



January 2023

## Performance Analysis Of Data-Driven Algorithms In Detecting Intrusions On Smart Grid

Tala Talaei Khoei

[How does access to this work benefit you? Let us know!](#)

Follow this and additional works at: <https://commons.und.edu/theses>

---

### Recommended Citation

Talaei Khoei, Tala, "Performance Analysis Of Data-Driven Algorithms In Detecting Intrusions On Smart Grid" (2023). *Theses and Dissertations*. 5707.  
<https://commons.und.edu/theses/5707>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, and Senior Projects at UND Scholarly Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UND Scholarly Commons. For more information, please contact [und.common@library.und.edu](mailto:und.common@library.und.edu).

PERFORMANCE ANALYSIS OF DATA-DRIVEN  
ALGORITHMS IN DETECTING INTRUSIONS ON  
SMART GRID

by

Tala Talaei Khoei

A Dissertation

Submitted to the Graduate Faculty of the  
University of North Dakota

In partial fulfillment of the requirements  
for the degree of  
Doctor of Philosophy

Grand Forks, North Dakota  
December 2023

Copyright 2023 Tala Talaei Khoei

Name: Tala Talaei Khoei  
Degree: Doctor of Philosophy

This document, submitted in partial fulfillment of the requirements for the degree from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

DocuSigned by:  
  
Naïma Kaabouch

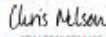
DocuSigned by:  
  
Wen-Chen Hu

DocuSigned by:  
  
Tarek Eiderini

DocuSigned by:  
  
Dr. Ertan Ozturk

DocuSigned by:  
  
Dr. Sattar Dorafshan

This document is being submitted by the appointed advisory committee as having met all the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

DocuSigned by:  
  
Chris Nelson  
Dean of the School of Graduate Studies  
12/1/2023  
Date

## PERMISSION

Title	Performance Analysis of Data-driven Algorithms in Detecting Intrusions on Smart Grid
Department	Electrical Engineering
Degree	Doctor of Philosophy

In presenting this dissertation in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University can make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my dissertation work or, in her absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this dissertation or part thereof for financial gain shall not be allowed without my written permission and that of my advisor. It is also understood that due recognition shall be given to me and to my advisor in any scholarly use which may be made of any material in my dissertation.

Tala Talaei Khoei

12/05/2023

# TABLE OF CONTENTS

LIST OF FIGURES .....	x
LIST OF TABLES.....	xii
ACKNOWLEDGEMENTS .....	xiii
ABSTRACT .....	xv
<b>Chapter 1 INTRODUCTION .....</b>	<b>1</b>
1.1 Motivation and Problem Statement .....	1
1.2 Dissertation Goal and Objectives .....	3
1.3 Contributions .....	3
1.4 Dissertation Organization.....	5
<b>Chapter 2 OVERVIEW OF SMART GRID .....</b>	<b>7</b>
2.1 Smart Grid Network .....	7
2.1.1 Smart Grid Architecture.....	8
2.1.2 Smart Grid Features .....	10
2.1.3 Smart Grid Applications .....	11
2.1.4 Cyber-Security Requirements in Smart Grid .....	13
2.2 Cyber-attacks on Smart Grid.....	15
2.2.1 Cyber-Attacks Targeting Sensor .....	15
2.2.1.1 Intrusion Attacks.....	15
2.2.1.2 Spoofing Attacks .....	16
2.2.1.3 Injection Attacks.....	16
2.2.1.4 Denial of Service Attacks .....	16
2.2.1.5 Time Synchronization Attacks .....	16

2.2.1.6 Jamming Attacks .....	17
2.2.2 Cyber-Attacks Targeting Computer .....	18
2.2.2.1 Brute-Force Attacks.....	19
2.2.2.2 Social Engineering Attacks.....	19
2.2.2.3 Viruses/Malware Attacks.....	20
2.2.2.4 Horse Trojan Attacks.....	20
2.2.2.5 Popping Human Interface Attacks .....	21
2.2.3 Cyber-Attacks Targeting Network Channel.....	21
2.2.3.1 Man-in-the-Middle Attacks .....	22
2.2.3.2 Eavesdropping Attacks .....	22
2.2.3.3 Puppet Attacks.....	24
2.2.3.4 Smurf Attacks .....	24
2.2.3.5 Masquerading Attacks .....	25
2.2.3.6 Smart Meter Tampering Attacks .....	26
2.2.3.7 Traffic Analysis Attacks .....	26
2.3 Scenarios of Intrusion Attacks .....	28
2.4 Literature Review of Detection Techniques on Smart Grid.....	29
<b>Chapter 3 DETECTION METHODOLOGIES .....</b>	<b>30</b>
3.1 Attack Detection Procedure Using Machine and Deep Learning .....	30
3.2 Intrusion Attack Detection .....	33
3.2.1 Dataset .....	34
3.2.2 Data Pre-Processing .....	35
3.2.2.1 Data Cleaning .....	36

3.2.2.2 Data Transformation.....	41
3.2.2.3 Handling Categorical Data .....	41
3.2.2.4 Class Rebalancing and Sample Size Reduction.....	42
3.2.2.5 Feature Engineering.....	42
3.2.2.6 Pearson’s Correlation Coefficient.....	43
3.2.2.7 Tree Feature Importance.....	43
3.3 Machine and Deep Learning .....	40
3.3.1 Supervised Learning Models .....	40
3.3.1.1 Traditional Machine Models.....	41
3.3.1.1.1 Support Vector Machine .....	41
3.3.1.1.2 Decision Tree .....	41
3.3.1.1.3 Logistic Regression.....	42
3.3.1.1.4 K-Nearest Neighbor .....	43
3.3.1.1.5 Naïve Bayes .....	43
3.3.1.2 Ensemble Machine Models.....	43
3.3.1.2.1 Bagging .....	48
3.3.1.2.1.1 Random Forest.....	49
3.3.1.2.2 Boosting .....	49
3.3.1.2.2.1 Gradient Boosting .....	50
3.3.1.2.2.2 Categorical Boosting.....	50
3.3.1.2.2.3 Light Gradient Boosting .....	51
3.3.1.2.2.4 Adaptive Boosting .....	51
3.3.1.2.3 Stacking.....	52



3.3.1.3 Deep Learning .....	53
3.3.1.3.1 Artificial Neural Network .....	53
3.3.1.3.1.1 Deep 1 Neural Network .....	54
3.3.1.3.2 Convolutional Neural Network .....	50
3.3.1.3.2.1 Alex Neural Network .....	50
3.3.1.3.2.2 Densely Connected Neural Network.....	51
3.3.1.3.2.3 Residual Neural Network.....	52
3.3.1.3.2.4 Capsule Neural Network.....	60
3.3.2 Unsupervised Learning Models .....	62
3.3.2.1 Traditional Models .....	62
3.3.2.1.1 K-means .....	63
3.3.2.1.2 Principal Component Analysis .....	63
3.3.2.2 Neural Network Models .....	64
3.3.2.2.1 Variational Auto-Encoder Models .....	64
3.4 Reinforcement Learning.....	60
3.4.1 Deep Q-Learning .....	62
3.4.2 Capsule Q-Network .....	63
3.5 Online Learning.....	69
3.5.1 Sequential Learning .....	70
3.5.2 Online Sequential Capsule Network .....	71
<b>Chapter 4 RESULTS AND DISCUSSIONS.....</b>	<b>72</b>
4.1 Performance Analysis Metrics .....	72
4.2 Intrusion Attack Detection Analysis .....	73

4.2.1 Results of Detecting Intrusion Attacks on Smart Grid .....	73
4.2.1.1 Analysis of Optimized Parameters .....	73
4.2.1.2 Results of Feature Selection in Detecting Intrusion Attacks .....	74
4.2.1.3 Results of Ensemble Models in Detecting Intrusion Attacks .....	75
4.2.1.3.1 Results of Ensemble Learning Models in Detecting Reflection Intrusion Attacks .....	77
4.2.1.3.2 Results of Ensemble Learning Models in Detecting Exploitation Intrusion Attacks .....	80
4.2.1.3.3 Results of Improved Boosting Learning Models in Detecting reflection and exploitation and all Intrusion Attacks .....	81
4.2.1.4 Results of Neural Models in Detecting Intrusion Attacks .....	84
4.2.1.4.1 Results of Residual Neural Network in Detecting Intrusion Attacks .....	85
4.2.1.4.2 Results of Densely Connected Neural Network in Detecting Intrusion Attacks .....	86
4.2.1.5 Results of Comparison between Supervised and Unsupervised ML and DL models in Detecting Intrusions .....	90
4.2.1.6 Results of Reinforcement Learning-based Model in Detecting Intrusion Attacks .....	95
4.2.1.7 Results of Online Deep Learning Model in Detecting Intrusion Attacks .....	99
<b>Chapter 5 CONCLUSION</b> .....	<b>106</b>
<b>BIBLIOGRAPHY</b> .....	<b>108</b>

## LIST OF FIGURES

Figure 2.1 Overview of Smart Grid.....	8
Figure 2.2 Main Sub-Networks in the Smart Grid Architecture.....	9
Figure 2.3 Critical Applications in Smart Grid.....	13
Figure 2.4 Classification of Cyber-attacks on Smart Grid.....	14
Figure 3.1 Supervised Learning Working Flow. ....	34
Figure 3.2 Unsupervised Learning Working Flow. ....	35
Figure 3.3 An Overview of Reinforcement Learning Working Flow.....	36
Figure 3.4 Attack Distribution in CICDDoS 2019. ....	38
Figure 3.5 Overview of Data Pre-Processing Steps.....	38
Figure 3.6 Examples of Different Categories of Feature Selection Techniques. ....	43
Figure 3.7 Architecture of DNN.....	50
Figure 3.8 Architecture of CNN. ....	52
Figure 3.9 Architecture of AlexNet. ....	52
Figure 3.10 Architecture of DenseNet. ....	54
Figure 3.11 Architecture of ResNet.....	60
Figure 3.12 Architecture of CapsNet.....	62
Figure 3.13 Architecture of Variational Auto-Encoder.....	61
Figure 3.14 General Architecture of DQN. ....	63
Figure 3.15 Architecture of Capsule Q-Network.....	64
Figure 3.16 Necessary Steps of the CapsNet Q-learning Model. ....	69
Figure 3.17 Overview of the Proposed Online Detection Architecture.....	68

Figure 3.18 Overview of Batch-based CapsNet. ....	73
Figure 3.19 Overview of CapsNet using Sequential Euclidean Distance Routing Algorithm.....	71
Figure 3.20 Schematic Overview of Capsulation Block.....	71
Figure 4.1 Pearson’s Correlation Coefficient Heatmap. ....	82
Figure 4.2 Importance of the features based on the Extra Tree classifier. ....	83
Figure 4.3 Accuracy of the Selected Models. ....	82
Figure 4.4 Detection Rate of the Selected Models.....	82
Figure 4.5 Misdetection Rate of the Selected Models. ....	83
Figure 4.6 False Alarm Rate of Selected Models. ....	84
Figure 4.7 Evaluation results in terms of accuracy, detection rate, misdetection rate, and false alarm rate. ....	84
Figure 4.8 Results of attacks in terms of accuracy, detection rate, misdetection rate, and false alarm rate.....	86
Figure 4.9 Results of the Attacks for the DenseNet Models in terms of Highlighted Metrics. ....	91
Figure 4.10 Performance evaluation of the ML models in terms of accuracy, detection rate, misdetection rate, and false alarm rate for Test Data. ....	93
Figure 4.11 Performance evaluation of cyber-attacks based on best ML models in terms of processing time, prediction time, training time per sample, and memory size. ....	95
Figure 4.12 Confusion Matrices of the Models with Respect to the discount factor. ....	97
Figure 4.13 Results of the Proposed Models in in terms of Accuracy, Detection Rate, Misdetection Rate, and False Alarm Rate. based on the Discount Factors of 0.001 and 0.9.....	98
Figure 4.14 Results of Highlighted models in terms of accuracy and detection rate, misdetection rate, and false alarm rate.....	101
Figure 4.15 Confusion Matrices of 2-Class CapsNet Models.....	101
Figure 4.16 Results of Highlighted Models in terms of Accuracy and Kappa with respect to the Sample Sizes. ....	109

## LIST OF TABLES

Table 2.1 Summary of Cyber-Attacks Targeting Smart Grid. ....	27
Table 3.1 List of Attacks. ....	38
Table 3.2 Short Descriptions of the Used Features in CICDDoS 2019. ....	40
Table 4.1 List of Parameters in Models. ....	80
Table 4.2 Evaluation Results for Reflection-based Attacks. ....	84
Table 4.3 Evaluation Results for Exploitation-based Attacks. ....	85
Table 4.4 Models' performance for Attacks in Terms of the considered Metrics (Best performances are in bold). ....	92
Table 4.5 The ML models' performance in Terms of Processing Time, Prediction Time, Training Time per Sample, and Memory Size for Test Data(Best performances are in bold). ....	93
Table 4.6 Performance of the ML Models In Terms Of PRT, PT, TPS, and M for Test data. ....	97
Table 4.7 Comparison of Different Timing Metrics with Respect to the Discount Factor of 0.001 and 0.9. ....	100
Table 4.8 Test Results of the Highlighted Models in terms of Training Time Per sample, Prediction Time, and Memory Size. ....	102

## ACKNOWLEDGEMENTS

First of all, my deepest appreciation goes to my academic advisor, Dr. Naima Kaabouch, who had a crucial role in helping me complete my Ph.D. Her continued support and guidance always provided me with invaluable insights and strengthened my abilities. This work would never have been possible without her extensive direction and professional feedback. I would also like to express my gratitude to all the committee members, Dr. Hu, Dr. Ozturk, Dr. Elderini, and Dr. Dorafshan for their time and feedback.

I would especially like to thank my family, especially my brothers and my husband, for their continuous support and encouragement throughout my entire course of study. I would never have been successful without them creating confidence in me and encouraging me to pursue my dreams.

This dissertation is dedicated to my parents, Roghayeh and Masoud.

*For their endless love, support, and encouragement*

## ABSTRACT

The traditional power grid is no longer a practical solution for power delivery due to several shortcomings, including chronic blackouts, energy storage issues, high cost of assets, and high carbon emissions. Therefore, there is a serious need for better, cheaper, and cleaner power grid technology that addresses the limitations of traditional power grids. A smart grid is a holistic solution to these issues that consists of a variety of operations and energy measures. This technology can deliver energy to end-users through a two-way flow of communication. It is expected to generate reliable, efficient, and clean power by integrating multiple technologies. It promises reliability, improved functionality, and economical means of power transmission and distribution. This technology also decreases greenhouse emissions by transferring clean, affordable, and efficient energy to users.

Smart grid provides several benefits, such as increasing grid resilience, self-healing, and improving system performance. Despite these benefits, this network has been the target of a number of cyber-attacks that violate the availability, integrity, confidentiality, and accountability of the network. For instance, in 2021, a cyber-attack targeted a U.S. power system that shut down the power grid, leaving approximately 100,000 people without power. Another threat on U.S. Smart Grids happened in March 2018 which targeted multiple nuclear power plants and water equipment. These instances represent the obvious reasons why a high level of security approaches is needed in Smart Grids to detect and mitigate sophisticated cyber-attacks.

For this purpose, the US National Electric Sector Cybersecurity Organization and the Department of Energy have joined their efforts with other federal agencies, including the Cybersecurity for Energy Delivery Systems and the Federal Energy Regulatory Commission, to investigate the security risks of smart grid networks. Their investigation shows that smart grid requires reliable solutions to defend and prevent cyber-attacks and vulnerability issues. This investigation also shows that with the emerging technologies, including 5G and 6G, smart grid may become more vulnerable to multistage cyber-attacks. A number of studies have been done to identify, detect, and investigate the vulnerabilities of smart grid networks. However, the existing techniques have fundamental limitations, such as low detection rates, high rates of false positives, high rates of misdetection, data poisoning, data quality and processing, lack of scalability, and issues regarding handling huge volumes of data. Therefore, these techniques cannot ensure safe, efficient, and dependable communication for smart grid networks.



Therefore, the goal of this dissertation is to investigate the efficiency of machine learning in detecting cyber-attacks on smart grids. The proposed methods are based on supervised, unsupervised machine and deep learning, reinforcement learning, and online learning models. These models have to be trained, tested, and validated, using a reliable dataset. In this dissertation, CICDDoS 2019 was used to train, test, and validate the efficiency of the proposed models. The results show that, for supervised machine learning models, the ensemble models outperform other traditional models. Among the deep learning models, densely neural network family provides satisfactory results for detecting and classifying intrusions on smart grid. Among unsupervised models, variational auto-encoder, provides the highest performance compared to the other unsupervised models. In reinforcement learning, the proposed Capsule Q-learning provides higher detection and lower misdetection rates, compared to the other model in literature. In online learning, the Online Sequential Euclidean Distance Routing Capsule Network model provides significantly better results in detecting intrusion attacks on smart grid, compared to the other deep online models.

# Chapter 1

## INTRODUCTION

### 1.1 Motivation and Problem Statement

The development of the smart grid has introduced significant advancements in the management and distribution of electrical power. This modernized electrical grid utilizes advanced technologies to enhance efficiency, reliability, and sustainability. However, along with its numerous benefits, smart grid also presents notable security challenges that require careful consideration and robust solutions. As the smart grid incorporates digital communication and intelligent devices, it becomes susceptible to cyber threats and vulnerabilities.

The interconnected nature of the grid, with its various components communicating and exchanging data, opens potential entry points for malicious actors. Cyber-attacks on the smart grid could lead to disruptions in power supply, unauthorized access to sensitive information, and even potential safety hazards. Ensuring the security of the smart grid involves addressing several key areas, such as data privacy, authentication and access control, encryption, anomaly detection, incident response plans, collaboration and standards, and continuous monitoring and updates. In the following, short descriptions of these security measures are provided.

- *Data Privacy*: Smart grid collects and analyzes vast amounts of data to optimize energy distribution. Protecting the privacy of this data, including personal and usage information, is crucial to maintain public trust and comply with regulations.
- *Authentication and Access Control*: Implementing strong authentication mechanisms and access controls helps prevent unauthorized individuals from gaining control over grid components. This includes securing communication channels and ensuring only authorized personnel can access critical systems.
- *Encryption*: The use of encryption safeguards data transmission and storage, making it challenging for attackers to intercept and exploit sensitive information.
- *Anomaly Detection*: Employing sophisticated anomaly detection systems allows for the early identification of unusual or suspicious activities within the grid. This enables rapid response and mitigation efforts.

- *Incident Response Plans:* Developing comprehensive incident response plans ensures a coordinated approach in the event of a security breach. Timely identification, containment, and recovery are vital components of such plans.
- *Collaboration and Standards:* Establishing industry-wide security standards and fostering collaboration among stakeholders, including utilities, technology providers, and regulatory bodies, can lead to a more secure and resilient smart grid ecosystem.
- *Continuous Monitoring and Updates:* Regularly monitoring the grid's components and systems for vulnerabilities and applying timely updates and patches is essential to address emerging security threats.

The smart grid's security challenges require a multidisciplinary approach, combining expertise in electrical engineering, computer science, cybersecurity, and policymaking. As the smart grid continues to evolve and play a pivotal role in modern energy infrastructure, a proactive and adaptive security strategy is paramount to ensure its reliable and secure operation. In general, the integration of advanced technologies and digital communication within the smart grid brings about numerous benefits, such as enhanced efficiency and better energy management. However, this digital transformation also introduces a heightened risk of cyber threats and attacks.

To safeguard the integrity, reliability, and security of the smart grid, the implementation of Intrusion Detection Systems (IDS) is of paramount importance. An IDS continuously monitors smart grid's network and systems for any suspicious or unauthorized activities. By analyzing network traffic, system logs, and behavior patterns, an IDS can promptly identify potential threats, including malware, unauthorized access attempts, and anomalous activities. In case of a security breach or cyber-attack, an IDS provides real-time alerts to grid operators and security personnel. This enables a swift and coordinated response to mitigate the impact of the attack, prevent further damage, and restore normal operations. In addition, smart grid encompasses critical infrastructure that, if compromised, could have severe consequences for energy distribution, public safety, and national security. An IDS helps protect key components, such as power generation, transmission, and distribution systems, from malicious intrusions that could disrupt services or cause widespread outages.

An IDS plays a crucial role in maintaining the integrity of data exchanged within the smart grid. By detecting unauthorized modifications or alterations of data, an IDS ensures the accuracy and reliability of information used for decision-making and energy optimization. Many regions have established regulations and standards for securing critical

infrastructure, including smart grid. Implementing an IDS not only helps utilities comply with these regulations but also demonstrates a commitment to cyber-security and public safety.

An IDS provides grid operators with a comprehensive view of network activities and potential threats. This enhanced visibility enables better situational awareness and informed decision-making, allowing operators to proactively address vulnerabilities and emerging risks. It also can learn and adapt to evolving cyber-threats. As attackers develop new techniques and tactics, an IDS can be updated to detect emerging intrusion patterns, ensuring a more robust defense against ever-changing cyber-security challenges. At the end, the presence of a well-implemented IDS can act as a deterrent to potential attackers. The knowledge that their activities will be promptly detected and responded to may discourage malicious actors from targeting the smart grid.

For this purpose, numerous approaches have been suggested to detect cyber-vulnerabilities targeting an IDS on smart grid. These techniques can be classified into localization, Artificial Intelligence, prediction models, filtering, and intrusion detection systems. These techniques primarily can detect and classify the cyber-attacks targeting an IDS on smart grid. Despite several studies having been conducted in this field, the current studies still deal with some limitations, such as low detection rate, high misdetection rate, and high dependency to other sensors or hardware. These limitations lead to inefficient techniques for smart grid cyber-attack detection. Additionally, these current techniques do not provide a secure, reliable, and optimal approach to detect cyber-attacks on smart grid.

## **1.2 Dissertation Goal and Objectives**

Considering security challenges and limitations of the existing techniques, the goal of this dissertation is to develop reliable, secure, and independent techniques to detect cyber-attacks that target Intrusion Detection system on smart grid. To achieve this goal, the following objectives are set and met:

- Identify cyber-vulnerabilities and threats of smart grid.
- Compare the efficiency of supervised and unsupervised machine learning models in detecting intrusion attacks.
- Develop a new reinforcement learning model for detecting intrusion attacks and compare its performance with the existing models.
- Develop an online learning model and compare its performance with the existing models.

### 1.3 Contributions

The contributions of this dissertation are described as follows:

- *Comprehensive analysis of cyber-attacks on Smart Grid and their proposed solutions*

Among the works done on the smart grid in the literature, less effort has been put on the analysis of intrusion detection systems and their vulnerabilities they deal on smart grid. Therefore, in this dissertation, cyber-attacks targeting intrusion detection systems, namely intrusions are comprehensively investigated along with discussions of their risk analysis. Additionally, a taxonomy of smart grid attacks is elaborated and discussed. This comprehensive analysis helps develop more efficient and reliable security solutions to intrusions. Moreover, various detection techniques are summarized in depth.

- *Analysis of supervised and unsupervised machine learning and deep learning techniques in detecting and classifying intrusions on smart grid*

In this dissertation, supervised and unsupervised machine learning and deep learning techniques are used to detect and classify intrusions on smart grid. Various conventional and ensemble supervised and unsupervised machine learning and deep learning techniques, such as support vector machine, decision tree, random forest, stacking, bagging, boosting, densely connected neural network, and residual neural network, Principal Component Analysis, K-means, and Variational Autoencoder are developed, analyzed, and compared. Every developed model consists of a set of parameters and hyperparameters, whose selection controls the overfitting/underfitting issues over the training data and provides the optimal results. As a result, these parameters have strong impacts on the efficiency, detection, and classification process.

- *Proposing a Reinforcement learning techniques for detecting intrusions on smart grid*

In this dissertation, Reinforcement learning techniques are used as an optimal solution for dealing with security issues in smart grid. Due to the complexities and uncertainty of the smart grid along with the high volume of data transfer, some artificial intelligence techniques cannot ensure optimal results in dealing with security issues in smart grid. For example, supervised learning techniques need a huge volume of data to provide accurate results. In addition, supervised and unsupervised learning techniques cannot handle a sequence of actions simultaneously, which may increase the cost of training and further processes. Moreover, the lack of adaptability of such techniques may increase the need for retraining processes. To address such challenges, a new type of model has been proposed, reinforcement learning, which decreases the cost of computational power. This type of learning is defined as the training of machine learning models to make a

sequence of decisions. In this learning, the agent learns to achieve a goal in an uncertain and complicated environment. Such learning is an integral part of the deep learning models that can maximize some portion of the cumulative reward. It can also help attain a complex objective or optimize a specific dimension over several steps.

To address this contribution, a Reinforcement learning technique, deep Q-network, was developed to minimize the detection delay, misdetection rates, and false alarm rates. One technical issue that faces such a proposed technique was the selection of appropriate parameters in the training process. It is important to note that the network's parameters were not trained; however, they were periodically synchronized with those of the main Q-networks. The use of such a strategy can increase the stability of the training; however, selecting ineffective parameters could result in inaccurate results. Additionally, the development of such a model required various design requirements, including the model's statistics and agent behavior. These challenges were empirically validated based on several metrics, such as accuracy, detection rate, probability of misdetection, probability of false alarm, training time, testing time, time complexity, and memory usage. The proposed deep Q network was also tested in a constructed environment, demonstrating that the proposed technique can detect intrusions.

- *Developing an online-based technique for classifying and detecting intrusions on smart grid*

In this dissertation, an online deep learning technique is used to effectively classify, detect, and mitigate different cyber-vulnerabilities. Despite the extensive research in literature, deep online learning has not been fully addressed by existing studies which will require further investigation by scholars in the future. In addition, while online deep learning has several benefits over batch deep learning, including scalability and efficiency, there are still limited studies that focus on issues related to high volume and high velocity data for online learning in large-scale environments. In addition to some preliminary research, there are still many unaddressed challenges, such as computational efficiency, learning scalability, and model complexity that need to be addressed.

To address deep online learning, a known family of deep learning models, convolutional neural network (CNN) models, were used. These models are widely used in literature for detecting attacks on smart grid. Despite their advantages, CNN-based techniques have some limitations. These techniques can replicate the same knowledge at all points in the spatial dimension of an input dataset compared with other neural network models. Hence, the features at one spatial location using translated replicas of feature detectors are available at other locations. CNN has local shared. The performance of this CNN-based model was compared with those of other deep learning models.

## **1.4 Dissertation Organization**

This dissertation is organized as follows:

Chapter 2 provides an overview of the smart grid, architecture, applications, cyber-attacks, and intrusion detection systems. This chapter explains the cyber-attacks targeting smart grid, and provides the aim, objective, and a risk analysis of these attacks. A comprehensive review of the existing literature about the detection techniques is discussed in this chapter. Also, intrusion detection systems and detailed description of the intrusion attacks are provided in this chapter.

Chapter 3 indicates the cyber-attacks detection techniques targeting smart grid. This chapter explains the cyber-attack detection models that were developed to detect and classify attacks on smart grid. In addition, description of the training dataset is highlighted in this chapter. At the end, this chapter provides the descriptions of the machine learning, deep learning, and reinforcement techniques, while it discusses a proposed technique on online deep learning.

Chapter 4 outlines and analyzes the results of the proposed techniques on intrusion attacks targeting smart grid. An in-depth analysis and a comparison of different algorithms are provided in this chapter. Also, this chapter discusses the impacts of algorithms' hyperparameters on each algorithm's detection performance. Chapter 5 finally summarizes this dissertation and provides future works and open research directions.

## Chapter 2

### OVERVIEW OF SMART GRID

In this chapter, overviews of smart grid network, intrusion detection systems, and cyber-attacks targeting intrusion detection systems are provided. Additionally, a brief description of these attacks, aims, impacts, and their security requirements are discussed.

#### **2.1 Smart Grid Network**

The power system, since its inception over a century ago, has transformed into one of the most intricate networks in human history. With rising demand, today's electrical grids have evolved into vast interconnected systems, managed by various power corporations. However, the complex interactions among these entities often lead to inefficient cross-region transmission and suboptimal power delivery. Traditional power grids are also grappling with new challenges stemming from technological advancements such as distributed renewable energy generation, electric vehicle charging systems, and smart meters. The increasing reliance on electricity and demands for better power quality necessitate improvements in power delivery, pricing flexibility, and restoration speed [1].

To address these challenges, the concept of a smart grid has emerged. This next-generation power system aims to modernize the traditional grid, enhancing its reliability, flexibility, and efficiency. The smart grid features advanced metering infrastructure (AMI) and intelligent devices to reduce demand and operational costs. It incorporates distributed automation for improved system reliability and automated controls for enhanced power management, as presented in Figure 2.1. Notably, the smart grid is characterized by a two-way communication network connecting power plants, transmission sensors, and consumers, facilitating optimized power delivery through computer-based automation. However, the smart grid offers economic benefits, it also introduces new cyber-security threats. Its distributed control reduces the vulnerability of central control centers but raises concerns about malicious actors exploiting the access points provided by



system-on-a-chip (SoC) electrical devices. The increased data flow along the power transmission network makes it susceptible to data interception and unauthorized modification, potentially disrupting operations or facilitating unfair practices [2]. The smart grid's distributed nature could lead to security breaches, jeopardizing its reliability and causing widespread impact.

One specific vulnerability is the potential for cascading failure events, where a few failed components trigger broader system collapses and blackouts. Historical examples, such as the 2003 blackout in the northeast U.S. the 2012 blackout in India, the 2016 power blackout in Turkey, and the 2020 power cut in France underscore the significant consequences of such failures. Despite progress in power grid stability and security, the complexity of smart grid security remains a challenge. Predicting the propagation of cascading effects and understanding attackers' strategies to maximize impact remain unclear. More effective modeling, simulation, and analysis tools are needed to comprehensively assess and defend against these smart cyber-threats [3, 4].

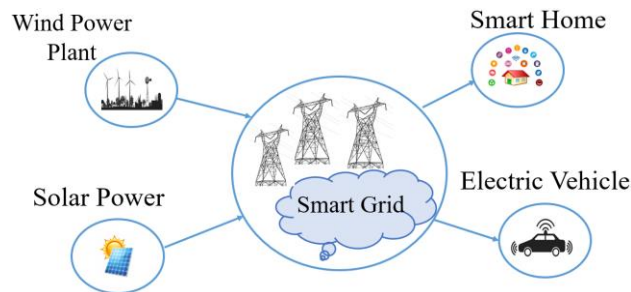


Figure 2.1 Overview of Smart Grid.

### 2.1.1 Smart Grid Architecture

The hierarchical architecture of the smart grid infrastructure is of paramount importance due to its ability to interconnect a diverse array of systems. Within this framework, distinct sub-networks play a pivotal role, each focused on specific geographical regions. The smart grid network encompasses three primary sub-networks: The Wide Area Network (WAN), the Neighborhood Area Network (NAN), and the Home Area Network (HAN), as shown in Figure 2.2.

The foundation of the smart grid’s hierarchical architecture is the WAN. This expansive network acts as the backbone, facilitating communication and coordination across extensive geographic areas. It manages high-level functions such as bulk power transmission, energy market operations, and system-wide monitoring. The WAN ensures efficient exchange of data and commands between power generation plants, substations, and control centers over vast territories. In contrast, situated one tier below the WAN, the NAN serves as an intermediary network that links together localized clusters of infrastructures. Operating within specific neighborhoods or districts, the NAN enables communication between distribution substations, residential areas, commercial zones, and industrial sectors. It enhances the distribution of power, supports demand response initiatives, and facilitates the integration of renewable energy sources within these localized regions [5].

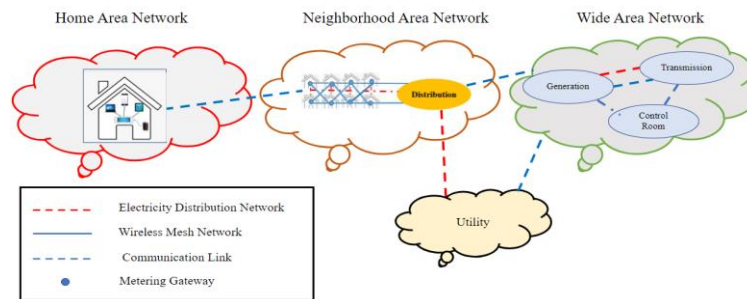


Figure 2.2 Main Sub-Networks in the Smart Grid Architecture.

The innermost layer of the smart grid’s hierarchical architecture is the HAN. This network operates within individual residences, connecting various smart devices, appliances, and energy management systems. Through the HAN, homeowners gain real-time insights into their energy consumption, allowing them to optimize usage patterns, participate in demand response programs, and contribute to overall grid stability. Smart meters, thermostats, and home automation systems are key components of the HAN. This tri-tiered structure ensures that the smart grid operates cohesively and efficiently, extending its reach from the macroscopic level of the WAN down to the microscopic granularity of the HAN. By integrating these distinct sub-networks, the smart grid optimizes energy distribution, enhances grid reliability, supports renewable energy integration, and empowers consumers to actively engage in energy management practices. As the smart

grid continues to evolve, this hierarchical architecture remains a fundamental framework for enabling the future of intelligent and sustainable energy systems [3, 4, 5].

### **2.1.2 Smart Grid Features**

The smart grid introduces a suite of innovative features that revolutionize the traditional power system. In general, the smart grid is anticipated to bring about several significant features aimed at enhancing grid resilience, self-healing capabilities, environmental performance, and overall system efficiency. Key among these features is enhanced grid resilience, which empowers the system to swiftly recover from disruptions, ensuring uninterrupted power supply. Grid resilience refers to the ability of the power grid to rapidly recover and continue functioning during power interruptions and outages. This can be achieved by incorporating additional distributed power supply and integrating modern resources into the power grid when disruptions occur. Complementing this is the self-healing capability, enabling rapid fault detection and isolation. The integration of renewable energy sources like solar and wind underscores the commitment to sustainability, while distributed energy resources introduce localized power generation for improved reliability. The self-healing feature allows the system to swiftly identify faults, reduce outage durations, and expedite system recovery. Consequently, the heightened flexibility and reliability afforded by the grid's resilience and self-healing capabilities profoundly impact the economy [1, 3, 5, 6].

Another pivotal feature expected from the smart grid is the enhancement of system performance. Within the traditional power grid, energy losses can occur due to various factors, including power station faults or damage to transmission lines. The smart grid holds the promise of elevating system performance by optimizing the utilization of assets and operations, thereby curbing energy costs and enabling efficient electricity transmission. These advantages are poised to directly enhance power quality and facilitate effective asset management, indicating an elevated level of overall system performance. Furthermore, the smart grid is projected to accelerate the transition from conventional vehicles to electric

vehicles, potentially enhancing environmental performance by reducing energy consumption among end-users and minimizing energy losses across the grid [5, 6, 7].

