CYBER-ATTACK DETECTION OF POWER CONVERTERS IN ISLANDED MICROGRID USING DEEP LEARNING APPROACHES

Nanthaluxsan E. – 190411U
Jalini S. – 190246R
Kopikanth K. – 190320N

Bachelor of Science in Engineering

Department of Electrical Engineering

University of Moratuwa
Sri Lanka

July 2024

CYBER-ATTACK DETECTION OF POWER CONVERTERS IN ISLANDED MICROGRID USING DEEP LEARNING APPROACHES

Nanthaluxsan E. – 190411U
Jalini S. – 190246R
Kopikanth K. – 190320N

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree Honours Degree of Bachelor of the Science of Engineering

Department of Electrical Engineering

University of Moratuwa
Sri Lanka

July 2024

DECLARATION

We declare that this is our own work and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, we hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. We retain the right to use this content in whole or part in future works (such as articles or books)

Name	**Index Number**	**Signature**
Nanthaluxsan E.	190411U	
Jalini S.	190246R	
Kopikanth K.	190320N	

Date: 13.07.2024

The above candidates have carried out research for the B.Sc. Engineering undergraduate design project under my supervision. I confirm that the declaration made above by the students is true and correct.

Supervisor Name	Date	Signature
Senior. Prof Sisil Kumarawadu		
Dr. Logeeshan Velmanickam	14/7/2024	

ABSTRACT

Islanded microgrids have emerged as essential components in enhancing the sustainability and efficiency of electricity distribution in modern power systems. They play a vital role in advancing renewable energy adoption and reducing transmission losses. However, as communication and control technologies in microgrids evolve, the susceptibility to cyber threats targeting power converters has increased. This study introduces a deep learning-based model designed to detect and classify cyber-attacks, specifically focusing on False Data Injection (FDI) and Denial of Service (DOS) attacks targeting solar inverters, battery inverters, and DC to DC boost converters.

The proposed model employs deep learning techniques to analyze real-time data and identify anomalies indicative of cyber-attacks. Three distinct cases of FDI and DOS attacks are considered, providing a comprehensive evaluation of the model's robustness and versatility. To validate the effectiveness of the approach, real-time tests are conducted using MATLAB, simulating various attack scenarios over time. The results demonstrate the model's capability to accurately detect and classify different types of cyber-attacks, ensuring the reliability and security of microgrid operations.

In the commercialization phase, this cybersecurity solution is particularly targeted at high-security locations that frequently utilize islanded microgrids, such as military bases and hospitals. The planned implementation at the University of Moratuwa's microgrid serves as a pilot project, showcasing the practical application and benefits of the technology. This deep learning-based solution provides a scalable and dependable method for safeguarding the electrical infrastructure of islanded microgrids, addressing a critical need in the sector.

By commercializing this cybersecurity solution, the project aims to deliver significant value to critical environments, enhancing the protection of essential services and infrastructure. The reliable detection and classification of cyber-attacks in real-time not only mitigate potential risks but also ensure the continuous and secure operation of power systems in high-security locations. The innovative approach and demonstrated effectiveness of the model are expected to attract stakeholders in the energy sector, fostering partnerships, investments, and widespread adoption in smart grid applications.

Overall, this project addresses a pressing need for robust cybersecurity measures in modern power systems. By integrating advanced deep learning techniques with real-time data analysis, it offers a practical and scalable solution for protecting islanded microgrids from cyber threats. The successful implementation and commercialization of this technology hold the potential to revolutionize cybersecurity in the energy sector, ensuring the resilience and reliability of critical power infrastructure.

Keywords: Islanded Microgrid, power converter, cyber-attack, False Data Injection(FDI), Denial of Service(DoS), Deep learning

ACKNOWLEDGEMENT

The successful completion and outcome of this final year design project required a lot of guidance and immense support from many people. We would like to take this opportunity to express our sincere gratitude to every person who guided and supported us throughout the project.

First and foremost, we would like to extend our heartfelt gratitude to our project supervisors Senior. Prof Sisil Kumarawadu and Dr. V. Logeeshan for the enormous support and encouragement they have given us throughout the project. We are greatly indebted to them for their continued guidance and useful advices provided to us.

Next, we owe our deep gratitude to Prof. Rodrigo W.D.A.S., Prof. Lidula N. Widanagama Arachchige and Senior. Prof Sisil Kumarawadu for their invaluable advice, encouragement, and support given to us during the project evaluations. Their feedback and suggestions on the project greatly facilitated us to reach the goals that we set out to achieve.

We also wish to extend our sincere gratitude to Prof. W.D.A.S. Wijayapala, Head of the Department of Electrical Engineering, and all the academic staff members for providing their expertise in the relevant areas of our project. We would also like to thank the technical officers of the laboratories who helped us.

Finally, we would like to give our warmest thank you to our parents and all our colleagues for their invaluable comments and feedback regarding our work for the project and for helping us in making this project a success

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

VSC - Voltage Source Converters

FDI – False Data Injection

DoS – Denial of Service

ANN – Artificial Neural Network

RNN – Recurrent Neural Network

LSTM – Long Short-Term Memory

CSIs – Current Source Inverters

DC – Direct Current

AC – Alternating Current

FNN – Feedforward Neural Network

PCS – Power Conditioning System

MSE – Mean Squared Error

MAE – Mean Absolute Error

DER – Distributed Energy Resources

DWT – Discrete Wavelet Transform

FN – False Negatives

FP – False Positives

TN – True Negatives

TP – True Positives

CHAPTER 1

1. INTRODUCTION

1.1 Background

The increasing integration of renewable energy sources and the evolution of microgrid technologies have led to significant advancements in modern energy systems. Microgrids, especially islanded microgrids, are essential for providing reliable and sustainable energy solutions, particularly in remote or isolated areas. These microgrids operate independently from the main power grid, relying on local energy generation, storage, and distribution to maintain a stable electricity supply. They offer numerous benefits, including enhanced energy security, reduced transmission losses, and the ability to incorporate diverse energy sources such as solar, wind, and battery storage.

