Deploying host-based IDSs

HIDSs can provide your systems with higher levels of security after network-based IDSs is installed and running. Nevertheless, because each IDS must be installed and configured for each unique server, establishing HIDSs on every host in the company might be a lengthy process.

Hence, it is advisable that the enterprise set up HIDSs first on crucial servers. Overall deployment costs will be reduced as a result, and inexperienced staff members will be able to concentrate on alerts issued by the most crucial hosts. Organizations that care more about security may think about putting HIDSs on the majority of their hosts once it operation is normal. In this situation, HIDSs solutions must have centralized administration and reporting capabilities. The complexity of managing warnings from a large number of hosts will be greatly reduced by these features. The capacity of the operator to distinguish between true and false alerts plays a significant role in an IDS's performance, but this is especially true for host-based IDSs. An operator using an IDS in a certain environment will eventually develop a sense of what is typical for that environment as monitored by the IDS [55].

III.8.3 Alarm strategies

The decision of which IDS alarm features to activate and when is crucial when installing IDSs. A broad range of alarm options, including email, paging, network management protocol traps, and even automatic attack source blocking, are available with the configurable alarm capabilities that are included with the majority of IDSs.

Even though these capabilities could seem alluring, it's best to use care until you have a reliable IDS installation and a general understanding of how the IDS behaves in your environment. IDS alarms shouldn't be activated for up to many months after installation, according to some experts [55].

III.8.4 Response options for IDSs

IDSs produce replies after collecting event data and analyzing it to look for signs of attacks. Some of these solutions ask the user to send their findings and results to a certain place. Others entail more responsive automated mechanisms. Although scientists are prone to underestimate the significance of effective response functions in IDSs, they are actually of utmost significance. Commercial IDSs allow a variety of choices for replies, which are usually divided into active responses, passive responses, or a combination of the two [55].

III.8.4.1 Active responses

When specific types of intrusions are discovered, active IDS responses are automatic actions that are done. The two types of active reactions are as follows:

- **Collect additional information:** The active response aims to gather more details regarding a possible attack. In the case of an IDS, this might entail turning up the sensitivity of data sources (such as increasing the number of events recorded by an operating system audit trail or turning up the sensitivity of a network monitor to capture all packets, rather than just those targeting a specific port or target system). Further data gathering is beneficial for a number of reasons. Having a system to determine if an attack occurred can aid in detecting attacks and collecting data for investigations and legal action [55].
- **Change the environment:** Stopping an attack in progress and subsequently refusing the attacker access is another active response. IDSs often only have the capacity to block IP addresses from which the attacker appears to be coming, rather than being able to deny access to a specific person [55]. Several strategies can be employed to counteract attacks including terminating the connection by injecting TCP reset packets into the attacker's connection to the victim system, reconfiguring firewalls and routers to block packets coming from the attacker's apparent location, blocking the network ports, protocols, or services being used by an attacker via setting routers and firewalls and reconfiguring routers and firewalls to turn off all connections that use particular network interfaces in emergency situations.

III.8.4.2 Passive responses

Users of the system receive information from passive IDS responses, and human beings are relied on us to take action in response to that information. Numerous commercial IDSs only accept passive responses.

- **Alarms and notifications:** IDSs send out alarms and notifications to users when attacks are detected. Commercial IDSs offer flexibility in how, when, and to whom alarms are produced. The most common type of alarm is an on-screen alert or popup window that can contain various levels of information about the attack. Remote alarm or alert notification is

also an option for large or distributed enterprises. Email is not a recommended notification mechanism due to the potential for hackers to block the message [55].

- **SNMP traps and plug-ins:** A network management system receives warnings and alarms from certain commercial IDSs that are designed to do this. They provide warnings and notifications through SNMP traps and messages to centralized network management consoles so that network operations staff may attend to them. This reporting scheme has a number of advantages, such as the capacity to modify the entire network infrastructure to respond to a detected attack, the capacity to shift the processing load associated with an active response to a system other than the one being targeted by the attack, and the capacity to utilize standard communications channels [55].

- **Reporting and archiving capabilities:** Commercial IDSs often provide the ability to produce regular reports and other in-depth information sheets. Some of the devices can provide reports detailing system activities and intrusions found within a certain reporting period (for example, a week or a month.) Some offer statistics or logs produced by the IDS in forms appropriate for use with report generation software or inclusion in database systems [55].

III.8.5 Conclusion

In this chapter, we have discussed the importance of IDS in detecting security breaches and unauthorized access due to the increasing threat of cyber-attacks. IDSs is particularly vital in areas such as IoT and intelligent vehicles where the potential for attacks is higher. We have explored various types of IDSs, including NIDS, HIDS, and APIDS, each utilizing distinct methods to identify potential threats. We have also emphasized the importance of combining IDSs with firewalls as an effective security measure and understanding the differences between IDS and IPS. Proper deployment of IDSs is critical to ensure optimal security and minimize the potential impact of attacks, making IDSs a crucial tool for safeguarding against cyber threats and protecting against potential breaches. In the next chapter, our objective is to develop a pre-trained model that exhibits the capability to accurately classify various types of attacks. This model will serve as a valuable tool in the field of cybersecurity, enabling us to detect and mitigate potential threats more effectively. We will make use of the power of transfer learning to improve the functionality and effectiveness of our model.

CHAPTER IV

IMPLEMENTATION OF THE PROPOSED CNN-BASED IDS METHOD

IV.1 Introduction

We will be focusing on building an Intrusion Detection System (IDS) model using Convolutional Neural Networks (CNN) and an autoencoder. In this endeavor, we will explore the power of deep learning techniques to enhance the security of computer networks. Our primary objective is to develop a robust IDS model capable of accurately identifying and mitigating potential intrusions. To achieve this, we will start by constructing a source model that combines the strengths of CNN and autoencoders. The CNN architecture enables us to effectively extract high-level features from network traffic data, while the autoencoder aids in learning useful representations. This integrated approach allows us to leverage the benefits of both models for improved performance.

By using DTL, we aim to exploit the knowledge gained by the source model and transfer it to the target model, enabling the latter to benefit from the insights and features extracted by the former. This process facilitates the creation of a more effective IDS model, capable of accurately detecting and classifying potential intrusions with reduced false alarm rates.

IV.2 Evaluation metrics

Evaluation metrics in the world of intrusion detection system refer to a set of measures used to evaluate the performance of a model that is trained to detect and classify different types of cyber attacks. These metrics help assess the model's ability to identify different types of attacks while minimizing false alarms and missed attacks. Common evaluation metrics in this field include accuracy, precision, recall and F1 score, and they are used to select the best model for a given task, optimize its parameters, and measure its effectiveness in real-world scenarios. A summary of evaluation metrics are provided in Table IV.1. true positive (TP), true negative (TN), false

positive (FP), and false negative (FN).

Table IV.1: Evaluation metrics.

Metric	Math description	Characteristic
Confusion matrix	TP	Correctly detects an attack as an attack.
	FP	Incorrectly detects a normal activity as an attack.
	TN	Correctly identifies a normal activity as normal.
	FN	Fails to detect an attack and classifies it as normal.
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Measures the percentage of correctly classified instances by a classification model.
Precision	$\frac{TP}{TP+FP}$	A measure of the model's ability to make accurate positive predictions.
Recall	$\frac{TP}{TP+FN}$	A measure of how often the model correctly identifies all positive instances
F1 Score	$\frac{2.precision.recall}{precision+recall}$	It combines precision and recall into a single metric that provides a balance between the two.
FAR	$\frac{FP}{FP+TN}$	A metric that measures the proportion of falsely identified positive instances among all the negative instances.

IV.3 Trained attacks for the proposed model

Because of the progress in ML and the emergence of DL field, as well as its ability to extract high-level features with high accuracy, these systems have been included to work with real network traffic for a wide range of intrusions and normal behavior as an ideal method of testing and evaluation [72]. In 2018, the Canadian Establishment for Cybersecurity (CIC) released the most recent and realistic cyber dataset called CSE-CIC-IDS2018. The fundamental goal of this dataset is to create an orderly technique to deal with producing various and extensive benchmark datasets for IDS on the development of client profiles that contain theoretical depictions of events and practices seen on the system. This dataset contains seven separate attack scenarios: botnet, heartbleed, brute-force, DoS and distributed denial of service (DDoS), inside network penetration, and web attacks. The assaulting framework contains 50 computers, while the victim association contains 5 divisions, thirty servers, and 420 machines. The dataset contains the captures, organized traffic, and framework logs of each machine, as well as 80 highlights extracted from the captured traffic using CICFlowMeter-V3 [73]. In is chapter, our studies and developed DL-based IDS models are all based on the aforementioned dataset. The following attacks are studied:

- **Benign:** In the world of IDS, benign refers to regular and non-threatening network activity or behavior. Web browsing, email communication, and file transfers between trusted users are examples of benign traffic. The capacity of IDS to tell the difference between benign and malicious traffic is crucial because it allows the system to recognize and respond to possible security threats.

The use of anomaly-based detection in IDS is a method for identifying benign traffic. The IDS monitors network traffic for deviations from established patterns of typical behavior in this method. If the traffic deviates significantly from these patterns, the system flags it as potentially malicious and investigates further. Because it does not rely on the presence of a known signature to identify malicious behavior [74].

- **SSH:** One common type of attack that an IDS may detect is an attack on the secure shell protocol (SSH) [75]. SSH is a cryptographic network protocol used for secure remote login and command execution over an insecure network. It is widely used by system administrators and developers to manage remote servers and devices. However, SSH is also a common target for attackers, who attempt to gain unauthorized access to systems or steal sensitive data [76], Figure IV.1 illustrate the SSH brute force attack.

  The SSH brute force attack method, involves systematically attempting a range of possible passwords or keys until the correct one is found. This type of attack can be effective against weak passwords or encryption keys, but it can also be time-consuming and resource-intensive. By studying and testing the brute force attack method, we aimed to gain a deeper understanding using DL, specifically CNN, to secure encryption keys in protecting against unauthorized access and data breaches [77].



Figure IV.1: An example of SSH brute force attack.

- **DoS GoldenEye:** A cyberattack known as a DoS attack seeks to obstruct regular traffic flow to a targeted system, server, or network. A specific sort of DoS attack known as "GoldenEye" attacks web servers by overloading them with HTTP requests, as illustrated in Figure IV.2 [78]. In order to identify DoS attacks, particularly GoldenEye attacks, IDS analyze network traffic and search for patterns that signal an attack is occurring. The following are a few examples of signs that IDS can check for:

  - Unusually high traffic: IDS can identify a GoldenEye attack as an unexpected traffic spike since it generates a lot of HTTP requests.
  - Repeated requests for the same page: GoldenEye attacks frequently target a particular web server page or URL. A huge number of repeated requests for the same page might be recognized by IDS as a sign of an attack.