Furthermore, the smart grid is fortified by additional vital attributes. Advanced metering infrastructure serves as a conduit for real-time communication, fostering seamless interaction between utility providers and consumers, thereby ensuring accurate billing and facilitating effective demand management. Additionally, robust communication networks bolster connectivity, while microgrids offer localized power supply autonomy. Integration of electric vehicles and adoption of energy storage solutions enhance versatility, optimizing resource deployment. Data analytics and cybersecurity measures culminate in a holistic framework, bestowing the smart grid with adaptability, efficiency, and ecological mindfulness, thereby propelling a sustainable energy paradigm [6, 7].

### **2.1.3 Smart Grid Applications**

The smart grid encompasses a diverse range of dispersed applications and capabilities, as shown in Figure 2.3, including [1-10]:

1. *Advanced Metering Infrastructure (AMI)*: AMI uses smart meters and communication systems to monitor and manage electricity consumption. Smart meters provide real-time data on energy usage from various devices, such as home appliances, heating systems, and solar panels. This data is transmitted back to the utility company, allowing for accurate billing, demand forecasting, and the ability to implement dynamic pricing strategies to encourage energy conservation.
2. *Supervisory Control and Data Acquisition (SCADA)*: SCADA systems play a crucial role in monitoring and controlling the electric power grid. These systems gather real-time data from sensors and devices across the grid, such as substations, transformers, and power lines. SCADA allows operators to remotely monitor grid conditions, detect faults, and respond quickly to restore power in case of outages. It also aids in optimizing grid performance and minimizing downtime.

3. *Demand Response Management (DRM)*: DRM focuses on managing and shaping electricity demand to match supply and maintain grid stability. During peak demand periods, when electricity consumption is high, DRM can reduce demand by implementing strategies such as load shifting, where non-essential devices are temporarily turned off or their energy consumption is reduced. This helps prevent grid overloads, lower costs, and enhance overall grid reliability.
4. *Substation Automation*: This involves automating and remotely controlling substations, which play a crucial role in transforming and distributing electricity across the grid.
5. *Electrical Vehicle (EV) Charging*: With the rise of electric vehicles, the smart grid can manage EV charging by optimizing charging times, coordinating with renewable energy sources, and ensuring that charging infrastructure is used efficiently.
6. *Outage Management (OM)*: OM systems help utilities detect, locate, and respond to power outages more efficiently. They enable rapid identification of affected areas and help prioritize restoration efforts.
7. *Distribution Management (DM)*: DM systems optimize the distribution of electricity by managing the flow of power and voltage levels across the distribution network. They enhance reliability and reduce losses.
8. *Home Energy Management (HEM)*: HEM systems provide homeowners with tools to monitor and manage their energy usage. Users can adjust settings and schedules for appliances, heating, and cooling systems to conserve energy and reduce costs.

These applications collectively contribute to building a smarter, more efficient, and resilient electrical grid that can adapt to changing energy demands, integrate renewable energy sources, and enhance overall sustainability.

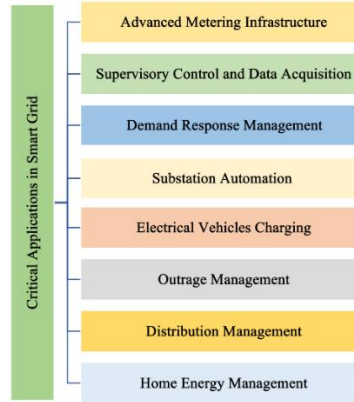


Figure 2.3 Critical Applications in Smart Grid.

### 2.1.4 Cyber-Security Requirements in Smart Grid

As the transition from conventional power grids to smart grids took place, security emerged as a critical challenge over the past few decades. To tackle this challenge, the system and its infrastructure must adhere to secure architectural principles. Consequently, cyber-security, as an essential and complementary process, must adhere to a comprehensive set of security requirements. Initially, the National Institute of Standards and Technology (NIST) outlined three fundamental security requirements for the smart grid: confidentiality, integrity, and availability. However, accountability also plays a significant role in smart grid security. In general, when unauthorized access occurs, confidentiality is compromised. Integrity ensures accurate data by safeguarding it against improper modifications or unauthorized data destruction. On the other hand, availability is a crucial aspect of smart grids, ensuring access to the system’s data [1-3, 11].

The absence of availability indicates that data is not accessible for users. In addition to the requirements, accountability is a key player in smart grid security, providing traceability of system activities recorded by individuals, devices, or public authorities. The recorded data can serve as evidence in case of an attack, validating actions taken by users, including administrators, and verifying the integrity of data collected from devices. By adhering to these four requirements, confidentiality, integrity, availability, and accountability, adequate protection for smart grid infrastructures can be achieved. Due to the inherent vulnerabilities in communication, smart grid networks are susceptible to various cyber-attacks, which

can be categorized in multiple ways. In the subsequent section, we delve into existing classifications of cyberattacks, along with our proposed classification. We describe potential attacks, their intentions, and the impacts they have on the networks [4, 8-11].

## 2.2 Cyber-attacks on Smart Grid

The constant improvement of power and energy systems involves integrating new power generation designs and the latest protection and control technologies. In fact, the use of digitalization techniques in the power system has become increasingly important due to our heavy reliance on electricity for various aspects of our lives. Smart technologies have been integrated into areas, such as smart grid. Hence, current research focuses on advancing data transmission through communication networks and protocols to enhance communication network security using intelligent systems. However, the adoption of modern technologies can introduce security and stability challenges, such as cyber-threats to energy systems [12].

Cyber-attacks targeting smart grid have become a pressing concern. As smart grid technology advances, incorporating digital communication and control systems, the risk of malicious cyber-threats grows. These attacks can exploit vulnerabilities in the grid's interconnected devices and software, potentially leading to disruptions in energy distribution, data compromise, and even widespread outages. To counter these cyber-attacks, it is critical to study and investigate them before implementing stringent protocols, encryption techniques, and intrusion detection systems, which can fortify smart grid against cyber-threats [13]. Thus, as illustrated in Figure 2.4, cyber-attacks can be divided into sensor-based, computer-based, and network channel-based attacks. These categories along with a brief description of the associated attacks are described as follows:

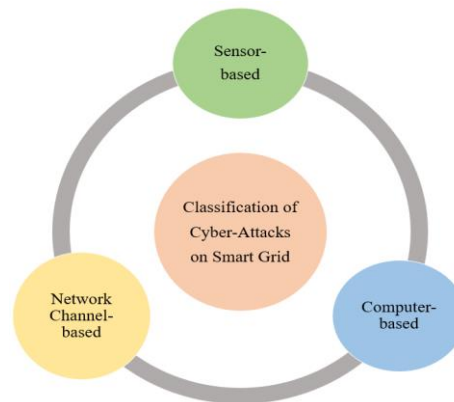


Figure 2.4 Classification of Cyber-attacks on Smart Grid.

### **2.2.1 Cyber-Attacks Targeting Sensor**

Cyber-attacks targeting sensors in smart grid networks are a form of attacks aimed at exploiting vulnerabilities in the sensors and communication systems that are an integral part of modern smart grid infrastructure. Smart grids are advanced electricity distribution systems that use digital technology to monitor, control, and optimize the flow of electricity from power plants to consumers. In general, smart grids rely on a wide range of sensors to gather data about electricity generation, distribution, consumption, and overall grid health. These sensors include devices like smart meters, phasor measurement units (PMUs), and other control systems. Cyber-criminals target vulnerabilities in these sensors to gain unauthorized access or control. Thus, these attacks targeting sensors may lead to grid instability, financial losses, safety risks, and low data privacy [14, 15].

Hence, manipulating sensor data can lead to incorrect decisions in grid management, potentially causing power outages, equipment damage, or even a full-scale blackout. In addition, disrupted electricity supply and operational downtime can result in financial losses for utilities, businesses, and consumers [16]. It is worth mentioning that a compromised smart grid can pose safety risks to both utility workers and the public, especially if critical systems fail to operate as expected. Also, breaches in smart grid sensors can expose sensitive customer data, leading to privacy concerns and regulatory compliance issues. In summary, cyber-attacks targeting sensors in smart grid networks pose serious threats to the stability, security, and reliability of our electricity distribution systems. Implementing comprehensive cybersecurity measures and staying vigilant against evolving threats are essential to safeguarding our critical infrastructure [16-20] These attacks targeting sensors in smart grid networks include intrusion, spoofing, injection, jamming, and time synchronization attacks, as discussed in the following.

#### **2.2.1.1 Intrusion Attacks**

Intrusion attacks are highly damaging cyber threats that exploit vulnerabilities in a network to gain unauthorized access to nodes. The primary objective of these attacks is to compromise the integrity and confidentiality of the smart grid network. Intruders seek to misuse available resources, disrupting the normal operations of the grid. Detecting and preventing intrusion attacks are crucial to maintaining the security and stability of the smart grid infrastructure [18-20].

#### **2.2.1.2 Spoofing Attacks**

Spoofing attacks target sensors within smart grid networks. These attacks involve various techniques, such as identity spoofing, ARP spoofing, GPS spoofing, IP spoofing, and MAC spoofing. In these attacks, malicious actors pretend to be



legitimate nodes, deceiving other nodes within the network. The result is a disruption of network security, reliability, stability, and overall operation. Spoofing attacks can lead to widespread confusion and unauthorized access, posing significant risks to the smart grid's functionality [21-25].

### **2.2.1.3 Injection Attacks**

Injection attacks are attempts by adversaries to manipulate data within the smart grid network. Attackers may delete, alter, or introduce falsified data, which can have severe consequences for the grid's operations. Such attacks can potentially cause blackouts, compromise data integrity, introduce corruption, and even create illegitimate nodes within the network. Preventing and mitigating injection attacks is crucial to maintaining the accuracy and effectiveness of the smart grid's data [18-20, 26].

### **2.2.1.4 Denial of Service Attacks**

Denial of Service (DoS) attacks are a common form of cyber-attack that aim to disrupt the availability of a smart grid network. Attackers flood the network with excessive signals, overwhelming its capacity and rendering it inaccessible to authorized users. A particular type of DoS attack involves jamming, where a malicious actor generates constant or random signals to occupy the network channel. This prevents legitimate devices from transmitting or receiving data, leading to service disruption [18-40].

### **2.2.1.5 Time Synchronization Attacks**

Time Synchronization attacks (TSA) are a significant concern for sensors in smart grids, particularly those reliant on accurate timing information. These attacks target the timing mechanisms used in phasor measurement units and wide area protection, monitoring, and control devices. For instance, GPS spoofing attacks can compromise the timing accuracy of devices that rely on GPS signals. Given that communication and control messages in a smart grid are time-sensitive, TSA and GPS spoofing attacks can have a direct impact on the grid's performance and coordination. In conclusion, the smart grid faces a diverse range of cyber threats, including intrusion attacks, spoofing attacks, injection attacks, denial of service attacks, and time synchronization attacks. Safeguarding against these threats requires robust cybersecurity measures, continuous monitoring, and proactive mitigation strategies to ensure the security, reliability, and stability of smart grid operations [27-30, 33].

### **2.2.1.6 Jamming Attacks**

Jamming attacks on a smart grid refers to malicious activities where an attacker intentionally disrupts or interferes with the wireless communication signals used within the smart grid network. These attacks aim to cause communication failures,

disrupt data transmission, and potentially compromise the operation of the grid. The attack process on smart grid consists of signal interference, and communication disruption.

The attacker generates strong radio frequency signals within the same frequency bands used by smart grid communication devices, such as smart meters, sensors, or control systems. Then, the attacker's interference overwhelms or drowns out legitimate communication signals, causing communication failures between various components of the smart grid network [31].

Jamming attacks have two potential consequences, namely communication breakdown, service outages, and operational disruption. Jamming attacks can lead to disruptions in communication between smart grid devices, making it difficult to monitor and control energy distribution effectively. The inability to transmit critical data, control commands, or operational information can disrupt the reliable and efficient functioning of the smart grid. Smart grid devices rely on communication for coordination and real-time decision-making. Jamming attacks can lead to service outages and power distribution issues. By implementing these mitigation strategies and maintaining a proactive approach to secure communication channels, smart grid operators can reduce the risk of jamming attacks and enhance the reliability and resilience of their energy distribution systems [32].

### **2.2.2 Cyber-Attacks Targeting Computer**

Cyber-attacks targeting computers are malicious activities carried out by hackers or cybercriminals to compromise the integrity, confidentiality, and availability of computer systems. These attacks can have various objectives, including stealing sensitive information, disrupting operations, gaining unauthorized access, and causing financial or reputational harm. Computers in critical infrastructures, such as smart grid networks, are especially attractive targets due to the potential impact of their compromise [33].

Cyber-attacks targeting computers are deliberate and malicious activities conducted by cybercriminals, hacktivists, or state-sponsored actors with the goal of exploiting vulnerabilities within computer systems. These attacks can have various motives, including financial gain, data theft, disruption of operations, espionage, or causing chaos. Computers in critical infrastructures like smart grids are prime targets due to their importance and potential impact on public services. These Cyber-attacks that target computers and servers in smart grid networks include brute force, social engineering, viruses/malware, horse trojan, and popping human interface attacks, as highlighted as following:

### **2.2.2.1 Brute-Force Attacks**

Brute-force attacks are a type of cyber-security threat that involves systematically attempting all possible combinations of passwords or encryption keys until the correct one is found. In the context of smart grids, where advanced technologies and communication systems play a crucial role, the risk of brute-force attacks poses a significant concern. Instances of these attacks are known as password guessing, encryption key cracking, authentication bypass. In password guessing, attackers use automated tools to repeatedly guess passwords for user accounts, control systems, or administrative interfaces within the smart grid network.

In encryption key cracking, brute-force attacks can also be used to crack encryption keys that protect sensitive data transmitted between components of the smart grid. In authentication bypass, attackers attempt various combinations of usernames and passwords to gain unauthorized access to critical systems, often exploiting weak or default credentials [34].

In addition, these attacks can have several implications, including unauthorized access, data exposure, and operational disruption. In unauthorized access, successful attacks can grant attackers unauthorized access to control systems, grid components, or sensitive data, potentially leading to manipulation or disruption of operations. In contrast, in data exposure, if attackers gain access to user accounts, they might exfiltrate sensitive data, compromising the confidentiality of customer information or grid operations. In Operational Disruption, compromised access can disrupt grid operations, affecting energy distribution, load management, and overall system stability [35-37].

### **2.2.2.2 Social Engineering Attacks**

Social engineering attacks pose a significant threat to smart grid systems, which are designed to enhance the efficiency and reliability of energy distribution through advanced technology. These attacks manipulate human psychology to deceive individuals into divulging confidential information, performing unauthorized actions, or compromising security protocols. In the context of smart grids, social engineering attacks can exploit vulnerabilities in human behavior to gain unauthorized access, disrupt operations, or compromise sensitive data. These attacks are known as phishing, pretexting, tailgating, baiting, quid pro quo, impersonation, and elicitation. In phishing attacks, attackers send deceptive emails or messages to employees, customers, or administrators of the smart grid system. These messages may appear to be from a legitimate source, such as a utility company or a trusted vendor. The recipients are then tricked into clicking malicious links, downloading malware, or revealing login credentials, which can lead to unauthorized access and control of the smart grid infrastructure. In pretexting, attackers impersonate authorized personnel, such as utility employees, contractors, or even

regulatory officials. They create elaborate scenarios or stories to convince employees to disclose sensitive information, provide access to critical infrastructure, or perform actions that compromise security [38].

In tailgating, an attacker gains physical access to restricted areas of a smart grid facility by following an authorized individual through access points, without proper verification. Once inside, the attacker can tamper with equipment, install malicious devices, or gain unauthorized control. In baiting, attackers leave physical devices, such as infected USB drives or seemingly innocent equipment, in strategic locations where they are likely to be found by employees or visitors. When someone plugs in or interacts with the device, it can introduce malware into the smart grid network. In quid pro quo, attackers offer something in exchange for information or access. For example, they may pose as technical support personnel and promise to solve a non-existent issue in exchange for login credentials or sensitive information. In impersonation, attackers may pretend to be a high-ranking executive, a supervisor, or a fellow employee to manipulate individuals into performing certain actions, such as transferring funds or altering system configurations. In elicitation, attackers engage in casual conversations with employees or personnel, gradually extracting information about the smart grid infrastructure, processes, or security measures. This information can then be used to plan and execute more targeted attacks [20-30].

#### **2.2.2.3 Viruses/Malware Attacks**

Viruses and malware attacks in the context of smart grids can have serious consequences, potentially disrupting the reliable and efficient distribution of energy. These malicious software threats target the digital components of the smart grid system, including computers, communication networks, sensors, and control systems. These attacks are known as ransomware, worms, botnets, malware for remote control, data exfiltration malware, and firmware attacks [21-25].

Ransomware is a type of malware that encrypts critical data and systems, rendering them inaccessible until a ransom is paid to the attackers. In a smart grid scenario, ransomware could lock down key components, such as energy management systems or distribution control centers, causing disruptions in energy distribution until the ransom is paid or the systems are recovered. Worms are self-replicating malware that spread across networks and devices. If a worm infects smart grid components, it can rapidly propagate through the system, causing widespread outages and potentially damaging physical equipment. Botnets are networks of compromised devices that are controlled by a central entity, often used for coordinated attacks. If a smart grid system becomes part of a botnet, the attackers can use it to launch large-scale distributed denial of service (DDoS) attacks or other forms of cyberattacks [27-28].

Malware designed for remote control can give attackers the ability to manipulate the smart grid's control systems. This could result in unauthorized changes to energy distribution, equipment damage, or even physical safety risks. Data

Exfiltration Malware aims to steal sensitive data from the smart grid, such as customer information or system configurations. Stolen data can be exploited for financial gain or used to gain a competitive advantage. Malware can target the firmware of devices within the smart grid infrastructure. Compromised firmware can lead to unstable or unpredictable behavior in devices, potentially causing disruptions in energy flow or control. By taking these precautions and continuously monitoring the smart grid infrastructure, utilities and organizations can reduce the risk of viruses and malware compromising the integrity and functionality of the energy distribution system [29].

#### **2.2.2.4 Horse Trojan Attacks**

A Horse Trojan, also known as a Trojan Horse, is a type of malicious software that disguises itself as a legitimate program or file but contains harmful code that can compromise the security and functionality of a system. In the context of smart grids, a Horse Trojan attack can have severe implications for the reliable distribution of energy and the overall operation of the grid. This attack includes infiltration, installation, backdoor access, manipulation of control systems, and data theft and espionage. In infiltration, the attacker gains access to the smart grid system, either through a vulnerable point in the network or by targeting a specific individual with a phishing or social engineering attack. The Trojan is disguised as a harmless or necessary software component, such as a firmware update or a control system software [30, 31].

In installation, once the Trojan is introduced into the smart grid network, it is executed, and its malicious code is activated. The Trojan may be designed to remain dormant for a certain period to avoid detection, making it challenging to identify its presence. In backdoor access, the Horse Trojan creates a backdoor, granting the attacker unauthorized access to the smart grid's control systems, sensors, and communication networks. This access allows the attacker to manipulate grid settings, disrupt energy distribution, or gather sensitive data.

In manipulation of control systems, with control over the smart grid's systems, the attacker can issue commands that manipulate energy distribution, alter voltage levels, or shut down critical components. This could result in widespread power outages, equipment damage, or even safety hazards. In data theft and espionage, the Horse Trojan might also be programmed to steal sensitive information, such as customer data, operational details, or grid configurations. The stolen data can be exploited for financial gain or used to gain a competitive advantage. By implementing these measures, smart grid operators can significantly reduce the risk of Horse Trojan attacks and enhance the security and resilience of the energy distribution system [33-36].

### **2.2.2.5 Popping Human Interface Attacks**

Pop-up (popping) human interface attacks in the context of a smart grid refer to a specific type of social engineering attack where an attacker manipulates the user interface or interactions with the system to deceive operators or personnel into making unintended and potentially harmful actions. These attacks exploit the human element in the operation of the smart grid, aiming to compromise its integrity and functionality. The examples of these attacks are deceptive interface, phony alerts and warnings, unauthorized actions, malicious payload, and exploiting cognitive biases [37-39].

In deceptive interface, the attacker creates a deceptive user interface that mimics the legitimate control panel or software used by smart grid operators. This interface may contain fake alerts, warnings, or requests for action that appear genuine. In phony alerts and warnings, the deceptive interface generates pop-up alerts or warnings that seem urgent and critical. For example, it might claim there's a system malfunction, impending equipment failure, or a cyber threat that requires immediate attention. In unauthorized actions, the pop-up alert prompts the operator to take specific actions, such as adjusting settings, rerouting energy flows, or shutting down critical components to mitigate the perceived threat. In reality, these actions could disrupt energy distribution or compromise the security of the smart grid [40].

In malicious payload, the deceptive interface could carry a payload that, when interacted with, executes malicious commands or introduces malware into the system. This malware could potentially gain unauthorized access, cause outages, or compromise data. In exploiting cognitive biases, the attacker may exploit cognitive biases, such as urgency bias (making rash decisions under pressure), authority bias (following instructions from perceived higher-ups), or confirmation bias (only seeking information that confirms preconceived notions). By raising awareness, implementing safeguards, and maintaining a vigilant approach to user interactions, smart grid operators can reduce the risk of falling victim to "popping" human interface attacks and ensure the secure and reliable operation of the energy distribution system [20-32, 37].

### **2.2.3 Cyber-Attacks Targeting Network Channel**

Cyber-attacks targeting network channels are a type of malicious activity aimed at exploiting vulnerabilities within the communication pathways that connect various devices, systems, and components in a network. These attacks can disrupt communication, compromise data integrity, and potentially grant unauthorized access to critical systems. There are several attacks targeting network channels, namely man-in-the-middle (MITM), eavesdropping, jamming, injection, puppet, Smurf, masquerading, and smart meter tampering attacks [25-30].

### **2.2.3.1 Man-in-the-Middle Attacks**

Man-in-the-Middle (MitM) attacks in the context of a smart grid refer to a type of cyber-attack where an unauthorized third-party intercept and potentially alters the communication between two legitimate parties within the smart grid network. These attacks exploit vulnerabilities in the communication channels between smart grid components, potentially compromising data integrity, stealing sensitive information, or even gaining unauthorized control over critical systems. In smart grid, this attack consists of two vulnerabilities, namely weak encryption and unauthenticated connections. In weak encryption, if the communication between smart grid devices is inadequately encrypted, the attacker can easily decrypt and understand the intercepted data. In contrast, in unauthenticated connections, lack of proper authentication mechanisms can allow the attacker to impersonate one or both parties and establish a connection without being detected [20-25].

MitM attacks have three potential consequences, namely data tampering, data theft, and control manipulation. In data tampering, the attacker can modify the intercepted data packets before forwarding them to the intended recipient, potentially causing unintended actions or disruptions. In data theft, sensitive information, such as customer data, financial records, or operational details, can be stolen and used for malicious purposes. In control manipulation, by altering control commands, the attacker could manipulate the behavior of smart grid components, affecting energy distribution and causing outages. As a result, MitM attacks pose a significant threat to the security and reliability of smart grid systems. By implementing these mitigation strategies and maintaining a proactive approach to network security, smart grid operators can minimize the risk of MitM attacks and ensure the integrity of their energy distribution infrastructure [27, 28].

### **2.2.3.2 Eavesdropping Attacks**

In the context of a smart grid, an eavesdropping attack refers to a malicious activity where an unauthorized entity intercepts and listens to communication within the smart grid network, with the intent of gathering sensitive information or compromising the integrity of data transmissions. Eavesdropping attacks in the smart grid can have serious implications for the security, privacy, and reliability of the energy distribution system. In this attack, the attack process consists of several steps, namely interception, packet capture, and data analysis. In interception, the attacker gains access to the communication channels within the smart grid network. This could involve exploiting vulnerabilities in network infrastructure, devices, or software. In packet capture, the attacker captures data packets that are being transmitted between different components of the smart grid, such as smart meters, sensors, control centers, and substations. In data analysis,

the attacker analyzes the captured data packets to extract valuable information, such as energy consumption patterns, control commands, operational details, or even encryption keys [30, 32, 37-40].

This attack also has several potential consequences, such as data theft, operational disruption, privacy violation, and cyber-espionage. In general, by eavesdropping on communication, attackers can steal sensitive data, including customer information, financial records, and proprietary grid configurations. In addition, eavesdropping attacks can lead to disruptions in the operation of the smart grid, as attackers may gain insights into system vulnerabilities or manipulate critical control commands. Moreover, eavesdropping compromises the privacy of individuals and organizations by exposing their confidential communications and activities within the smart grid network. Adversaries, such as competitors or nation-state actors, might engage in eavesdropping to gather intelligence on energy distribution strategies or national infrastructure. By implementing these mitigation strategies and maintaining a proactive approach to network security, smart grid operators can reduce the risk of eavesdropping attacks and safeguard the confidentiality, integrity, and reliability of their energy distribution systems [11, 18, 29, 40].

### **2.2.3.3 Puppet Attacks**

Puppet attacks are not a commonly recognized term in the field of cybersecurity or smart grids. However, based on your description, it appears that you may be referring to a scenario where an attacker gains control over certain components or systems within a smart grid network and uses them as puppets to carry out malicious actions. This could involve manipulating smart grid devices to execute unauthorized commands or behaviors. Puppet attacks in the context of a smart grid involve an attacker gaining unauthorized control over specific devices or components within the grid and using them as "puppets" to carry out malicious actions. These actions may include altering energy distribution, disrupting operations, or compromising the security of the smart grid network. These attacks consist of the following process, unauthorized access, remote control, and malicious actions [25, 26].

As a matter fact, the attacker gains access to vulnerable or poorly secured smart grid devices, such as smart meters, sensors, or controllers. Once access is established, the attacker takes control over these compromised devices, effectively turning them into puppets under their command. The attacker uses the compromised devices to carry out malicious actions, such as manipulating energy flow, causing power outages, or interfering with grid operations. It also has several consequences, namely operational disruption, safety risks, and data compromise. Puppet attacks can disrupt the reliable operation of the smart grid by causing unexpected changes in energy distribution or control systems. Manipulating smart grid components could lead to safety hazards for personnel, consumers, and the environment. Attackers could use



compromised devices to access sensitive data or exploit vulnerabilities in the grid's communication infrastructure. By adopting these mitigation strategies and maintaining a proactive approach to network security, smart grid operators can reduce the risk of puppet attacks and ensure the reliable and secure operation of their energy distribution systems [27].

#### **2.2.3.4 Smurf Attacks**

A Smurf attack is a type of cyber-attack that involves flooding a target network with a large volume of Internet Control Message Protocol (ICMP) packets. These packets are typically sent using IP broadcast addresses, causing the target's network to become overwhelmed with traffic and potentially leading to disruption or denial of service. In the context of a smart grid, a Smurf attack could have detrimental effects on the communication infrastructure and operations. A Smurf attack in a smart grid involves an attacker sending a large number of ICMP packets to smart grid devices or communication infrastructure using broadcast addresses. This flood of packets overwhelms the network's capacity and can lead to communication breakdowns or service disruptions. These attacks consist of the following process, namely ICMP packet generation, broadcast amplification, and network overload [28, 29].

The attacker sends a flood of ICMP Echo Request packets to a broadcast address within the smart grid network. Since the broadcast address is used, all devices within the broadcast domain respond to each packet, amplifying the traffic and consuming network resources. The high volume of ICMP responses saturates the network bandwidth, causing congestion and potential denial of service for legitimate communication. These attacks have some potential consequences, namely communication disruption, operational impact, and resource consumption. Smart grid devices rely on effective communication to coordinate operations. A Smurf attack can disrupt communication channels and lead to service outages.

Overwhelmed networks can lead to delays or failures in transmitting critical data, control commands, and operational information. Network resources, such as bandwidth and processing power, are consumed by the attack, affecting the overall performance of the smart grid. By implementing these measures and maintaining vigilant network security practices, smart grid operators can reduce the risk of Smurf attacks and ensure the reliable and secure operation of their energy distribution systems [30].

#### **2.2.3.5 Masquerading Attacks**

Masquerading attacks, also known as identity spoofing or impersonation attacks, involve a malicious actor pretending to be a legitimate user, device, or entity to gain unauthorized access or carry out malicious actions within a network or system. In the context of a smart grid, masquerading attacks can undermine the security and integrity of the energy distribution system. Masquerading attacks in a smart grid involve an attacker pretending to be a trusted entity or device

within the grid's communication network. By impersonating legitimate participants, the attacker gains access to sensitive information, control commands, or other resources, potentially leading to operational disruptions or security breaches. The attack process consists of identity falsification, unauthorized access, and malicious actions [32].

In general, the attacker falsifies identification credentials, such as IP addresses, device IDs, or authentication tokens, to mimic a legitimate user or device. Using the falsified identity, the attacker enters the smart grid network, often bypassing authentication and access controls. Once inside the network, the attacker can execute unauthorized actions, such as manipulating energy distribution, altering configurations, or stealing sensitive data. These attacks have some consequences, such as operational disruptions, data compromise, and unauthorized control. Masquerading attacks can disrupt the smooth operation of the smart grid by introducing unauthorized changes or commands. Attackers can access and steal sensitive information, such as customer data or grid configurations. By masquerading as an authorized entity, attackers can gain control over critical smart grid components or systems. By implementing these measures and maintaining a proactive approach to network security, smart grid operators can reduce the risk of masquerading attacks and ensure the secure and reliable operation of their energy distribution systems [35].

#### **2.2.3.6 Smart Meter Tampering Attacks**

Smart meter tampering attacks in a smart grid involve malicious actions aimed at manipulating or altering the functionality of smart meters. These attacks can lead to inaccurate measurement of energy consumption, fraudulent activities, or disruptions in the energy distribution system. Smart meter tampering attacks target the integrity of smart meters, which are essential components of a smart grid used to measure and monitor energy consumption in real time. Attackers attempt to manipulate or compromise these meters to achieve various malicious goals. These attacks consist of physical tampering, firmware manipulation, and hacking interfaces. Attackers physically manipulate smart meters by bypassing measurement components, altering calibration settings, or disconnecting sensors to underreport energy usage. Attackers modify the firmware or software of smart meters to manipulate measurement algorithms, leading to inaccurate readings. Also, if smart meters have remote access interfaces (e.g., wireless or network connections), attackers might exploit vulnerabilities to gain unauthorized access and tamper with meter functionality [1, 18, 29, 34, 40].

The potential consequences of these attacks are known as fraudulent consumption, service disruption, revenue loss, and security breach. Tampered smart meters can underreport energy consumption, leading to financial losses for utility providers and incorrect billing for consumers. Manipulated smart meters might disrupt energy distribution, impact load balancing, or cause grid instability due to inaccurate measurements. Utilities can suffer revenue loss if tampered meters

result in unaccounted-for energy usage. Tampering could potentially compromise the security of the entire smart grid network, providing attackers with a foothold for broader cyber-attacks. By implementing these mitigation strategies and maintaining a vigilant approach to smart meter security, utility providers and smart grid operators can minimize the risk of smart meter tampering attacks and ensure the accuracy, reliability, and security of their energy distribution systems [1, 17, 20].

### **2.2.3.7 Traffic Analysis Attacks**

A traffic analysis attack in the context of a smart grid refers to a type of cyberattack where an adversary intercepts and analyzes the communication traffic between different components and devices within the smart grid infrastructure. This attack aims to gather valuable information about the operations, behaviors, and patterns within the smart grid network, which can then be exploited for malicious purposes. In a smart grid, various devices such as smart meters, sensors, substations, and control systems communicate with each other to monitor and manage the distribution and consumption of electrical power. These communications often involve sensitive data, including real-time power consumption information, operational commands, and system status updates. A traffic analysis attack seeks to exploit the information transmitted between these devices, even if the attacker cannot directly access the content of the messages [18, 28, 30, 40-43].

The goal of a traffic analysis attack may include inference of consumer behavior, operational insights, privacy violations, and critical infrastructure. Generally, by analyzing patterns in power consumption data, attackers may infer when individuals are present at their homes, when they are likely to be away, and other behavioral patterns. This information can be used for activities such as burglary or social engineering attacks. Additionally, adversaries can gain insights into the operational behavior of the smart grid, identifying vulnerabilities or weaknesses that can be exploited to disrupt power distribution or manipulate the grid's functionality. If attackers can link specific consumption patterns to individual households, they can breach privacy by revealing personal habits and activities. Moreover, attackers may identify critical substations or components within the smart grid to target, aiming to cause widespread power outages or other disruptions [2, 5, 10, 40].

As a result, all these cyber-attacks may violate different security requirements, as shown in Table I. For example, jamming attacks may disrupt the transmission and reception of data by violating network availability. Intrusion attacks may use the network's available resources, jeopardizing the network's confidentiality and integrity. Spoofing attacks can mislead legitimate nodes in the network, leading to confidentiality, integrity, availability, and accountability violation. More details of cyber-attacks, their impacts, purposes, and security requirements can be found in Table 2.1.

Table 2.1 Summary of Cyber-Attacks Targeting Smart Grid.