– Use of anomalous user-agents: To avoid being detected, attackers may include customized user-agent strings in their HTTP requests. IDS may be set up to recognize requests with suspicious or harmful user-agents.

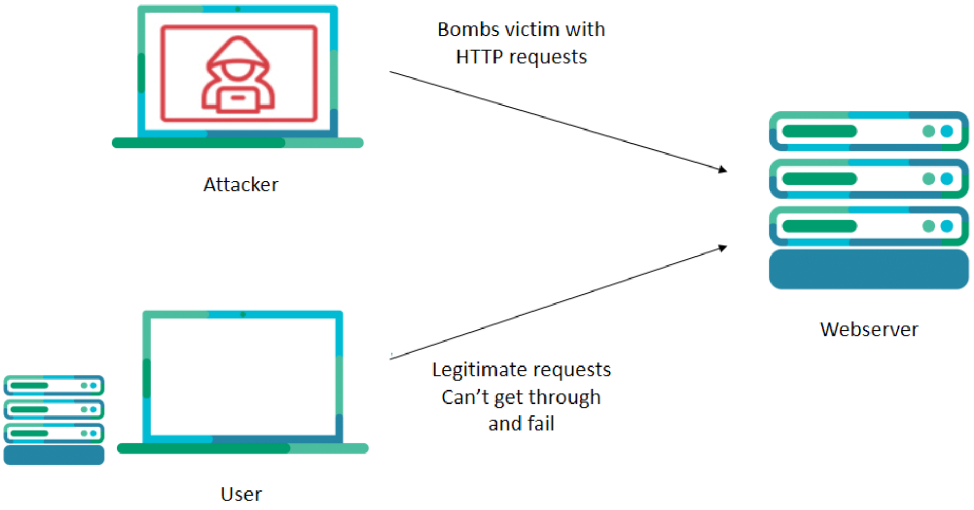


Figure IV.2: An example of dos GoldenEye attack.

IDS can provide warnings for system administrators to examine and take action to mitigate the attack if it finds any of these patterns. Implementing web application firewalls, rate limiting traffic, and restricting traffic from particular IP addresses are some typical mitigating strategies for GoldenEye attacks [79].

- **DoS Slowloris:** The Slowloris attack is a particular kind of DoS assault that targets web servers by creating several connections to the server and holding them open for as long as possible, depleting the server's resources and preventing it from responding to valid requests [80]. IDS can look for a number of indicators, including:

  – A high number of open connections: attacks by Slowloris create many connections to the server, which IDS can identify as an unusually high number of open connections.

  – Slow or incomplete requests: IDS can see slowloris attacks as an anomaly since they deliver HTTP requests very slowly or in fragmented, incomplete chunks.

  – Connection timeouts: slowloris attacks attempt to keep connections to the server open for as long as possible, which might create timeouts on reliable connections. An attack can be recognized by IDS by detecting an unusually high number of timeouts.

  To stop or lessen slowloris assaults, there are further strategies that may be implemented. Web servers can be set up to, for instance, restrict the number of connections that can be made from a single IP address or limit the amount of time a connection can be idle. Traffic from slowloris can also be blocked using network-level security measures like load balancers and firewalls.

- **FTP:** File transfer protocol (FTP) is a protocol used to transfer files between servers and clients. FTP attacks are a common type of attack that exploits vulnerabilities in FTP servers and clients to gain unauthorized access, steal data, or launch further attacks.

  IDS can detect FTP attacks by analyzing the traffic for specific signatures and anomalies, such as failed login attempts, abnormal login patterns, or unauthorized file transfers. IDS can also use protocol analysis to detect FTP commands that are outside of the expected behavior, such as unusual file types or commands that are not typically used in normal FTP sessions [81].

  IDS can respond to FTP attacks by generating alerts, blocking traffic, or triggering other security measures to prevent the attack from succeeding. IDS can also be configured to take specific actions based on the severity of the threat or the type of attack detected [81]. In our project, we provide the FTP brute force attack dataset, which includes methodically trying different usernames and passwords until the correct combination is found to gain unauthorized access to an FTP server.

In Figure IV.3, data visualization using bar plot is used to summarize the amount of attacks used in our dataset. This latter includes 1663703 instances of benign, 193360 instances of attack type FTP-BruteForce, 187589 instances of attack type SSH-Bruteforce, 41508 instances of attack type DoS-GoldenEye, and 10990 instances of attack type DoS-Slowloris.

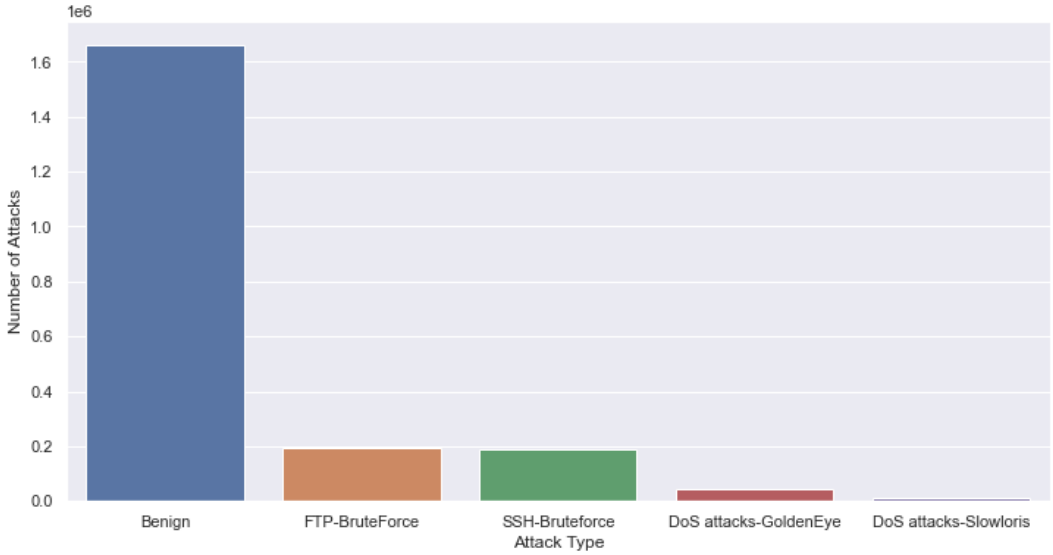


Figure IV.3: Attacks used in our source model.

To avoid biasing in data, we need to use data argumentation to balance the data and make equal distributions, resampling can help to address class imbalance, which is a common problem in intrusion detection. By oversampling the minority class or undersampling the majority class, the model can learn to better distinguish between different types of attacks and improve its overall accuracy. This can ultimately lead to better detection and mitigation of security threats. After resampling the attack types we got this representation in Figure IV.4.

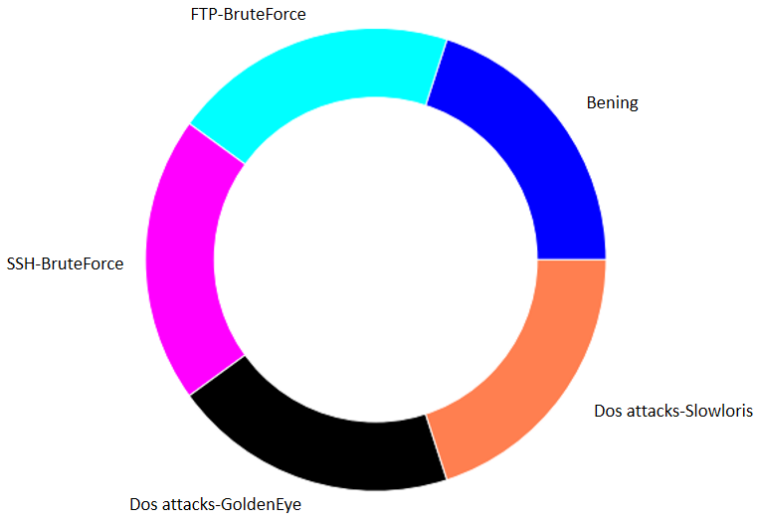


Figure IV.4: Intrusion attack type distribution.

IV.4 Proposed CNN architecture

Our proposed model has two inputs to get different representation of the same dataset. In the first input, by using the CNN, we aim to extract meaningful features from the input data. we used several layers including Conv1D, Batch Normalization, MaxPooling, Flatten, and Dense (cf. Chapter II). The initial layer in our source model is known as the input layer. Its primary purpose is to receive input data and transmit it to the subsequent layer for further processing. To enhance the model's performance, we integrated Conv1D, a commonly used convolutional layer for processing one-dimensional data. Additionally, we employed batch normalization, a technique that normalizes the inputs of a neural network layer to enhance optimization stability and efficiency. Furthermore, we utilized MaxPooling to reduce the dimensionality of the feature maps generated by the convolutional layers. We also incorporated Dense layers, which are fully connected layers that take the flattened output from the preceding layer and apply a set of learned weights to produce an output vector of a specific size. Lastly, the Flatten layer converts the multidimensional tensor output from a convolutional layer into a single one-dimensional vector, which can then be inputted into a final Dense layer. Together, these layers form a robust model capable of accurately classifying various types of data.

In the second input of the model, a 1D auto-encoder has been used, wish is based on the idea of learning to represent the input data in a compressed form, where the most important features are preserved while the redundant or irrelevant features are removed. we used only the encoder part of the 1D autoencoder, it takes the input data and compresses it into a low-dimensional representation [82]. This compressed representation can be used for various purposes, such as data compression and feature extraction. In the proposed model, the input layer is the first layer, followed by Conv1D, Batch Normalization, MaxPooling layers, and then we added a Flatten layer as the last layer before the output.