<b>Cyber-Attacks</b>	<b>Objectives/Purpose</b>	<b>Impacts</b>	<b>Security Requirements</b>
Jamming Attacks	<ul style="list-style-type: none"> <li>Disrupting the transmission and the reception of data.</li> </ul>	<ul style="list-style-type: none"> <li>Blocking one or several nodes to transmit and receive information collisions.</li> </ul>	<ul style="list-style-type: none"> <li>Availability</li> </ul>
Spoofing Attacks	<ul style="list-style-type: none"> <li>Pretending to be a legitimate node to compromise the system.</li> </ul>	<ul style="list-style-type: none"> <li>Misleading other nodes.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> <li>Availability</li> <li>Confidentiality</li> <li>Accountability</li> </ul>
Injection Attacks	<ul style="list-style-type: none"> <li>Injecting false/untrusted data packets into a network.</li> </ul>	<ul style="list-style-type: none"> <li>Injecting false data</li> <li>Corrupting the legitimate processes and operations</li> <li>Appearance of illegitimate nodes in the network.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> </ul>
Flooding Attack	<ul style="list-style-type: none"> <li>Depleting, and exhausting system resources.</li> </ul>	<ul style="list-style-type: none"> <li>Malfunction of nodes and loss of availability in a network.</li> </ul>	<ul style="list-style-type: none"> <li>Availability</li> </ul>
Man-in-the-Middle Attacks	<ul style="list-style-type: none"> <li>Preventing, or modifying data during transmission through the network.</li> </ul>	<ul style="list-style-type: none"> <li>Unauthorized access to sensitive information.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> <li>Confidentiality</li> </ul>
Social Engineering Attacks	<ul style="list-style-type: none"> <li>Manipulating users to reveal sensitive information.</li> </ul>	<ul style="list-style-type: none"> <li>Violation of users' privacy.</li> <li>Temporary or permanent damage to the system.</li> <li>Steal sensitive and private information.</li> <li>Identity theft.</li> </ul>	<ul style="list-style-type: none"> <li>Confidentiality</li> </ul>
Eavesdropping Attack	<ul style="list-style-type: none"> <li>Monitoring and capturing all network traffic.</li> </ul>	<ul style="list-style-type: none"> <li>Loss of privacy.</li> </ul>	<ul style="list-style-type: none"> <li>Confidentiality</li> </ul>
Intrusion Attack	<ul style="list-style-type: none"> <li>Gain illegal access to the node or network.</li> </ul>	<ul style="list-style-type: none"> <li>Misusing available resources in the network.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> <li>Confidentiality</li> </ul>
Brute Force Attacks	<ul style="list-style-type: none"> <li>Cracking usernames and passwords.</li> </ul>	<ul style="list-style-type: none"> <li>Gaining unauthorized access to users' systems or accounts.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> <li>Confidentiality</li> </ul>
Time synchronization Attack	<ul style="list-style-type: none"> <li>Targeting timing data and disrupting the time synchronization between nodes.</li> </ul>	<ul style="list-style-type: none"> <li>Compromising events, such as location estimation and fault detection</li> <li>Performance degradation.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> <li>Availability</li> </ul>
Traffic Analysis Attack	<ul style="list-style-type: none"> <li>Control the hosts and the devices that are connected to the network.</li> </ul>	<ul style="list-style-type: none"> <li>Sniff and analyze the message to achieve information about the patterns of communications between nodes.</li> </ul>	<ul style="list-style-type: none"> <li>Confidentiality</li> </ul>
Masquerade Attack	<ul style="list-style-type: none"> <li>Pretend to be an authorized user.</li> </ul>	<ul style="list-style-type: none"> <li>Gaining unauthorized access to users' systems.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> <li>Availability</li> <li>Confidentiality</li> <li>Accountability</li> </ul>
Smart Meter Tampering Attack	<ul style="list-style-type: none"> <li>Modification of the transmitted data for any customers.</li> </ul>	<ul style="list-style-type: none"> <li>Pay higher or lower electricity bills.</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> </ul>
Buffer Overflow Attack	<ul style="list-style-type: none"> <li>Sending improper or incorrect data to the specific system.</li> </ul>	<ul style="list-style-type: none"> <li>System crashes or exhaust resources.</li> </ul>	<ul style="list-style-type: none"> <li>Availability</li> </ul>
Puppet Attack	<ul style="list-style-type: none"> <li>Sending fake data in the Advanced Metering Infrastructure network.</li> </ul>	<ul style="list-style-type: none"> <li>Reduce packet delivery to 10% or 20%</li> </ul>	<ul style="list-style-type: none"> <li>Availability</li> </ul>

		<ul style="list-style-type: none"> <li>Exhaust the communication network bandwidth.</li> </ul>	
Smurf Attack	<ul style="list-style-type: none"> <li>Modifying the traffic of an entire system.</li> </ul>	<ul style="list-style-type: none"> <li>Replay and saturate the target network.</li> </ul>	<ul style="list-style-type: none"> <li>Availability</li> </ul>
Popping the HMI attack	<ul style="list-style-type: none"> <li>Get unauthorized access</li> </ul>	<ul style="list-style-type: none"> <li>Controlling the compromised system</li> </ul>	<ul style="list-style-type: none"> <li>Integrity</li> <li>Availability</li> <li>Confidentiality</li> <li>Accountability</li> </ul>

### 2.3 Scenarios of Intrusion Attacks

One damaging impact of cyber-attacks on smart grid is blackouts. A blackout refers to the complete loss of electrical power in a specific geographic area, often resulting from a failure or disruption within the power generation, transmission, or distribution system. In the context of smart grids, blackouts can have significant and far-reaching consequences due to the intricate interconnection and interdependence of modern power systems [43]. This impact, blackout, is considered as one of the key factors that occurred by intrusion attacks. Smart grids, as technologically advanced and interconnected systems, are designed to enhance the efficiency, reliability, and flexibility of electricity distribution [44]. They achieve this through real-time monitoring, data analysis, and automated control mechanisms. However, despite their advancements, smart grids are not immune to vulnerabilities, such as intrusion attacks, and blackouts can occur as a result of intrusion attacks on smart grids [45,46]. The example of intrusion attacks disruptions on smart grid is physical infrastructure failure, human error, and overloading and imbalance in electricity demands.

The consequences of blackouts on smart grids can be severe and extend beyond mere inconvenience. They can disrupt essential services, cause financial losses, compromise industrial processes, and even endanger public safety. Additionally, blackouts can have cascading effects, impacting neighboring regions and potentially leading to longer recovery times. Therefore, motivated by the importance of intrusion attacks, detailed descriptions of intrusion attacks targeting smart grid with respect to the blackouts. In the following, several scenarios of these attacks that occurred in the last few years are provided.

- In January 2020, a deliberate intrusion attack was launched, targeting the power systems and natural gas infrastructure spanning the states of Oklahoma, Texas, and Kansas. In the course of this audacious attack, the perpetrators managed to seize control over the critical network components, wielding an unprecedented level of unauthorized authority. As a result of this breach, the assailants not only gained command over the intricate web of interconnected systems but also managed to exfiltrate a portion of sensitive data. This incident underscores the

alarming potential for cyber adversaries to manipulate and compromise essential energy-related assets, thereby highlighting the pressing need for heightened cybersecurity measures within the realm of critical infrastructure.

- In the spring of 2021, an alarming intrusion attack unfolded within the smart grid system of Houston, Texas – a bustling metropolis renowned for its energy prominence. This pivotal city, nestled within the heart of the Lone Star State, encountered a formidable cyber assault that targeted its cutting-edge smart grid infrastructure. The attackers exploited a previously undiscovered vulnerability within Houston's smart grid control software, gaining surreptitious access to critical nodes that regulated the city's energy distribution network. This breach set the stage for a well-orchestrated attack that had the potential to disrupt not only Houston's power flow but also to trigger cascading effects that reverberated across the broader Texas power grid.
- In 2021, a significant intrusion attack unfolded targeting the smart grid infrastructure of Atlanta, Georgia – a bustling hub of technology and commerce. Amidst the digital shadows, malicious actors seized upon a vulnerability within Atlanta's smart grid control systems, breaching its defenses and gaining unauthorized access to critical control points. This breach ushered in a concerning phase where the attackers wielded control over essential aspects of the city's energy distribution network. Exploiting this newfound access, the attackers executed a calculated manipulation of grid operations. Subtly altering control settings, they induced localized power fluctuations that served as a stark reminder of the potential chaos their actions could unleash. While the immediate impact was constrained, the strategic nature of the attack raised alarm bells about the broader ramifications.
- In 2022, a concerning intrusion attack unfolded, targeting the smart grid infrastructure of Portland, Oregon. This vibrant urban center found itself thrust into the midst of a disruptive digital assault that sent shockwaves through its energy distribution networks. The attack's origins lay in the exploitation of a previously undetected vulnerability within Portland's smart grid control architecture. Adversaries deftly breached the city's defenses, gaining illicit access to pivotal nodes within the energy distribution network. This unauthorized access provided them with an ominous degree of control over critical components.
- In 2022, an intrusion attack within Miami, Florida's smart grid happened. The genesis of this attack lay in the exploitation of a hitherto undiscovered vulnerability within Miami's smart grid control infrastructure. Adversaries deftly navigated through the city's defenses, circumventing safeguards to gain surreptitious access to pivotal junctions within the energy distribution network. This unauthorized access bestowed upon them a formidable degree of control over critical operational elements. The attackers embarked on manipulation of grid operations.

By subtly altering control parameters, they orchestrated localized power fluctuations that served as a stark demonstration of their ability to disrupt the status quo. While the immediate impact was managed, the calculated nature of the attack sounded an urgent alarm about the broader implications that could unfold.

- In April 2023, an intrusion attack targeting smart grid happened in Chicago, Illinois. Malicious actors exploited a vulnerability in the city's smart grid communication system, gaining unauthorized access to key control nodes. This breach allowed the attackers to manipulate power distribution across the city. The impact was felt by residents and businesses as sporadic power outages occurred over the course of several hours. Rapid response teams were deployed to isolate the compromised components and restore normal operations, highlighting the continued vulnerability of even advanced smart grid systems.
- In July 2023, New York City fell victim to an intrusion attack on its smart grid. Cybercriminals exploited a weakness in the city's smart metering system, enabling them to remotely manipulate energy consumption data. This manipulation led to an imbalance in the grid's load distribution, triggering a domino effect that resulted in localized blackouts in certain neighborhoods. Prompt intervention by utility personnel and cybersecurity experts helped rectify the situation and restore power within a few hours. The incident underscored the susceptibility of smart grid systems to targeted attacks aimed at disrupting energy distribution.
- In August 2023, an intrusion attack struck San Francisco's smart grid infrastructure. The attackers exploited a vulnerability in the city's smart grid control software, granting them unauthorized access to critical control systems. This breach enabled the perpetrators to remotely alter power settings, leading to voltage instability and disruptions in energy supply to homes and businesses. The incident prompted a citywide response, with emergency measures enacted to stabilize the grid and restore power. It also highlighted the need for ongoing cybersecurity enhancements to safeguard smart grid systems against evolving threats.

As a result of these instances, after an investigation about intrusion attacks on smart grid, a concerning statistical overview came to light [43-46]. In fact, more than 60% of the power grids across the United States found themselves susceptible to the ominous specter of intrusion attacks. This unsettling statistic underscored a sobering reality that the systems underpinning the nation's energy infrastructure were exposed to a growing digital threat. Also, in June 2021, the Energy Chief of the U.S. mentioned intrusion attacks as the biggest threat to the U.S. power systems in the last few months, since smart power grids are exposed to multiple cyber intrusions.

The adversaries had access to the U.S. power system, shut it down for a few hours, and gained access to the network and users' data. Intrusion attacks are estimated to cost over six trillion dollars in utility services all over the world. The rate of these attacks in Mach 2023 has an 62% increase in comparison with the rate of these attacks in March 2022. Thus, to these examples and the statistical overview This dissertation also attempts to fill the existing gap in the literature by developing and proposing techniques to reduce the rates of intrusion attacks on smart grids.

## **2.4 Literature Review of Detection Techniques on Smart Grid**

For detecting cyber-attacks, machine learning category received more attention from researchers. For example, the authors of [47] used machine-learning to detect jamming attacks, namely random forest, support vector machine, and neural network. Their numerical results show that the proposed technique based on random forest achieves high accuracy. In [48], the authors also used machine learning algorithms to detect social engineering attacks. The technique performs based on unsupervised learning, which means that there is no previous knowledge about the observed cyber-attacks. They compared the performance of different machine learning algorithms (support vector machine, biased support vector machine, artificial neural, scaled conjugate gradient, and self-organizing map) in terms of reliability, accuracy, and speed. Their simulations proved that support vector machine give better results compared to other algorithms.

Another study in the literature that exploited machine-learning is detailed in [49]. In this study, the authors highlighted a concept from statics and economics, named "first difference," which led them to develop a classifier to detect time synchronization attacks in the network. Their results show that Artificial Neural Networks (ANN) are the best choice for detecting these attacks in the network. In [50], the authors used an ANN model to detect MITM attacks and their results did provide a good detection rate. In [51], the authors used machine learning techniques to detect and locate intruders in smart grids. The simulation results of this study showed that the proposed method could achieve a good detection rate. Deep learning techniques has also been used to detect cyber-attacks targeting smart grid infrastructure. For instance, in [52], the authors proposed ensemble deep learning techniques, using deep neural network (DNN) and decision tree. The proposed model was evaluated based on the 10-fold cross validation. The evaluation results showed that the proposed model outperforms other traditional techniques, including random forest, AdaBoost, and DNN.

In [53], the authors applied a deep reinforcement learning based technique to identify the physical tripped line and the fake outage line. Another study [54] also employed a deep learning technique, called encoders to reduce dimensions and feature extraction, followed by an advanced Generative Adversarial Network (GAN) to detect false data injection attacks. In this study, due to expensive costs of labelling in power systems, the collected data is partially labelled, therefore,



numerical results show the effectiveness and high rate of accuracy of this model. Another type of AI-based category is data mining algorithms that can be useful for detecting cyber-attacks in the smart grid.

Fuzzy logic-based methods have also been proposed to effectively detect various attacks in a network. For example, the authors of [55] proposed artificial immune systems and fuzzy logic in order to detect flooding attacks in a network. In this study, the aim of using fuzzy logic is to reduce uncertainty whenever there is no clear line between malicious and legitimate traffic. Another study that applied fuzzy logic in cyber-attack detection is described in [56], in which the authors developed a detection technique based on fuzzy logic for jamming attacks. This technique uses the clear channel assessment, bad packet ratio, and received strength signal parameters to detect link loss due to jamming or other causes. The efficiency of their proposed techniques for constant and random jamming was high.

Other authors [57] combined fuzzy logic with other approaches, such as genetic algorithms and Hidden Markov Model (HMM), to detect various cyber-attacks. Another important type in AI-based techniques is that of evolutionary algorithms, which are widely used for global optimizations. One popular instance of evolutionary algorithms is genetic algorithms. These algorithms can mimic the evolution and natural selection process. In [58], the authors proposed a technique based on a genetic algorithm that consists of two stages, training and detection. In their work, they applied a genetic algorithm for reducing the set of features in the detection stage. According to the authors' results, this technique provides a high level of accuracy for various cyber-attacks in networks.

In [59], the authors also analyzed the impact of genetic algorithms on various machine-learning techniques, such as SVM, KNN, and ANN. The simulation results show that genetic algorithms and these three machine learning techniques effectively detect FDIA in smart grids. However, KNN and SVM were found more efficient in detecting these attacks than some existing techniques. In another work, the authors proposed a hybrid technique based on Genetic algorithms and fuzzy logic [60]. They developed a multi-objective genetic-fuzzy model for detecting anomalies in networks. The numerical results show that the proposed method is not suitable for detecting attacks in networks. In [61], the authors also analyzed a hybrid framework for detecting different cyber-attacks that apply both genetic and fuzzy logic techniques. This method provided good accuracy and better results compared to some other existing techniques.

In recent years, Reinforcement Learning (RL) techniques have been capable of learning in an environment with an unmanageable huge number of states to address critical shortcomings of RL. Deep Reinforcement Learning (DRL) methods, such as Deep Q-Networks (DQN), have been shown to be promising techniques to handling these scenarios by leveraging DL models during the learning process. For example, the authors of [62] proposed four RL-based approaches,

namely Deep Q-Network, Double Deep Q-Network, Policy Gradient, and Actor Critic, to detect intrusions in supervised scenarios. They used a simple neural network as DL model to integrate with Q-network. In [63], the authors proposed a DRL-based technique adaptive cloud IDS system to classify complex attacks on the network. In [64], the authors also developed a DQN model, using an automated trial-error approach and continuously enhancing its detection capabilities. In this study, the authors mostly focused on using feed-forward neural network as DL model in the proposed DQN model. Another study [65] was conducted to propose anomaly-based IDS using DRL technique. In this study, the authors applied a simple DQN model to have the ability to self-update the behavior of the network. In [66], the authors proposed a network IDS using adversarial RL with DQN to detect intrusions on the network.

## Chapter 3

### DETECTION METHODOLOGIES

As it was outlined in the previous chapter, the increasing need for detecting intrusion attacks in smart grids stems from the growing integration of modern technologies into critical infrastructure, such as electrical grids. Smart grids leverage advanced communication and control systems to optimize energy distribution, enhance grid stability, and enable two-way communication between utilities and consumers. While these advancements offer numerous benefits, they also introduce new vulnerabilities and security challenges, making intrusion detection crucial. To detect intrusion attacks in smart grid, utilities and organizations operating smart grids need to implement robust cyber-security measures. For this purpose, comprehensive and proactive approaches to detect and classify intrusion attacks is crucial to safeguarding the integrity, reliability, and security of smart grid operations are required. To contribute to filling this gap, in this chapter, three different machine learning-based detection techniques, namely supervised, unsupervised, and reinforcement learning models are discussed. These techniques are investigated to demonstrate their effectiveness to detect and classify intrusions on smart grids. Additionally, necessary steps in detection process, dataset acquisition, data pre-processing and hyperparameter tuning techniques are highlighted.

#### **3.1 Attack Detection Procedure Using Machine and Deep Learning**

Machine Learning (ML) and Deep Learning (DL) are classified into three main categories, namely Supervised, unsupervised, and reinforcement learning. These categories represent different approaches to how the learning models are trained and the type of data they work with. In supervised learning, the algorithm is trained on a labeled dataset, where each input data point is associated with a corresponding target or label [67]. The goal of the algorithm is to learn a mapping from inputs to outputs so that it can make predictions or classifications on new, unseen data. An overview of supervised learning technique is indicated in Figure 3.1.

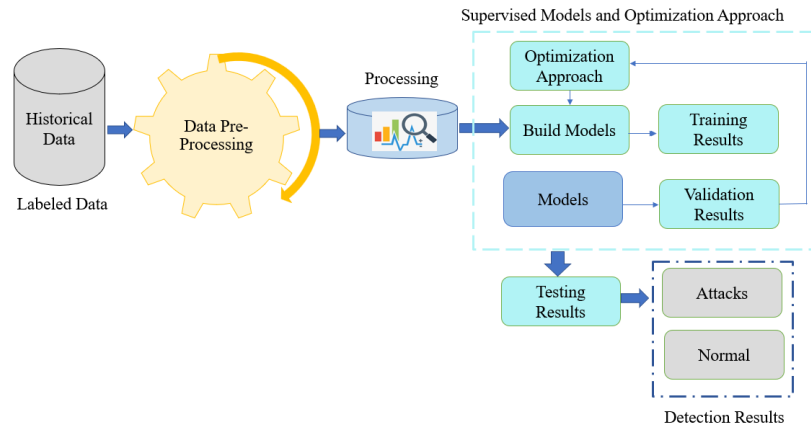


Figure 3.1 Supervised Learning Working Flow.

In supervised learning, the workflow for building and training a supervised machine learning model consists of several key steps, from data acquisition to model optimization, as shown in Figure 3.1. Data acquisition involves gathering the necessary data (historical data) for detecting and classifying intrusion attacks on smart grid. In data pre-processing step, it is essential to assess data quality, completeness, and relevance. In fact, this step is a fundamental phase in the learning pipeline that involves a series of systematic transformations applied to historical data to enhance its quality, relevance, and compatibility for subsequent analysis. This encompassing process encompasses techniques such as data balancing to address class imbalance, imputation to handle missing values, normalization to standardize feature scales, and encoding to convert categorical variables into numerical representations. By carefully curating and refining the data through these steps, data preprocessing ensures that the resulting dataset is well-structured, representative, and conducive to effective model training, ultimately fostering improved model accuracy, robustness, and interpretability [67-69].

Next phase, model selection and optimization approach, are integral components of the learning model development process, aimed at optimizing model performance and generalization. Model selection involves the meticulous evaluation of diverse algorithms to identify the most suitable one for a specific task, considering factors such as algorithm complexity and compatibility with the problem domain. Optimization approach (hyperparameter tuning) on the other hand, revolves around refining the internal settings of a chosen model to achieve optimal results. By iteratively adjusting hyperparameters, such as learning rates, regularization strengths, and network architectures, practitioners seek to strike a delicate balance between underfitting and overfitting, ultimately enhancing the model's capacity to capture intricate patterns within the data while minimizing errors. This intricate interplay between model selection and hyperparameter tuning is pivotal in yielding models that robustly and accurately generalize to unseen data, thus driving the efficacy of machine learning endeavors [68].

Unsupervised learning constitutes a foundational facet of machine learning where the principal emphasis lies in extracting inherent structures and patterns from unlabeled data. Through a process of autonomous exploration, unsupervised learning algorithms endeavor to comprehend the underlying relationships and dependencies within the dataset. This is chiefly achieved through techniques such as clustering, wherein similar data points are grouped together, and dimensionality reduction, which aims to reduce the complexity of the data while preserving its salient characteristics. Unconstrained by predefined output labels, unsupervised learning serves as a potent tool for data exploration, visualization, and feature extraction, enabling the discovery of hidden insights and driving advancements in diverse domains ranging from data compression to anomaly detection [50]. An overview of unsupervised learning is indicated in Figure 3.2.

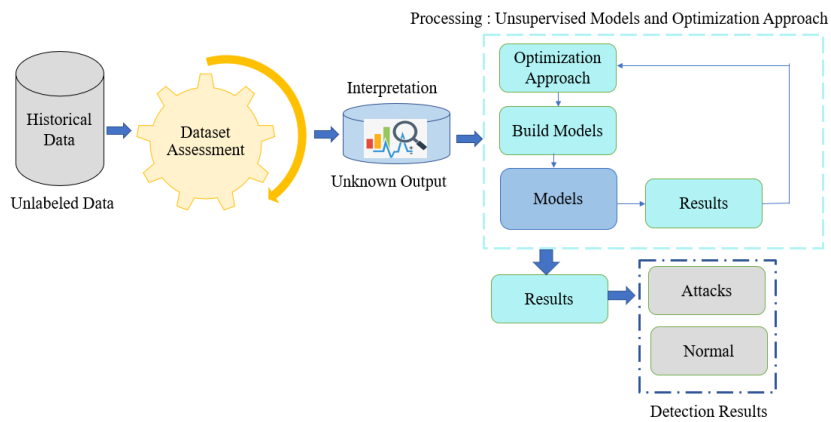


Figure 3.2 Unsupervised Learning Working Flow.

Unsupervised learning encompasses a sequence of distinct phases designed to unveil meaningful patterns from unlabeled historical data. The initial phase involves data acquisition and data pre-processing, where historical data is collected and refined to ensure compatibility with subsequent analysis. Following this, the exploration and clustering phase ensues, where algorithms autonomously identify natural groupings within the data, facilitating the organization of similar instances into clusters. Simultaneously, dimensionality reduction techniques are employed to compress the data while retaining essential information. The subsequent phase involves interpreting and validating the results, where domain expertise aids in comprehending the significance of the identified clusters and reduced dimensions. Throughout these phases, the iterative and interactive nature of unsupervised learning guides the refinement of analysis parameters and techniques, culminating in a deeper understanding of the data's underlying structure and potential applications [67-68].

Reinforcement learning constitutes a dynamic branch of machine learning centered on developing and refining intelligent agents that interact with an environment to attain specific goals. Characterized by the pursuit of sequential decision-making, reinforcement learning hinges on the concept of an agent receiving feedback through rewards or penalties

based on its actions. This feedback mechanism guides the agent's learning process, enabling it to iteratively optimize its decision-making strategy over time. Through a sequence of actions and observations, the agent learns to navigate complex environments and discern optimal policies that maximize cumulative rewards. Reinforcement learning finds application in diverse domains, including robotics, game playing, finance, and autonomous systems, embodying a potent paradigm for imbuing agents with the ability to learn and adapt through interaction [70]. An overview of the reinforcement learning approach is provided in Figure 3.3.

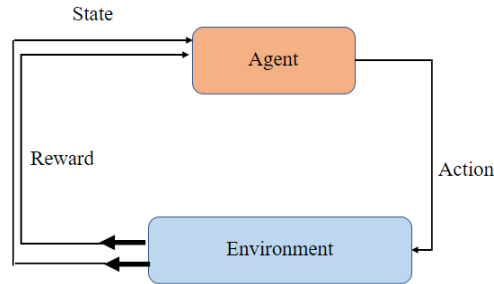


Figure 3.3 An Overview of Reinforcement Learning Working Flow.

Reinforcement learning operates through a series of well-defined phases, collectively facilitating the agent's acquisition of optimal strategies in interacting with an environment. The initial phase entails defining the problem's state space, action space, and the rules governing transitions between states upon taking actions. The agent then embarks on an iterative exploration-exploitation process, making decisions based on learned policies and exploring new actions to balance knowledge acquisition and reward maximization. Through these interactions, the agent receives feedback in the form of rewards or penalties, enabling it to fine-tune its decision-making over time using techniques such as temporal difference learning or Monte Carlo methods. The learning process converges as the agent continually updates its value function or policy, aiming to converge upon an optimal policy that yields the maximum cumulative reward over the long term. This sequence of phases encapsulates the essence of reinforcement learning, where intelligent agents progressively harness insights from interactions to evolve strategies that excel in achieving predefined objectives [70].

In the following sections, training data corresponding to intrusion attacks are explained. In addition, descriptions of different machine, deep, and reinforcement learning algorithms used in this work are provided. Also, different hyperparameter tuning and optimization techniques are briefly discussed.

### 3.2 Intrusion Attack Detection

In this section, descriptions of how the training dataset is built, data pre-processing techniques, and how features are extracted are given.

### 3.2.1 Dataset

In this dissertation, the dataset used is known as CICDDoS 2019 [71], created through a collaborative effort between the Canadian Institute of Cyber-security and the University of New Brunswick. This dataset encompasses over 1 million samples of authentic traffic and 30 million samples of malicious attacks. This dataset is one of the newest datasets associated with intrusion attacks [71]. The CICDDoS 2019 contains training data corresponding to 13 different intrusion attacks, as shown in Table 3.1.

Table 3.1 List of Attacks.

Attacks	Number of Samples
Total Normal	5693110
Domain Name System (DNS)	5071011
User Datagram Protocol (UDP)	3134645
Simple Network Management Protocol (SNMP)	5159870
Trivial File Transfer Protocol (TFTP)	20082580
Lightweight Directory Access Protocol (LDAP)	2179930 232
Network Basic Input/Output System (Netbios)	4092937
Microsoft SQL To Server (MSSQL)	5781928
Simple Service Discovery Protocol (SSDP)	2610611
Network Time Protocol (NTP)	1202649
Port Scanning (Portscan)	2312
Simple Service Discovery Protocol (SSDP)	2610611
User Datagram Protocol Link Aggregation (UDP-Lag)	366461
Synchronized Flooding (SYN Flooding)	4444750
Character Generator (Chargen)	232

The intrusion attacks in the given data are classified into the two categories, as presented in Figure 3.4, namely reflection-based and exploitation-based attacks. In reflection-based attacks, malicious user's identity remains hidden by using standard third-party devices. These types of attacks primarily operate within application layer protocols, employing transport layer protocols such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or a combination of both. Noteworthy instances of reflection-based attacks encompass MSSQL, SSDP, DNS, LDAP, NetBIOS, SNMP,

PortMap, CharGen, NTP, and TFTP. Exploitation-based attacks are similar to reflection-based attacks, but with slight differences. Exploitation-based attacks are only triggered by specific features of protocols and bug implementations. In this case, a malicious user initializes the attack by sending many messages to the victim. Therefore, when the victim replies to these messages, the malicious user refuses to respond. Ultimately, the victim is exhausted, and no connection can be continued. The samples of these attacks include SYN Flood, UDP, and UDP Lag attacks [70].

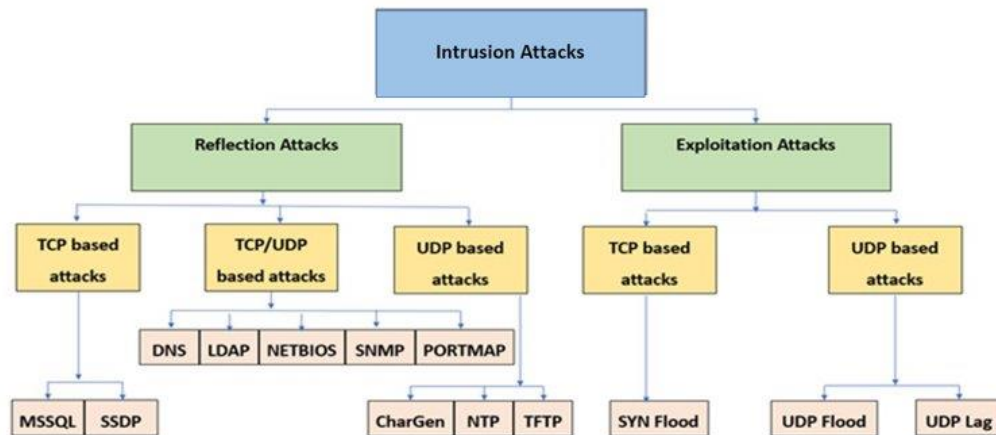


Figure 3.4 Attack Distribution in CICDDoS 2019.

The CICDDoS 2019 consists of 88 features extracted using CICFlowMeter. In this used dataset, the feature dimension was reduced to 39, which are considered as the most relevant features in detection process. For instance, several features, such as Flow ID, Source IP, Destination IP, and Timestamp were removed from the given dataset. In other words, Source IP and Destination IP do not affect the training of any models, while they may increase the overfitting issues for authentic and attack signals since they use the same IP address. In this corresponding dataset, the selected features are classified into the X classes, namely flow-based, packets' characteristics-based, forward direction-based, backward direction-based, and time-based. The list of these features along with their abbreviations are provided in Table 3.2.

Table 3.2 Short Descriptions of the Used Features in CICDDoS 2019.

Feature	Abbreviations
Forward arrival time total	Fwd IAT Total
Packets Length Variance	Packet Length Variance
Forward packets	Fwd Packets
Forward arrival time mean	Fwd IAT Mean
Forward arrival time standard deviations	Fwd IAT Std



Flow arrival time standard deviations	Flow IAT Std
Flow arrival time maximum	Flow IAT Max
Forward arrival time minimum	Fwd IAT Min
Minimum packet length	Min Packet Length
Packet length mean	Packet Length Mean
Backward packet length mean	Bwd Packet Length Mean
Backward packet length standard deviations	Bwd Packet Length Std
Flow arrival time mean	Flow IAT Mean
Flow arrival time minimum	Flow IAT Min
Backward arrival time mean	Bwd IAT Mean
Backward arrival time standard deviations	Bwd IAT Std
Total length of Forward packets	Total Length of Fwd Packets
Flow byte(s)	Flow Bytes
Backward packets	Bwd Packets
Maximum packets length	Max Packets Length
Total forward packets	Total Fwd Packets
Total backward packets	Total Bwd Packets
Forward packet length standard deviations	Fwd Packet Length Std
Backward packet length minimum	Bwd Packet Length Min
Backward arrival time total	Bwd IAT Total
Backward arrival time minimum	Bwd IAT Min
Flow packet(s)	Flow Packets
Flow Duration(s)	Flow Duration
Total Length of Fwd Packet(s)	Total Length of Fwd Packets
Total Length of Bwd Packet(s)	Total Length of Bwd Packets
Backward Packet(s)	Backward Packets
Backward arrival time maximum	Bwd IAT Max
Forward arrival time maximum	FWD IAT Max
Backward Packet length Maximum	BWD Packet length Max
Forward Packet length Maximum	FWD Packet length Max

Forward Packet length Mean	FWD Packet length Mean
Average Packets Size	Average Packet Size
Packet Length standard deviations	Packet Length Std
Flow arrival time Mean	Flow IAT Mean

### 3.2.2 Data Pre-Processing

Data pre-processing is a crucial phase in the data analysis pipeline that significantly impacts the quality and effectiveness of any analytical process. Its importance stems from the fact that raw data collected from various sources often contain inconsistencies, noise, missing values, and other irregularities that can negatively affect the performance and reliability of subsequent analyses and models [72]. Data pre-processing, as presented in Figure 3.5, involves a series of steps aimed at cleaning, transforming, and organizing the data into a suitable format for analysis or modeling.

#### 3.2.2.1 Data Cleaning

Data collected from real-world sources can be messy and contain errors or outliers. Data cleaning involves identifying and correcting errors, inconsistencies, and inaccuracies in the dataset. This step ensures that the data is reliable and accurate. Techniques used in data cleaning include removing duplicate entries, handling missing values through imputation, and identifying and addressing outliers [73-76].

#### 3.2.2.2 Data Transformation

Data transformation is the process of converting data into a more suitable format for analysis or modeling. This can involve scaling numerical features to a common range, transforming skewed distributions using techniques like logarithmic or Box-Cox transformations, and encoding categorical variables into a numerical format through techniques like one-hot encoding or label encoding [73-76]. For normalization of CICDDoS 2019, the standard scalar was used as a normalization technique that rescales features to unit variance.

#### 3.2.2.3 Handling Categorical Data

Many real-world datasets contain categorical variables, which require special treatment. Categorical variables need to be converted into numerical values for most analytical and machine learning algorithms [73-76]. One-hot encoding and label encoding are common techniques used to handle categorical data. For this purpose, all intrusion attacks were encoded as 1 and normal traffic as 0.

### 3.2.2.4 Class Rebalancing and Sample Size Reduction

In classification problems, datasets might have imbalanced class distributions, where one class is significantly more prevalent than others. This can lead to biased models that perform poorly on underrepresented classes [73-76]. Data pre-processing methods such as oversampling, under sampling, or generating synthetic samples using techniques can help balance the class distribution. To provide a balance class in CICDDoS 2019, a random equal number of different intrusion attacks from CICDDoS 2019 were selected. Since the least amount of normal traffic for one intrusion attack, as shown in Table 3.2 is 29,808 samples, hence 29,808 attacks samples were selected from the given dataset. As a result, the dataset consists of 59,616 samples were selected to train, test, and validate machine and deep learning models.

To conclude, data pre-processing is a critical phase that lays the foundation for accurate and effective data analysis and modeling. By ensuring data quality, transforming variables appropriately, handling categorical data, reducing dimensionality, and addressing class imbalance, data pre-processing helps create a clean and structured dataset that enhances the performance and interpretability of subsequent analyses and models [73].

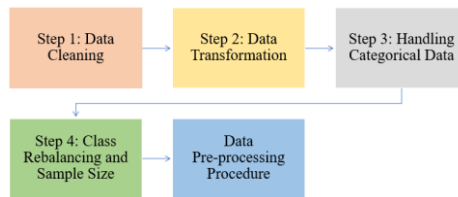


Figure 3.5 Overview of Data Pre-Processing Steps.

### 3.2.2.5 Feature Engineering

Feature engineering (known as feature selection) is a critical aspect in machine learning model development, involving the deliberate process of identifying and choosing a subset of relevant input variables or features from the original dataset. The primary goal of feature selection is to enhance model performance, reduce computational complexity, and mitigate the risk of overfitting by eliminating irrelevant or redundant features that may introduce noise or unnecessary complexity into the modeling process. The importance of feature selection lies in its capacity to improve the interpretability and generalization of machine learning models. By reducing the number of input features, the model becomes more focused on the most informative variables, thus simplifying the underlying relationships and making the model's decision-making process more transparent. Additionally, feature selection can lead to faster training times and improved model efficiency, which is particularly crucial when dealing with large datasets or computationally intensive algorithms [77].

Feature selection methods can be broadly categorized into filter, wrapper, and embedded approaches, as presented in Figure 3.6. Filter methods assess the relevance of features independently of the chosen machine learning algorithm. Wrapper methods involve training and evaluating the model using different subsets of features to select the most effective set. Embedded methods, on the other hand, integrate feature selection into the training process of the machine learning algorithm itself. The choice of method depends on the specific problem and dataset characteristics. Effective feature selection demands careful consideration of the trade-off between reducing dimensionality and maintaining the relevant information needed for accurate model predictions. In this dissertation, to identify important features, two techniques of Pearson’s Correlation Coefficient and Tree feature importance were applied to discard the correlated and low importance features from the corresponding dataset [78].

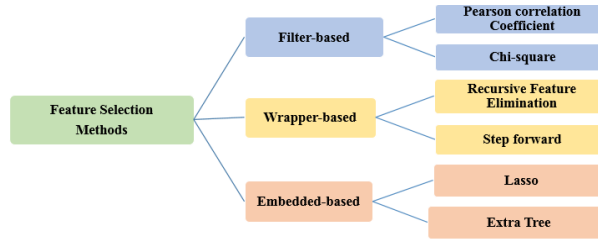


Figure 3.6 Examples of Different Categories of Feature Selection Techniques.

### 3.2.2.6 Pearson’s Correlation Coefficient

Pearson's correlation coefficient,  $\rho_{x,y}$ , is a statistical measure that quantifies the strength and direction of a linear relationship between two continuous variables. It assesses how well the data points of two variables fit a straight line, indicating the degree of association between them. Pearson's correlation coefficient ranges between -1 and +1, where -1 represents a perfect negative correlation, +1 represents a perfect positive correlation, and 0 indicates no linear correlation between the variables [77, 78]. This technique is shown as:

$$\rho_{x,y} = \frac{COV(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \quad (1)$$

where  $COV(X,Y)$  is the covariance of X and Y, and  $\sigma_X$  and  $\sigma_Y$  are the deviations of X and Y, respectively. In this dissertation, two variables were considered highly correlated when their correlation coefficient is higher than 0.9. Therefore, when the correlation between two variables is greater than 0.9, one of the features is removed from the given benchmark.

### 3.2.2.7 Tree Feature Importance

In this dissertation, a tree feature importance was used to present the importance of the used feature in the given dataset and discard the low importance features from the dataset. This technique involves using the Gini importance metric to

assess the significance of different features and subsequently eliminate those that are considered irrelevant for the task at hand. The Gini importance is a measure of how much a particular feature contributes to the overall reduction in impurity (Gini impurity) as decision nodes split in a tree-based model. Tree-based feature importance consists of several algorithms, namely decision trees, random forests, or gradient boosting machines, which are designed to automatically determine feature importance during the model-building process. The Gini importance provides a quantifiable way to rank the features based on their contribution to making accurate predictions based on random forest. This ranking reflects the extent to which each feature helps differentiate between different classes or outcomes within the dataset [77].

In this dissertation, the Gini importance is used to assess the relevance of each feature in relation to the target variable. Features with a Gini importance score below 0.01 are considered to have minimal impact on the model's predictive performance. As a result, these features are deemed less influential in distinguishing between different classes or outcomes and are thus considered candidates for removal. By discarding features with Gini importance scores below the specified threshold, the dataset is refined to include only those features that make a more meaningful contribution to the model's accuracy. The removal of these less impactful features serves to simplify the model, reduce noise, and potentially enhance its interpretability. Importantly, this feature selection process is carefully designed to not significantly impact the model's efficiency, ensuring that its predictive capabilities remain intact while achieving a more streamlined and focused representation of the data [79].

### **3.3 Machine and Deep Learning**

In this section, supervised and unsupervised machine learning and deep learning techniques are discussed in detail.

#### **3.3.1 Supervised Learning Models**

Supervised learning models are a fundamental category of machine learning algorithms that learn from labeled training data to make predictions or decisions about new, unseen data. In supervised learning, the algorithm learns from the relationship between input features and corresponding target outputs provided in the training dataset [80]. These models encompass two primary types of tasks, namely classification and regression. Classification involves predicting a categorical or discrete label. Algorithms like logistic regression, decision trees, support vector machines, and neural networks are commonly used for classification tasks. For example, a classification model can predict whether an email is spam or not spam. In contrast, regression focuses on predicting continuous numerical values. Linear regression, polynomial regression, decision trees, and neural networks are often used for regression tasks. An example could be predicting the

price of a house based on its features like area, number of bedrooms, and location. In this dissertation, only classification models were focused. The list of these models along with a short description are provided in the following:

### **3.3.1.1 Traditional Machine Models**

Traditional machine learning models are fundamental algorithms used to build predictive models by learning patterns and relationships from data. These models have been established in the field for a considerable time and are widely applied to various problems. They operate within the framework of supervised learning, where they learn from labeled training data to make predictions or classifications on new, unseen data [80-86]. Short descriptions of the traditional models that were used in this dissertation is outlined as follows:

#### **3.3.1.1.1 Support Vector Machine**

A Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm primarily used for classification and regression tasks. It works by finding an optimal hyperplane that best separates different classes or predicts continuous values. In the context of classification, an SVM aims to find the hyperplane that maximizes the margin between two classes. The margin is the distance between the hyperplane and the closest data points of each class [81].

SVMs can handle linear and nonlinear separable data by using different techniques, namely linear SVM and non-linear SVM. When data can be separated by a straight line or plane, a linear SVM finds the best-fitting hyperplane that separates the classes while maximizing the margin. For complex data that can't be separated linearly, SVMs use the kernel trick. This involves transforming the original feature space into a higher-dimensional space where the classes become separable by a hyperplane. SVMs also offer a unique property known as the "soft-margin" approach. This allows some misclassification to occur, accommodating noisy data or overlapping classes. The trade-off between maximizing the margin and allowing misclassification is controlled by the hyperparameter called the "C" parameter. SVMs have gained popularity due to their ability to handle high-dimensional data, few support vectors, and the kernel trick for nonlinear patterns. They are used in various domains like image classification, text categorization, bioinformatics, and finance. However, SVMs can be sensitive to the choice of kernel and require careful tuning of hyperparameters to achieve optimal results [82].

#### **3.3.1.1.2 Decision Tree**

A Decision Tree is a fundamental machine learning algorithm used for both classification and regression tasks. It operates by recursively partitioning the data into subsets based on the values of input features, ultimately making decisions or predictions. The process of building a decision tree involves several steps, namely root node, splitting, child nodes, and leaf nodes. In root node, the algorithm selects the feature that best separates the data based on a certain criterion (e.g., Gini impurity or information gain). This feature becomes the root node of the tree. The data is divided into subsets based on the values of the chosen feature at the root node. Each subset corresponds to a branch leading to a child node.

In child node, the algorithm repeats the process of selecting the best feature and splitting for each child node, creating further branches in the tree. This recursive process continues until a stopping criterion is met, such as reaching a maximum depth or a minimum number of instances in a node. The final nodes in the tree are called leaf nodes and represent the predicted class or value. In the case of classification, the majority class in a leaf node is assigned as the prediction. In regression, the mean or median of the target values in the leaf node is used as the prediction [83].

Decision Trees are known for their interpretability, as they allow you to visualize the decision-making process. However, they tend to overfit the data, capturing noise and outliers. To mitigate this, techniques like pruning and using ensembles like Random Forests are often employed. Despite their limitations, decision trees are valuable for understanding data relationships, creating simple models, and serving as building blocks for more complex algorithms [81].

### **3.3.1.1.3 Logistic Regression**

Logistic Regression is a fundamental statistical and machine learning algorithm used primarily for binary classification tasks. Despite its name, logistic regression is a classification algorithm, not a regression algorithm. It estimates the probability that a given input instance belongs to a particular class. The main idea behind logistic regression is to model the relationship between the input features and the probability of an instance belonging to a certain class. The output of the logistic regression model is a probability score between 0 and 1. To convert this probability score into a class prediction, a threshold (usually 0.5) is chosen. If the predicted probability is above the threshold, the instance is classified as one class; otherwise, it's classified as the other class [83].

Logistic regression employs the logistic function (also known as the sigmoid function) to model the relationship between the input features and the predicted probability. The logistic function transforms any input into a value between 0 and 1, which is interpreted as the probability of an instance belonging to the positive class. The algorithm learns the optimal coefficients (weights) for each feature through a process called optimization, often using techniques like gradient

descent. These coefficients are used to calculate the log-odds (logarithm of the odds ratio) of an instance belonging to the positive class.

Logistic regression is straightforward to implement, interpretable, and works well when the relationship between input features and the log-odds is roughly linear. It's commonly used in applications such as email spam detection, medical diagnosis, credit scoring, and many other binary classification tasks. Extensions like multinomial logistic regression allow for multiclass classification, making logistic regression a versatile and widely used algorithm in machine learning [80-83].

#### **3.3.1.1.4 K-Nearest Neighbor**

K-Nearest Neighbor (KNN) is a straightforward machine learning algorithm used for classification and regression tasks. In KNN, the prediction for a new data point is based on the majority class (in classification) or the average value (in regression) of its K closest neighbors in the training dataset. This algorithm operates under the assumption that similar data points tend to have similar outcomes. To make a prediction with KNN, the algorithm calculates the distances between the new data point and all data points in the training set.

The K data points with the shortest distances to the new point are considered its nearest neighbors. For classification, the most common class among these neighbors becomes the predicted class for the new point. For regression, the algorithm averages the target values of the K neighbors to predict the new point's value. However, choosing the appropriate value of K is crucial, as a small K might lead to noisy predictions, while a large K can blur class boundaries. KNN is a simple algorithm to grasp and implement, though it might struggle with high-dimensional data and requires careful consideration of distance metrics and K value for optimal results [84].

#### **3.3.1.1.5 Naïve Bayes**

Naïve Bayes is a probabilistic machine learning algorithm often used for classification tasks. It is based on Bayes' theorem, which calculates the probability of an event occurring given the probabilities of related events. Naïve Bayes assumes that the features describing an instance are independent of each other, even though this might not be the case in reality. Despite this simplification, Naïve Bayes can be surprisingly effective. In Naïve Bayes, for a new instance, the algorithm calculates the probability of each class based on the features associated with that instance. It then selects the class with the highest probability as the predicted class for instance. This is done by multiplying the probabilities of each feature belonging to a particular class. Despite its simplicity and the "naïve" assumption of feature independence, Naïve Bayes often works well for text classification, spam detection, and other applications where the feature independence



assumption is reasonable. However, its performance might suffer when dealing with highly correlated features or cases where feature independence does not hold [85].

### **3.3.1.2 Ensemble Machine Models**

Ensemble learning is a machine learning technique that aims to enhance the predictive performance and robustness of models by combining the outputs of multiple individual models. It leverages the concept that a group of models, when aggregated, can often make more accurate predictions than any individual model. In ensemble learning, various strategies are employed to create a diversified set of models. These strategies include training models on different subsets of the dataset (bagging), iteratively adjusting models to focus on previously misclassified instances (boosting), or using a meta-learner to combine the predictions of different models (stacking). By harnessing the strengths of different algorithms or settings, ensemble learning can effectively capture complex patterns, mitigate overfitting, and enhance generalization capabilities. Ensemble learning is widely used across various domains and has proven to be particularly beneficial in scenarios where individual models might struggle due to noisy data, limited predictive power, or the presence of outliers. The overall predictive accuracy and stability achieved through ensemble learning make it a valuable technique for improving the reliability and effectiveness of machine learning models [86]. In the following different types of ensemble models are described briefly.

#### **3.3.1.2.1 Bagging**

Bagging or Bootstrap Aggregating is a prominent ensemble learning technique designed to enhance the performance and robustness of machine learning models. In bagging, a diverse set of models is created by training multiple instances of the same base learning algorithm on different subsets of the training data. These subsets are generated through random sampling with replacement, allowing each subset to have instances that might appear more than once or not at all. Each model in the bagging ensemble is trained independently on its respective subset, leading to a collection of models that have been exposed to different portions of the data.

To make predictions, the outputs of these individual models are combined, often through a process of averaging (for regression tasks) or voting (for classification tasks). This combination of predictions helps to alleviate the impact of outliers and reduces variance, resulting in a more stable and accurate overall prediction. Bagging is particularly effective when dealing with high-variance algorithms, like decision trees, that tend to overfit the training data. By creating an ensemble of such models and then aggregating their predictions, bagging mitigates overfitting and improves the ensemble's ability

to generalize to new, unseen data. This technique has applications in various fields, providing a powerful tool to enhance the reliability and performance of machine learning models [80-86].

#### **3.3.1.2.1.1 Random Forest**

Random Forest is a widely used and powerful ensemble learning method that extends the concept of bagging. It's particularly effective for improving the accuracy, stability, and generalization of machine learning models. Random Forest builds upon decision trees, a common base model in machine learning. In a Random Forest, multiple decision trees are generated, each trained on a different subset of the training data using the bagging technique.

However, there's an additional layer of randomness introduced during the construction of each decision tree. During the process of splitting nodes in the tree, instead of considering all features, Random Forest randomly selects a subset of features to make the best split at each node. This introduces diversity among the trees, ensuring they capture different aspects of the data. When making predictions with a Random Forest, the outputs of all individual trees are combined. For regression tasks, the predictions are often averaged, and for classification tasks, the class with the most votes across all trees is selected. The aggregated prediction benefits from the wisdom of multiple trees, which collectively yield a more accurate and robust result [80-82].

Random Forests offer several advantages, including the ability to handle high-dimensional data, noisy data, and complex relationships. They are less prone to overfitting compared to a single decision tree, making them a popular choice across various domains, from finance to healthcare and beyond. Their ease of use, ability to capture intricate patterns, and capability to deal with both numerical and categorical features make Random Forests a valuable tool in the machine learning toolbox.

#### **3.3.1.2.2 Boosting**

Boosting is an ensemble learning technique aimed at improving the performance of machine learning models by sequentially refining their predictive abilities. Unlike other ensemble methods that create independent models, boosting focuses on iteratively improving a base model's weaknesses. Boosting begins by training a base model on the entire training dataset. The subsequent models, referred to as weak learners, are trained to emphasize the instances that the previous models struggled with. In each iteration, the weak learner is trained on a modified version of the training data, where the weights of the misclassified instances are increased. This iterative process allows the ensemble to increasingly focus on the more challenging examples. As the boosting process continues, the individual weak learners are combined to form a strong learner that effectively corrects errors made by its predecessors.

During the prediction phase, each weak learner contributes a weighted vote or prediction, and their collective output is combined to produce the final prediction. The combination of these weighted predictions results in a boosted model that significantly improves accuracy, particularly when dealing with complex data distributions or noisy datasets [80-86]. In this dissertation, several boosting algorithms were used, namely Gradient Boosting, Categorical Boosting, Light Gradient Boosting, and Adaptive Boosting, as described as below:

#### **3.3.1.2.2.1 Gradient Boosting**

Gradient Boosting is a boosting ensemble technique that is designed to create powerful predictive models by iteratively refining the predictions of a base model. Gradient Boosting is especially effective in improving the accuracy and predictive performance of models for various tasks like regression and classification. In Gradient Boosting, the process begins with an initial model, often a simple one like a decision tree with shallow depth, which is called a weak learner. This weak learner is trained on the training dataset to make predictions. However, unlike traditional boosting techniques that directly modify instance weights, Gradient Boosting focuses on the model's errors.

In each iteration, a new weak learner is trained to predict the residual errors (the differences between the actual outcomes and the predictions of the current ensemble). The weak learner is built to minimize these residual errors using gradient descent optimization. The ensemble then incorporates the predictions of this new model, adjusting the weights of the existing models' predictions to correct for the remaining errors.

By repeating this process over several iterations, the ensemble continually reduces the errors and refines its predictive capabilities. The final prediction is achieved by aggregating the predictions of all the weak learners, with each prediction being weighted according to its contribution to reducing the overall errors. Gradient Boosting is highly flexible and can work with a variety of base models, including decision trees and linear regression. Due to its iterative error-correction process and adaptive learning, Gradient Boosting excels at capturing complex relationships within data, effectively handling noisy data, and producing models with superior predictive accuracy. It is widely used in machine learning competitions and real-world applications across industries [80-86].

#### **3.3.1.2.2.2 Categorical Boosting**

Categorical Boosting is a specialized technique within the realm of machine learning that specifically caters to datasets containing categorical features. Categorical features are attributes that take on discrete values representing different categories or labels, as opposed to continuous numerical values. In traditional machine learning algorithms, dealing with categorical features can be challenging because they require numerical inputs. Categorical Boosting algorithms have been

developed to effectively handle these types of features. These algorithms implement strategies, such as target encoding, one-hot encoding, and ordinal encoding to transform categorical variables into numerical representations that can be used by gradient boosting algorithms.

This transformation process enables the algorithm to harness the predictive power of categorical features without losing their inherent meaning. By understanding and effectively utilizing categorical features, Categorical Boosting algorithms enhance the accuracy of predictions, particularly in scenarios where categorical attributes play a crucial role in making informed decisions. This specialized approach ensures that gradient boosting techniques can be applied to a broader spectrum of datasets, leading to improved performance and more accurate predictive models [83].

### **3.3.1.2.2.3 Light Gradient Boosting**

Light Gradient Boosting (LightGBM) is a high-performance and efficient gradient boosting framework that has gained significant popularity in the field of machine learning. It is designed to handle large datasets, complex relationships, and categorical features while delivering fast training speeds and competitive predictive accuracy. LightGBM improves upon traditional gradient boosting algorithms by introducing innovative techniques such as the Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). These techniques optimize the sampling of data instances and the grouping of features, resulting in faster training times and reduced memory usage [80-86].

One of LightGBM's key features is its ability to handle categorical features without the need for one-hot encoding. It implements a unique "Histogram-based Learning" approach, which discretizes continuous values into discrete bins, allowing it to directly work with categorical data. This not only saves memory but also preserves the interpretability of categorical features. Moreover, LightGBM employs a leaf-wise tree growth strategy, which constructs trees by splitting the leaf with the maximum gain in predictive accuracy. This approach often leads to shallower trees and better model performance. Additionally, LightGBM provides options for handling class imbalance and custom loss functions, allowing for fine-tuning to specific problem domains. The combination of these innovations makes LightGBM well-suited for a wide range of machine learning tasks, including classification, regression, and ranking. Its speed and efficiency make it a popular choice for both practitioners and researchers who aim to achieve accurate predictions with minimal computational resources [86].

### **3.3.1.2.2.4 Adaptive Boosting**

Adaptive Boosting, commonly known as AdaBoost, is a powerful ensemble learning technique that focuses on iteratively improving the performance of weak learners to create a strong predictive model. AdaBoost is particularly

effective in enhancing the accuracy of models for classification tasks, making it a widely used technique in machine learning. The core idea behind AdaBoost is to sequentially train a series of weak learners, which are models that perform slightly better than random guessing. In each iteration, the weak learner is trained on a modified version of the training data, where the emphasis is placed on the instances that were misclassified by previous models. This adaptive approach ensures that subsequent models pay more attention to the challenging instances, gradually improving the ensemble's overall accuracy.

During the final prediction phase, the outputs of all weak learners are combined to produce the ensemble's prediction. Each weak learner's contribution is weighted based on its accuracy, with more accurate models having a greater influence on the final decision. This weighting mechanism enables AdaBoost to give higher importance to the models that perform better on difficult examples. AdaBoost is particularly effective at handling complex data distributions, reducing both bias and variance. It works well with a variety of base classifiers, such as decision trees, and can adapt to different problem domains by using appropriate weight adjustments.

However, it is important to note that AdaBoost can be sensitive to noisy data and outliers, as these instances receive increased emphasis during training. In summary, AdaBoost is a dynamic ensemble technique that iteratively enhances the predictive power of weak learners by focusing on instances that are harder to classify. Its ability to adapt and learn from mistakes, combined with its improved overall accuracy, makes it a valuable tool in the machine learning toolkit [80-86].

### **3.3.1.2.3 Stacking**

Stacking, a sophisticated ensemble learning technique, involves combining the predictions of multiple diverse models through a two-level process to create a more robust and accurate predictive model. Stacking goes beyond basic ensemble methods by utilizing the strengths of individual models and leveraging their collective intelligence. In the first level, a variety of base models are trained on the same dataset. These models can be different algorithms or even the same algorithm with different hyperparameters. Each model learns unique patterns and captures distinct information from the data. Once trained, these base models make predictions on the same set of data. In the second level, a meta-learner, often referred to as the "stacker," is trained on the predictions made by the base models. The stacker learns how to optimally combine these predictions to produce the final ensemble prediction. This higher-level model can be a simple linear regression, a decision tree, or any other algorithm that learns from the base models' outputs [79-80].

Stacking aims to exploit the complementary strengths of different models and to improve overall predictive performance. By combining models with diverse perspectives, stacking can often achieve higher accuracy than any

individual model. It's worth noting that stacking requires careful tuning and validation to avoid overfitting, as both base models and the stacker need to generalize well to new data. Stacking is particularly useful when dealing with complex problems where no single model excels. It can also handle a wide range of data types, making it versatile in various domains such as image recognition, natural language processing, and financial forecasting. While more complex to implement than some other ensemble methods, stacking's potential for significant performance improvement justifies its popularity in advanced machine learning scenarios [85-86].

### **3.3.1.3 Deep Learning**

Deep supervised learning-based models are one of the main categories of deep learning models that use a labeled training dataset to be trained. These models measure the accuracy through a function, loss function and adjust the weights till the error has been minimized sufficiently. Among the supervised deep learning category, three important models are identified, namely Deep neural networks, convolutional neural networks, and recurrent neural network-based models [70-73]. In the following, these categories along with their used models in this dissertation are discussed.

#### **3.3.1.3.1 Artificial Neural Network**

Artificial Neural Network (ANN) is a computational model composed of interconnected units, or neurons, organized in layers. It consists of an input layer that receives data, one or more hidden layers that process information, and an output layer that produces results. Each neuron is associated with a weight that represents the strength of its connection to other neurons. During training, these weights are adjusted iteratively to minimize the difference between the network's predictions and the actual target outputs, typically utilizing a process called backpropagation. This involves calculating the gradient of the error and updating the weights in reverse order, allowing the network to learn relationships and patterns within data [85-91].

ANNs are versatile tools used in various machine learning tasks, from classification and regression to complex tasks like image recognition and natural language processing. Their architecture flexibility, along with different neuron activation functions and network topologies, allows them to model intricate relationships in data. The network's ability to generalize from training data to new, unseen data makes ANNs powerful tools in modern AI, enabling them to capture intricate patterns and perform tasks that are challenging to address using traditional programming approaches. One of the known ANN-based models is Deep Neural Network, which discussed in the following:

### 3.3.1.3.1.1 Deep Neural Network

Deep Neural Networks (DNNs) are one type of ANN-based models, which are characterized by multiple hidden layers between the input and output layers, as shown in Figure 3.8. These layers, often referred to as the "deep" layers, allow DNNs to model increasingly complex and abstract representations of data. Each layer consists of interconnected neurons, and the connections between neurons have associated weights that are adjusted through training. DNNs leverage a hierarchy of learned features, enabling them to automatically extract intricate patterns and features from input data. Training DNNs typically involves backpropagation, where errors are propagated backward through the layers to update the weights and fine-tune the network's performance.

DNNs have demonstrated remarkable success in various domains, including image and speech recognition, natural language processing, and more. Their ability to learn hierarchical representations of data has fueled breakthroughs in research. However, the increased depth and complexity of DNNs demand significant computational resources for training and can be susceptible to issues like vanishing gradients and overfitting. Nonetheless, these challenges have led to the development of techniques like batch normalization, skip connections, and various regularization methods, contributing to the ongoing advancements in deep learning and its applications [90-93].

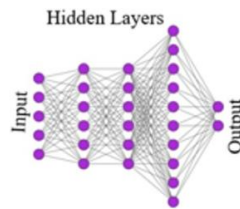


Figure 3.7 Architecture of DNN.

### 3.3.1.3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) has become increasingly popular in a number of fields, including security. A main advantage of such networks is the low number of parameters compared to other types of neural networks, such as ANNs. Another important characteristic of CNNs is that they can extract abstract information when their input data grows into deeper layers. The basic architecture of CNN is shown in Figure 3.8. This architecture typically consists of several convolutional layers, pooling layers, and fully connected layers. Convolutional operations are one of the main elements of CNN. The sparse connection and weight sharing strategy in CNN guarantee high computational efficiency and robustness. In a convolutional layer, given an input containing one or more sets of 1-D feature matrix, the output feature matrix of this

layer is created by convolution with multiple sets of convolution kernels [74-75]. This convolution operation can be expressed as follows:

$$x_j^L = \mathcal{F}(\sum_{i=1}^I w_{ij}^L * x_i^{L-1} + b_j^L) \quad (2)$$

Where  $x_j^L$  is the  $j$ th output feature map of convolution layer  $L$ ,  $*$  is the convolution operation used for the  $i$ th feature matrix  $x_i^{L-1}$  of the input layer  $L-1$  and the corresponding  $j$ th convolutional kernel  $w_{ij}^L$ ,  $b_j^L$  denotes an additional bias matrix,  $\mathcal{F}(\cdot)$  is a nonlinear activation function for each feature mapping value. The activation functions usually are hyperbolic Tangent, sigmoid, and Rectified linear unit (ReLU).

The operation of the pooling layer permits the training parameter to be moderately reduced and ensures the spatial invariance of the feature map, resulting in prevention of overfitting. Generally, the pooling layer can reduce the size of the convolved feature map with the aim of reducing the computational cost. This process can be performed by reducing the connections between layers while the pooling layer can independently perform on every feature map. The pooling operations have several types: MaxPooling, AveragePooling, and SumPooling. MaxPooling can take the largest element of the feature map, whereas AveragePooling can calculate the average of the elements in a feature map. SumPooling can calculate the total sum of the elements in the feature map. In this study, we use the MaxPooling operation as it provides better results than the other pooling types.

In CNN, the last several layers are usually connected by fully connected layers (FC) to classify features. In these layers, the output matrix is flattened into a vector. After weighted summation and nonlinear activation, the output  $x^{L+2}$  can be expressed as follows:

$$x^{L+2} = \sigma[(w^{L+2})^T x^{L+1} + b^{L+2}] \quad (3)$$

Where  $w^{L+2}$  is the weight matrix of the fully connected layer,  $b^{L+2}$  is the bias vector, and  $\sigma(\cdot)$  is a nonlinear activation function, such as Tanh or ReLu. Finally, CNN uses a SoftMax function to learn any kind of continuous and complex relationship between network variables. In this study, there are two FC layers, where the classification process takes place. CNN faces a critical challenge in preventing overfitting. The dropout technique is used here, where a few neurons are dropped from the neural network during the training process. A constant dropout rate of 0.7 is found adopted, which provides the optimum predictions [95].



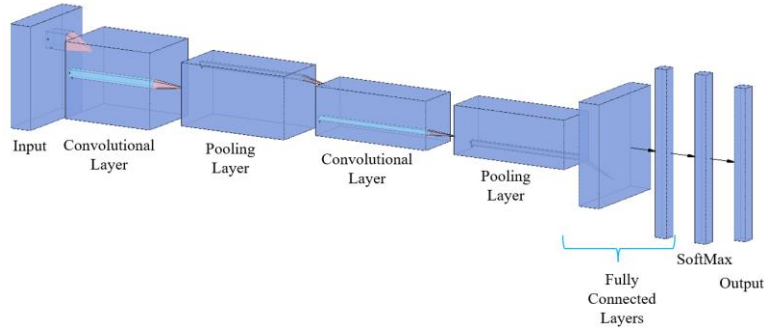


Figure 3.8 Architecture of CNN.

### 3.3.1.3.2.1 Alex Neural Network

AlexNet is a convolutional neural network (CNN) architecture that played a pivotal role in advancing the field of deep learning, particularly in computer vision tasks like image classification. The architecture of AlexNet consists of multiple convolutional layers that learn hierarchical features from raw pixel values, as shown in Figure 3.9. It includes Rectified Linear Unit (ReLU) activation functions to introduce non-linearity and employs max-pooling layers to down-sample the learned features while preserving important information. One of the key innovations of AlexNet was its use of dropout, a regularization technique that helps prevent overfitting by randomly "dropping out" a fraction of neurons during training. The network also introduced the concept of using GPUs (Graphics Processing Units) to accelerate deep learning computations, which significantly sped up training times [96-97]. Overall, AlexNet's success marked a turning point in the use of deep neural networks for image recognition tasks, paving the way for more advanced architectures and models in the years that followed.

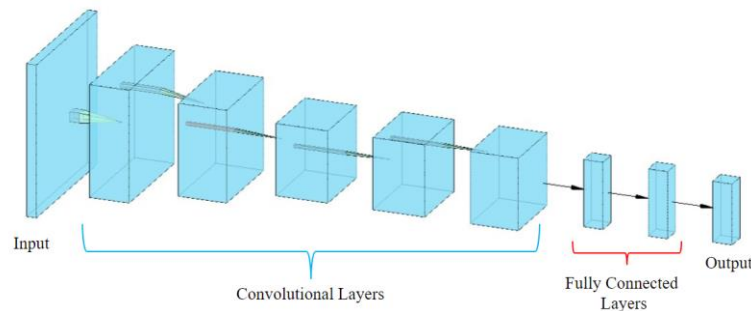


Figure 3.9 Architecture of AlexNet.

### 3.3.1.3.2.2 Densely Connected Neural Network

Densely Connected Neural Network (DenseNet) is a deep learning architecture that has gained prominence for its innovative approach to information flow and feature reuse within neural networks. Created to address the vanishing

gradient problem and enhance training efficiency, DenseNet introduces the concept of dense connections between layers, where each layer is directly connected to every subsequent layer in a feedforward manner. A basic architecture of this model is presented in Figure 3.10. The DenseNet model is a type of convolutional neural network, which uses a dense connection among layers via dense blocks. This model architecture is divided into several dense blocks, where the features' dimensions are constant within a block, however, the layers between dense blocks, known as transition layers, perform the down sampling, using batch normalization, convolutional layers, and loss functions [96-98].

DenseNet has four forms, DenseNet-121, 169, 201, and 264. They have the same characteristics, but different numbers of layers. For instance, in DenseNet with 121 layers, each layer is connected to the remaining layers in a feed-forward manner. This model consists of four dense blocks, three transition layers, and final 121 layers. These final layers consist of 117 convolutional layers, 3 transition layers, and 1 classification output layer. Each convolutional layer corresponds to a set sequence of operations, such as batch normalization and Rectified Linear Unit function. The classification output layer involves several operations, regularization L2-norm, global average pooling, and SoftMax. DenseNet-169 has 165 convolutional layers, 3 transition layers, and 1 classification output layer, while DenseNet-201 consists of 197 convolutional layers, 3 transition layers, and 1 classification output layer [96-98].

DenseNets exhibit impressive performance in various computer vision tasks, such as image classification and object detection, due to their ability to capture intricate patterns and exploit shared information effectively. Additionally, they address common issues like overfitting by encouraging feature diversity and enabling more efficient gradient flow. This innovation in architecture has contributed to the advancement of deep learning models and their applications across different domains [99].

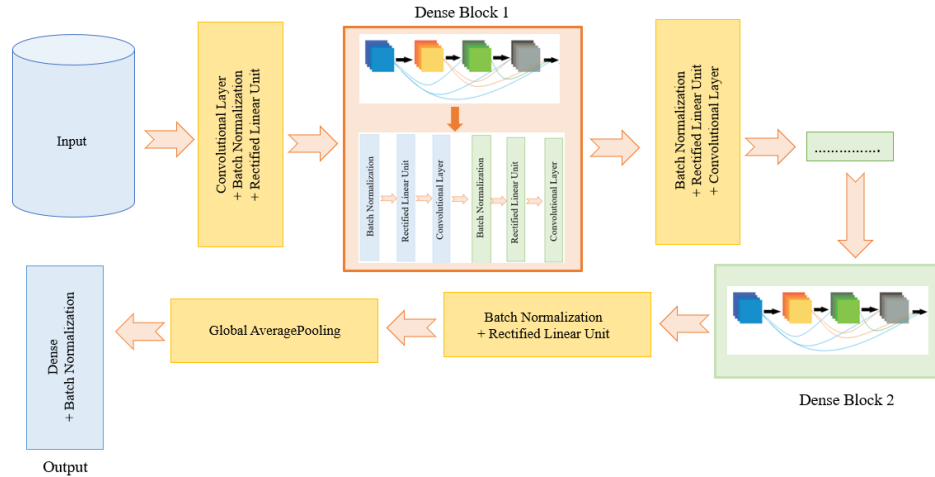


Figure 3.10 Architecture of DenseNet.

### 3.3.1.3.2.3 Residual Neural Network

Residual Neural Network (ResNet) is one of the most known within ResNet family that can accelerate the speed of training, increase the network's depth without requiring any parameters, and decrease the effect of vanishing gradient problems. The core idea of ResNet revolves around residual blocks, as illustrated in Figure 3.11, which enable the network to learn residual mappings instead of attempting to directly learn the desired underlying mapping. This is achieved by introducing "skip connections" or "identity shortcuts" that allow the network to learn and optimize the difference between the desired output and the current layer's output. These skip connections help prevent the vanishing gradient problem, which can hinder training in very deep networks. ResNet's architecture consists of multiple residual blocks stacked together, each containing convolutional layers and skip connections.

By utilizing these residual blocks, ResNet models can extend to hundreds of layers while maintaining the ability to efficiently train and converge. This architectural innovation has led to significant improvements in image classification, object detection, and other computer vision tasks. ResNet's impact on the field of deep learning is profound, as it introduced a practical solution for training exceptionally deep neural networks and paved the way for subsequent advancements in network architecture design. ResNet-50 is a member of the ResNet family with a 50-layer residual network consisting of 48 convolutional layers, 2 max-pooling layers, and  $3.8 \times 10^9$  floating points operations. As shown, the ResNet-50 architecture consists of the following elements:

- A convolution with a kernel size of  $7 \times 7$  and 64 different kernels of size 2 and a Max pooling with a size of 2, resulting in layer 1.

- In the convolution layer 2, there are a  $1 \times 1$  with a kernel size of 64,  $3 \times 3$  with a kernel size of 64 and  $1 \times 1$  with a kernel size of 64, resulting in 9 layers.
- In the convolution layer 3, there are a  $1 \times 1$  with a kernel size of 64,  $3 \times 3$  with a kernel size of 64 and  $1 \times 1$  with a kernel size of 64, resulting in 12 layers.
- In the next convolution layer 4, there are a  $1 \times 1$  with a kernel size of 64,  $3 \times 3$  with a kernel size of 64 and  $1 \times 1$  with a kernel size of 64, resulting in 18 layers.
- In the next convolution layer 5, there are a  $1 \times 1$  with a kernel size of 64,  $3 \times 3$  with a kernel size of 64 and  $1 \times 1$  with a kernel size of 64, resulting in 9 layers.
- In the last layer, there are an average pooling with a kernel size of  $7 \times 7$ , a fully connected layer with 1000 nodes, and a SoftMax function [65-77].