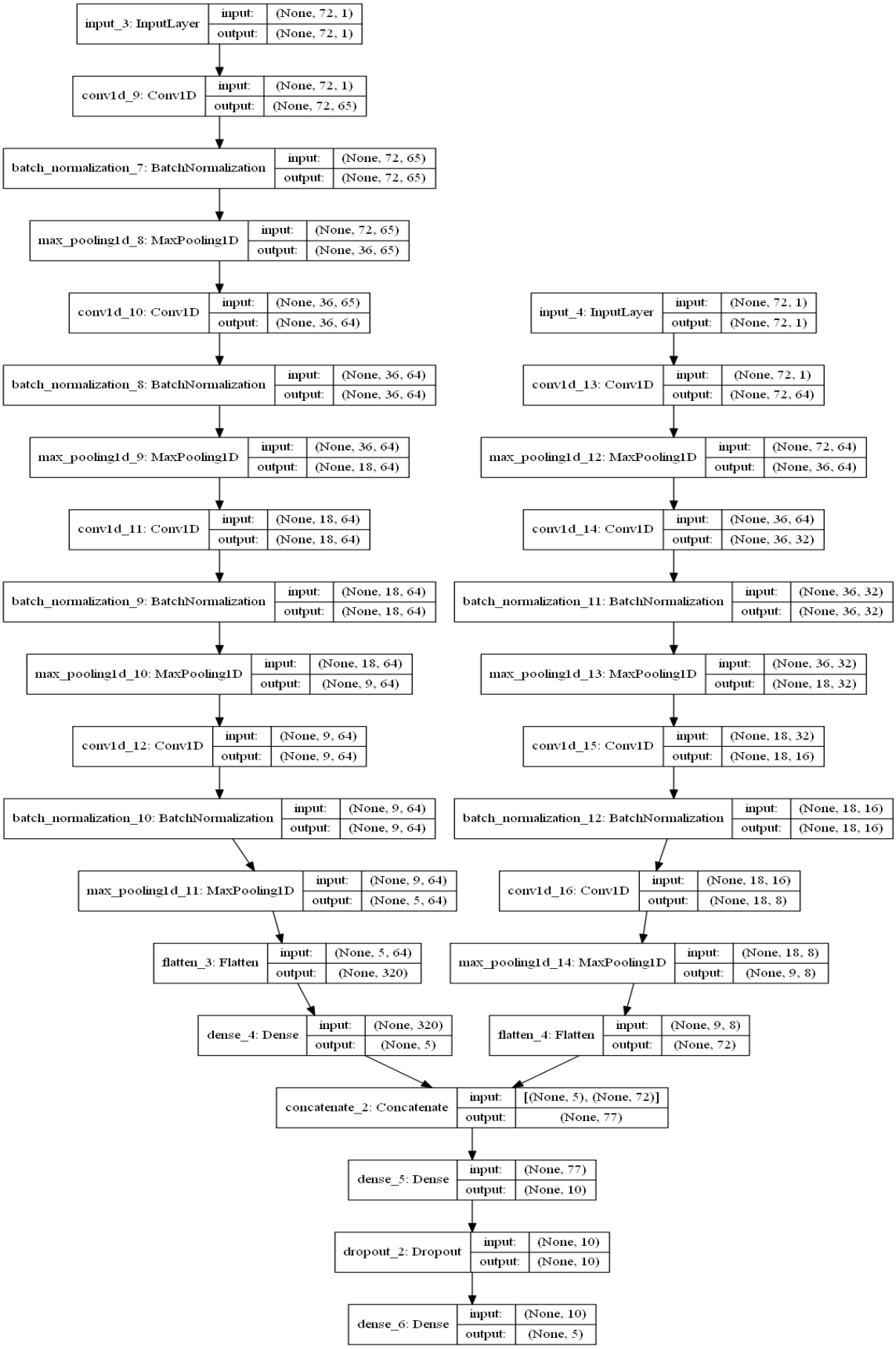


Figure IV.5: Our proposed source model.

Finally, we concatenated the output of the two models and added additional layers containing dropout. The final layer is a dense layer with 5 classifications representing our 5 attacks (Benign, Ssh-brute force, DoS goldeneye, DoS slowloris, FTP-brute force). The benefit of concatenating our two models is that it allows us to gain a deeper understanding of the features in our data and we can effectively leverage the advantages of both models to improve the accuracy of our classification task. The additional layers containing dropout help to prevent overfitting, which can lead to better generalization performance on new, unseen data. The softmax activation function has been used, which is employed particularly for multi-class classification problems. It is typically applied to the output layer of a neural network model. The softmax function takes a vector of real-valued inputs and normalizes them into a probability distribution over multiple classes. Figure IV.5 illustrates the proposed CNN architecture for our source model.

Model summary:

This summary result is showing the total number of parameters in a model. Specifically, it indicates that the model has a total of 86,731 parameters, which can be thought of as the dials that the model adjusts to learn from the input data and make predictions.

```
Total params: 86,731.
Trainable params: 86,121.
Non-trainable params: 610.
```

Out of these parameters, 86,121 are trainable, which means that they can be updated during the training process to improve the model's performance. On the other hand, 610 parameters are non-trainable, which means that they are fixed and cannot be updated during training.

Model build and fit:

In the compile layer, the Adam optimizer is a popular algorithm for stochastic optimization, well-suited for training deep neural networks. It combines the advantages of two other optimization methods, namely Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSprop). Adam adapts the learning rate for each parameter, providing efficient updates that can handle sparse gradients and noisy objective functions. By utilizing adaptive learning rates, Adam helps the model converge faster and achieve better performance. The choice of 'categorical_crossentropy' as the loss function is appropriate when dealing with multi-class classification problems. In our case, an IDS typically involves classifying network traffic data into different intrusion categories. Categorical cross-entropy measures the dissimilarity between the predicted probability distribution and the true distribution of the classes. As for the metrics, setting it to ['accuracy'] allows us to monitor the model's performance during training. Accuracy is a widely used evaluation metric for classification tasks. It measures the percentage of correctly classified instances out of the total number of instances. By tracking accuracy, we can assess how well our model is performing in terms of classifying network traffic data correctly.

In the fit layer of the deep learning model, the batch_size parameter determines the number of samples that will be propagated through the network at each training iteration. Here, we have set the batch_size to 36. The choice of an appropriate batch size can have an impact on the training

process and overall performance of the model.

```
model.compile( optimizer='Adam', loss='categorical_crossentropy', metrics=
['accuracy'])
model.fit([X_train1,X_train11], y_train1, validation_data =([X_test1
,X_test11], y_test1), epochs=80, batch_size=36)
```

Model validation accuracy:

Achieving a validation accuracy of 99.97% after 80 epochs as represented in the Figure IV.6, indicates that the model has been well-trained and is able to generalize well to new data. It also suggests that the model has learned the underlying patterns and features in the training data, and is able to apply this knowledge to make accurate predictions on unseen data.

As shown in Figure IV.6, the accuracy and validations accuracy are congruent to each other.

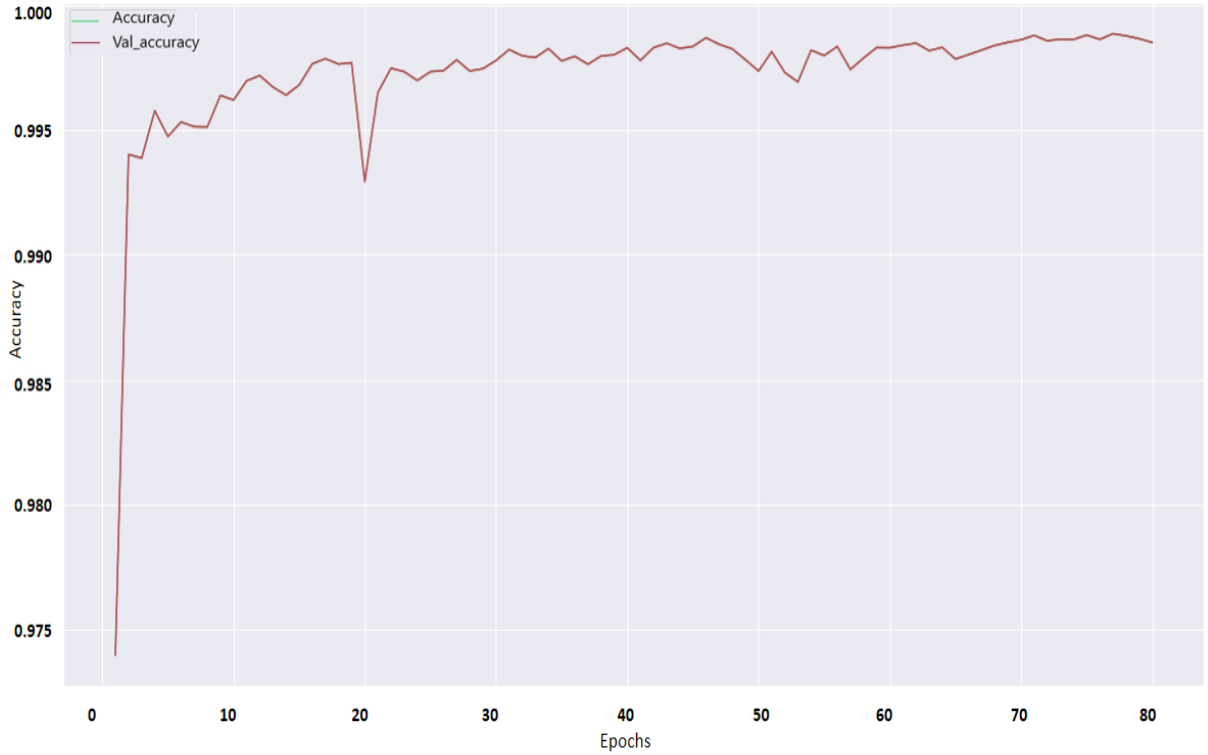


Figure IV.6: The obtained accuracy and validation accuracy of the proposed model.

Model validation loss:

Attaining a validation loss of 0.0022 as shown in Figure IV.7, indicates that the model's predictions are very close to the actual values, A low loss value also suggests that the model is generalizing well and is not overfitting the training data. This outcome is a testament to the efficacy of the model architecture and the high quality of the training data. It should be noted that our model has destabilized, resulting in a need to reassess its performance for the remaining epochs. To mitigate this, activating the early stopping option can be effective.

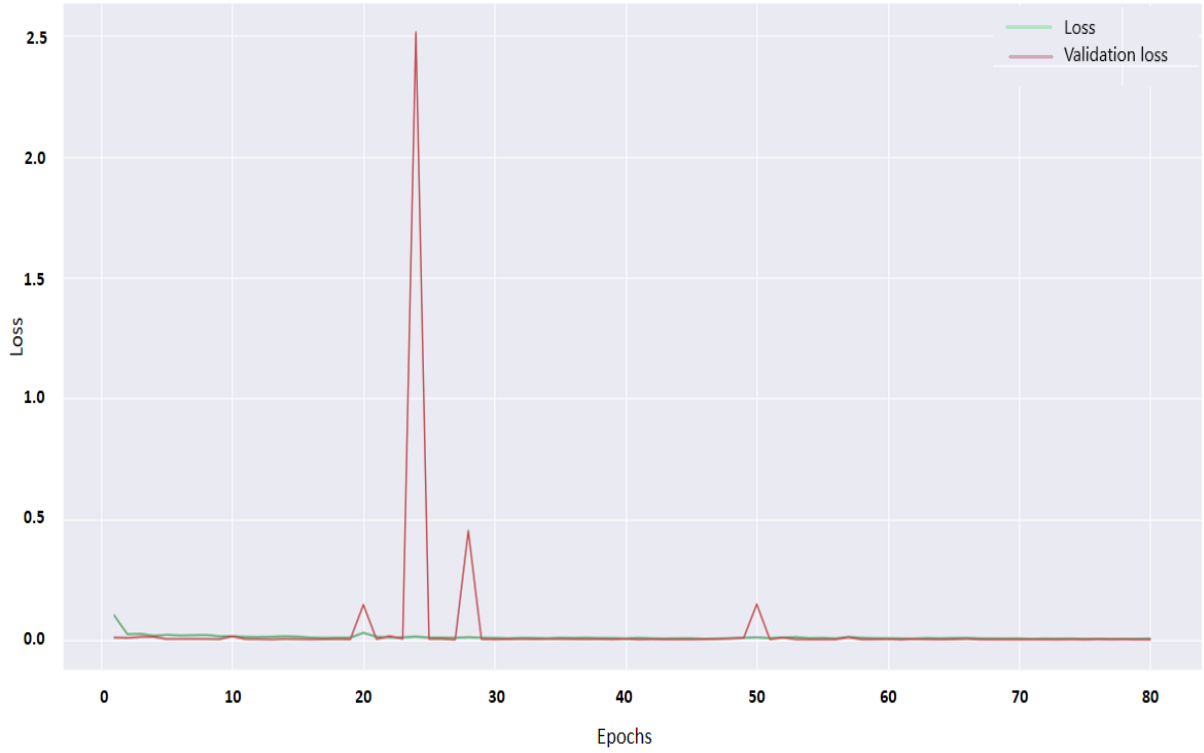


Figure IV.7: Loss and validation loss of the proposed model.