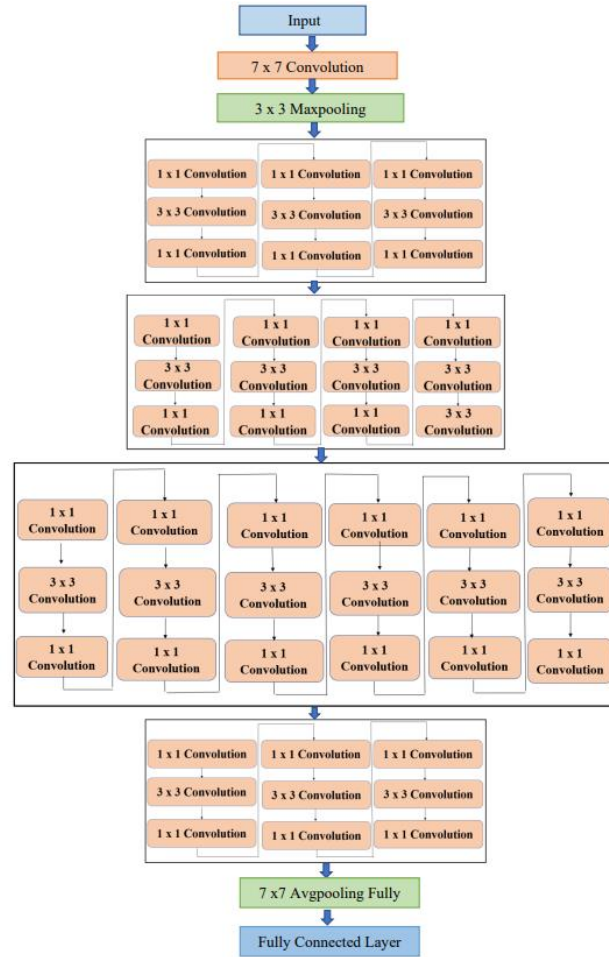


Figure 3.11 Architecture of ResNet.

### 3.3.1.3.2.4 Capsule Neural Network

The basic Capsule Neural Network (CapsNet) architecture is shown in Figure 3.12. This architecture is a novel type of neural network, which uses a vector-in and vector-out to transmit information. Unlike the traditional neural network that are embedded with scalar neurons, the typical information unit in capsule network is a vectorized capsule that consists of multiple scalars. Each capsule has different features with particular characteristics. These characteristics can have particular pre-defined parameters, such as position, size, direction, velocity, reflectivity, and texture. The module length of a capsule also provides special meaning, such as probability of the feature existence. CapsNet has three important layers, namely convolutional layer, primary capsule layer, and digit capsule layer. The convolutional layer is used to extract the primary features from the input.

The primary capsule layer presents the convolution process and the process of constructing the capsules. The convolution product is reformed into a capsule with a particular length to get the primary features in the vectorized form. The primary capsule layer transfers characteristic information to digit capsule layer through a dynamic attention routing approach. The number of capsules in this layer is the number of classification and the length of the capsules presents the probability of the classifications. In addition, two main functions, attention and Squashing functions, in CapsNet are briefly discussed [94].

An attention function is integrated into the model to make the classification model focus on the main critical and distinguishable information for the classification results. In other words, these functions are mainly used to find the information that has the most important and relevant information, helping the network to focus on specific parts of the data rather than the entire data. The attention function can also help distinguish the best features corresponding to the target variables. This function involves mapping a vector with key-value pairs to the output vector. It is worth mentioning that the attention function can be different in different types of capsule networks. For instance, in CapsNet, a traditional attention function is applied to the model. In such function, the capsules attempt to find an agreement in high dimensional space; hence, low level capsules vote for higher level capsules, which is computationally expensive [95].

The CapsNet architecture uses the length of the output vector to show the probability of a feature's existence. Therefore, a compression function is required to narrow the length of the capsules between 0 and 1 [80]. A function, squashing function, makes the long vector length close to 1 and short vector length to 0, while the direction of the vector is unchanged and calculated as following:

$$V_j = \text{squash}(s_j) = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \quad (4)$$

where  $V_j$  is the output vector of capsule  $j$ ,  $s_j$  is the  $j$ th capsule after dynamic routing, and  $\|\cdot\|$  means 2 norm. In addition, the vectorized feature transfer methods of CapsNet can extract the information, while it defines a substantial computational overhead. Additionally, it premises the CapsNet architecture to work effectively with sufficient features, which highly depends on the first few neural layers of the model. Thus, these challenges, computational overhead and performance dependency on first few layers, motivated us to develop other DL models based on capsule architecture to solve the limitations of CapsNet and CNN [95-100].

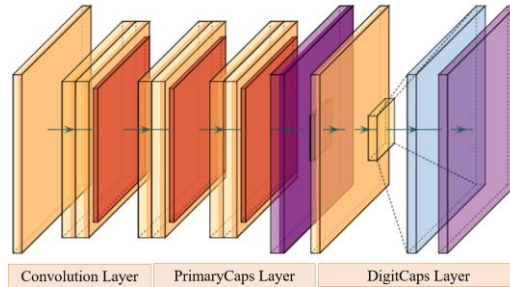


Figure 3.12 Architecture of CapsNet.

### 3.3.2 Unsupervised Learning Models

Unsupervised learning models are a category of machine learning algorithms that aim to uncover patterns and structure in data without the use of labeled target outputs. Unlike supervised learning, where the algorithm is trained on input-output pairs, unsupervised learning operates solely on input data. The primary goal of these models is to reveal underlying relationships, groupings, or representations within the data.

One common approach in unsupervised learning is Clustering, where the algorithm groups similar data points together into clusters, with the aim of maximizing intra-cluster similarity and minimizing inter-cluster similarity. This can help in discovering natural groupings within the data. Another key concept is Dimensionality Reduction, which involves reducing the number of features (dimensions) in the data while retaining its essential information.

This can help in visualizing high-dimensional data, speeding up processing, and mitigating the curse of dimensionality. In summary, unsupervised learning models play a crucial role in exploring and understanding data without the need for labeled examples. They are applied in diverse areas such as customer segmentation, image compression, anomaly detection, and more, contributing to a deeper comprehension of the inherent structure within complex datasets [100-104]. In this dissertation, several unsupervised traditional and neural network-based models were investigated, as follows:

#### 3.3.2.1 Traditional Models

Traditional unsupervised learning models are foundational techniques used to uncover patterns, structures, and relationships within data without relying on labeled outputs. These models have been widely used in various fields to extract meaningful insights from datasets. Two prominent types of traditional unsupervised models are clustering and dimensionality reduction. These traditional unsupervised models are foundational tools for understanding data's intrinsic properties and organizing it in meaningful ways. They've been applied in various domains, such as customer segmentation, document clustering, image compression, and more. However, with the advent of deep learning and advanced techniques,

traditional methods are often complemented or extended by newer approaches that can capture more complex patterns and relationships in data [100-104].

#### **3.3.2.1.1 K-means**

K-means clustering is an unsupervised machine learning algorithm used for grouping a collection of data points into distinct clusters. The K in K-means represents the number of clusters that the algorithm aims to identify within the dataset. The algorithm operates through iterative steps: it starts by randomly placing K cluster centroids in the data space. In each iteration, data points are assigned to the nearest centroid, forming clusters. The centroids are then recalculated based on the data points in each cluster. This process continues until the centroids stabilize or a predefined number of iterations is reached.

K-means seeks to minimize the sum of squared distances between data points and their assigned cluster centroids. The resulting clusters represent groups of similar data points, and the centroids act as representative points for these clusters. While K-means is efficient and widely used, it has limitations like sensitivity to initial centroid placement and assumptions about cluster shapes. Despite these constraints, K-means remains a fundamental tool for tasks like customer segmentation, image compression, and data organization, offering a straightforward way to uncover patterns and structure within data without the need for labeled examples [100-102].

#### **3.3.2.1.2 Principal Component Analysis**

Principal Component Analysis (PCA) is an unsupervised model that is used in data analysis and machine learning to simplify and transform high-dimensional datasets into a lower-dimensional space while retaining as much meaningful information as possible. It achieves this by identifying new axes, called principal components, that capture the maximum variance in the data. These components are orthogonal to each other, meaning they are uncorrelated, allowing for efficient representation.

PCA operates through a process of eigenvalue decomposition on the covariance matrix of the data. The eigenvalues and associated eigenvectors determine the directions of highest variance in the data. By selecting a subset of the principal components that explain the most variance, you can effectively reduce the dimensionality of the dataset. This is particularly useful for visualization, noise reduction, and improving computational efficiency in subsequent analyses. However, it's important to note that while PCA provides simplification, the interpretability of the transformed features



may decrease, and careful consideration is needed to ensure that vital information isn't lost during dimensionality reduction [102-104].

### **3.3.2.2 Neural Network Models**

Unsupervised neural network models are a subset of ANN models that are designed for learning from unlabeled data without explicit target outputs. These models are structured to identify and capture underlying patterns, relationships, and representations in the data, aiding in data exploration and feature extraction. One significant category within unsupervised neural networks is Auto-Encoders. Autoencoders consist of an encoder and a decoder network. The encoder maps the input data into a lower-dimensional latent space, compressing the information, while the decoder reconstructs the original data from this representation.

By training the network to minimize the reconstruction error, the encoder learns to extract essential features that characterize the data, effectively performing unsupervised dimensionality reduction and feature learning. Unsupervised neural network models have diverse applications, from data augmentation and anomaly detection to data visualization and transfer learning. They play a crucial role in extracting meaningful information from unlabeled data and continue to drive advancements in understanding complex datasets and improving downstream machine learning tasks. In this dissertation, one common type of Auto-Encoders, namely Variational Auto-Encoder were used [94-106]. A description of this model is provided as following:

#### **3.3.2.2.1 Variational Auto-Encoder Models**

Variational Autoencoders (VAEs) are a class of generative unsupervised models and a type of neural network architecture designed for unsupervised learning tasks, particularly for learning the latent representations of complex data distributions. VAEs combine the principles of autoencoders and probabilistic modeling to capture the underlying structure of data and generate new samples that resemble the original data distribution. In a VAE, the model consists of two main parts: an encoder and a decoder, as shown in Figure 3.13. The encoder takes input data and maps it to a distribution in the latent space, which is typically a multivariate Gaussian distribution. The key innovation of VAEs lies in introducing a probabilistic interpretation to the encoding process. This means that rather than producing a single point in the latent space, the encoder generates a probability distribution, capturing the uncertainty associated with the latent representation [102-104].

During training, the VAE seeks to minimize the reconstruction loss, which encourages the decoder to generate outputs that closely resemble the input data. Simultaneously, VAEs also minimize the Kullback-Leibler divergence between the

learned latent distribution and a chosen prior distribution, often a standard Gaussian. This regularization term encourages the latent space to be well-structured and continuous, facilitating smooth interpolation and controlled generation of new data samples. The result is a model that learns a continuous and probabilistic mapping from the data space to the latent space, enabling both data compression (via the encoder) and generation (via the decoder). VAEs have found applications in various domains, including image generation, data augmentation, and disentangled representation learning. Their probabilistic nature allows for principled uncertainty estimation, making VAEs a powerful tool in modern machine learning research and applications [104].

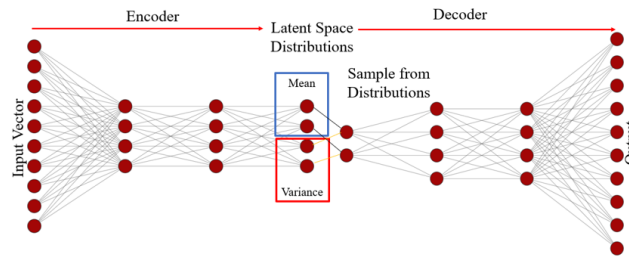


Figure 3.13 Architecture of Variational Auto-Encoder.

### 3.4 Reinforcement Learning

Suppose the system is modeled as Markov Decision Process (MDP), so the agent can communicate with the environment based on the discrete time steps. This communication can be useful in designing decision-making-related problems [104-108]. MDP is presented as a 5-tuple  $M = (S, A, T, R, \delta)$ , where  $S$  is a set of possible states,  $A$  is the set of possible actions that the agent can perform on the environment,  $T$  is the transition function from a state to another state,  $R$  shows the reward function, and  $\delta$  is the discount factor with possible values of 0 and 1. A policy  $\varphi$  can be defined as the conditional probability distribution of choosing different actions based on the  $S$ . Distribution of the reward sequence can be detected when the stationary policy has been selected.

So, the policy is evaluated by the action-value function. The action-value function is specified under the policy as the sum of the discounted reward from  $S$  and follows the related policies. Thus, the optimal policy can maximize the expected cumulative reward according to all the states. The given optimal action can be calculated using the following Equation:

$$Q^*(s, a) = \max_{\varphi} Q^{\varphi}(s, a) \quad (5)$$

More accurately, the agent can communicate with the environment to explore and investigate different transitions, and the achieved reward function is performed based on the action taken. Through communication in the environment at  $t$ , the agent can monitor the data about the state  $S$ , while it selects the action based on the policy. After that, it receives a reward  $r$  from the environment according to the action and transfers to the next state. The agent improves itself by the gained experiences based on cyclic agent-environment communication.

Such a learning process is based on the transition probabilities and reward functions to learn the MDP model and then find an optimal approximating policy using planning in the MDP (known as model-based technique). Additionally, learning the optimal value functions directly without learning the model and deriving the optimal policy is the second result of the learning process, which is called a model-free technique. Q-learning is one of the widely used model-free techniques which modifies the Q-values estimation according to the sample's experiences on every time step. In the next section, we discuss in detail this technique [104]. In the following, we discuss deep Q-learning and the proposed technique in detail.

### 3.4.1 Deep Q-Learning

Deep Q-Learning is a prominent technique in the field of reinforcement learning, designed to enable agents to learn optimal policies for sequential decision-making tasks. It combines the Q-learning algorithm with deep neural networks, enabling it to handle high-dimensional state spaces. A general architecture of DQN is presented in Figure 3.14. The fundamental idea behind deep Q-learning is to approximate the Q-function, which represents the expected cumulative future rewards of taking a specific action in a particular state. Q-learning is a classic reinforcement learning algorithm that aims to learn the optimal action-value function, The Q-function provides an estimate of the expected cumulative reward when taking action "a" in state "s" and following an optimal policy thereafter [108-110].

In this context, Deep Q-Network(DQN) extends Q-learning by using deep neural networks to approximate the Q-function. A neural network is employed to approximate  $Q(s, a)$  for all possible actions in a given state. The network takes the state as input and produces Q-values for each action as output. The Bellman equation is a key principle in reinforcement learning, defining the relationship between the current Q-value, the immediate reward, and the maximum Q-value of the next state. The Q-learning update rule involves minimizing the difference between the current Q-value estimate and the updated estimate based on the Bellman equation. In addition, DQN employs experience replay to improve stability and learning efficiency. It stores agent experiences (state, action, reward, next state) in a replay buffer and samples batches of

experiences randomly during training. This reduces the correlation between consecutive experiences, leading to more stable learning [110-116].

A target network is introduced to stabilize learning. It is a copy of the Q-network with delayed updates. During training, the target network's parameters are updated less frequently, reducing the risk of divergence caused by rapidly changing Q-value estimates. Moreover, the loss function used in DQN is typically the mean squared error (MSE) between the predicted Q-values and the target Q-values, which are computed using the Bellman equation and the target network. Stochastic gradient descent or similar optimization methods are used to update the Q-network's weights. Balancing exploration (trying new actions) and exploitation (choosing the best-known actions) is crucial. Epsilon-greedy exploration, where the agent chooses the optimal action most of the time and a random action with a small probability, is a common strategy.

DQN has been highly influential but faces challenges like overestimation of Q-values and instability during training. Various enhancements have been proposed, including Double Q-learning to mitigate overestimation and Dueling DQN to separately estimate state value and action advantages. In the last few years, DQN and its variants have been successfully applied to various domains, including playing video games, robotic control, recommendation systems, and more. As a result, as a fusion of Q-learning and deep neural networks, has demonstrated its capacity to tackle complex decision-making tasks by approximating optimal action-value functions. Its innovations, such as experience replay and target networks, contribute to more stable and effective learning in high-dimensional state spaces.

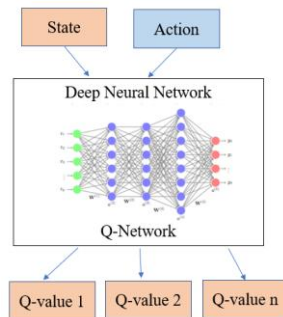


Figure 3.14 General Architecture of DQN.

### 3.4.2 Capsule Q-Network

Figure 3.15 shows the architecture of a Capsule Q-network, where the outputs are the Q-values, and the actions are calculated for the current state. As one can observe, the features of the given data denote the state of the Capsule Q-network. In this context, there are 21 features in the given dataset; the first 20 features are used as states, whereas the last feature

(the label of the data) can be used for computing the award of the model. In addition, the agent is applied based on the network structure; hence there should be at least one agent for the network. The agent can communicate with the environment and uses rewards according to the rewards based on the current state and the chosen action.

The agent can train the model and estimate future reward values. In the training process of the proposed model, the agent needs to explore the action space using a policy, namely epsilon-greedy exploration. This method helps the agent to choose a random action with a probability of  $e$  or action based on the value function with the probability of  $1-e$ . Additionally, the states can describe the inputs by the environment to the agent for acting. In this architecture, the dataset features are used as state parameters for the model; hence, 20 features are used as input for training and classifying the model. An action also is the decision selected by the agent after processing the environment during the time. The agent creates a list of actions based on the input of the neural network [102-104].

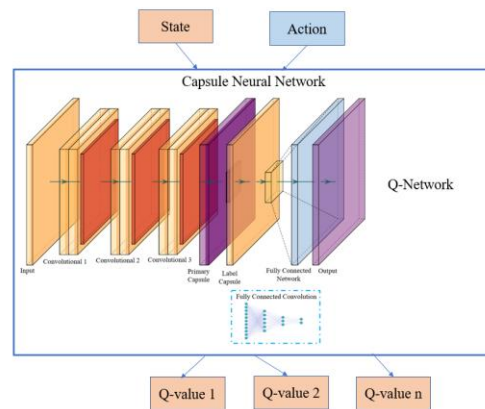


Figure 3.15 Architecture of Capsule Q-Network.

Although the general DQN uses the DNN model as a function approximator, there are several shortcomings of DNN in the architecture of DQN, including being computationally slow and working with infinite and discrete environments. To address this issue, we used a Capsule Neural Network (CapsNet), instead of DNN as a function approximator. The CapsNet agent can leverage a CapsNet as the function approximator to calculate Q-values based on the states and actions. The final Q-values are mostly used to present if an attack happened successfully [103].

This process feeds the state vector to the current model while the agent compares the output of the current model based on the Q-values and the Q-threshold values for classifying the attack classes. The feedback from the environment for an action by an agent is known as a reward [86-88]. The output values of the model is a reward. The agent can consider a

positive reward when the results of the classification model are compatible with the actual results based on the labels of the given data. In contrast, a negative reward happens when the classification model is not compatible with the real results based on the labels of the corresponding data. Furthermore, the reward value is highly dependent on the probability of the prediction by the classifier. This value can be adjusted according to the Q-values to improve the classifier's performance [102-115]. The main steps of the proposed technique are summarized in Figure 3.16.

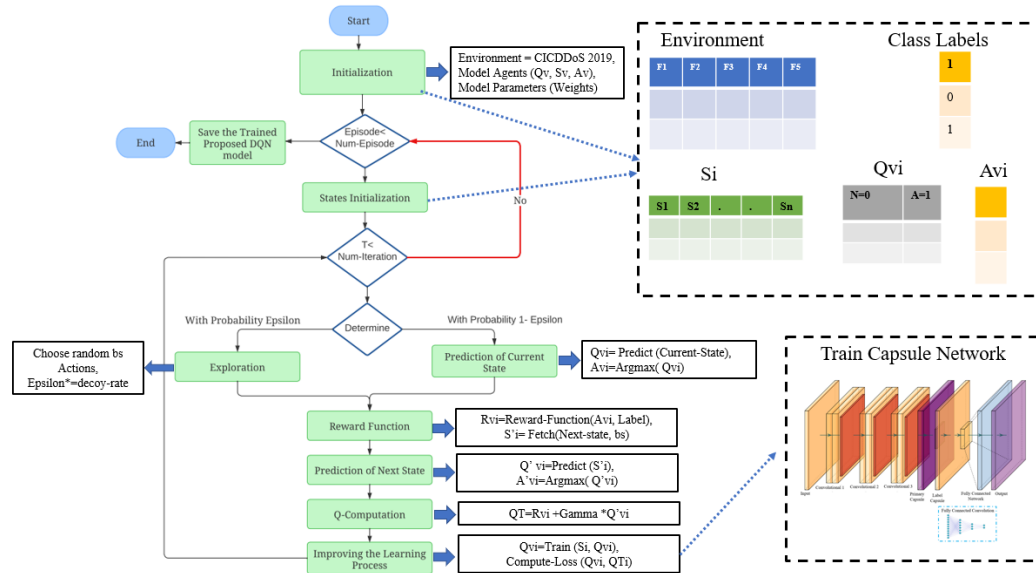


Figure 3.16 Necessary Steps of the CapsNet Q-learning Model.

### 3.5 Online Learning

Online learning is a dynamic approach in machine learning where a model is continuously updated as new data streams in. Instead of waiting to gather a complete dataset, the model learns from individual data points or small subsets as they arrive. This process allows the model to adapt quickly to changing trends and patterns in the data. Online learning is particularly useful for scenarios involving real-time data streams, like social media posts or sensor readings. While this approach offers agility and responsiveness, it can also be sensitive to noisy or biased data due to its rapid updates. Online learning's key advantage lies in its ability to handle streaming data in real-time. It's ideal for applications where data arrives continuously, such as monitoring stock prices or analyzing user activity on a website. The model's adaptive nature means it can quickly respond to shifts in the data distribution. Online learning is memory-efficient as it doesn't require storing the entire dataset, and it allows for incremental updates, saving time and resources [108-110].

In contrast, batch learning involves training a machine learning model on a fixed batch of data all at once. The model is fed the entire dataset, and its parameters are adjusted based on the collective information from the batch. This traditional approach is well-suited for situations where the data can be collected and processed before training. Batch learning tends to yield stable and accurate models, especially when the dataset is preprocessed effectively. However, it can be computationally intensive, requiring substantial resources to process large datasets. Batch learning shines in scenarios where the entire dataset can be collected and processed as a whole. This approach tends to produce more reliable models due to its comprehensive view of the data. It's commonly used in situations with structured data, where analysis can be performed offline, like credit scoring or medical research. While it requires more computational power and may not adapt as rapidly as online learning, it ensures a more controlled training environment [108].

In general, the choice between online and batch learning depends on the nature of the data and the problem at hand. Online learning is favored when dealing with continuous data streams, enabling adaptability and prompt responses to changes. Batch learning is suitable for scenarios with well-defined data batches, where a thorough analysis and model training can be conducted offline. In some cases, a hybrid approach might be employed, combining both methods to balance real-time adaptability and stable model training. Ultimately, the decision should align with the specific requirements and constraints of the project [109].

### 3.5.1 Sequential Learning

In the batch learning setting, a whole dataset consists of two sections, namely training and testing sets. We consider the length of training and testing sets as  $A$  and  $B$  respectively. A model  $P$  is applied to learn on training set  $L_{train} = \{(X_i, Y_i) | i \in \{1 \dots A\}\}$ , where  $X_i \in \mathbb{R}^n$ ,  $n$  is  $n$ -dimensional feature vector, and  $Y_i$  is a target variable. In the testing sets, the model  $P$  predicts a label for every  $X_i \in L_{train}$ . In addition, the models' parameters can be adjusted based on particular loss function to improve the model  $P$  performance.

A data stream is defined as an infinite order sequence of samples  $(X_i, Y_i)$ , shown as  $D_{stream} = \{D_1, D_2, D_3, \dots\}$ . In online setting data is read only one time due to limited computing and storage issues. For this purpose, to train the model  $P$ , we need to incrementally train the model to adapt the new data samples without forgetting the current information. Therefore, in incoming data sequence, the true label  $Y_i$  is discovered, and the loss function is defined. Each sample is employed to train and test the model. Thus, model  $P$  can test unknown samples. Before moving forward with the next incoming data streams, an online incremental learning algorithm creates a new model  $P$  according to the previous model

and existing samples. In sequential learning, data has to be evaluated over time in a data stream, whereas data distribution may change dynamically during time [110].

### **3.5.2 Online Sequential Capsule Network**

The framework of the proposed online technique is shown in Figure 1. The proposed framework consists of several phases, namely data pre-processing, model classification and prediction, and evaluation metrics. It is worth to mention that the CPU and GPU that we used in this study are provided in Figure 1. In data pre-processing phase, we used raw data to train, test, and validate the proposed model. For this purpose, we need initially identify the missing values, and normalize the given data. Finally, we have to generate a sequence of data streams to make the data ready for online sequential training. In general, in offline learning, the training of the models are performed at regular intervals and the results at periodic intervals are accumulated. In the contrary, in online learning, the training process can be performed in an incremental manner, continuously feeding data as it arrives to the model in groups. In model classification, we selected CapsNet as core model and fed the data in sequential and online manner. To address such challenges, we needed to make some modifications on CapsNet structure and solve the network issues regarding to catastrophic forgetting. Thus, we proposed a routing algorithm, namely Sequential Euclidean Distance Routing Algorithm, and updated the model architecture based on an algorithm, learning without forgetting, and regularization technique to train, test, and validate the given sequential data over particular time stamp. Consequently, the model are evaluated in terms of accuracy, probability of detection, probability of misdetection, probability of false alarm, processing time, training time per sample, testing time, and memory size. We also provided a comparison of the proposed model, Online CapsNet using Sequential Euclidean Distance Routing Algorithm with a simple online CapsNet using dynamic routing algorithm.



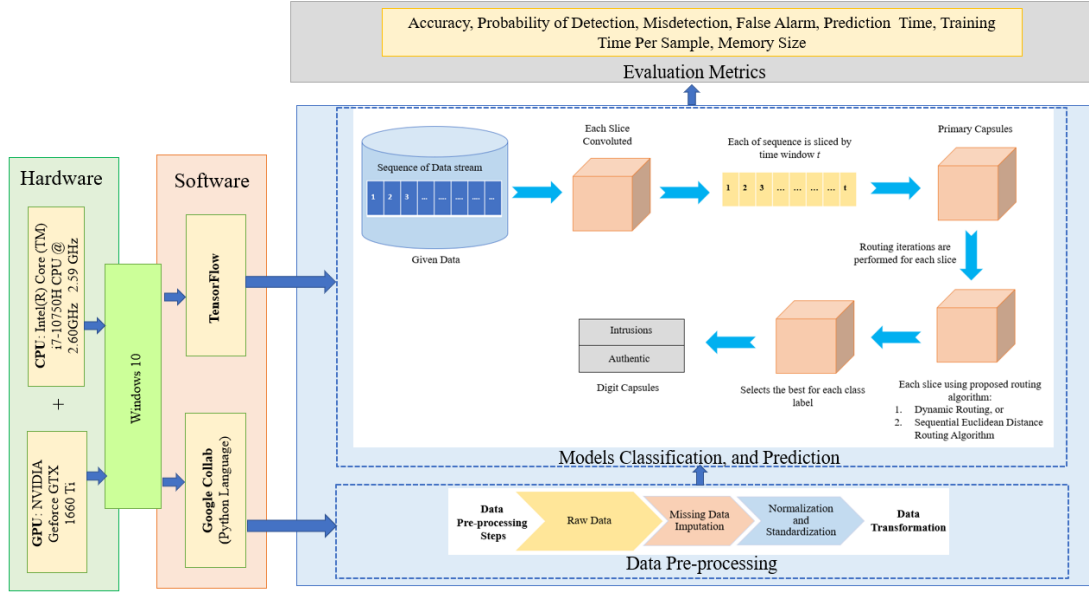


Figure 3.17 Overview of the Proposed Online Detection Architecture.

The CapsNet models have currently indicated a great impact in the field of DL. For instance, each object type in computational graphics has several instantiation parameters, which can be converted into different shapes using rendering functions. The CapsNet performs as opposite of this operation; hence, if an input of a function is a signal (or an image), the output generates the abstract representations. In this context, CapsNet can keep the specific feature parameters, including rational and translational relationships of an object. A simple architecture of CapsNet is illustrated in Figure 3.18. In CapsNet models, the groups in the lowest, in the between, and the highest are known as primary, convolutional, and class capsule groups. As presented, a CapsNet for classification problems is defined with a parameter  $\alpha$  which is mapped as  $\partial_{\alpha}: (\mathbb{R}^{X_w}) \mapsto \mathbb{R}^V$ , where an input with a size of  $X_w$  to a  $V$ -dimensional probability vector can be mapped.

To define, the routing mechanism, the input is converted to a capsule group,  $C = \{A, U\}$ . In such process,  $X_w$  is converted to the width  $P_w$  of the lowest capsule group. Then,  $C$  has a set of activation groups, where  $A \in \mathbb{R}^{P_w}$  and a set of instantiation parameter vectors,  $U \in \mathbb{R}^{P_w * P_D}$ , where  $P_D$  is the dimension of parameter vectors in the lowest level of capsules. In addition,  $A$  can be calculated from  $U$  or any other neural layers. It is noticeable that the routing process can be done between higher level capsules and the prediction vectors, known as  $\widehat{U}_{jt}$ . Each of these prediction vectors represents a path from the  $i$ th lower-level capsule to  $j$ th higher level capsule. Through the training process, for any particular entity, every  $U$  and  $A$  elements are trained to show an existence probability and characteristics of the classes. In general, CapsNet models can show the invariance of the existence probabilities along with the their equivariance of an entity type.

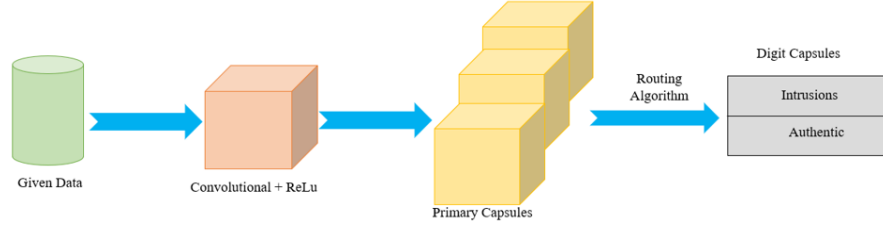


Figure 3.18 Overview of Batch-based CapsNet.

In the original CapsNet model, Dynamic Routing Algorithm is mainly used as a routing approach for working in a non-parametric expectation and maximization manner according to the available similarities between capsules. For this purpose, the activation scalar  $\alpha_j$  is computed as follows:

$$\alpha_j = \text{length} (O_j) = \sqrt{\sum_{d=1}^D O_{j_d}^2} \quad (6)$$

Where  $U$  and  $O$  outline the lower and higher-level instantiation vectors, while  $i$  and  $j$  indicate the index of the vectors in lower and the higher levels. For normalization of  $\alpha_j$  to a valid probability, a nonlinear function is used. The nonlinear function refers to a squash function, which is defined as following:

$$O_j = \text{squash} (s_j) = \frac{\|s_j\|^2}{1+\|s_j\|^2} \frac{s_j}{1+\|s_j\|^2} \quad (7)$$

Where  $s_j$  defines as an unnormalized instantiation parameter vector for the  $j$ th capsule in the higher level. The squash function also can make the activations recognizable using values to around zero or one. Also,  $\widehat{U}_{j|l}$  is measured as  $\widehat{U}_{j|l} = U_i * W_{ij}$ . Such vector can represent a path from  $i$ th capsule in lower level to  $j$ th capsule in higher level. In this way, a transformation matrix,  $W_{ij}$ , is the only needed parameter for a routing approach. Algorithm 1 demonstrates the main technique of dynamic routing algorithm with iteration number  $A$  and level index  $I$ . For CapsNet with  $L$  layers,  $I$  is ranged between 0 to  $L$  where  $I$  of primary capsules is equal to 0. We also can mention that the primary capsules are computed via several convolutional layers. Routing coefficients,  $r$ , are initiated to zero, whereas the  $r$  is updated to increase the chance of arrangements between  $\widehat{U}_{j|l}$  and  $O_j$ .

**Pseudocode Algorithm 1: Dynamic Routing Algorithm**

1. **procedure ROUTING** ( $\hat{w}_{ji}$ ,  $r$ ,  $L$ )
2. for Capsules  $i$  in layer  $L$  and Capsules  $j$  in layer  $(L+1)$  in  $t$ -th window:  
 $B_{ij} \beta 0$
3. for  $r$  iterations, do:
4. for all capsules  $i$  in layer  $L$ :  $C_i \beta \text{SoftMax}(B_{ij})$ ,  $K_i \beta \text{SoftMax}(B_i)$ ,  
SoftMax calculates  $K_i$
5. for all capsules  $j$  in layer  $(L+1)$ :  $S_j \beta \sum_i K_i \cdot \hat{w}_{ji}$
6. for all capsules  $j$  in layer  $(L+1)$ :  $\hat{S}_j \beta \text{Squash}(S_j)$ , Squash calculates  $S_j$
7. for all capsules  $i$  in layer  $(L)$  and for all capsules  $j$  in layer  $(L+1)$ :  
 $B_{ij} \beta B_{ij} + \hat{w}_{ji} \cdot \hat{S}_j$
8. end for
9. return  $\hat{S}_j$
10. end **procedure**

In this dissertation, to address the online learning issues, we use a sequential routing framework for a sequence of data, and modify the routing algorithm of CapsNet architecture. As we discussed, the original CapsNet uses Dynamic Routing Algorithm as routing mechanism; however, in this study, we propose a Sequential Euclidean Distance Routing Algorithm as routing mechanism. In general, to train our proposed model, the input sequences are initially transformed to three dimensional sequences via convolutional and linear projection layers, whereas the encoded sequences are divided into the time windows  $t$  and several routing iterations are conducted for each slice to classify the given label. Dividing each capsule group may lead to use existing routing algorithm for fixed data. Additionally, the proposed model can train a limited context in encoding process of every frame. Thus, the online capacities can be applied to the training process; however, the loss function of Connectionist Temporal Classification (CTC) is employed. Figure 4 provides an overview of Sequential Euclidean Distance Routing CapsNet and discusses more details in the following.