Confusion matrix for model test:

As shown in Table IV.2, the confusion matrix for the classification results indicates that the model achieved an accuracy of almost 100% for all the classes. For each class, the recall and precision are both also almost 100%, which indicates that the model correctly identified all instances of each class, and did not mistakenly classify any non-instances as belonging to any of the classes. The F1 score is also approximately 100% for each class, which implies that the model's performance is optimal, with a balance between precision and recall. It is evident from the calculated values that the False Alarm Rate (FAR) for each class is approximately 0.00, signifying an exceptionally low occurrence of false positives in the classification results. This implies that the model has demonstrated a high level of accuracy in correctly identifying negative instances for all classes.

Looking at the Table IV.2, we can see that for the Benign class, the model correctly identified 4892 instances and did not falsely classify any non-Benign instances as benign. However, the model mistakenly classified 6 Benign instances as non-Benign. For the FTP-BruteForce class, the model correctly identified 4943 instances and only falsely classified 2 non-FTP-BruteForce instances as FTP-BruteForce. Similarly, for the SSH-Bruteforce class, the model correctly identified 5124 instances and only falsely classified 1 non-SSH-Bruteforce instance as SSH-Bruteforce.

For the DoS attacks-GoldenEye class, the model correctly identified 4974 instances and only falsely classified 1 non-DoS attacks-GoldenEye instance as DoS attacks-GoldenEye. However, for the DoS attacks-Slowloris class, the model falsely classified 3 non-DoS attacks-Slowloris instances as DoS attacks-Slowloris and also missed 1 instance of DoS attacks-Slowloris.

Table IV.2: Confusion matrix and classification results for the model test.

Class	Benign	FTP-BruteForce	SSH-Bruteforce	DoS-GoldenEye	DoS-Slowloris
TP	4892	4943	5124	4974	5060
FP	0	2	1	1	3
FN	6	0	0	0	1
TN	20102	20055	19875	20025	19936
Accuracy	0.99976	0.99994	0.99995	0.99995	0.99988
Precision	1.00	0.99960	0.99980	0.99980	0.99941
Recall	0.99877	1.00	1.00	1.00	0.99980
F1 Score	0.99939	0.99980	0.99990	0.99990	0.99961
FAR	0	0.0001	0.00005	0.00005	0.00015

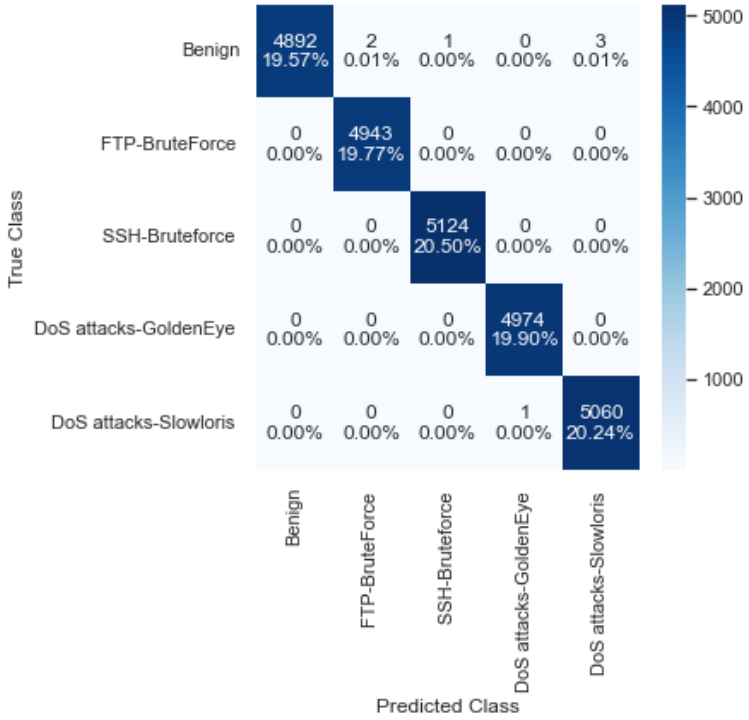


Figure IV.8: Confusion matrix of the proposed model.

IV.5 Transfer learning to other model

Transfer learning (TL) is a powerful machine learning technique that involves utilizing the knowledge gained from training one model to improve the performance of another model on a related task. It is particularly useful in the context of deep learning, where training models with a large number of parameters can be computationally expensive and time-consuming. In TL, we can reuse the pre-trained layers of an existing model as a starting point for training a new model on a different task. This allows the new model (so-called target model) to leverage the learned feature representations of the pre-trained model (so-called source model), which are typically generaliz-

able across different tasks [83, 84], One of the primary benefits of using transfer learning is the reduced training time required to achieve good results. By reusing pre-trained layers, we can significantly reduce the number of parameters that need to be trained, which can make the training process faster and more efficient. Additionally, transfer learning can improve the accuracy of our model, as the learned feature representations can be highly informative for the new task. This can be especially beneficial when working with small datasets, as the pre-trained model can help the new model generalize better to new data.

IV.5.1 Trained attacks

In our target model, we have incorporated five different attack types, three of which were already present in our source model. These attacks include benign, FTP-bruteForce, and SSH-bruteForce. While their meanings are already familiar, we have also introduced a new attack type called DDoS, which stands for Distributed Denial of Service, is a malicious attempt to disrupt the regular functioning of a network, service, or website by overwhelming it with a flood of internet traffic. Attackers typically use multiple compromised devices to launch a coordinated attack, causing the targeted system to become inaccessible to legitimate users. By including the DDoS attack type in our target model, we have expanded our understanding and preparedness to combat this specific form of cyber threat.

We have included two specific types of DDoS attacks: DDoS attack-HOIC and DDoS attack-LOIC-UDP.

- **DDoS attack-HOIC:** The DDoS attack-HOIC refers to the high orbit ion cannon attack, which is a type of DDoS attack commonly employed by malicious actors to disrupt the normal operations of a network, service, or website. HOIC functions by coordinating a large number of computers, forming a botnet, to simultaneously flood the target with an overwhelming amount of traffic. This attack type aims to exhaust the target's bandwidth, server resources, or other critical components, rendering the system unresponsive to legitimate users. The HOIC attack is typically characterized by its high intensity and volume of traffic, making it challenging for targeted entities to handle the influx effectively. The use of this attack can lead to severe disruptions, financial losses, and reputational damage for the affected organization.

- **DDoS attack-LOIC-UDP:** The DDoS attack-LOIC-UDP refers to the low orbit ion cannon attack that specifically targets the user datagram protocol (UDP) during a DDoS attack. LOIC-UDP is a variant of the LOIC tool, which is widely known for its use in launching DDoS attacks. Unlike other protocols, UDP does not establish a connection before sending data, making it vulnerable to exploitation in a DDoS attack. The LOIC-UDP attack floods the target's network connections with a massive volume of UDP packets, overwhelming the network infrastructure and causing disruptions in the target's services. By saturating the target's network capacity and consuming resources, the LOIC-UDP attack aims to render the system unresponsive and inaccessible to legitimate users.

The key difference between a DoS attack and a DDoS attack lies in the scale and method of execution. Corresponding to figure IV.9, a DoS attack is carried out by a single source, typically a single computer or network, with the intention of overwhelming a targeted system or network to the point where it becomes inaccessible to legitimate users. In a DoS attack, the attacker exploits vulnerabilities in the target's resources, such as bandwidth, processing power, or memory, by flooding it with an excessive amount of traffic or by exploiting specific weaknesses in the target's software or configuration. DoS attacks can disrupt services temporarily or indefinitely, depending on the target's ability to handle the attack.

On the other hand, a DDoS attack is more sophisticated and powerful because it leverages a distributed network of compromised computers, known as a botnet, to launch the attack. These compromised machines, under the control of the attacker, are coordinated to flood the target with a massive volume of traffic simultaneously, overwhelming its resources. The distributed nature of a DDoS attack makes it more challenging to mitigate and defend against since the traffic originates from various sources, making it harder to identify and block.

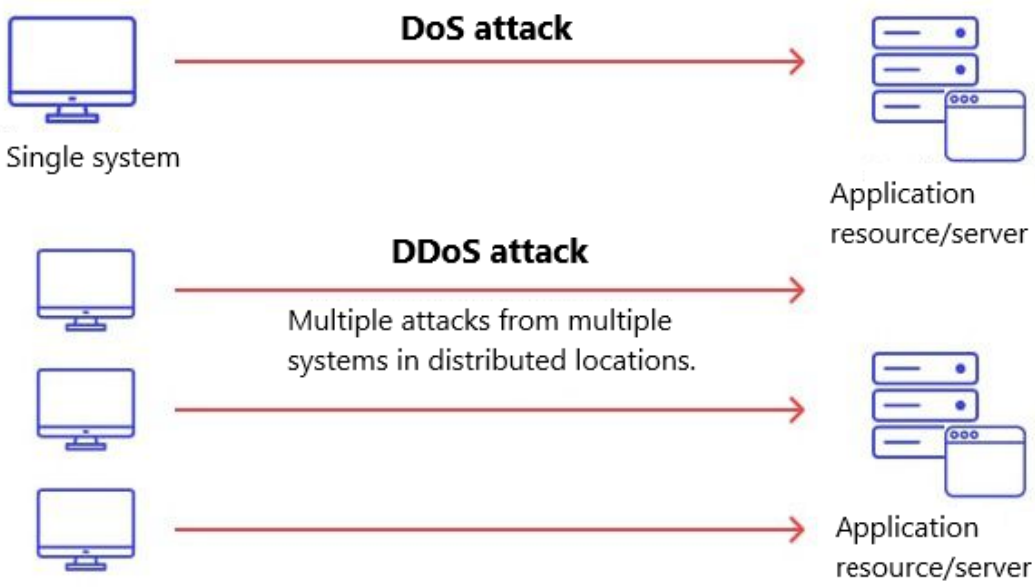


Figure IV.9: The difference between DoS attack and DDoS attack.