As can be seen in Figure 3.19, sequential routing mainly works on real valued feature sequence,  $x$ , with a length  $\hat{T}$  and a feature with dimensions  $\hat{F}$  to a  $V$  dimensional probability vector sequence, namely  $\hat{y}$  with a length  $T$ . An input feature

sequence  $x$  is converted into the primary capsule  $C_0$  via capsulation layer. In capsulation layer,  $A$  has a width  $T$ , and height  $P_H$ , while  $U$  has a width  $T$ , height  $P_H$ , and depth  $P_D$ . In this layer, the Maxout is used as activation function in convolutional layers, moving to primary capsule. The detailed process of capsulation layer is provided in Figure 3.20. Then, primary capsule has a width  $T$ , height  $P_H$ , and depth  $P_D$  respectively via capsulation layer. Therefore,  $C_0$  has to be fed into the lowest layer of capsule, encoding a convolutional capsule group  $C_1$  with the width, height, and depth of  $T$ ,  $M_H$ , and depth  $M_D$  respectively. In addition, all the convolutional capsule group are known as  $C_{1..L-1}$  have the similar shapes. Also, a class capsule group  $C_L$  has a width, height, and depth of  $T$ ,  $V$ ,  $V_D$  respectively. In this context, an activation vector sequence is a sequence of probability vectors which every element shows a probability of a corresponding label symbol. As a final step, a loss function namely CTC is used to calculate the loss between  $\hat{y}$  and a label sequence.

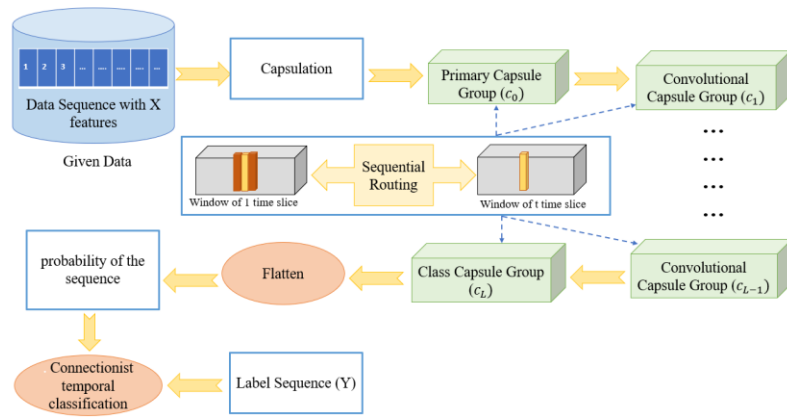


Figure 3.19 Overview of CapsNet using Sequential Euclidean Distance Routing Algorithm.

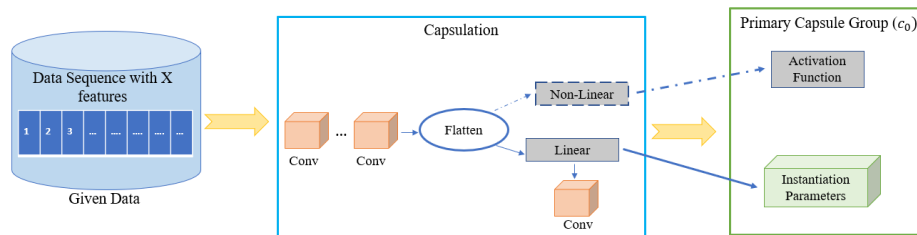


Figure 3.20 Schematic Overview of Capsulation Block.

As previously mentioned, in this study, the used routing algorithm is known as Sequential Euclidean Distance Routing Algorithm, which uses Euclidean Distance as initial metric. In such an algorithm, the training steps can make the embedded sample vectors close to their own category labels and far to other category labels. Suppose

$D = \{(X_1, Y_1), \dots, (X_N, Y_N)\}$  is a group with  $N$  labeled samples and  $X_i \in \mathbb{R}^D$ , and  $Y_i \in \{1, \dots, k\}$ , where  $d_k$  is considered as samples labeled with  $k$ . Thus, this network maps the needed set to the  $M$  dimensional space via an embedding function. Moreover, such algorithms can compute a vector  $C_k$  for every category.  $C_k$  can be measured as following:

$$C_k = \frac{1}{|d_k|} \sum_{(X_i, Y_i) \in d_k} f_{\theta}(X_i) \quad (8)$$

After that, this algorithm uses a SoftMax function to normalize the function of distance from its output  $f_{\theta}$  to  $C_k$  using following formulation.

$$P_{\theta}(Y=k|X) = \frac{\exp(-d(f_{\theta}(X), C_k))}{\sum_{k'} \exp(-d(f_{\theta}(X), C_{k'}))} \quad (9)$$

Such network is updated by decreasing the negative probability of  $\log$  based on the label  $k$ . In learning process, the classes subsets can be chosen randomly from the training set, while samples subset within every class is selected as the support set and the remaining can be chosen as a subset of query points. Algorithm 2 discusses the required process of Sequential Euclidean Distance Routing Algorithm.

**Pseudocode Algorithm 2: Proposed Sequential Euclidean Distance Routing Algorithm**

1. **procedure ROUTING** ( $P^{t-1}, \hat{w}_{ji}^t, r, L$ )
2. for Capsules  $i$  in layer  $L$  and Capsules  $j$  in layer  $(L+1)$  in  $t$ -th window:  $B_{ij} \beta 0$
3. for Capsules  $j$  in layer  $(L+1)$ :  $P_j^t \leftarrow P_j^{t-1}$
4. **for**  $r$  iterations **do**:
5. for Capsules  $i$  in layer  $L$  and Capsules  $j$  in layer  $(L+1)$  in  $t$ -th window:  $B_{ij} \leftarrow B_{ij} + \hat{w}_{ji}^t \cdot P_j^{t-1}$
6. for Capsules  $i$  in layer  $L$  in  $t$ -th window:  $K_i \beta \text{SoftMax}(B_i)$ , SoftMax calculates  $K_i$
7. for Capsules  $j$  in layer  $(L+1)$  in  $t$ -th window:  $S_j \beta \sum_i K_{ij} \cdot \hat{w}_{ji}^t$
8. for Capsules  $j$  in layer  $(L+1)$  in  $t$ -th window:  $\hat{S}_j \beta \text{Squash}(S_j)$ , Squash calculates  $S_j$
9. for Capsules  $j$  in layer  $(L+1)$ :  $B_{ij} \beta B_{ij} - \|w_{ji}\| \cdot S_j^2$
10. end for
11. return  $\hat{S}_j$
12. **end procedure**

As a consequence, online models can learn and memorize the given data, which is defined as continual learning. In continual learning, the previously obtained data can be neglected by the overlapping of the subsequent training tasks. Such a process in continual learning tasks is known as catastrophic forgetting. In the last few years, several techniques have been used to address the catastrophic forgetting issues in neural networks; however, there are several bottlenecks in dealing such issue.

For this purpose, we used two approaches, namely regularization and episodic memory. In regularization, the changes of influential parameters have to be changed in the learning process of new tasks. There are various techniques to evaluate the importance of parameters, such as Fisher information or the impacts of changes in loss function over a period of time. Despite these solutions providing a practical solution for parameters and tasks, they do not guarantee the models' generations [108]. In regularization approach, after using Fisher information, two important parameters of dropout and weight decay were regularized; however, they did not solve the generalization regarding the catastrophic forgetting. Thus, we used an episodic memory technique, namely learning without forgetting.

In general, in episodic memory, a small amount of the given data is stored from previously obtained tasks and combined with the given data from the existing tasks [109-112]. In particular, when a new data is fed to the model, the model can forget the previous tasks as it was trained for; resulting in catastrophic forgetting. To solve this issue, previous tasks can be used as constraints for model optimizations during the learning process. Then, in learning without forgetting, new tasks are used to train the network, while the original characteristics of network are stored in each iteration. Additionally, this technique attempts to remember the previous tasks when the new tasks are used. Ideally, the new tasks could be learned while sharing parameters from previous tasks, minimizing the chance of catastrophic forgetting.

To perform learning without forgetting, a CapsNet is given with a particular task parameter,  $\theta_o$ , and a shared parameter,  $\theta_s$ , respectively. The main aim is to add particular task parameters,  $\theta_n$ , for a new task using labels from the new tasks without using any data from the existing tasks. Algorithm 3 describes the structure of learning without forgetting. Initially, every new task is stored as  $y_0$  from the original network for the old tasks,  $\theta_s$  and  $\theta_o$ , outputs. Then, a set of label probabilities for each training sample is considered as a response. The nodes of every class are added to the output layer as a fully connected layer to the lower layer with a random weight  $\theta_n$ . The new parameter numbers are the node numbers time to the new classes number in the previous shared layer. After that, the network has to be trained to reduce the loss functions of all the tasks along with the regularized parameter  $R$  using Stochastic Gradient Descent (SGD). We first freeze the  $\theta_s$  and  $\theta_o$  respectively, then train jointly all the weights  $\theta_s$ ,  $\theta_o$ , and  $\theta_n$  until it reaches convergence.

**Pseudocode Algorithm 3: Learning Without Forgetting**

**Start with:**

1. Shared parameters
2. Task parameters for every out of dated tasks

3. Training data and ground truth on the new tasks

**Initialize:**

4.  $y_0 \beta$  CapsNet ( $X_n, T_S, T_0$ ) , Compute the output of the old task from new data

5.  $T_n \beta$  RANDINT( $|T_n|$ ), Randomly initialize new parameters Train:

6. Define  $\widehat{y}_0 : \text{CapsNet} (X_n, T_S, T_0)$  , old task output

7. Define  $\widehat{y}_n : \text{CapsNet} (X_n, T_S, T_n)$  , new task output

$$T_S^*, T_0^*, T_n^* \beta \underbrace{\text{argmin}}_{T_S, T_0, T_n} (\lambda_0 \tau_{old} (y_0, \widehat{y}_0) + \lambda_0 \tau_{new} (y_n, \widehat{y}_n) + R (T_S, T_0, T_n))$$

8. **end**

## Chapter 4

### RESULTS AND DISCUSSIONS

This chapter provides the results of the detection models discussed in the previous chapter to detect intrusion attacks that target smart grid networks. For these attacks, different machine and deep learning algorithms were analyzed and compared extensively. Also, a reinforcement learning model and an online learning-based algorithm proposed.

#### 4.1 Performance Analysis Metrics

To evaluate and compare the performance of the proposed models, 2 types of metrics of metrics are used. These metrics are related to the detection, and computational cost. The metrics related to the detection are accuracy (*ACC*), probability of detection (*PD*), probability of misdetection (*PMD*), probability of false alarm (*PFA*). They are defined as follows:

$$ACC = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} * 100 \quad (10)$$

$$PD = \frac{T_P}{T_P + F_N} * 100 \quad (11)$$

$$PMD = \frac{F_N}{T_P + F_N} * 100 \quad (12)$$

$$PFA = \frac{F_P}{T_P + F_N} * 100 \quad (13)$$

Where *TP* denotes as true positive, *TN* presents as true negative, *FP* is the false positive, and *FN* is false negative. The metrics related to computational cost are:

- *Processing Time*: It refers to the duration or time taken to train, test, and predict the results. In fact, it specifically represents the time required to predict or analyze malicious signals compared to non-malicious signals.
- *Training Time per Sample*: It refers to the duration or time taken to train a model on a single sample of the dataset. It represents the amount of time required to process and learn from the features and labels of an individual sample during the training phase.
- *Memory Usage*: It is the amount of memory or storage space that is required by a model during its entire process. It represents the amount of memory resources utilized to store and process the model's parameters, intermediate computations, and other relevant data.



## 4.2 Intrusion Attack Detection Analysis

In this section, the performance of machine learning algorithms in detecting intrusion attacks targeting smart grid were investigated and analyzed in depth, as follows:

### 4.2.1 Results of Detecting Intrusion Attacks on Smart Grid

In this section, the results of detecting intrusion attacks are provided. A number of machine learning algorithms are first compared. Then, a detailed analysis of the most efficient algorithms, namely supervised and unsupervised in detecting these attacks are provided, along with investigation of several deep learning models. Also, a reinforcement learning model and online learning-based technique are discussed. In the following, the results of different machine learning category are provided.

#### 4.2.1.1 Analysis of Optimized Parameters

In this study, a 5- fold cross-validation method is used to train 80% of the data and test 20% of the remaining given dataset. Table 4.1 provides the best hyperparameters of each model based on the tuning technique used to guarantee the optimal results of the selected models. Selecting the best parameters using tuning technique can prevent overfitting or high variance of data.

Table 4.1 List of Parameters in Models.

Model	Tuning Technique	Best Parameter Setting
Random Forest	Grid Search	Criterion = 'gini', n_estimators = 5.
Naïve Bayes	Grid Search	var_smoothing = 0.001.
K Nearest Neighbor	Grid Search	n_neighbors = 5, weights: 'uniform', leaf_size = 20
Classification and Regression Decision Tree	Grid Search	Criterion = 'gini', max depth = 36, splitter = 'best', max_features = 'log2'
Logistic Regression	Grid Search	Max_iter = 100, penalty = 'l2'.
Bagging	Grid Search	final estimator verbose= 1.
Boosting	Grid Search	max depth= 10, min impurity decrease = 10.
Stacking	Grid Search	n estimators = 42.
Support Vector Machine	Tree-structured Parzen Estimator	C = 4, penalty = 'l2'.

Adaptive Boosting	Tree-structured Parzen Estimator	'n_estimators': 200.0, Algorithm: 'SAMME'
Gradient Boosting	Tree-structured Parzen Estimator	'n_estimators': 187.0, loss: 'deviance', learning_rate: 0.65
Categorical Boosting	Tree-structured Parzen Estimator	n_estimators': 100, Learning_rate: 0.29
Light Gradient Boosting	Grid Search	Boosting_type = 'gbdt', max_depth = 10, learning_rate = 0.1, n_estimators = 100
Alex Neural Network	ADAM	Epoch = 100, momentum = 0.9, Batch size = 128, learning_rate = 0.01.
Densely Connected Neural Network-121	ADAM	Learning rate= 0.01, Feature maps= 0.5, growth rate =12
Densely Connected Neural Network-169	ADAM	Learning rate= 0.01, Feature maps= 0.5, growth rate =12
Densely Connected Neural Network-201	ADAM	Learning rate= 0.01, Feature maps= 0.5, growth rate =24
Densely Connected Neural Network-264	ADAM	Learning rate= 0.01, Feature maps= 0.5, growth rate =40
Residual Neural Network-50	Stochastic gradient descent	Momentum = 0.9, Learning Rate = 0.0001
Principle Component Analysis	Grid Search	max-depth = 10, Max-features = 'sqrt', splitter = 'best', Criterion = 'entropy'
K-means	Grid Search	n-clusters = 2, algorithm = 'auto', random-state = 0.
Variational-Auto Encoder	ADAM	Loss = 'mse', Activation = 'Relu', Epoch = 100
Capsule Q-Network	Manual	Num_episode=100, Num_iteration = 100, Hidden layers = 2, Initial Weight Values = Normal, Epsilon = 0.95, Decoy Rate = 0.99, Gamma = 0.01 and 0.9, Batch Size = 500.

#### 4.2.1.2 Results of Feature Selection in Detecting Intrusion Attacks

The results of Pearson's correlation coefficient are illustrated in Figure 4.1. As one can observe, several features (in green) are highly correlated with a coefficient  $> 0.9$ ; thus, these features are removed from our dataset. As a result, ten features are considered highly correlated. Thus, 27 features (Fwd IAT Total, Packet Length Variance, Fwd Packets, Fwd IAT Mean, Fwd IAT Std, Flow IAT Std, Flow IAT Max, Fwd IAT Min, Min Packet Length, Packet Length Mean, Bwd Packet Length Mean, Bwd Packet Length Std, Flow IAT Mean, Flow IAT Min, Bwd IAT Mean, Bwd IAT Std, Total Length of Fwd Packets, Flow Bytes, Flow Packets, Bwd Packets, Max Packets Length, Total Fwd Packets, Total Bwd Packets, Fwd Packet Length Std, Bwd Packet Length Min, Bwd IAT Total, and Bwd IAT Min) remain as a selected feature.

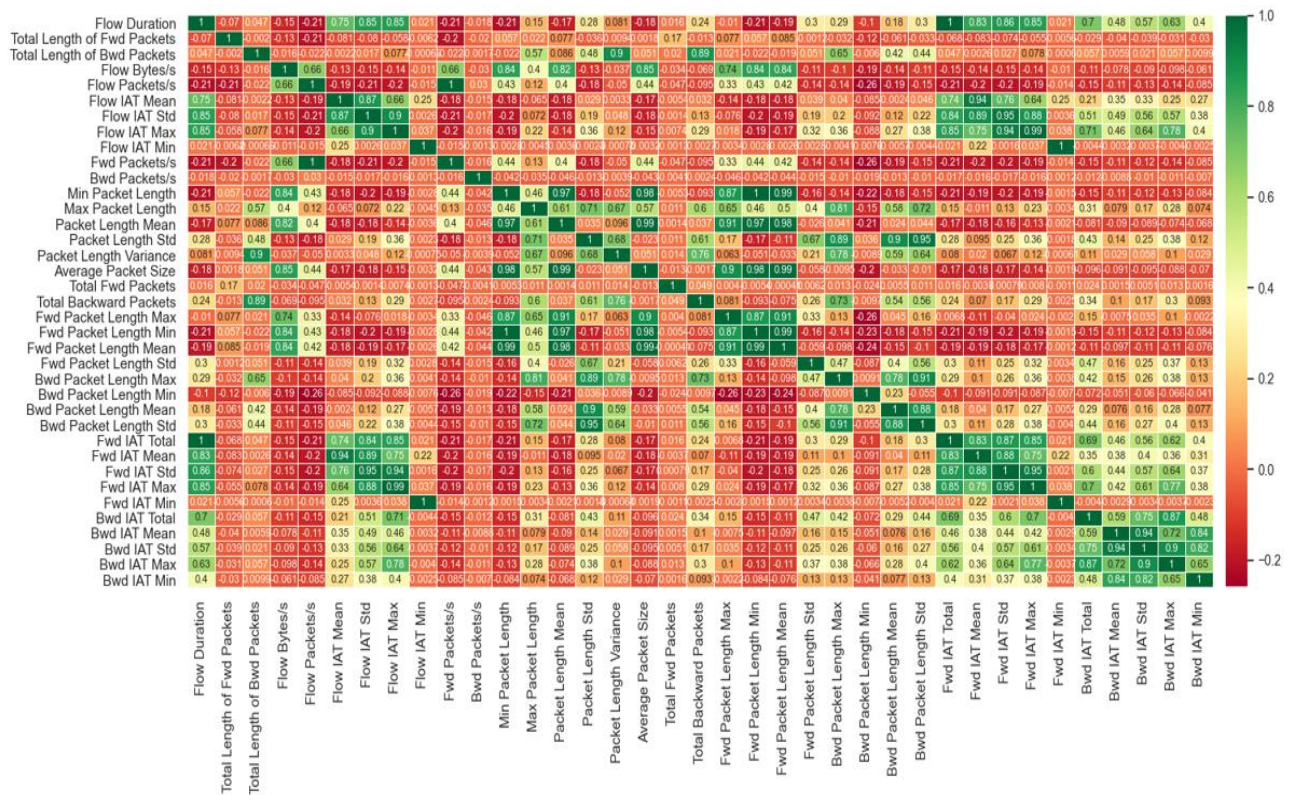


Figure 4.1 Pearson's Correlation Coefficient Heatmap.

Figure 4.2 provides the feature importance score for intrusion attacks, according to the Extra Tree classifier. As shown in this figure, Min Packet Length is the most important feature based on this technique; however, other features (Fwd Packets, Flow Packets, Packet Length Mean, Flow Max Packet Length, Total Length of Fwd Packets, Total Fwd Packets, Fwd IAT Total, Flow IAT Std, Fwd Packet Length Std, Flow IAT Max, Flow IAT Mean, Fwd IAT Std, Bwd Packet Length Min, Packet Length Variance, Fwd IAT Mean, Total Backward Packets, Bwd Packet Length Mean, Bwd IAT Total, Bwd IAT Mean, Bwd IAT Std, Bwd IAT Max, and Bwd IAT Min) remain as a selected feature.

and Bwd Packets) are also considered important features. In this technique, features that have a feature importance score less than 0.01 are removed. These features (Flow IAT Min, Bwd Packet Length Std, Bwd IAT Std, Bwd IAT Mean, Bwd IAT Min, and Fwd IAT Min) are discarded from the final dataset. Therefore, 21 features can be considered as significant features for detection process.

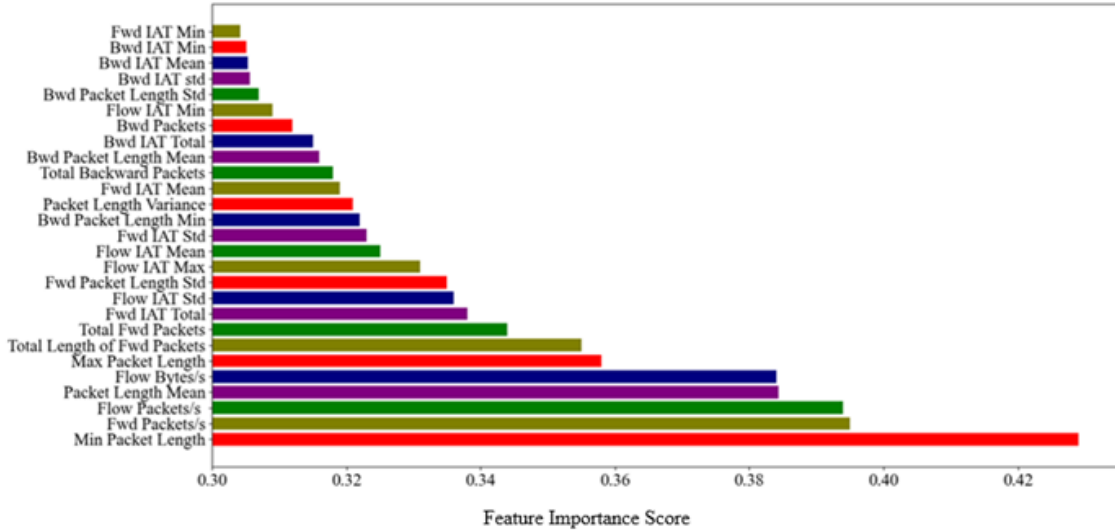


Figure 4.2 Importance of the features based on the Extra Tree classifier.

#### 4.2.1.3 Results of Ensemble Models in Detecting Intrusion Attacks

In this section, the results of detecting intrusion attacks using the different ensemble models are provided.

##### 4.2.1.3.1 Results of Ensemble Learning Models in Detecting Reflection Intrusion Attacks

In this dissertation, a comparative analysis of traditional machine learning and ensemble learning models using accuracy, detection rate, misdetection rate, and false alarm rate is provided. As one can observe, the results of reflection-based attacks are highlighted in Table 4.2. According to this table, the stacking-based classifier provides the best performance results in terms of accuracy, detection rate, misdetection rate, and false alarm rate, followed by the bagging, boosting, KNN, Random Forest, and Naïve Bayes. To be precise, Stacking-based, using Flow Bytes as the most important feature, achieves the best results for reflection-based attacks. As shown in Table 4.2, the accuracy, detection rate, misdetection rate, and false alarm rate of the stacking-based classifier are 96%, 4.1%, 8.9%, and 93.4%, respectively; however, it is apparent that bagging-based classifier achieves better results in comparison with the boosting-based ensemble technique.

Table 4.2 Evaluation Results for Reflection-based Attacks.

<b>Classifier</b>	<b>Detection Rate (%)</b>	<b>Misclassification Rate (%)</b>	<b>False Alarm Rate (%)</b>	<b>Accuracy (%)</b>
<b>Stacking</b>	<b>96</b>	<b>4.1</b>	<b>8.9</b>	<b>93.4</b>
Bagging	94.8	5.2	9.5	93
Boosting	94.04	5.9	9.3	92.2
Random Forest	92.2	6.7	8.2	90
Naïve Bayes	90	7	27.1	83.4
KNN	93.4	6.5	9	91.2

For example, the detection rate, misclassification rate, false alarm rate, and accuracy in the bagging-based model are 94.8%, 5.2%, 9.5%, and 93%, which are considered better results compared to those of the boosting technique. Moreover, the boosting technique has the worse results compared to other ensemble techniques. Moreover, it is apparent from the table that traditional machine learning methods, random forest and KNN, provide better results compared to the Naive Bayes technique for the reflection-based attacks. For instance, the detection rate, misclassification rate, false alarm rate, and accuracy of KNN are 93.1%, 6.5%, 9%, and 91.9%, respectively, which are relatively good results. In addition, Table 4.2 shows that Naïve Bayes provides the worst detection rate, misclassification rate, false alarm rate, and accuracy. As seen in this table, this classifier provides a detection rate of 90%, a misclassification rate of 7%, a false alarm rate of 27.1%, and an accuracy of 83.4%. Consequently, a comparison of traditional machine learning techniques with ensemble techniques for reflection attacks shows that ensemble techniques overall perform better in terms of the four-evaluation metrics.

#### **4.2.1.3.2 Results of Ensemble Learning Models in Detecting Exploitation Intrusion Attacks**

In this section, a comparative performance analysis of models, convolutional machine learning and ensemble, in detecting intrusion attacks in smart grids is provided. The convolutional models are support vector machine, naïve Bayes, and K nearest neighbor, and the boosting algorithms are AdaBoost, Gradient Boosting, and CatBoost. The hyperparameter tuning technique we apply is Tree-structured Parzen Estimator (TPE) optimization is applied for convolutional and boosting models to present optimal results.

The simulation results of the selected classifiers in terms of detection rate, misdetection rate, false alarm rate, and accuracy are illustrated in Table 4.3 for exploitation attacks. As shown in this table, the Stacking-based ensemble learning technique provides the best performance results among all the other classifiers for exploitation attacks. As can be seen, this technique obtains 96% detection rate, 1 % misdetection rate, 0.7 % , false alarm rate, and 97.3 % accuracy, respectively. In addition, other ensemble techniques, namely boosting and bagging for exploitation attacks provide acceptable results based on used evaluation metric. In fact, the boosting methods particularly perform better than the bagging techniques for exploitation attacks. For example, the boosting-based method provides 95.9% detection rate, 0.9% misdetection rate, 1.2 % false alarm rate, and 96.7% accuracy, while the bagging-based method achieves 95% detection rate, 1.2% misdetection rate, 1 % false alarm rate, and 95 % accuracy. In exploitation attacks, the random forest method, compared to other traditional machine learning techniques, generally performs much better. For example, It achieves 95% detection rate, 1.2% misdetection rate 1.9 % false alarm rate, and 94 % accuracy. Naïve Bayes classifier performs the worst in comparison to the other classifiers, as shown in Table 4.3. To conclude, the comparison of traditional machine learning techniques with ensemble techniques for exploitation attacks shows that ensemble techniques overall perform better.

Table 4.3 Evaluation Results for Exploitation-based Attacks.

<b>Classifier</b>	<b>Detection Rate (%)</b>	<b>Misdetection Rate (%)</b>	<b>False Alarm Rate (%)</b>	<b>Accuracy (%)</b>
<b>Stacking</b>	<b>96</b>	<b>1</b>	<b>0.7</b>	<b>97.3</b>
Bagging	95.0	1.2	1	95
Boosting	95.9	1.0	0.9	96.7
Random Forest	94	1.9	1.2	94
Naïve Bayes	87	13	27	77.1
KNN	94.4	2	1.4	94.6

#### 4.2.1.3.3 Results of Improved Boosting Learning Models in Detecting reflection and exploitation and all Intrusion Attacks

Figure 4.3 provides the accuracy values of the six selected models for the reflection and exploitation attacks. As can be seen, the CatBoost classifier achieves an accuracy of 97.71% for all attacks, 97.8% for the reflection attacks, and 97.9% for the exploitation attacks, which presents the highest accuracy in comparison with the other classifiers. The Gradient boosting classifier also sustains acceptable results, followed by AdaBoost, KNN, SVM, and NB for all attack cases. It has an accuracy of 94.61% for all attacks, 94.5% for the reflection attacks, and 94.63 % for the exploitation attacks. The NB

classifier provides the lowest accuracy for all cases compared to other models, with an accuracy of 85.06% for all attacks, 84.33% for the reflection attacks, and 85.8% for the exploitation attacks. Therefore, the boosting models perform better in terms of accuracy than the other convolutional machine learning models. Furthermore, among the boosting models, the CatBoost classifier provides the best accuracy for detecting intrusion attacks. Among the convolutional machine learning models, the KNN provides the best results in terms of accuracy. However, the accuracy is not enough for evaluating models in detecting cyberattacks.

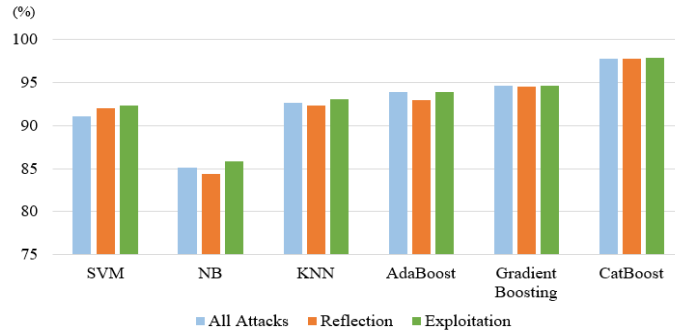


Figure 4.3 Accuracy of the Selected Models.

Figure 4.4 provides detection rate of the six selected models. This figure shows that the CatBoost classifier has the highest detection rate, followed by AdaBoost, Gradient Boosting, KNN, SVM, and NB for all scenarios, respectively. The CatBoost classifier has a detection rate of 96.89% for all attacks, 96.7% for the reflection attacks and 97.1% for the exploitation attacks. The other boosting classifiers, Gradient Boosting and AdaBoost, have also good probabilities of detection for all cases. But, AdaBoost has a better detection rate than that of the Gradient Boosting for all scenarios. The results also show that the boosting models have better results than those of the convolutional models in terms of detection rate of for detecting intrusion attacks in smart grid as shown in Fig. 3. Hence, As discussed, the CatBoost classifier is the best model in terms of detection rate, although the NB classifier is considered as one of the weak models for detecting intrusions in the network. Among the convolutional models, KNN has the best results in terms of detection rate compared to other convolutional models and Naïve Bayes has the poorest results for reflection and exploitation attacks.

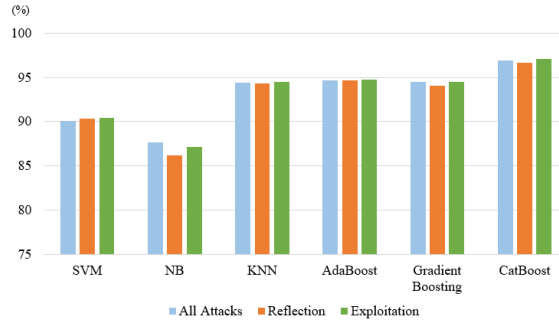


Figure 4.4 Detection Rate of the Selected Models.

Misdetection rate is an important metric for evaluating models in detecting attacks. Figure 4.5 illustrates the results of the misdetection probability of all models. As one can see, boosting models have better results, lower misdetection rate than the convolutional machine learning models. Among all models, CatBoost classifier presents the lowest misdetection rate of 5.06% for all attacks, 5.1% for reflection attacks, and 4.8% for exploitation attacks. The other boosting classifiers, Gradient Boosting and AdaBoost, have slightly higher results in terms of misdetection rate for all scenarios. For example, the Gradient Boosting model has a misdetection rate of 5.21% for all attacks, 5.4% for the reflection attacks, and 4.9% for the exploitation attacks, which are relatively better results compared to those of the AdaBoost classifier. Among convolutional machine learning models, KNN has the lowest probability of misdetection for all scenarios in terms of the probability of misdetection, and the Naïve Bayes classifier has the highest misdetection rate of 9.83% for all attacks, 9.9% for the reflection attacks, and 8.9% for the exploitation attacks.

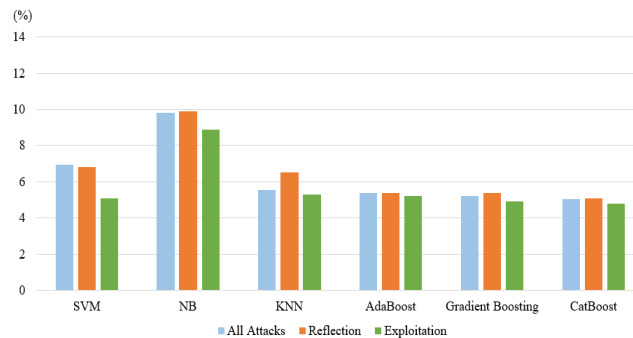


Figure 4.5 Misdetection Rate of the Selected Models.

Figure 4.6 shows the false alarm rate of the six models. As one can observe, the boosting models give better results than the convolutional models. Among all these models, CatBoost classifier has the best results in terms of the false alarm rate in all scenarios. This classifier has a false alarm rate of 3.98% for all attacks, 4.1% for reflection attacks, and 2.7% for exploitation attacks. The other boosting classifiers, Gradient Boosting and AdaBoost, also provide good results for the



false alarm rate. Gradient Boosting performs better than AdaBoost for all cases. Among the convolutional models, SVM performs better than the two other models from the same category. It has a probability of false alarm of 8.45% for all attacks, 8.3% for reflection, and 7.6% for exploitation attacks. The worst results belong to the Naïve Bayes classifier with a false alarm rate of 11.47% for all attacks, 11.1% for reflection attacks, and 10.4% for exploitation attacks.

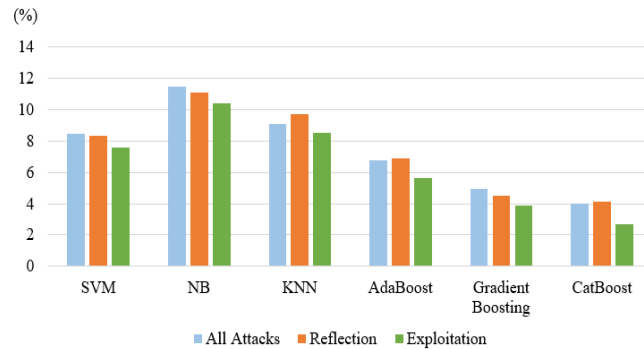


Figure 4.6 False Alarm Rate of Selected Models.

In short, the key insights of these results are discussed as following:

- Eight features of low importance were discarded from the dataset as a result of the RefiefF feature selection that gives the most significant features in training a successful model.
- Boosting classifiers provide higher-performance results in comparison with convolutional machine learning models.
- Among boosting models, the CatBoost classifier provides the best results in terms of accuracy, detection rate, false alarm rate, and misdetection rate.
- Among the convolutional machine learning models, SVM provides the best results and Naïve Bayes gives the worst results in terms of the four metrics.

#### 4.2.1.4 Results of Neural Models in Detecting Intrusion Attacks

In this section, the results of neural network-based models are presented.

##### 4.2.1.4.1 Results of Residual Neural Network in Detecting Intrusion Attacks

In this dissertation, a deep learning approach, Residual Neural Network with 50 layers (ResNet-50) was proposed, to detect and classify attacks targeting intrusion systems in smart grid. The evaluation is performed based on several metrics, namely accuracy, detection rate, misdetection rate, and false alarm rate. The performance of the proposed model is

compared with those of several traditional and conventional ML models, namely Gaussian Naïve Bayes, C-support vector machine, logistic regression, K-nearest neighbor, and random forest.