In Figure IV.10, data visualization using bar plot is used to summarize the amount of attacks used in our dataset, This latter includes 1028459 instances of benign, 193360 instances of attack type FTP-BruteForce, 187589 instances of attack type SSH-Bruteforce, 686012 instances of attack type DDoS-HOIC, and 1730 instances of attack type DDoS-LOIC-UDP.

To mitigate data bias, it is essential to employ data augmentation techniques as we used on the source's studied attacks , which aid in eliminating bias and achieving equitable distributions. After performing resampling on the attack types, we observed the resulting representation depicted in Figure IV.11.

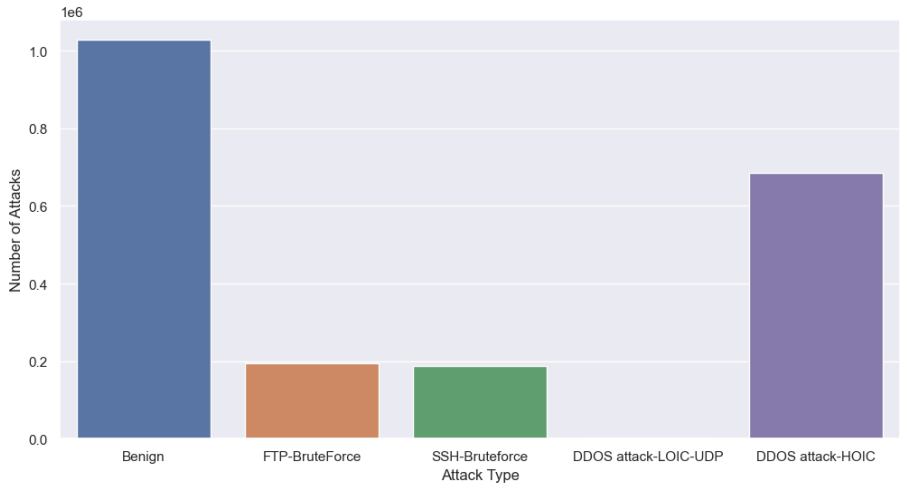


Figure IV.10: Attacks used in the target model.

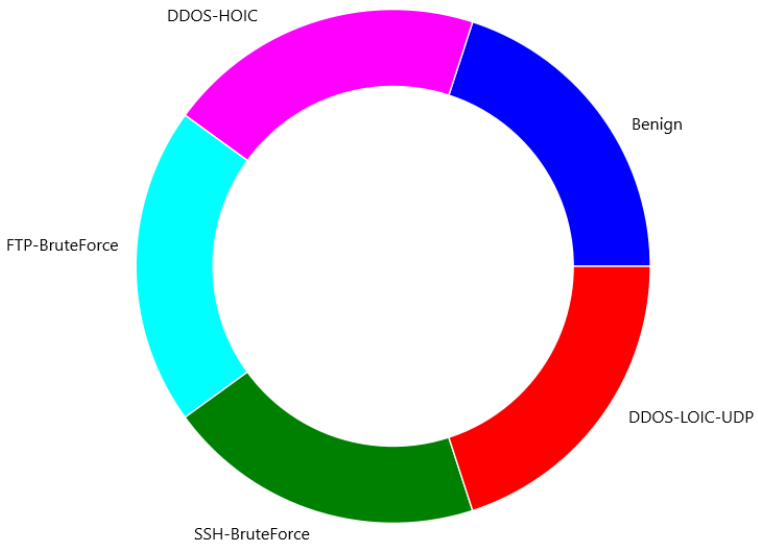


Figure IV.11: Intrusion attacks distribution for the target model.

IV.5.2 Proposed architecture for the target model

In our proposed architecture, we have used Inductive TL where knowledge gained from a source task is transferred to a target task, typically in a different domain. The goal is to improve the performance of the target task by leveraging the knowledge learned from the source task [83,84]. We used a pre-trained model and added two dense layers. The dense layer is the final layer in the model, and it has five classifications for new attacks with a softmax activation. The dense layer is responsible for learning high-level representations of the input data and making predictions based on those representations.

In our target model, as shown in Figure IV.12, we have a total of 91,600 parameters, with 4,869 of them being trainable and 86,731 being non-trainable. Before using fine-tuning, the

non-trainable parameters may have been pre-trained on a large dataset with a similar task. These pre-trained parameters may capture generic patterns in the data that can be useful for the specific task at hand.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 72, 1)]	0	[]
input_2 (InputLayer)	[(None, 72, 1)]	0	[]
model_6 (Functional)	(None, 5)	86731	['input_1[0][0]', 'input_2[0][0]']
dense (Dense)	(None, 64)	384	['model_6[0][0]']
dense_1 (Dense)	(None, 64)	4160	['dense[0][0]']
dense_2 (Dense)	(None, 5)	325	['dense_1[0][0]']

Total params: 91,600
Trainable params: 4,869
Non-trainable params: 86,731

Figure IV.12: Target model summary before using fine-tuning.

Model build and fit:

```
source_model.trainable=False
opt=keras.optimize.Adam(lr=0.001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics= ['accu-
racy'])
model.fit([X_train, X_train], y_train, validation_data=([X_test, X_test],
y_test), epochs=50, batch_size=128)
```

This code involves training a neural network model using Keras. It starts by setting the trainable attribute of a pre-existing model named source model to false, indicating that its weights and parameters will not be updated during training. Next, an Adam optimizer with a learning rate of 0.001 is instantiated and assigned to the variable opt. The model is then compiled using this optimizer, specifying the loss function as categorical cross-entropy and the evaluation metric as accuracy. The model is trained using the fit function, providing the training data, and validation data, the number of training epochs is 50, and the batch size is 128.

Model validation accuracy :

The model validation accuracy before fine-tuning for our target model was 60.41% after 50 epochs, as shown in Figure IV.13. This indicates that the model has some level of proficiency in the given task, but the accuracy is relatively low, suggesting that the model may not have learned the underlying patterns and features in the training data well enough to generalize to new data.

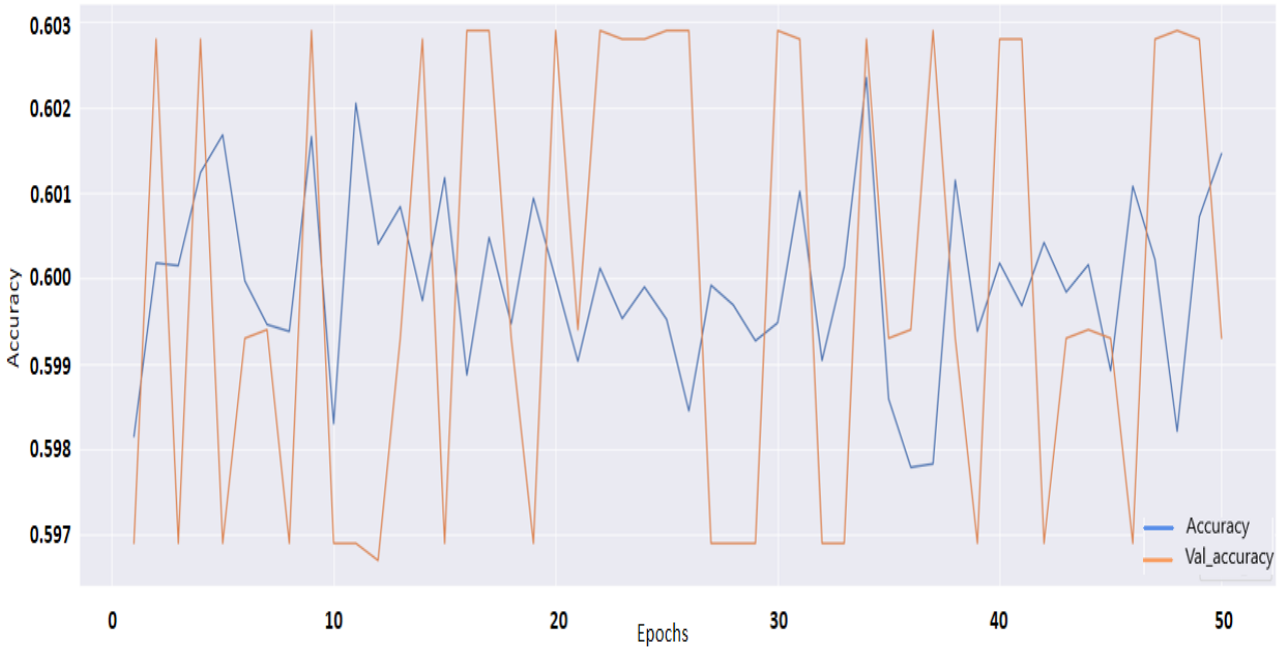


Figure IV.13: Accuracy and validation accuracy before using fine-tuning.

Model validation loss :

The model validation loss before fine-tuning for our target model was 0.6600 after 50 epochs, as shown in Figure IV.14, indicating that the model may still have the need for improvement.

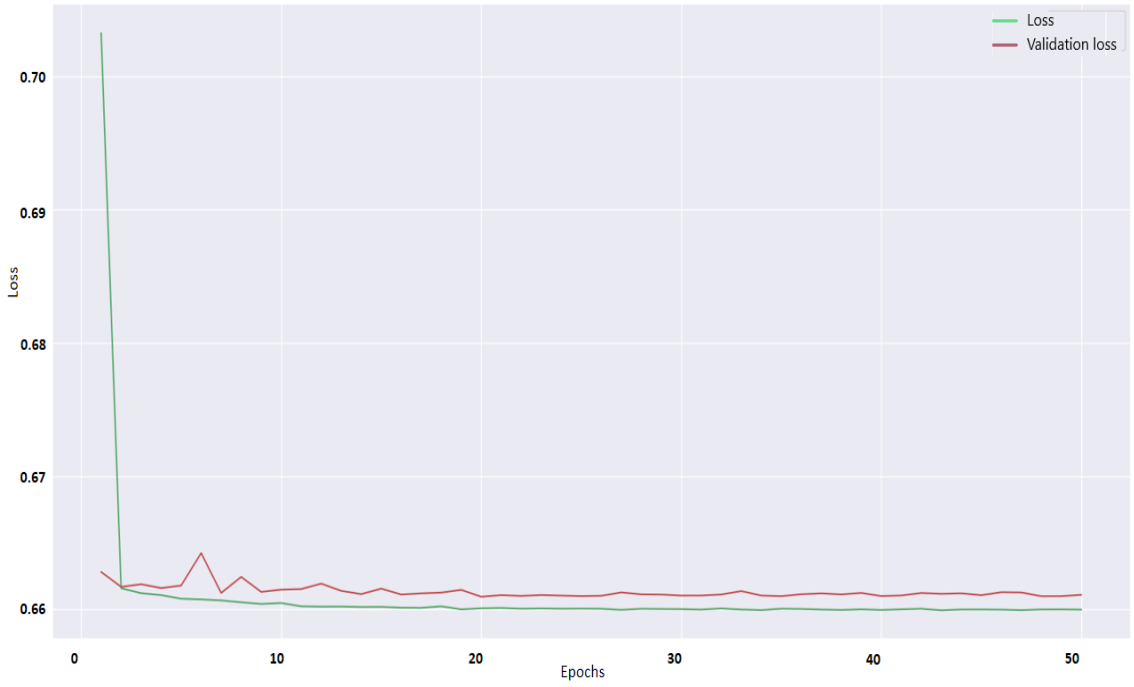


Figure IV.14: Loss and validation loss before using fine-tuning.

IV.5.3 Fine-tuning

In our TL model, we applied fine-tuning with two dense layers. Fine-tuning includes unfreezing some or all of the model's pre-trained layers before retraining it for a target task at a slower learning rate [85].

One of the advantages of fine-tuning is that it can help the model achieve even better performance, as it allows the model to adapt the learned feature representations to the specific requirements of the task. This can be especially useful when working with complex or highly specific tasks, where the learned feature representations may need to be refined to capture the nuances of the data. Additionally, fine-tuning can help prevent overfitting, as the pre-trained layers provide a regularization effect that can help the model generalize better to new data.

After using fine-tuning, the target model has undergone significant changes. In our case, we have taken a pre-trained model with a total of 91,600 parameters, of which only 4,869 were trainable, and we have adapted it to our specific task by fine-tuning it. As a result, the number of trainable parameters has increased significantly to 90,990, indicating that the model has adapted more closely to our specific task. At the same time, the number of non-trainable parameters has decreased from 86731 to 610, indicating that the model has retained much better as can be seen in Figure IV.15. Overall, these changes suggest that our target model has been optimized for our specific task through fine-tuning.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 72, 1)]	0	[]
input_2 (InputLayer)	[(None, 72, 1)]	0	[]
model_6 (Functional)	(None, 5)	86731	['input_1[0][0]', 'input_2[0][0]']
dense (Dense)	(None, 64)	384	['model_6[0][0]']
dense_1 (Dense)	(None, 64)	4160	['dense[0][0]']
dense_2 (Dense)	(None, 5)	325	['dense_1[0][0]']

Total params: 91,600
Trainable params: 90,990
Non-trainable params: 610

Figure IV.15: Target model summary after using fine-tuning.

Model build and fit:

The pre-existing model source model is set as trainable in this code, allowing for the updating of its parameters and weights throughout training. A variable called opt is given an adam opti-

mizer that has a learning rate of 0.0001. This optimizer is then used to build the model, with accuracy set as the evaluation metric and categorical cross-entropy as the loss function. With the training data, validation data, number of training epochs 150, and batch size 128 provided, the model is trained using the fit function.

```
source_model.trainable = True
opt=keras.optimizers.Adam(lr=0.0001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics= ['accu-
racy'])
model.fit([X_train, X_train], y_train, validation_data=([X_test, X_test],
y_test), epochs=150, batch_size=128)
```

Model validation accuracy:

After applying fine-tuning to our transfer learning model, the proposed model achieved a significantly improved validation accuracy of 99.85% after 150 epochs, as demonstrated in Figure IV.16. However, it is worth noting that epoch 20 serves as a significant achievement where we have already achieved a high level of accuracy, providing an opportunity to pause and consider the potential benefits of further training. This high validation accuracy suggests that the model has learned the underlying patterns and features of the training data, and exhibits strong generalization capabilities when applied to new data. The fine-tuning process allowed the model to adapt its pre-trained feature representations to the specific requirements of the target task, resulting in improved performance.

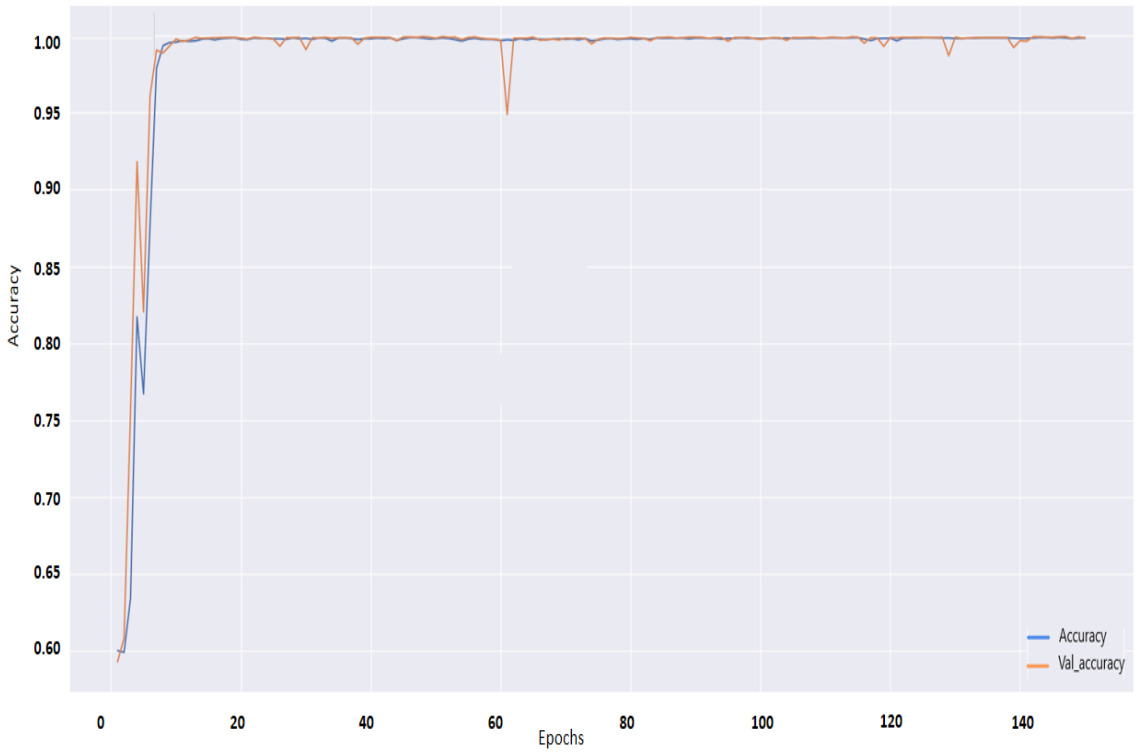


Figure IV.16: Accuracy and validation accuracy of target model after using fine-tuning.

Model validation loss:

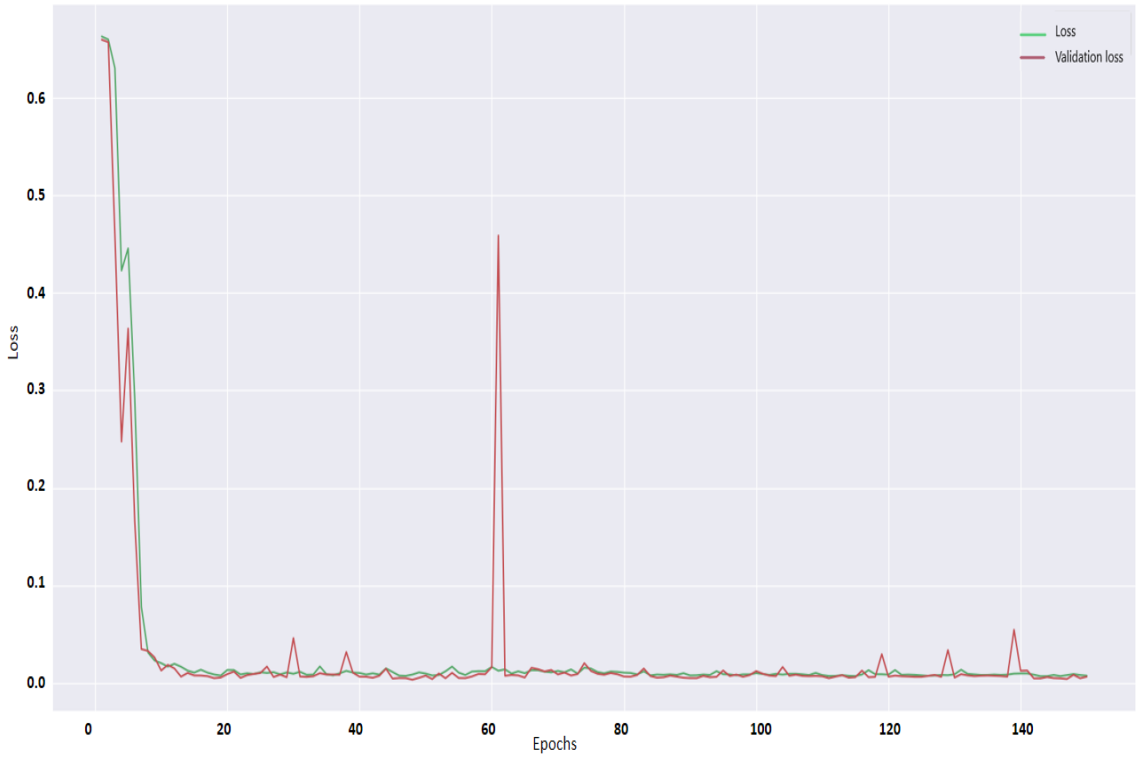


Figure IV.17: Loss and validation loss after using fine-tuning.

Applying fine-tuning to our TL model not only significantly improved the model's validation accuracy, but also reduced the validation loss to 0.0068, as depicted in Figure IV.17. This lower validation loss suggests that the model has improved its ability to generalize and make accurate predictions on unseen data., suggesting that it has learned more robust and accurate feature representations. Notice that our model has destabilized on epoch 61 and recap it performance for the rest of epochs. This latter can be prevented when early stopping option is activated.

Confusion matrix for model test:

When analyzing the confusion matrix shown in Table IV.3, for the Benign class, the model correctly identified 2029 instances as benign TP, with no FP. There were 15 instances of benign traffic misclassified as non-benign FN, while 7956 instances were correctly classified as non-benign TN. The accuracy for this class was 0.9985, indicating that the model achieved perfect classification accuracy. The precision, recall, and F1 score were also high, with values of 1.00, 0.99269, and 0.99634, respectively, reflecting the model's ability to accurately classify benign traffic. Moving to the DDoS attack-HOIC class, the model correctly identified 1919 instances of this attack type TP and had 12 FP, indicating instances of non-attack traffic being misclassified. There were no FN recorded, indicating that all instances of this attack type were correctly classified. The TN count was 8069. The accuracy, precision, recall, and F1 score for this class were 0.9988, 0.9938, 1.00, and 0.9969, respectively, demonstrating the model's high performance in detecting DDoS attack-HOIC. For the FTP-BruteForce and SSH-Bruteforce classes, the model achieved perfect accuracy, precision, recall, and F1 score. Both classes had no FP or FN, with all instances correctly classified.