Figure 4.7 shows the results of the six models for detecting all the attacks, Fuzzers, DoS, Exploits, and Genetic attacks only in terms of the four metrics. As one can observe, the ResNet-50 model has the highest accuracy, followed by random forest, K- nearest neighbor, C-support vector machine, logistic regression, and gaussian naïve Bayes. ResNet-50 model has an accuracy of 98.1%, while the worst accuracy belongs to the gaussian naïve bayes classifier with an accuracy of 86.8%. The random forest classifier has an accuracy of 97.66%, while the K-nearest neighbor classifier shows an accuracy of 97%, C-support vector machine achieves an accuracy of 95.5%, and logistic regression classifier provides an accuracy of 89.3%.

The ResNet-50 model also has the highest detection rate of 99.12%, while the logistic regression classifier has the lowest detection rate of 81.9%. The random forest, K-nearest neighbor, Gaussian naïve Bayes, and C-support vector machine classifiers have a detection rate of 99%, 98.24%, 94.43%, 96.3%, respectively, which are considered acceptable results. In addition, ResNet-50 provides the lowest misdetection rate in comparison with the other models. This model has a misdetection rate of 0.88%, while the worst misdetection rate belongs to logistic regression with 19.13%. Random forest, C-support vector machine, and K-nearest neighbor, have a misdetection rate less than 2%, which is considered a very good result. However, Gaussian naïve Bayes has a high misdetection rate of 5.26%.

Regarding false alarm probability, ResNet-50 has the lowest false alarm rate with 1.03%. In contrast, gaussian naïve bayes has the highest false alarm rate of 14.5%, which is considered the worst. The other models provide acceptable false alarm rate. For instance, the neighbor, logistic regression, and C-support vector machine random forest classifier have a false alarm rate of 1.97%, which is slightly higher than logistic regression and K- nearest neighbor (1.88%).

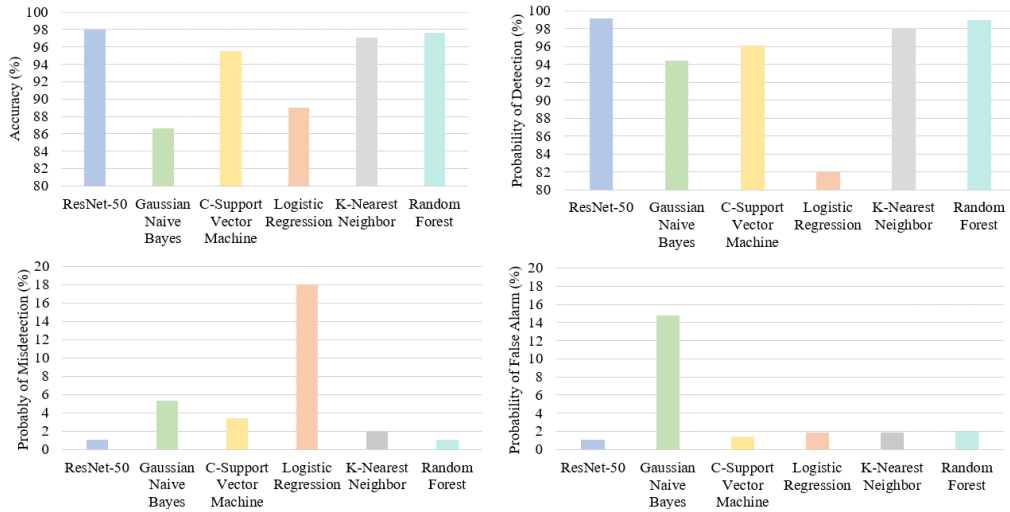


Figure 4.7 Evaluation results in terms of accuracy, detection rate, misdetection rate, and false alarm rate.

Figure 4.8 presents the results of the highlighted models for detecting attacks. As this figure shows, attacks are detected by ResNet-50 with an accuracy of 98.62%, while logistic regression model has an accuracy of 90.15%. The other models, random forest and K-nearest neighbor, provide acceptable results, which are slightly lower than the accuracy of ResNet-50.

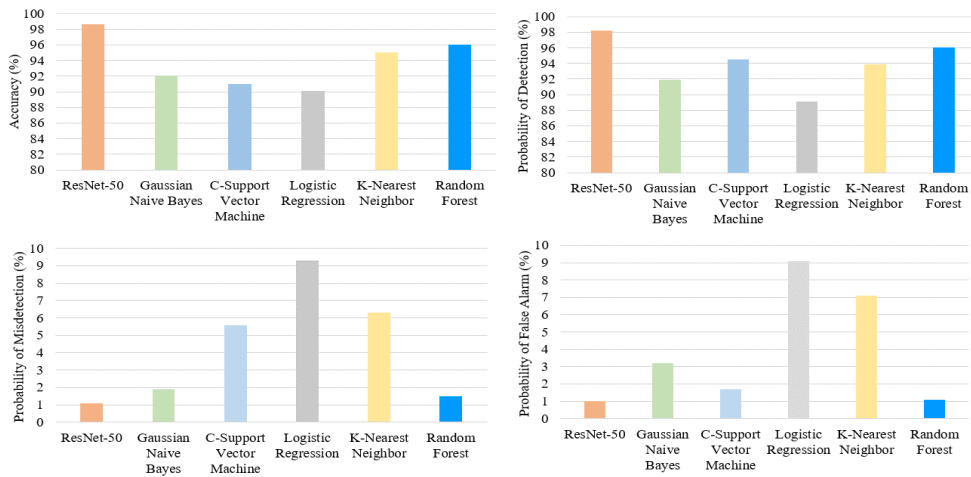


Figure 4.8 Results of attacks in terms of accuracy, detection rate, misdetection rate, and false alarm rate.

As shown in Figure 4.8, ResNet-50 detects attacks with the highest detection rate 98.21%, followed by random forest, C-support vector machine, K-nearest neighbor, Gaussian naïve bayes, and logistic regression. The lowest results belong to the logistic regression mode with a detection rate of 89.1%. ResNet-50 detects attacks with a lowest misdetection rate of 1.09%. Random forest model can detect these attacks with a misdetection rate of 1.5%, which is slightly higher than that of ResNet-50. In contrast, the logistic regression model has the highest and worst misdetection rate of 9.3%.

#### 4.2.1.4.2 Results of Densely Connected Neural Network in Detecting Intrusion Attacks

The DenseNet models' results for the detection of attacks in terms of accuracy, detection rate, misdetection rate, and false alarm rate are illustrated in Figure 4.9. It can be seen that the DenseNet-264 provides the best performance in comparison with the other DenseNet models. The attacks are detected using the DenseNet-264 model with an accuracy of 98.55%, a detection rate of 99.35%, a misdetection rate of 0.65%, and a false alarm rate of 0.9%. In contrast, the DenseNet-121 has the lowest performance results compared to the other DenseNet models.

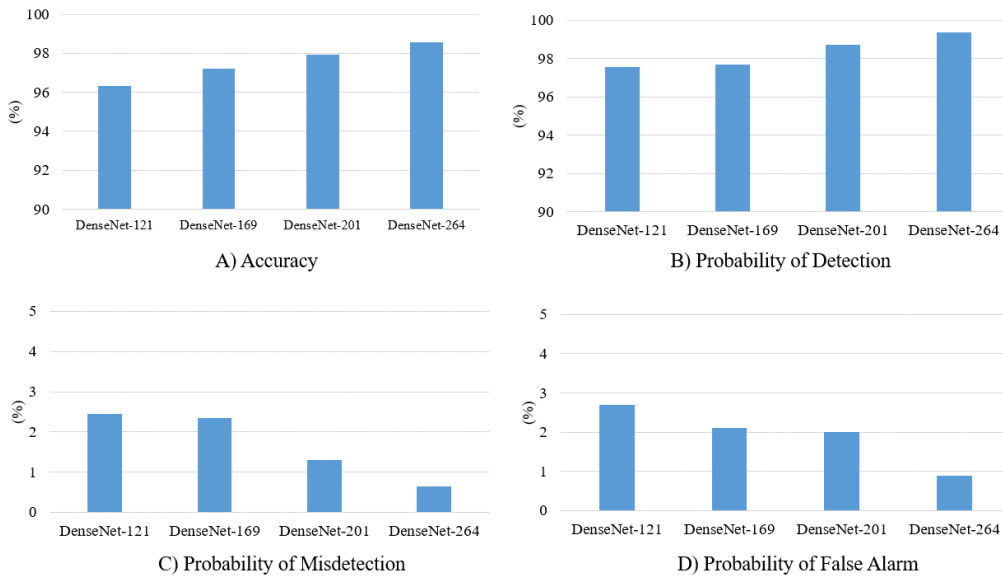


Figure 4.9 Results of the Attacks for the DenseNet Models in terms of Highlighted Metrics.

With this model, attacks can be detected with an accuracy of 96.34%, a detection rate of 97.35%, a misdetection rate of 2.45%, and a false alarm rate of 2.7%.

Table 4.4 illustrates the performance of the models for detecting attacks in terms of the processing time, training time per sample, prediction time, and memory size. As can be seen, DenseNet-264 detects attacks with the lowest prediction time, training time per sample, and processing time. However, the memory size needed by this model is slightly larger than those of DenseNet-121 and DenseNet-201. DenseNet-121 performs poorly in comparison with the other DenseNet models in terms of these metrics, except for the memory size.

Table 4.4 Models' performance for Attacks in Terms of the considered Metrics (Best performances are in bold).

<b>Model</b>	Processing Time (Sec)	Training Time Per Sample (Sec)	<b>Prediction Time</b> (Sec)	<b>Memory Size</b> (MiB)
DenseNet-121	410	0.59	0.89	157
DenseNet-169	402	0.57	0.73	183
DenseNet-201	390	0.51	0.66	177
<b>DenseNet-264</b>	<b>388</b>	<b>0.49</b>	<b>0.62</b>	<b>182</b>

#### 4.2.1.5 Results of Comparison between Supervised and Unsupervised ML and DL models in Detecting Intrusions

Figures 4.10 and 4.11 present the results of the learning models in terms of accuracy, detection rate, misdetection rate, and false alarm rate. The AlexNet model yielded the best results in terms of the selected metrics among supervised models. LightGBM yielded a slightly lower accuracy, detection rate and higher misdetection rate, and false alarm rate compared to the AlexNet model. The other supervised models, CART and C-SVM, also had satisfactory results. The LR and GNB models yielded the worst results among the supervised models. The VA-encoder model yielded the highest performance compared to the other unsupervised models. In contrast, the unsupervised models exhibited significantly lower performance in terms of the same metrics: the PCA model yielded considerably lower performance than the VA-Encoder. The K-means model had the lowest accuracy and detection rate and the highest misdetection rate, and false alarm rate. The AlexNet model yielded the best results compared to all supervised and unsupervised models in terms of the tested metrics, followed by LightGBM, VA-Encoder, CART, C-SVM, PCA, GNB and LR, and K-means.

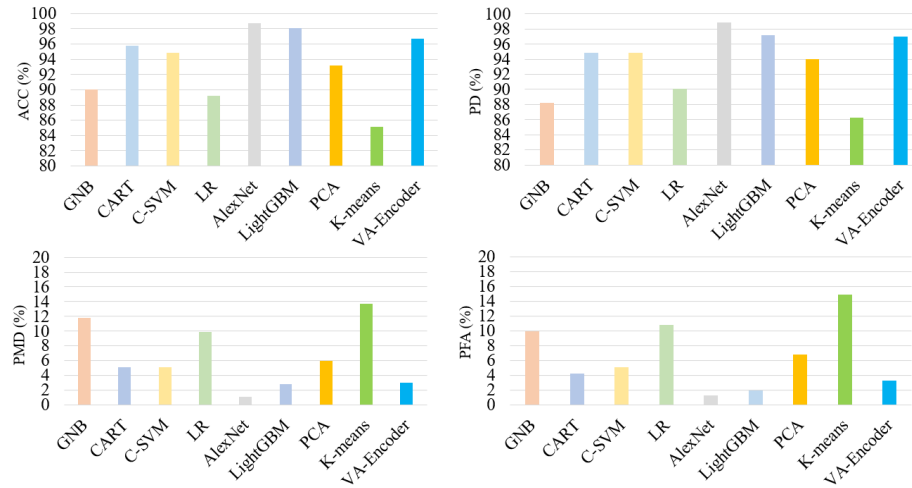


Figure 4.10 Performance evaluation of the ML models in terms of accuracy, detection rate, misdetection rate, and false alarm rate for Test Data.

Table 4.5 presents the model results in terms of the tested metrics. The AlexNet model has the lowest processing time, prediction time, training time per sample, and memory size compared to the other supervised and unsupervised models, while the Gaussian Naïve Bayes model had the worst performance compared to the other models. The CART model yields slightly higher processing time, prediction time, training time per sample, and memory size. The LightGBM model has acceptable results among the supervised models, whereas the VA-encoder model yields the best performance among the unsupervised models. K-means has the lowest performance compared to the other unsupervised models.

Table 4.5 The ML models' performance in Terms of Processing Time, Prediction Time, Training Time per Sample, and Memory Size for Test Data(Best performances are in bold).

Model	Processing Time (Sec)	Prediction Time (Sec)	Training Time per Sample (Sec)	Memory Size (MiB)
GNB	4.33	4.15	0.82	245
CART	1.2	1.1	0.2	132.
C-SVM	2.9	1.8	0.39	236
LR	1.6	1.2	0.51	223
<b>AlexNet</b>	<b>1.01</b>	<b>1</b>	<b>0.01</b>	<b>102</b>
LightGBM	1.4	1.3	0.09	112
PCA	1.9	0.91	0.89	164
K-means	1.9	1.4	0.81	180

<b>VA-Encoder</b>	<b>1.77</b>	<b>1.2</b>	<b>0.5</b>	<b>144</b>
-------------------	-------------	------------	------------	------------

As a consequence, the AlexNet model had the best results among the supervised models, whereas the VA-Encoder yielded the best results among the unsupervised models in terms of accuracy, detection rate, misdetection rate, and false alarm rate, processing time, prediction time, training time per sample, and memory size; therefore, the performance of these two models was investigated and compared in terms of classifying and detecting different types of cyber-attacks attacks on the smart grid.

Figure 4.11 represents the results of individual attacks for the two best selected models. AlexNet and VA-Encoder, in terms of accuracy, detection rate, misdetection rate, and false alarm rate. AlexNet outperformed the VA-Encoder model when detecting cyber-attacks. For example, the DNS attacks were detected with better performance using AlexNet compared to VA-Encoder. AlexNet detected these attacks with an accuracy of 99.13%, a detection rate of 99.81%, a misdetection rate of 0.19%, and a false alarm rate of 0.93%. VA-Encoder detected the same attacks with considerably lower performance: an accuracy of 96.83%, a detection rate of 97.11%, a misdetection rate of 2.89%, and a false alarm rate of 3.23%. The VA-encoder detected and classified UDP attacks with the highest performance. AlexNet detected MSSQL attacks with the lowest performance; however, the VA-Encoder detected SSDP, NTP, and TFTP with the lowest performance. AlexNet and VA-Encoder classified and detected all cyber-attacks with satisfactory results.

Table 4.6 presents the results of AlexNet and VA-Encoder in terms of processing time, prediction time, training time per sample, and memory size. AlexNet outperformed VA-Encoder for detecting cyber-attacks. For example, DNS attacks could be detected and classified using AlexNet with significantly lower processing time, prediction time, training time per sample, and memory size than VA-encoder. AlexNet detected the DNS attacks with a processing time of 1.1 seconds, a Prediction time of 0.9 seconds, a training time per sample of 0.3 seconds, and a memory size of 149 MiB. AlexNet detected NetBIOS attacks with the highest PRT, PT, TPS, and M compared to other attacks; however, VA-Encoder detected SSDP attacks with the highest processing time, prediction time, training time per sample, and memory size compared to other attacks. AlexNet could detect LDAP, DNS, SNMP, MSSQL, NetBIOS, NTP, SSDP, TFTP, UDP, UDP-Lag attacks, and Benign traffic better than VA-Encoder. The key points of this study are:

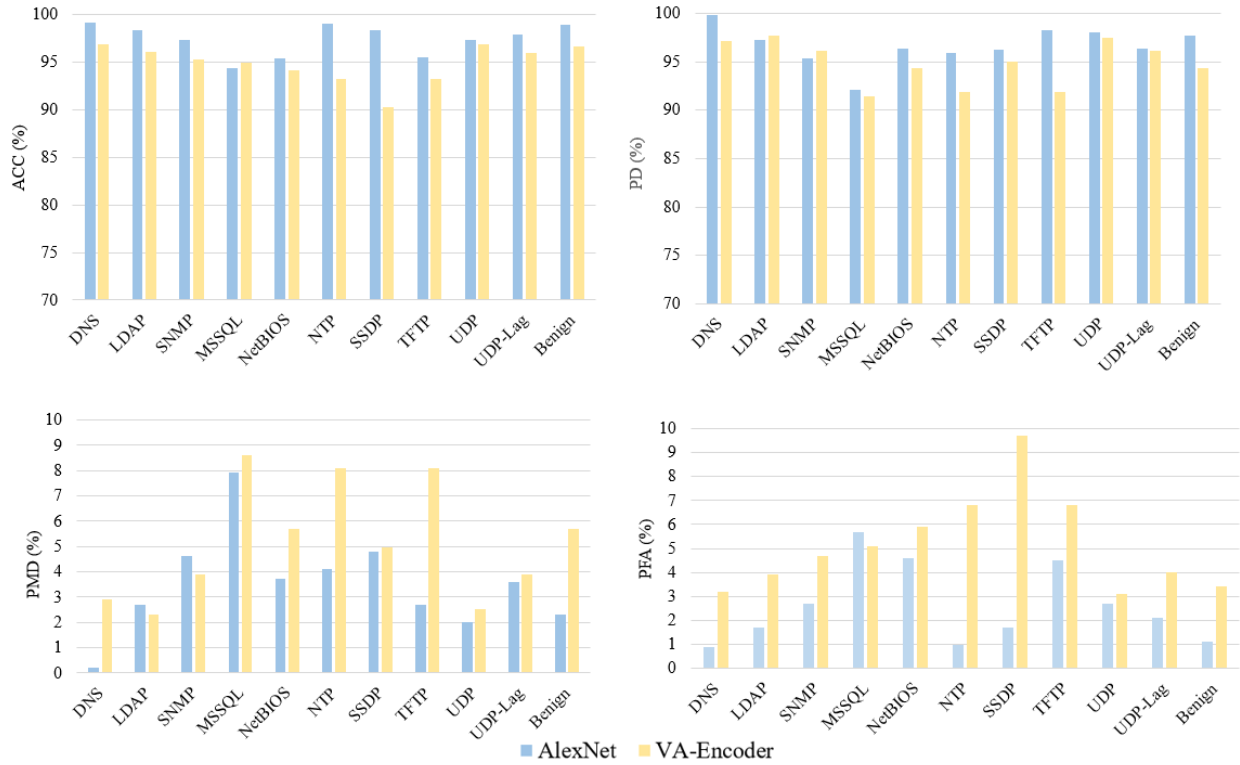


Figure 4.11 Performance evaluation of cyber-attacks based on best ML models in terms of processing time, prediction time, training time per sample, and memory size.

Table 4.6 Performance of the ML Models in Terms Of PRT, PT, TPS, and M for Test data.

Models	Attacks	PRT (Sec)	PT (Sec)	TPS (Sec)	M (MiB)
AlexNet	LDAP	1.4	1.2	0.7	125
	DNS	1.1	0.9	0.3	149
	SNMP	1.9	1.2	0.4	123
	MSSQL	1.3	1.2	0.1	177
	NetBIOS	1.9	1.4	0.9	191
	NTP	1.2	1.7	0.6	182
	SSDP	1.1	1	0.5	173
	TFTP	1.4	1.1	0.7	167
	UDP	1.8	1.2	0.5	166
	UDP-Lag	1.3	1.1	0.2	161
	DNS	1.4	1.2	0.7	125



	Benign	1.8	1.4	0.4	180
VA-Encoder	LDAP	3.5	3.1	0.4	290
	DNS	3.4	2.9	0.4	278
	SNMP	3.2	2.3	0.9	276
	MSSQL	2.9	2.3	0.3	254
	NetBIOS	2.9	2.3	0.4	246
	NTP	2.9	1.3	0.6	277
	SSDP	3.9	2.1	0.5	297
	TFTP	3.1	2.9	0.3	289
	UDP	3.8	3.1	0.7	290
	UDP-Lag	2.9	2.7	0.2	214
	Benign	3.1	2.9	0.2	212

- The AlexNet model yielded the best results of all supervised and unsupervised learning techniques in terms of the highlighted metrics.
- GNB and LR models yielded the worst results of the supervised models.
- The VA-Encoder model yielded the highest-performance results of the unsupervised models.
- The worst performance model among the unsupervised models was K-means.
- Several models, such as CART, C-SVM, and PCA, yielded satisfactory results

#### 4.2.1.6 Results of Reinforcement Learning-based Model in Detecting Intrusion Attacks

In this dissertation, the confusion matrices for the CapsNet Q-learning model based on two different discount factors of 0.001 and 0.9 are visualized. The confusion matrix represents the evaluation of the proposed model on the test data. In general, the row in this matrix presents the predicted class, and the column shows the true class. The confusion matrix on the main diagonal proves the correctly predicted class, such as TP and TN, whereas the incorrectly classified is in the off-diagonal cells, such as FP and FN. According to Figure 4.12, we can see that the true positive rate in the CapsNet Q-learning with a discount factor of 0.001 equals to 200 for the attacks, while it decreases to 190 for the attacks in a discount factor of 0.9. These results show that the CapsNet Q-learning agent performs much better with smaller discount values.

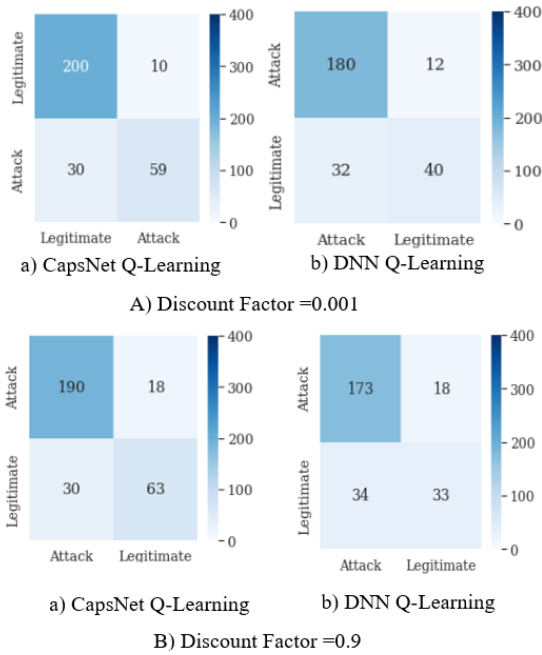


Figure 4.12 Confusion Matrices of the Models with Respect to the discount factor.

Figure 4.13 presents the results of the proposed CapsNet Q-learning model in comparison with the DNN Q-learning model in terms of accuracy, detection rate, misdetection rate, and false alarm rate. based on the two different values of discount factors of 0.001 and 0.9. It is evident from this figure that the proposed CapsNet Q-learning model with a discount factor of 0.001 provides the higher performance with an accuracy of 86.96%, a detection rate of 86.62%, a false alarm rate of 14.49%, and a misdetection rate of 13.04%. However, the DNN Q-learning with the same discount factor has an accuracy of 83.33%, a detection rate of 84.90%, a false alarm rate of 23.07%, and a misdetection rate of 15.10%, respectively. Additionally, the CapsNet Q-learning model with a discount factor of 0.9 has lower performance with an accuracy of 84.05%, a detection rate of 86.36%, a false alarm rate of 22.22%, and a misdetection rate of 13.64%. As a result, the two Q-learning models have better results with a lower discount factor, compared to the models with a higher discount factor. Moreover, as the discount factor is lower, better performance can be seen for detecting and classifying attacks for IDS on the smart grid.

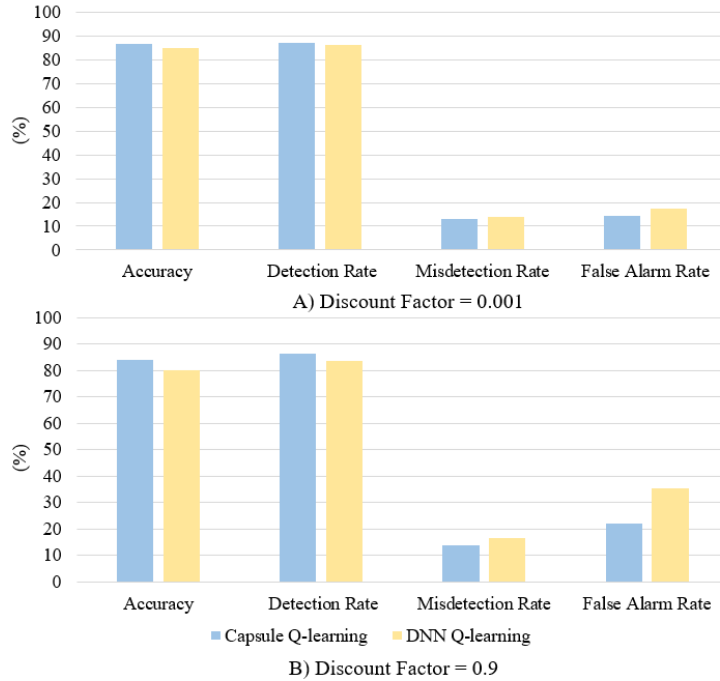


Figure 4.13 Results of the Proposed Models in in terms of Accuracy, Detection Rate, Misdetction Rate, and False Alarm Rate. based on the Discount Factors of 0.001 and 0.9.

Table 4.7 highlights the results of CapsNet Q-learning and DNN Q-learning models in terms of training time and *PT* with respect to the discount factors of 0.001 and 0.9. As one can see, the CapsNet Q-learning with a discount factor of 0.001 has the lowest training time and prediction time with 300.53 seconds (about 5 minutes) and 0.54 seconds, respectively. In contrast, the same model with a discount factor of 0.9 provides worse training time and prediction time. It is worth mentioning that the DNN Q-learning model reaches significantly higher training time and prediction time compared to the CapsNet Q-learning model for both values of discount factors. DNN Q-learning achieves a training time of 540.09 seconds, and prediction time of 1.03 seconds for a discount factor of 0.001, whereas the same model has training time of 587.54 seconds (about 10 minutes) and prediction time of 1.54 seconds in a discount factor of 0.9. As a result of this table, it is concluded that the lower discount factors provide lower training time and prediction time, while the lowest results in terms of training time and prediction time belong to the CapsNet Q-learning model.

Table 4.7 Comparison of Different Timing Metrics with Respect to the Discount Factor of 0.001 and 0.9.

Model	Discount Factor =0.001		Discount Factor = 0.9	
	Training Time (Sec)	Prediction Time (Sec)	Training Time (Sec)	Prediction Time (Sec)
CapsNet Q-learning	300.53	0.54	340.90	0.78

DNN Q-learning	540.09	1.03	587.54	1.54
----------------	--------	------	--------	------

As a result, the proposed model uses Q-learning RL-based approach and Capsule Network to communicate with the network environment, where the network traffic is captured and analyzed to detect and classify attacks for an IDS on the smart grid. This proposed model can payload in a self-learning fashion using an agent with no human knowledge. The proposed techniques are discussed in detail, including the agent, environment, Q-values, exploration, and rewards. To improve the learning capacity of an RL-based model, a parameter, namely the discount factor used to analyze the results. The experiment results depict the exponential progress and satisfactory performance of this proposed technique over DNN Q-learning approach. Finally, through extensive experiments of the results, the highest performance can be achieved with a discount factor of 0.001 in 100 episodes of learning.

#### 4.2.1.7 Results of Online Deep Learning Model in Detecting Intrusion Attacks

In this dissertation, the performance of our proposed online model, Online Sequential Euclidean Distance Routing CapsNet is assessed, and its results are compared with the Online Dynamic Routing CapsNet. It is worth mentioning that regularization (Reg) and learning without forgetting (LF) were used to address catastrophic forgetting. For this reason, the online models with respect to these techniques are discussed; hence, the evaluated models are defined as following:

- Online Sequential Euclidean Distance Routing CapsNet with learning without forgetting and regularization (OS-CapsNet-LF-Reg)
- Online Dynamic Routing CapsNet with learning without forgetting and regularization (OD-CapsNet-LF-Reg)
- Online Sequential Euclidean Distance Routing CapsNet with no learning without forgetting and regularization (OS-CapsNet-Reg)
- Online Dynamic Routing CapsNet with no learning without forgetting and regularization (OD-CapsNet-Reg)
- Online Sequential Euclidean Distance Routing CapsNet with learning without forgetting and no regularization (OS-CapsNet-LF)
- Online Dynamic Routing CapsNet with learning without forgetting and no regularization (OD-CapsNet-LF)
- Online Sequential Euclidean Distance Routing CapsNet with no learning without forgetting and no regularization (OS-CapsNet)

- Online Dynamic Routing CapsNet with no learning without forgetting and no regularization (OD-CapsNet)

In addition, several evaluation metrics are used, namely accuracy, detection rate, misdetection rate, false alarm rate, processing time, training time per sample, prediction time, and memory size. For validation of our proposed technique, confusion matrices with respect of routing algorithms, learning without forgetting algorithm and regularization techniques are provided.

Figures 4.14, 4.15, and 4.16, and Table 4.8 illustrate the results of the proposed models in terms of metrics mentioned. As shown in Figure 4.14, the OS-CapsNet-LF-Reg model provides the best results in terms of accuracy, detection rate, misdetection rate, false alarm rate, for detecting and classifying intrusion attacks. It is also worth to mention that this model has an accuracy of 98.63%, a detection rate of 99.56%, a misdetection rate of 0.43%, and a false alarm rate of 4.68% respectively. In contrast, the OS-CapsNet-LF-Reg model has a lower accuracy of 97.61%, a detection rate of 98.38%, a misdetection rate of 1.62%, and false alarm rate of 11.53%. In addition, it can be seen that the learning without forgetting algorithm and regularization techniques have strong impacts on the performance of the CapsNet. For instance, the OS-CapsNet and OD-CapsNet models, models without learning without forgetting algorithms and regularization techniques, provide the lowest performance in comparison with other models.

Moreover, the figure indicates that learning without forgetting algorithm affects the online CapsNet model stronger than the regularization technique, as OS-CapsNet-LF and OD-CapsNet-LF have better results, compared to the OS-CapsNet-Reg and OD-CapsNet-Reg models. Furthermore, the results prove that the OS-CapsNet models with or without learning without forgetting algorithms and regularization techniques have more acceptable results than the OD-CapsNet models. The true positive, true negative, false positive, and false negative of these models are shown in Figure 4.15 A to F as confusion matrices to prove the efficiency of these models.

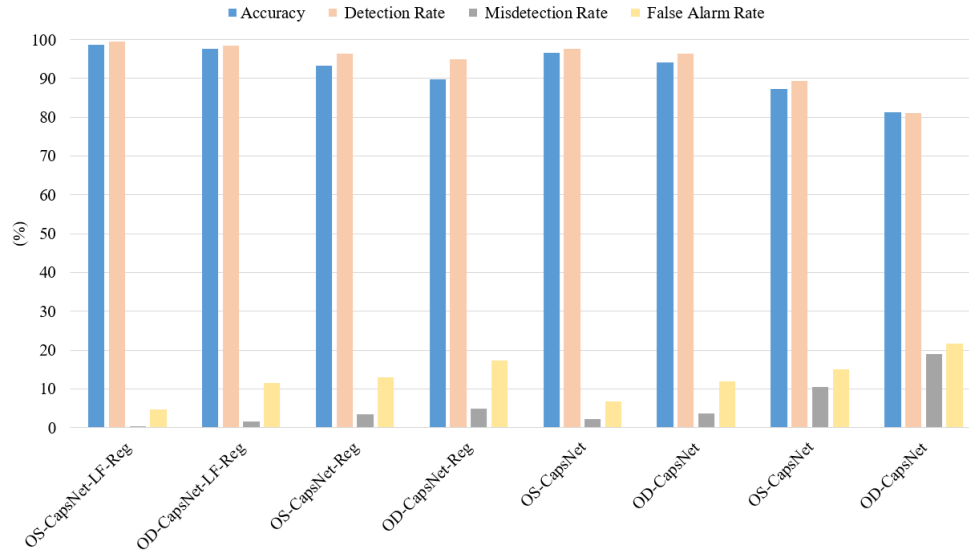


Figure 4.14 Results of Highlighted models in terms of accuracy and detection rate, misdetection rate, and false alarm rate.

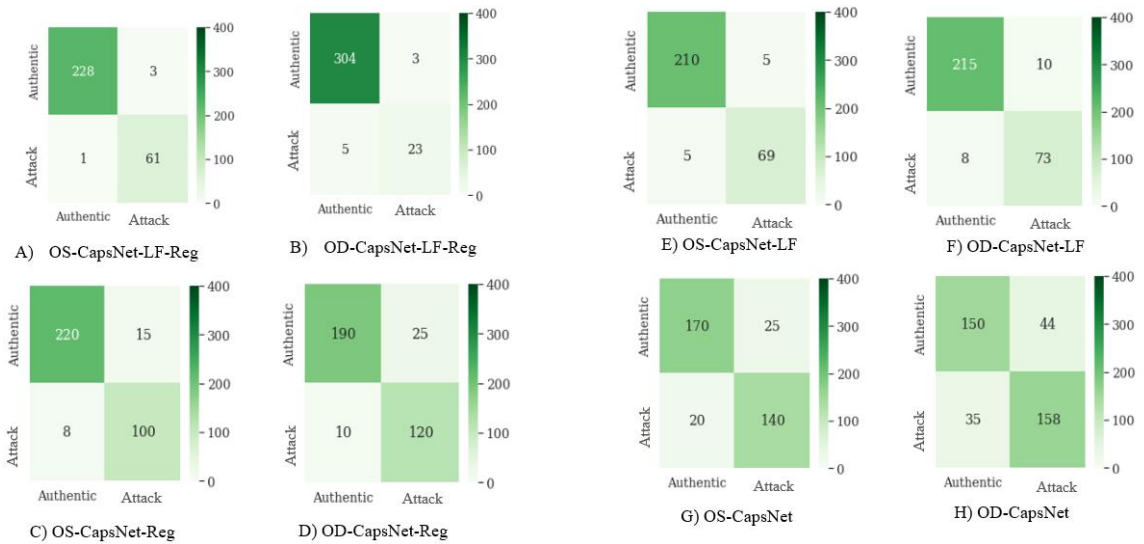


Figure 4.15 Confusion Matrices of 2-Class CapsNet Models.