The TP DDoS counts were 2035 and 2022, while the TN counts were 7965 and 7976, respectively. These results indicate the model's exceptional capability to accurately identify instances of FTP and SSH brute-force attacks. Lastly, the DDoS attack-LOIC-UDP class had 1980 TP, 1 FP, and no FN. The TN count was 8019. The model achieved perfect accuracy, precision, recall, and F1 score for this class, indicating its effectiveness in identifying instances of DDoS attack-LOIC-UDP.

Table IV.3: Confusion matrix and classification results for the model test.

Class	**Benign**	**DDOS-HOIC**	**FTP-BruteForce**	**SSH-Bruteforce**	**DDOS-LOIC-UDP**
TP	2029	1919	2035	2022	1980
FP	0	12	0	2	1
FN	15	0	0	0	0
TN	7956	8069	7965	7976	8019
Accuracy	0.9985	0.9988	1.00	0.9998	0.9999
Precision	1.00	0.9938	1.00	0.9990	0.9995
Recall	0.99269	1.00	1.00	1.00	1.00
F1 Score	0.99634	0.9969	1.00	0.9995	0.9997
FAR	0	0.0015	0	0.00025	0.000125

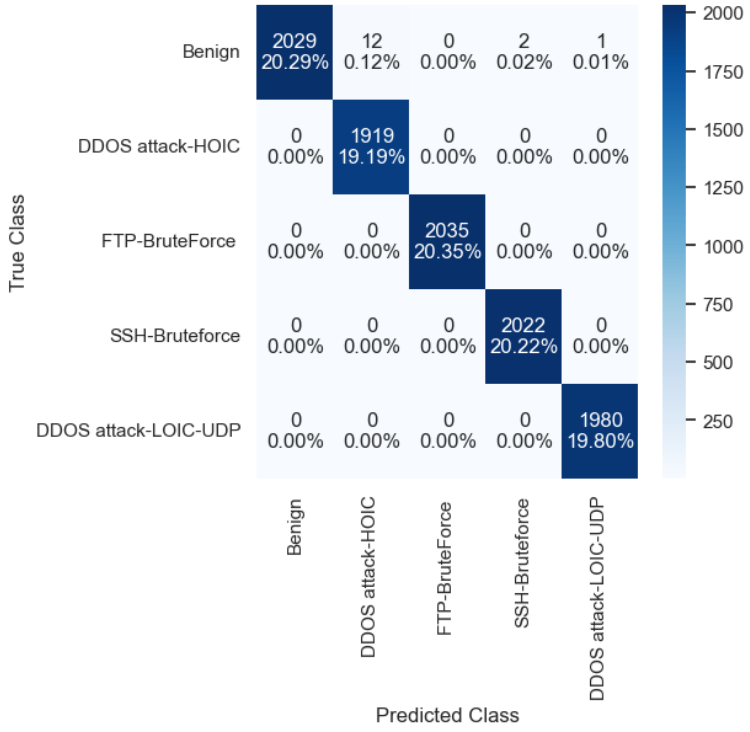


Figure IV.18: Confusion matrix for the proposed target after fine-tuning.

IV.6 Conclusion

Our objective was to enhance the efficacy of the IDS by developing a pre-trained model capable of accurately classifying five distinct types of attacks. Through our efforts, we achieved remarkable results in our proposed model, with a validation accuracy of 99.97% and an impressively low validation loss of 0.0022. Furthermore, we extended the utility of this model by leveraging TL techniques. By employing TL, we effectively harnessed the knowledge and expertise acquired from the pre-trained model and adapted it to the detection of smaller-sized attacks while significantly reducing the required training time. This approach allowed us to overcome the limitations of traditional models that struggle with variations in attack sizes. Additionally, we successfully employed to classify two more new attacks, enabling continuous detection of novel attack variants and further enhancing the capabilities of our model.

Initially, when we applied TL without fine-tuning, the validation accuracy of our target model was 60.41%, accompanied by a validation loss of 0.6600. However, by implementing fine-tuning techniques, we observed a significant improvement in performance. The proposed model, after undergoing fine-tuning, exhibited a remarkable validation accuracy of 99.85%, with a further reduction in validation loss to 0.0068. These findings highlight the effectiveness of fine-tuning in optimizing the model's performance and achieving outstanding accuracy.

Overall, our approach combining pre-training, TL, and fine-tuning has resulted in a highly accurate IDS model capable of accurately classifying attacks. The use of TL not only enabled us to detect a wider range of attacks but also reduced training time, making our system more efficient and versatile in handling various attack types.

CHAPTER V

GENERAL CONCLUSION

Ultimately, the three previous chapters have covered a wide range of topics, delving into the realms of ML, DL, IDS, and the proposed CNNs-based IDS method. In the first chapter, we explored the fundamental concepts of ML and DL. ML involves the development of algorithms that enable computers to learn from data and make predictions or decisions. DL, on the other hand, is a subfield of ML that focuses on training ANNs with multiple layers to extract complex patterns and features from data. Moving forward, the second chapter examined the field of IDS. These systems play a critical role in network security by detecting and responding to unauthorized access or malicious activities. We briefly explored the various techniques and approaches used in intrusion detection, highlighting the importance of accurate and timely detection to mitigate potential threats. In the final chapter, we presented our proposed CNN-based IDS method. Recognizing the limitations of traditional approaches, we leveraged the power of CNNs to enhance the accuracy and effectiveness of intrusion detection. Our approach involved employing an autoencoder to learn representative features from the network traffic data and constructing a CNN architecture for the source model. We then utilized TL to adapt the source model to different target environments. Through our comprehensive analysis and evaluation, we achieved satisfactory results. Our proposed CNN-based IDSs method demonstrated notable improvements in detecting various types of attacks, showcasing the potential for advanced DL techniques in the field of intrusion detection. The integration of autoencoders and TL allowed for the extraction of valuable insights and knowledge from diverse datasets, enhancing the adaptability and robustness of the IDS. Overall, our research has shed light on the significance of ML and DL in the realm of network security. By exploring IDSs and proposing an innovative CNN-based IDS method, we have contributed to the advancement of effective and intelligent detection mechanisms. These findings serve as a foundation for further research and development in the field, enabling us to continually improve network security and protect against emerging threats in the digital landscape.

BIBLIOGRAPHY

[1] Thuy TT Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, 10(4):56–76, 2008.

[2] Hanane Touati, Hamida Rahali, and Sofiane TCHOKETCH KEBIR. Développement d'une technique de reconnaissance gestuelle améliorée basée sur des signaux emg en utilisant les méthodes d'apprentissage en profondeur. 2021.

[3] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[4] Jun Yuan, Changjian Chen, Weikai Yang, Mengchen Liu, Jiazhi Xia, and Shixia Liu. A survey of visual analytics techniques for machine learning. *Computational Visual Media*, 7:3–36, 2021.

[5] Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. *Advances in neural information processing systems*, 28, 2015.

[6] Andrew Ng. Machine learning and ai via brain simulations. *Accessed: May*, 3:2018, 2013.

[7] Shixia Liu, Gennady Andrienko, Yingcai Wu, Nan Cao, Liu Jiang, Conglei Shi, Yu-Shuen Wang, and Seokhee Hong. Steering data quality with visual analytics: The complexity challenge. *Visual Informatics*, 2(4):191–197, 2018.

[8] Milan Picard, Marie-Pier Scott-Boyer, Antoine Bodein, Olivier Périn, and Arnaud Droit. Integration strategies of multi-omics data for machine learning analysis. *Computational and Structural Biotechnology Journal*, 19:3735–3746, 2021.

[9] Jordan Ott, Mike Pritchard, Natalie Best, Erik Linstead, Milan Curcic, and Pierre Baldi. A fortran-keras deep learning bridge for scientific computing. *Scientific Programming*, 2020:1–13, 2020.

[10] Neil D Bennett, Barry FW Croke, Giorgio Guariso, Joseph HA Guillaume, Serena H Hamilton, Anthony J Jakeman, Stefano Marsili-Libelli, Lachlan TH Newham, John P Norton, Charles

Perrin, et al. Characterising performance of environmental models. *Environmental modelling & software*, 40:1–20, 2013.

[11] Tal Yarkoni and Jacob Westfall. Choosing prediction over explanation in psychology: Lessons from machine learning. *Perspectives on Psychological Science*, 12(6):1100–1122, 2017.

[12] A. Géron. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.

[13] Yong Wang, Margaret Martonosi, and Li-Shiuan Peh. A supervised learning approach for routing optimizations in wireless sensor networks. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 79–86, 2006.

[14] Asmaa KOSSEIR and Mohamed MAAZOUZ. Développement d'un système de contrôle d'accès à un parc privé. 2020.

[15] Chérifa BOUCETTA, Hassine MOUNGLA, and Hossam AFIFI. L'apport de l'apprentissage automatique pour résoudre les contraintes liées à la mobilité dans le cadre des communications d2d. *Gestion du niveau de service dans les environnements émergents*, page 211, 2020.

[16] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons. b*, 4:51–62, 2017.

[17] Dastan Maulud and Adnan M Abdulazeez. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 1(4):140–147, 2020.

[18] Subal C Kumbhakar. Modeling allocative inefficiency in a translog cost function and cost share equations: an exact relationship. *Journal of Econometrics*, 76(1-2):351–356, 1997.

[19] Aatila Mustapha, Lachgar Mohamed, and Kartit Ali. An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field. In *Smart Applications and Data Analysis: Third International Conference, SADASC 2020, Marrakesh, Morocco, June 25–26, 2020, Proceedings 3*, pages 349–359. Springer, 2020.

[20] Eva Ostertagová. Modelling using polynomial regression. *Procedia Engineering*, 48:500–506, 2012.

[21] Matjaž Kukar and Igor Kononenko. Reliable classifications with machine learning. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*, pages 219–231. Springer, 2002.

[22] Ibrahim Mahmood Ibrahim and Adnan Mohsin Abdulazeez. The role of machine learning algorithms for diagnosing diseases. *learning*, 4(5):6, 2021.

[23] Shihab Elbagir Saad and Jing Yang. Twitter sentiment analysis based on ordinal regression. *IEEE Access*, 7:163677–163685, 2019.

[24] S Reema Sree, SB Vyshnavi, and N Jayapandian. Real-world application of machine learning and deep learning. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 1069–1073. IEEE, 2019.