Table 4.8 also indicates the test results of these models in terms of Training Time Per sample, Prediction Time, and Memory Size. It can be seen that the OS-CapsNet-LF-Reg model provides the best performance in terms of these metrics for detecting attacks. In fact, intrusion attacks using Online Sequential Euclidean Distance Routing CapsNet model with learning without forgetting and regularization techniques can be detected and classified with a training per sample of 1.01 seconds, a prediction time of 1.88 seconds, and a memory size of 339 mebibyte. In the contrary, these attacks using Online Dynamic Routing CapsNet with learning without forgetting and regularization techniques can be detected and classified with an exponential increase in training time per sample, prediction time, and memory size.

Additionally, these attacks using the same model has acceptable results with a training time per sample of 1.39 seconds, a prediction time of 2.05 seconds, and a memory size of 567 mebibyte. It can be seen that models using Learning without forgetting technique, namely OS-CapsNet-LF and OD-CapsNet-LF outperform the models that only used regularization technique, namely OS-CapsNet-Reg and OD-CapsNet-Reg. It is also worth mentioning that Online Sequential Euclidean Distance Routing CapsNet provides better results in comparison of Online Dynamic Routing CapsNet for all the cases. Therefore, we can highlight that the attacks using Online Sequential Euclidean Distance Routing CapsNet and learning without forgetting algorithm along with the regularization technique indicate better performance in terms of this metrics in comparison with these attacks in other cases.

Table 4.8 Test Results of the Highlighted Models in terms of Training Time Per sample, Prediction Time, and Memory Size.

<b>Models</b>	<b>Training Time Per Sample (Sec)</b>	<b>Prediction Time (Sec)</b>	<b>Memory Size (MiB)</b>
<b>OS-CapsNet-LF-Reg</b>	<b>1.01</b>	<b>1.88</b>	<b>339</b>
OD-CapsNet-LF-Reg	1.39	2.05	567
OS-CapsNet-Reg	2.6	3.8	367
OD-CapsNet-Reg	2.7	3.9	601
OS-CapsNet-LF	1.67	2.7	376
OD-CapsNet-LF	1.75	2.9	587
OS-CapsNet	6.98	8.2	387
OD-CapsNet	7.8	8.99	589

Figure 4.16 illustrates the results of the used models in terms of accuracy and Kappa with respect to the sample size. The results demonstrate that the OS-CapsNet-LF-Reg model has an accuracy of above 97% for different sample sizes; however, the OD-CapsNet-LF-Reg model provides slightly lower accuracy for different sample sizes. In addition, It can be seen that the performance of the OS-CapsNet-LF-Reg model is at a high level for different sample sizes. More specifically, the performance of this model remains extremely high during the simulations. However, the performance of the OD-CapsNet-LF-Reg model provides satisfactory results. In this context, this model accuracy and Kappa score is slightly lower than the OS-CapsNet-LF-Reg model through the simulations. It is observable that other models with learning without forgetting algorithms have higher accuracy and Kappa score in comparison with the models without learning

without forgetting algorithms or with regularization algorithms. As a result, the OS-CapsNet-LF-Reg model is not the only model with the best performance but also the one that has a nearly perfect performance in all evaluation metrics for detecting and classifying intrusion attacks on smart grid.

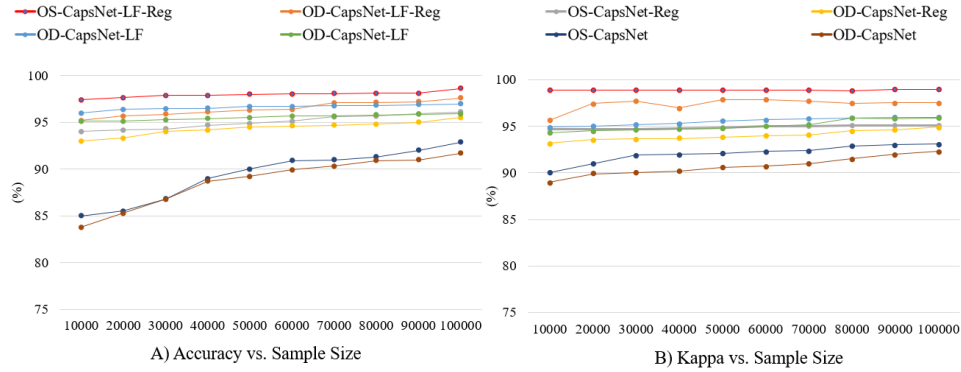


Figure 4.16 Results of Highlighted Models in terms of Accuracy and Kappa with respect to the Sample Sizes.

To conclude, the Online Sequential Euclidean Distance Routing CapsNet model outperforms the other developed online original CapsNet, Online Dynamic Routing CapsNet, to detect and classify attacks on smart grid. Our results also demonstrate the efficiency of our proposed model in terms of the metrics mentioned.

As previously outlined, the Online Sequential Euclidean Distance Routing CapsNet model provides significantly better results in detecting attacks in our network. In contrast, a limited number of current studies in the field of online deep learning make it difficult to compare our model with other existing studies. In fact, the majority of these studies used different evaluation metrics solvedolve different challenges in online learning; however, in this study, we aim at improving the efficiency of our model with focusing in dealing with catastrophic forgetting. We also modified the structure of routing algorithm in original CapsNet to verify the high performance of our proposed model. As discussed, the Online Dynamic Routing CapsNet provides a competitive performance compared to other existing routing mechanisms in CapsNet. To this end, we also attempt to answer the following academic questions:

1. How do we develop an online deep learning detection method to detect attack scenarios instead of batch-based scenarios?
2. How can we deal with catastrophic forgetting issues in CapsNet models



3. How changing the routing algorithm in capsule network can impact the performance of online models? We believe to the best of our knowledge that the Online Sequential Euclidean Distance Routing CapsNet model can represent new improvement in field of detecting and classifying attacks on smart grid.

## Chapter 5

### CONCLUSION

Smart grid offers a promising alternative to the traditional power grid by intelligently managing electricity distribution through two-way communication with consumers. This modern grid comes with several advantages, including improved reliability, affordability, the ability to integrate green energy sources, and efficient use of renewable power. However, it still faces several challenges, primarily related to security. Smart grid networks are susceptible to various cyber threats, such as intrusion attacks, which have targeted about 50% of U.S. power systems in recent years.

Intrusion attacks in the context of a smart grid pertain to deliberate and unauthorized attempts to compromise the security and functionality of the intelligent electricity distribution network. These attacks pose significant threats, potentially leading to disruptions in power supply, unauthorized access to critical infrastructure, or the compromise of sensitive data within the grid. Intruders may employ techniques where they overwhelm communication channels or components of the smart grid, thereby disrupting its normal operations. Additionally, malicious actors may seek unauthorized access to the grid's control systems, exploiting vulnerabilities to manipulate or interfere with power distribution. As the smart grid relies on complex computer systems and communication networks, safeguarding against intrusion attacks is paramount to ensuring the reliability and security of the modern electrical infrastructure.

Therefore, the goal of this dissertation is to investigate the efficiency of data-driven techniques in detecting cyberattacks on Smart Grid. To develop and meet the objectives of this dissertation, in Chapter 2, Objective 1, smart grid and cyber-attacks on smart grid, are explored and reviewed. In addition, a brief discussion of smart grid architecture, its characteristics, features, applications are provided. Moreover, the state-of-the-art detection solutions are investigated and discussed. It is also shown that limited security in smart grid is a primary concern to provide a secure network.

In Chapter 3, the methodologies of attack detection scenarios are explained. The training dataset, its features, mathematical models of machine learning, deep learning, reinforcement learning, and online learning are discussed. In general, Chapter 3 summarizes the required materials and methodology used for Objective 2, 3, and 4. In Chapter 4 of this dissertation, the proposed supervised, unsupervised, and reinforcement models along with online-based models are extensively tested and their efficiency are evaluated in terms of accuracy, probability of detection, probability of misdetection, probability of false alarm, processing time, prediction time, training time per sample, memory size, and

confusion matrix. It is also worth mentioning that the models' parameters are optimized, using Grid Search, ADAM optimizer, Stochastic Gradient Descent, Tree-structured Parzen Estimator.

Among the supervised machine learning models, ensemble learning provides the higher performance for detecting and classifying intrusions on smart grid. The results show that categorical boosting ensemble obtain an accuracy of over 97%, using Tree-Structured Parzen Estimator optimization technique. In contrast, the naïve Bayes model has the lowest performance for detecting and classifying intrusions with accuracy of around 85%.

Among deep learning models, the densely connected neural network with 264 layers have the highest performance with an accuracy of over 98%. It is worth mentioning that other neural networks, such as Alex Neural Network also provide high performance results for detecting these attacks. For instance, Alex Neural Network can detect DNS attacks with an accuracy of 99.13%, a probability of detection of 99.81%, a misdetection of 0.19%, a false alarm of 0.93%, a processing time of 1.1 seconds, a prediction time of 0.9 seconds, a training time per sample of 0.3 seconds, and a memory usage of 149 Mebibytes.

Among unsupervised models, the variational auto-encoder has the highest performance results for detecting these attacks with an accuracy around 96%. In contrast, the k-means model has the lowest performance among unsupervised models with an accuracy of around 82%. Also, the results of unsupervised models for investigating every attack performance is discussed. The results indicated the unsupervised models provide lower performance, compared to supervised models. For example, the DNS attack can be detected using variational auto-encoder with an accuracy around 95%.

In reinforcement learning, the proposed capsule deep Q-Network with lower discount factor provides higher performance, compared to traditional deep Q-learning model in literature. Also, the results show that as the discount factor decreases, the performance of the proposed model increases. According to the results, the proposed capsule deep Q-Network with the discount factor of 0.001 detects intrusions with an accuracy of over 86%. In the contrary, this technique can provide an accuracy of over 83% with the discount factor of 0.9.

In online learning, the sequential online capsule network, the Euclidean Distance Routing algorithm with catastrophic forgetting and regularization methods outperform other models in literature with an accuracy of over 98%. According to

the results, the proposed model is not the only model with the best performance but also the one that has a nearly stable performance for all evaluation metrics for detecting and classifying intrusion attacks on smart grid.

In conclusion, the proposed Artificial Intelligence-based attack detection methods were shown to be suitable candidates for use in smart grid networks as they can effectively detect intrusion attacks in near real-time with minimal to no modifications to the smart grid infrastructure. These methods can provide a high detection rate and a low false alarm rate compared to the existing methods. However, there are still several unsolved problems related to the smart grid security that must be filled with appropriate methods. In the following, some potential future works are described.

One future research direction is to investigate and develop countermeasure techniques after intrusion attacks were detected. In this regard, a potential research direction is to investigate the efficiency of blockchain [reference] in enhancing the smart grid security. More importantly and as discussed previously, although the existing solutions provide some level of security, there is still no single method to fully secure the smart grid networks. Thus, another potential future work is to develop multi-layer security frameworks composed of several methods to detect and mitigate different attacks targeting the smart grid infrastructure.

Moreover, Artificial Intelligence security solutions, namely deep learning models are susceptible to adversarial attacks, a concerning vulnerability that highlights the fragility of their decision boundaries. Adversarial attacks involve making small, carefully crafted perturbations to input data, often imperceptible to humans, which can lead to misclassification or erroneous outputs from the models. Adversarial attacks not only challenge the reliability of deep learning systems in critical applications such as smart grids, but also underscore the need for developing advanced defense mechanisms and more resilient models that can withstand these intentional manipulations. Therefore, developing robust models that can withstand such attacks and maintaining model security and data is of high importance.

## BIBLIOGRAPHY

- [1] M. Ghiasi, T. Niknam, Z. Wang, M. Mehrandezh, M. Dehghani, and N. Ghadimi, "A comprehensive review of cyber-attacks and defense mechanisms for improving security in smart grid energy systems: Past, present and future," *Electric Power Systems Research*, vol. 215, p.108975, 2023.
- [2] A. Khan, A.A. Laghari, M. Rashid, H. Li, A.R. Javed, and T.R. Gadekallu, "Artificial intelligence and blockchain technology for secure smart grid and power distribution Automation: A State-of-the-Art Review," *Sustainable Energy Technologies and Assessments*, 57, p.103282, 2023.
- [3] M. K. Hasan, A. A Habib, Z. Shukur, Z., F. Ibrahim, S. Islam, and M.A. Razzaque, "Review on cyber-physical and cyber-security system in smart grid: Standards, protocols, constraints, and recommendations," *Journal of Network and Computer Applications*, 209, p.103540, 2023.
- [4] S. Vahidi, M. Ghafouri, M. Au, M. Kassouf, A. Mohammadi and M. Debbabi, "Security of Wide-Area Monitoring, Protection, and Control (WAMPAC) Systems of the Smart Grid: A Survey on Challenges and Opportunities," in *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1294-1335, Secondquarter 2023, doi: 10.1109/COMST.2023.3251899.
- [5] F. Norouzi, T. Hoppe, L.M. Kamp, C. Manktelow, and P. Bauer, P., "Diagnosis of the implementation of smart grid innovation in The Netherlands and corrective actions," *Renewable and Sustainable Energy Reviews*, 175, p.113185, 2023.
- [6] F. Norouzi, T. Hoppe, L.M. Kamp, C. Manktelow, and P. Bauer, "Diagnosis of the implementation of smart grid innovation in The Netherlands and corrective actions," *Renewable and Sustainable Energy Reviews*, vol. 175, p.113185, 2023.
- [7] F. Dewangan, A. Y. Abdelaziz, and M. Biswal, "Load Forecasting Models in Smart Grid Using Smart Meter Information: A Review," *Energies*, vol. 16, no. 3, p. 1404, Jan. 2023, doi: 10.3390/en16031404.
- [8] M. Jafari, A. Kavousi-Fard, T. Chen and M. Karimi, "A Review on Digital Twin Technology in Smart Grid, Transportation System and Smart City: Challenges and Future," in *IEEE Access*, vol. 11, pp. 17471-17484, 2023, doi: 10.1109/ACCESS.2023.3241588.
- [9] M. Waseem, M. Adnan Khan, A. Goudarzi, S. Fahad, I. A. Sajjad, and P. Siano, "Incorporation of Blockchain Technology for Different Smart Grid Applications: Architecture, Prospects, and Challenges," *Energies*, vol. 16, no. 2, p. 820, Jan. 2023, doi: 10.3390/en16020820.
- [10] T. Mazhar *et al.*, "Electric Vehicle Charging System in the Smart Grid Using Different Machine Learning Methods," *Sustainability*, vol. 15, no. 3, p. 2603, Feb. 2023, doi: 10.3390/su15032603.

- [11] T.Ahmad, R. Madonski, D. Zhang, C. Huang, and A. Mujeeb, "Data-driven probabilistic machine learning in sustainable smart energy/smart energy systems: Key developments, challenges, and future research opportunities in the context of smart grid paradigm," *Renewable and Sustainable Energy Reviews*, vol. 160, p.112128, 2022.
- [12] B.B. Gupta, and A. Dahiya, "*Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges and Countermeasures*," CRC Press, 2020.
- [13] V. Y. Pillitteri, V.Y. and T. L. Brewer, "The Smart Grid Interoperability Panel – Cyber Security Working Group, Smart Grid Cyber Security Guidelines," pp. 1–597, 2014.
- [14] S. N. Islam, Z. Baig and S. Zeadally, "Physical Layer Security for the Smart Grid: Vulnerabilities, Threats, and Countermeasures," *Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6522-6530, 2019.
- [15] A. Sinha, R. Vyas, V. Subramanian, and O.P. Vyas, "Critical Infrastructure Security: Cyber-Physical Attack Prevention, Detection, and Countermeasures," *Quantum Cryptography and the Future of Cyber Security*, pp. 134-162, 2020.
- [16] D. Kemi, X. Ren, D. E. Quevedo, S. Dey, and Ling Shi, "Defensive Deception against Reactive Jamming Attacks in Remote State Estimation," *Automatica*, vol. 113, pp. 108680, 2020.
- [17] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos and G. Pantziou, "A Survey on Jamming Attacks and Countermeasures in WSNs," *Communications Surveys and Tutorials*, vol. 11, no. 4, pp. 42-56, 2009.
- [18] Y. Arjoun, F. Salahdine, M. S. Islam, E. Ghribi and N. Kaabouch, "A Novel Jamming Attacks Detection Zubair Approach Based on Machine Learning for Wireless Communication," *International Conference on Information Networking (ICOIN)*, pp. 459-464, 2020.
- [19] M. Z. Gunduz and R. Das, "Analysis of Cyber-attacks on Smart Grid Applications," *International Conference on Artificial Intelligence and Data Processing (IDAP)*, pp. 1-5, 2018.
- [20] D. Karagiannis, and A. Argyriou, "Jamming Attack Detection in a Pair of RF Communicating Vehicles using Unsupervised Machine Learning," *Vehicular Communications*, vol. 13, pp.56-63, 2018.
- [21] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni and N. Kaabouch, "Detection of GPS Spoofing Attacks on Unmanned Aerial Systems," *Consumer Communications and Networking Conference (CCNC)*, pp. 1-6, 2019.
- [22] J. Sengupta, S. Ruj, and S. D. Bit, "A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT." *Journal of Network and Computer Applications*, VOL. 149, pp.102481, 2020.

- [23] V. Delgado-Gomes, J. F. Martins, C. Lima, and P. N. Borza, "Smart Grid Security Issues," International Conference on Compatibility and Power Electronics (CPE), pp. 534–538, 2015.
- [24] L. Xiao, Y. Li, G. Han, G. Liu and W. Zhuang, "PHY-Layer Spoofing Detection with Reinforcement Learning in Wireless Networks," *Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10037-10047, 2016.
- [25] M. Mavani, and K. Asawa, "Modeling and Analyses of IP Spoofing Attack in 6LoWPAN Network," *Computers and Security*, vol. 70, pp. 95-110, 2017.
- [26] W. Yin, P. Hu, J. Wen, and H. Zhou, "Ack Spoofing on Mac-layer Rate Control: Attacks and Defenses," *Computer Networks*, vol.171, pp.107133, 2020.
- [27] Y. Wang, T. T. Gamage and C. H. Hauser, "Security Implications of Transport Layer Protocols in Power Grid Synchrophasor Data Communication," *Transactions on Smart Grid*, vol. 7, no. 2, pp. 807-816, 2016.
- [28] T. Lieskovan, J. Hajny and P. Cika, "Smart Grid Security: Survey and Challenges," *International Congress on UltraModern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 1-5, 2019.
- [29] G. Dileep, "A survey on smart grid technologies and applications," *Renewable Energy*, vol. 146, pp.2589-2625, 2020.
- [30] K. Tazi, F. Abdi and M. F. Abbou, "Review on Cyber-physical Security of the Smart Grid: Attacks and Defense Mechanisms," *International Renewable and Sustainable Energy Conference (IRSEC)*, pp. 1-6, 2015.
- [31] J. Kim and L. Tong, "On Topology Attack of a Smart Grid: Undetectable Attacks and Countermeasures," *Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1294-1305, 2013.
- [32] R. Oppliger, R. Hauser, and D. Basin, "SSL/TLS Session-aware User Authentication—Or How to Effectively Thwart the Man-in-the-middle," *Computer Communications*, vol. 29, no. 12, pp.2238-2246, 2006.
- [33] S. Fatima, and N. Kaabouch. "Social Engineering Attacks: A Survey." *Future Internet*, vol. 11, no. 4, pp. 89, 2019.
- [34] K. N. Ambili, and J. Jose, "Trust Based Intrusion Detection System to Detect Insider Attacks in IoT Systems," *Information Science and Applications*, pp. 631-638, 2020.
- [36] S. Wang, S. Zhu and Y. Zhang, "Blockchain-based Mutual Authentication Security Protocol for Distributed Radio Frequency Identification (RFID) Systems," *Symposium on Computers and Communications (ISCC)*, pp. 00074-00077, 2018.
- [37] H. K. Patil, D. Wing, and T. M. Chen, "VoIP Security," *Computer and Information Security Handbook*, pp. 871-886, 2013.

- [38] W. Han, and Y. Xiao, "Privacy Preservation for V2G Networks in Smart Grid: A Survey," *Computer Communications*, vol. 91, pp. 17-28, 2016.
- [39] D. Mukherjee, "Detection of data-driven blind cyber-attacks on smart grid: A deep learning approach," *Sustainable Cities and Society*, 92, p.104475, 2023.
- [40] T.O. Olowu, S. Dharmasena, A. Hernandez, and A. Sarwat, "Impact analysis of cyber-attacks on smart grid: A review and case study," *New Research Directions in Solar Energy Technologies*, pp.31-51, 2021.
- [41] M. I. Oozeer and S. Haykin, "Cognitive Risk Control for Mitigating Cyber-Attack in Smart Grid," in *IEEE Access*, vol. 7, pp. 125806-125826, 2019.
- [42] B.R. Amin, S. Taghizadeh, M. S. Rahman, M.J. Hossain, V. Varadharajan, and Z. Chen, "Cyber-attacks in smart grid—dynamic impacts, analyses and recommendations," *IET Cyber-Physical Systems: Theory & Applications*, 5(4), pp.321-329, 2020.
- [43] J. Shi, S. Liu, B. Chen and L. Yu, "Distributed Data-Driven Intrusion Detection for Sparse Stealthy FDI Attacks in Smart Grids," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 3, pp. 993-997, 2021.
- [44] Q. Liu, V. Hagenmeyer and H. B. Keller, "A Review of Rule Learning-Based Intrusion Detection Systems and Their Prospects in Smart Grids," in *IEEE Access*, vol. 9, pp. 57542-57564, 2021.
- [45] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Securing the Smart Grid: A Comprehensive Compilation of Intrusion Detection and Prevention Systems," in *IEEE Access*, vol. 7, pp. 46595-46620, 2019.
- [46] Z. El Mrabet, N. Kaabouch, H. El Ghazi, and H. El Ghazi. "Cyber-security in Smart Grid: Survey and Challenges," *Computers and Electrical Engineering*, vol. 67, pp.469-482, 2018.
- [47] W. Wang, and Z. Lu, "Cyber Security in the Smart Grid: Survey and Challenges," *Computer networks*, vol. 57, no. 5, pp.1344-1371, 2013.
- [48] C. Smutz, and A. Stavrou, "Malicious PDF Detection using Metadata and Structural Features," *Annual computer security applications conference*, pp. 239-248. 2012.
- [49] J. Wang, W. Tu, L. C. K. Hui, S. M. Yiu and E. K. Wang, "Detecting Time Synchronization Attacks in Cyber-Physical Systems with Machine Learning Techniques," *Conference on Distributed Computing Systems (ICDCS)*, pp. 2246-225, 2017.
- [50] R. A. Sowah, K. B. Ofori-Amanfo, G. A. Mills, and K. M. Koumadi, "Detection and Prevention of Man-in-the-middle Spoofing Attacks in MANETs using Predictive Techniques in Artificial Neural Networks (ANN)," *Journal of Computer Networks and Communications*, 2015.



- [51] G. Prasad, Y. Huo, L. Lampe and V. C. M. Leung, "Machine Learning Based Physical-Layer Intrusion Detection and Location for the Smart Grid," Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1-6, 2019.
- [52] A. Al-Abassi, H. Karimipour, A. Dehghantanha, and R. M. Parizi, "An Ensemble Deep Learning-Based Cyber-Attack Detection in Industrial Control System," in IEEE Access, vol. 8, pp. 83965-83973, 2020, doi: 10.1109/ACCESS.2020.2992249.
- [53] Z. Wang, H. He, Z. Wan, and Y. Sun, "Coordinated Topology Attacks in Smart Grid Using Deep Reinforcement Learning," in IEEE Transactions on Industrial Informatics, vol. 17, no. 2, pp. 1407-1415, 2021, doi: 10.1109/TII.2020.2994977.
- [54] Y. Zhang, J. Wang, and B. Chen, "Detecting False Data Injection Attacks in Smart Grids: A Semi-Supervised Deep Learning Approach," in IEEE Transactions on Smart Grid, vol. 12, no. 1, pp. 623-634, 2021, doi: 10.1109/TSG.2020.3010510.
- [55] G. F. Scaranti, L. F. Carvalho, S. Barbon and M. L. Proença, "Artificial Immune Systems and Fuzzy Logic to Detect Flooding Attacks in Software-Defined Networks," Access, vol. 8, pp. 100172-100184, 2020.
- [56] H. Reyes, N. Kaabouch, "Jamming and Lost Link Detection in Wireless Networks with Fuzzy Logic," International Journal of Scientific and Engineering Research, vol. 4, no. 2, pp. 1-7, 2013.
- [57] J. Gomez, and D. Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection," Workshop on information assurance, vol. 6, no. 3, pp. 321-323. 2002.
- [58] S. Lysenko, K. Bobrovnikova, R. Shchuka and O. Savenko, "A Cyberattacks Detection Technique Based on Evolutionary Algorithms," Conference on Dependable Systems, Services and Technologies (DESSERT), pp. 127-132, 2020.
- [59] J. Sakhnini, H. Karimipour and A. Dehghantanha, "Smart Grid Cyber Attacks Detection Using Supervised Learning and Heuristic Feature Selection," Conference on Smart Energy Grid Engineering (SEGE), pp. 108-112, 2019.
- [60] C. H. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy Rule Mining Approach and Evaluation of Feature Selection Techniques for Anomaly Intrusion Detection," Pattern Recognition, vol. 40, no. 9, pp. 2373- 2391, 2007.
- [61] R. Elhefnawy, H. Abounaser and A. Badr, "A Hybrid Nested Genetic-Fuzzy Algorithm Framework for Intrusion Detection and Attacks," Access, vol. 8, pp. 98218-98233, 2020.
- [62] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," Expert Systems with Applications, Vol. 141, p.112963, 2020.
- [63] K. Sethi, R. Kumar, N. Prajapati and P. Bera, "Deep Reinforcement Learning based Intrusion Detection System for Cloud Infrastructure," 2020 International Conference on COMMunication Systems & NETworks (COMSNETS), pp. 1-6, 2020.

- [64] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard, "Deep QLearning Based Reinforcement Learning Approach for Network Intrusion Detection," *Computers*, vol. 11, no. 3, p. 41, 2022.
- [65] Y. -F. Hsu and M. Matsuoka, "A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System," 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), pp. 1-6, 2020.
- [66] E. Suwannalai, and C. Polprasert, "Network intrusion detection systems using adversarial reinforcement learning with deep Q-network," In 2020 18th International Conference on ICT and Knowledge Engineering (ICT&KE), pp. 1-7, 2020.
- [67] A. Si-Ahmed, M.A. Al-Garadi, and N. Boustia, "Survey of Machine Learning based intrusion detection methods for Internet of Medical Things," *Applied Soft Computing*, p.110227, 2023.
- [68] V. Jain, and R. Bhatia, "A survey on machine learning schemes for fiber nonlinearity mitigation in radio over fiber system," *Journal of Optical Communications*, Vol. 0, 2023.
- [69] A. Gaurav, B. Gupta, and P.K. Panigrahi, "A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system," *Enterprise Information Systems*, 17(3), p.2023764, 2023.
- [70] Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, and J. Morimoto, "Deep learning, reinforcement learning, and world models," *Neural Networks*, 152, pp.267-275, 2022.
- [71] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," *International Carnahan Conference on Security Technology*, 2019.
- [72] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," *International conference on information systems security and privacy (ICISSP)*, pp. 407–414, 2016.
- [73] K.B. Soulami., N. Kaabouch, M. N. Saidi, and A. Tamtaoui., "Breast cancer: One-stage automated detection, segmentation, and classification of digital mammograms using UNet model based-semantic segmentation," *Biomedical Signal Processing and Control*, 66, p.102481, 2021.
- [74] K. Sharifani, and M. Amini, "Machine Learning and Deep Learning: A Review of Methods and Applications," *World Information Technology and Engineering Journal*, 10(07), pp.3897-3904, 2023.
- [75] H. S. Obaid, S. A. Dheyab and S. S. Sabry, "The Impact of Data Pre-Processing Techniques and Dimensionality Reduction on the Accuracy of Machine Learning," 2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON), pp. 279-283, 2019.

- [76] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin, "When machine learning meets privacy: A survey and outlook," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp.1-36, 2021.
- [77] A. Thakkar and R. Lohiya, "Role of the swarm and evolutionary algorithms for intrusion detection system: A survey," *Swarm Evolutionary. Computation.*, vol. 53, pp. 100631, 2020.
- [78] D. E. Hinkle, W. Wiersma, and S. G. Jurs, "Applied statistics for the behavioral sciences," Houghton Mifflin College Division, vol. 663, 2003.
- [79] S. Chesney, K. Roy, and S. Khorsandroo. "Machine Learning Algorithms for Preventing IoT Cybersecurity Attacks," *Intelligent Systems Conference*, pp. 679-686, 2020.
- [80] A. Campagner, D. Ciucci, and F. Cabitza. "Aggregation models in ensemble learning: A large-scale comparison," *Information Fusion*, 90, pp.241-252, 2023.
- [81] Y. Yang, H. Lv, and N. Chen, "A survey on ensemble learning under the era of deep learning," *Artificial Intelligence Review*, 56(6), pp.5545-5589, 2023.
- [82] J. Dong, Q. Zhang, X. Huang, Q. Tan, D. Zha, and Z. Zihao, "Active ensemble learning for knowledge graph error detection," *In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining* (pp. 877-885), 2023.
- [83] Y. Zhang, J. Liu, and W. Shen, "A review of ensemble learning algorithms used in remote sensing applications," *Applied Sciences*, 12(17), p.8654, 2023.
- [84] A. Rana, A. Dumka, R. Singh, M.K. Panda, N. Priyadarshi, and B. Twala, "Imperative role of machine learning algorithm for detection of Parkinson's disease: review, challenges and recommendations," *Diagnostics*, 12(8), p.2003, 2023.
- [85] Y. Fu, A.R. Downey, L. Yuan, T. Zhang, A. Pratt, and Y. Balogun, "Machine learning algorithms for defect detection in metal laser-based additive manufacturing: A review," *Journal of Manufacturing Processes*, 75, pp.693-710, 2022.
- [86] F.K. Alarfaj, I. Malik, H.U. Khan, N. Almusallam, M. Ramzan, and M. Ahmed, "Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms," *IEEE Access*, 10, pp.39700-39715, 2022.
- [87] W. Li, W. Tao, J. Qiu, X. Liu, X. Zhou and Z. Pan, "Densely Connected Convolutional Networks with Attention LSTM for Crowd Flows Prediction," *IEEE Access*, vol. 7, pp. 140488-140498, 2019.
- [88] K. Ibtissam, M. S. Abdelrahman, A. Alrashide and O. A. Mohammed, "Assessment of Protection Schemes and their Security under Denial-of-Service Attacks," *2022 IEEE International Conference on Environment and Electrical Engineering and 2022 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, pp. 1-6, 2022.

- [89] L. Zhou, X. Ouyang, H. Ying, L. Han, Y. Cheng, and T. Zhang, "Cyber-attack classification in smart grid via deep neural network," *Proceedings of the 2nd international conference on computer science and application engineering*, pp. 1-5, 2018.
- [90] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016.
- [91] M. Lopez-Martin, Manuel, C. Belen Carro, A. Sanchez-Esguevillas, and J. Lloret. "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," *Expert Systems with Applications* 124,196-208, 2019.
- [92] N. Elmrabit, F. Zhou, F. Li and H. Zhou, "Evaluation of Machine Learning Algorithms for Anomaly Detection," *International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp.1-8, 2020.
- [93] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, 25(1-3), pp. 18-31, 2018.
- [94] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826. 38.
- [95] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, 115, 1–37, 2014.
- [96] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 9, 1–14. 40, 2014.
- [97] K. He, X. Zhang, S. Ren, S. Jian, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1–11. 41, 2015.
- [98] K. He, X. Zhang, S. Ren, J. Sun, "Identity mappings in deep residual networks," In *Proceedings of the 2014 European Conference on Computer Vision (ECCV)*, Amsterdam, The Netherlands, 11–14 October 2015; pp. 630–645, 2021.
- [99] V. Mazzia, F. Salvetti, and M. Chiaberge, "Efficient-capsnet: Capsule network with self-attention routing," *Scientific Reports*, 11(1), pp.1-13, 2021.
- [100] W. Huang, F. and Zhou, "DA-CapsNet: dual attention mechanism capsule network," *Scientific Reports*, 10(1), pp.1-13, 2020.
- [101] D. T. Pham, S. S. Dimov, N.D. Chi, "Selection of K in K-means clustering," *Proc. Inst. Mech. Eng. Part C Journal of Mechanical Engineering Science*, 219, 103–119, 2005.
- [102] T.I. Jolliffe, C. Jorge, "Principal component analysis: A review and recent developments," *A Mathematical Physical Engineering Science*, P. 374, 2016.

- [103] S. Bock, M. Weiß, "A Proof of Local Convergence for the Adam Optimizer," In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, 2019.
- [104] F. Jafari, S. Dorafshan, "Comparison between Supervised and Unsupervised Learning for Autonomous Delamination Detection Using Impact Echo," *Remote Sensing*, 14, 6307, 2022.
- [105] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard, "Deep QLearning Based Reinforcement Learning Approach for Network Intrusion Detection," *Computers*, vol. 11, no. 3, p. 41, 2022.
- [106] Y. -F. Hsu and M. Matsuoka, "A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System," 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), pp. 1-6, 2020.
- [107] E. Suwannalai, and C. Polprasert, "Network intrusion detection systems using adversarial reinforcement learning with deep Q-network," In 2020 18th International Conference on ICT and Knowledge Engineering (ICT&KE), pp. 1-7, 2020.
- [108] K. Ren, Y. Zeng, Z. Cao, and Y. Zhang, "ID-RDRL: a deep reinforcement learning-based feature selection intrusion detection model," *Scientific Reports*, Vol. 12, No. 1, pp.1-18, 2022.
- [109] Q. Xu, K. Chen, G. Zhou, and X. Sun, "Change Capsule Network for Optical Remote Sensing Image Change Detection," *Remote Sensing*, vol. 13, no. 14, p. 2646, Jul. 2021, doi: 10.3390/rs13142646
- [110] K. Lee, H. Joe, H. Lim, K. Kim, S. Kim, C.W. Han, and Kim, H.G., 2021. Sequential routing framework: fully capsule network-based speech recognition. *Computer Speech & Language*, 70, p.101228.
- [111] P. Aggleton, A.J. Nelson, and S.M. O'Mara, "Time to retire the serial Papez circuit: Implications for space, memory, and attention." *Neuroscience & Biobehavioral Reviews*, p.104813, 2022.
- [112] Z. Li and D. Hoiem, "Learning without Forgetting," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935-2947, 1 Dec. 2018, doi: 10.1109/TPAMI.2017.2773081.