[25] Shweta Srivastava. Weka: a tool for data preprocessing, classification, ensemble, clustering and association rule mining. *International Journal of Computer Applications*, 88(10), 2014.

[26] Mikko I Malinen and Pasi Fränti. Balanced k-means for clustering. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+ SSPR 2014, Joensuu, Finland, August 20-22, 2014. Proceedings*, pages 32–41. Springer, 2014.

[27] Mohssen Mohammed, Muhammad Badruddin Khan, and Eihab Bashier Mohammed Bashier. *Machine learning: algorithms and applications*. Crc Press, 2016.

[28] Elisabeth Sulmont, Elizabeth Patitsas, and Jeremy R Cooperstock. What is hard about teaching machine learning to non-majors? insights from classifying instructors' learning goals. *ACM Transactions on Computing Education (TOCE)*, 19(4):1–16, 2019.

[29] S Sivagama Sundhari. A knowledge discovery using decision tree by gini coefficient. In *2011 International Conference on Business, Engineering and Industrial Applications*, pages 232–235. IEEE, 2011.

[30] Amani Hanaa MADANI, Hamza KHEDDAR, et al. Steganalysis of the rlpfs method using support vector machine. 2020.

[31] Hamza Kheddar, Merouane Bouzid, and David Megías. Pitch and fourier magnitude based steganography for hiding 2.4 kbps melp bitstream. *IET Signal Processing*, 13(3):396–407, 2019.

[32] S Amarappa and SV Sathyanarayana. Data classification using support vector machine (svm), a simplified approach. *Int. J. Electron. Comput. Sci. Eng*, 3:435–445, 2014.

[33] Ed Burns. In-depth guide to machine learning in the enterprise. *Techtarget. March*, page 17, 2021.

[34] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Eric Horvitz. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[35] Mohammad Hasanzadeh Mofrad, Rami Melhem, Yousuf Ahmad, and Mohammad Hammoud. Multithreaded layer-wise training of sparse deep neural networks using compressed sparse column. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2019.

[36] Mourad Benaferi Mourad, Youcef Benabderrahmane, and AHMED CHERIF MAZARI. *MALARIA DETECTION IN HUMAN CELLS USING DEEP LEARNING*. PhD thesis, 2020.

[37] Anders Krogh. What are artificial neural networks? *Nature biotechnology*, 26(2):195–197, 2008.

[38] Surajit Chattopadhyay. Feed forward artificial neural network model to predict the average summer-monsoon rainfall in india. *Acta Geophysica*, 55:369–382, 2007.

[39] Radoslaw M Cichy and Daniel Kaiser. Deep neural networks as scientific models. *Trends in cognitive sciences*, 23(4):305–317, 2019.

[40] Abdulmajid Murad and JY Pyun. Deep recurrent neural networks for human, 2017.

[41] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[42] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

[43] Shui-Hua Wang, Khan Muhammad, Jin Hong, Arun Kumar Sangaiah, and Yu-Dong Zhang. Alcoholism identification via convolutional neural network based on parametric relu, dropout, and batch normalization. *Neural Computing and Applications*, 32:665–680, 2020.

[44] Asmida Ismail, Siti Anom Ahmad, Azura Che Soh, Khair Hassan, and Hazreen Haizi Harith. Improving convolutional neural network (cnn) architecture (minivggnet) with batch normalization and learning rate decay factor for image classification. *International Journal of Integrated Engineering*, 11(4), 2019.

[45] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[46] Mohamed Elleuch, Rania Maalej, and Monji Kherallah. A new design based-svm of the cnn classifier architecture with dropout for offline arabic handwritten recognition. *Procedia Computer Science*, 80:1712–1723, 2016.

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

[48] Mukul Soni, Mayank Singhal, Rahul Katarya, et al. A systematic survey on recent deep learning trends in intrusion detection system. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 1927–1932. IEEE, 2021.

[49] Chaitanya Gupta, Ishita Johri, Kathiravan Srinivasan, Yuh-Chung Hu, Saeed Mian Qaisar, and Kuo-Yi Huang. A systematic review on machine learning and deep learning models for electronic information security in mobile networks. *Sensors*, 22(5):2017, 2022.

[50] Mostaque Md Hassan et al. Current studies on intrusion detection system, genetic algorithm and fuzzy logic. *arXiv preprint arXiv:1304.3535*, 2013.

[51] Eirini Anthi, Lowri Williams, Małgorzata Słowińska, George Theodorakopoulos, and Pete Burnap. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, 2019.

[52] Ansam Khraisat and Ammar Alazab. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity*, 4:1–27, 2021.

[53] Laisen Nie, Zhaolong Ning, Xiaojie Wang, Xiping Hu, Jun Cheng, and Yongkang Li. Data-driven intrusion detection for intelligent internet of vehicles: A deep convolutional neural network-based method. *IEEE Transactions on Network Science and Engineering*, 7(4):2219–2230, 2020.

[54] Rashmi Ravindra Chaudhari and Sonal Pramod Patil. Intrusion detection system: classification, techniques and datasets to implement. *International Research Journal of Engineering and Technology (IRJET)*, 4(2):1860–1866, 2017.

[55] Rebecca Gurley Bace, Peter Mell, et al. Intrusion detection systems. 2001.

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

[57] Rebecca Bace and Peter Mell. Nist special publication on intrusion detection systems. Technical report, Booz-allen and Hamilton Inc MCLEAN VA, 2001.

[58] Giovanni Vigna and Richard A Kemmerer. Netstat: A network-based intrusion detection system. *Journal of computer security*, 7(1):37–71, 1999.

[59] EFE Ahmet and İrem Nur ABACI. Comparison of the host based intrusion detection systems and network based intrusion detection systems. *Celal Bayar University Journal of Science*, 18(1):23–32, 2022.

[60] Tiina Kovanen, Gil David, and Timo Hämäläinen. Survey: Intrusion detection systems in encrypted traffic. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 16th International Conference, NEW2AN 2016, and 9th Conference, ruSMART 2016, St. Petersburg, Russia, September 26-28, 2016, Proceedings 16*, pages 281–293. Springer, 2016.

[61] Om Ambavkar, Prathmesh Bharti, Amit Chaurasiya, Roshan Chauhan, and Mahalaxmi Palinje. Review on ids based on ml algorithms.

[62] V Kotov and V Vasilyev. A survey of modern advances in network intrusion detection. In *13th International Workshop on Computer Science and Information Technologies (CSIT'2011)*, pages 18–21. Citeseer, 2011.

[63] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.

[64] Suzan Hajj, Rayane El Sibai, Jacques Bou Abdo, Jacques Demerjian, Abdallah Makhoul, and Christophe Guyeux. Anomaly-based intrusion detection systems: The requirements, methods, measurements, and datasets. *Transactions on Emerging Telecommunications Technologies*, 32(4):e4240, 2021.

[65] Asma Shaikh and Preeti Gupta. Advanced signature-based intrusion detection system. In *Intelligent Communication Technologies and Virtual Mobile Networks: Proceedings of ICICV 2022*, pages 305–321. Springer, 2022.

[66] Yanxin Wang. *A hybrid intrusion detection system*. Iowa State University, 2004.

[67] Praveen Kumar Rajendran, B Muthukumar, and G Nagarajan. Hybrid intrusion detection system for private cloud: a systematic approach. *Procedia Computer Science*, 48:325–329, 2015.

[68] Ehab S Al-Shaer and Hazem H Hamed. Modeling and management of firewall policies. *IEEE Transactions on network and service management*, 1(1):2–10, 2004.

[69] Abid Hussain. Use of firewall and ids to detect and prevent network attacks. *International Journal of Technical Research & Science*, 3:289–292, 2018.

[70] Nilotpal Chakraborty. Intrusion detection system and intrusion prevention system: A comparative study. *International Journal of Computing and Business Research (IJCBR)*, 4(2):1–8, 2013.

[71] Basant Subba, Santosh Biswas, and Sushanta Karmakar. A neural network based system for intrusion detection and attack classification. In *2016 Twenty Second National Conference on Communication (NCC)*, pages 1–6. Ieee, 2016.

[72] Rawaa Ismael Farhan, Abeer Tariq Maolood, and NidaaFlaih Hassan. Performance analysis of flow-based attacks detection on cse-cic-ids2018 dataset using deep learning. *Indonesian Journal of Electrical Engineering and Computer Science*, 20(3):1413, 2020.

[73] V Kanimozhi and T Prem Jacob. Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. *International Journal of Engineering Applied Sciences and Technology*, 4(6):2455–2143, 2019.

[74] Brian Lee, Sandhya Amaresh, Clifford Green, and Daniel Engels. Comparative study of deep learning models for network intrusion detection. *SMU Data Science Review*, 1(1):8, 2018.

[75] Laurens Hellemons, Luuk Hendriks, Rick Hofstede, Anna Sperotto, Ramin Sadre, and Aiko Pras. Sshcure: a flow-based ssh intrusion detection system. In *Dependable Networks and Services: 6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2012, Luxembourg, Luxembourg, June 4-8, 2012. Proceedings 6*, pages 86–97. Springer, 2012.

[76] Jeonghoon Park, Jinsu Kim, BB Gupta, and Namje Park. Network log-based ssh brute-force attack detection model. *CMC-Computers Materials & Continua*, 68(1):887–901, 2021.

[77] Dawn Xiaodong Song, David A Wagner, Xuqing Tian, et al. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001.

[78] Etienne van de Bijl, Jan Klein, Joris Pries, Rob van der Mei, and Sandjai Bhulai. Detecting novel variants of application layer (d) dos attacks using supervised learning.

[79] Edwin Fenil and P Mohan Kumar. Survey on ddos defense mechanisms. *Concurrency and Computation: Practice and Experience*, 32(4):e5114, 2020.

[80] Muhammad Yeasir Arafat, Muhammad Morshed Alam, and Mohammad Fakrul Alam. A practical approach and mitigation techniques on application layer ddos attack in web server. *International Journal of Computer Applications*, 131(1):13–20, 2015.

[81] John Hancock, Taghi M Khoshgoftaar, and Joffrey L Leevy. Detecting ssh and ftp brute force attacks in big data. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 760–765. IEEE, 2021.

[82] Hamza Kheddar and David Megías. High capacity speech steganography for the g723. 1 coder based on quantised line spectral pairs interpolation and cnn auto-encoding. *Applied Intelligence*, 52(8):9441–9459, 2022.

[83] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[84] Hamza Kheddar, Yassine Himeur, Somaya Al-Maadeed, Abbes Amira, and Faycal Bensaali. Deep transfer learning for automatic speech recognition: Towards better generalization. *arXiv preprint arXiv:2304.14535*, 2023.

[85] Abdel-Aziz Binguitcha-Fare and Prince Sharma. Crops and weeds classification using convolutional neural networks via optimization of transfer learning parameters. *Int J Eng Adv Technol (IJEAT)*, 8(5):2249–8958, 2019.