Power converters are pivotal components within islanded microgrids, facilitating the conversion and regulation of electrical energy. Key types of power converters used in these systems include Voltage Source Converters (VSCs), Current Source Inverters (CSIs), DC-DC boost converters, and battery chargers. These converters manage the flow of electricity, balance loads, and ensure optimal power quality. However, their increasing digitization and connectivity have exposed them to a range of cyber threats.

Cyber-attacks on power converters can disrupt the normal operation of islanded microgrids, leading to severe consequences such as power outages, equipment damage, and compromised energy delivery. Among the various types of cyber threats, Denial of Service (DOS) attacks and False Data Injection Attacks (FDI) are particularly concerning. DOS attacks can overwhelm control systems, rendering converters inoperative, while FDI can manipulate sensor data to mislead control algorithms, causing incorrect responses and destabilizing the microgrid.

Existing research has predominantly focused on specific power converters within DC microgrids, analyzing vulnerabilities and developing detection methods tailored to DOS or FDI attacks. However, there is a need to expand this scope to encompass AC islanded microgrids, which present unique challenges due to different operational characteristics and connectivity requirements. AC microgrids often involve more complex interactions between power converters and other components, making them susceptible to a broader range of cyber-attacks.

Deep learning models represent a promising approach to detecting and mitigating cyber threats in power converters, enhancing the security and reliability of islanded microgrids. Deep learning, a subset of machine learning, uses neural networks to learn intricate patterns and correlations from data. These models have shown significant potential in various fields, including image recognition, natural language processing, and anomaly detection. By training deep learning models on extensive datasets simulating various cyber-attack scenarios, it is possible to develop algorithms capable of swiftly identifying anomalous behaviors indicative of DOS or FDI attacks.

This project aims to develop a robust cyber-attack detection system for power converters within AC islanded microgrids using deep learning. By creating a simulation model representing the University of Moratuwa (UOM) in MATLAB, the project will generate healthy and attack data to train the deep learning models. The mathematical models will represent various types of cyber-attacks, ensuring realistic simulation scenarios. The goal is to build a deep learning model that accurately detects and classifies different types of attacks, enhancing the security and resilience of islanded microgrid infrastructures.

This project highlights the critical importance of safeguarding power converters in islanded microgrids from cyber threats. By expanding the focus to include AC microgrids and leveraging deep learning approaches, the project seeks to contribute significant insights and methodologies to bolster the security and reliability of future energy systems. Through rigorous experimentation and validation, the project aims to establish a framework that detects and effectively mitigates cyber threats, ensuring sustained energy delivery and system integrity.

1.2 Motivation and Problem Statement

In the realm of modern energy systems, the vulnerability of power converters to cyber threats has emerged as a critical issue, particularly in islanded microgrid environments. This project seeks to address these concerns by developing an islanded microgrid system fortified against cyber threats through the application of deep learning models. The ultimate goal is to ensure the security and reliability of microgrid operations amidst potential cyber-attacks targeting various power converters.

Power converters such as Voltage Source Converters (VSCs), Current Source Inverters (CSIs), DC to DC boost converters, and battery chargers are integral components of islanded microgrids. However, their interconnected nature and reliance on digital control systems make them susceptible to cyber-attacks. Common types of cyber threats in microgrid contexts include Denial of Service (DOS) attacks and False Data Injection Attacks (FDI), which can compromise the stability and functionality of the microgrid.

Existing research has predominantly focused on specific power converters within DC microgrids, analyzing their vulnerabilities and developing detection methods tailored to DOS or FDI attacks. However, this project aims to expand this scope by investigating the broader impact of cyber-attacks across a variety of power converters within AC islanded microgrids. This broader approach is essential as AC microgrids present unique challenges due to their different operational characteristics and connectivity requirements compared to DC systems.

To achieve robust cyber-attack detection and mitigation, the project proposes leveraging deep learning models. Deep learning, a subset of machine learning that utilizes neural networks to learn intricate patterns and correlations from data, has shown promise in enhancing the precision of cyber threat detection. By training deep learning models on extensive datasets that simulate various cyber-attack scenarios,

researchers aim to develop algorithms capable of swiftly identifying anomalous behaviors indicative of DOS or FDI attacks.

Furthermore, the project intends to explore the combined effects of DOS and FDI attacks on multiple interconnected power converters within the microgrid. Understanding these combined effects is crucial for devising comprehensive security measures that can preemptively mitigate potential cascading failures or disruptions across the microgrid.

This represents a proactive approach to safeguarding islanded microgrid operations against cyber threats by harnessing the capabilities of deep learning. By broadening the focus to encompass AC microgrids and investigating the interplay of different types of cyber-attacks, researchers aim to contribute significant insights and methodologies to enhance the security and resilience of future energy systems. Through rigorous experimentation and validation, the project endeavors to establish a framework that not only detects but also effectively mitigates cyber threats, thereby bolstering the overall stability and reliability of islanded microgrid infrastructures.

1.3 Aim of the Project

The aim of this project is to develop and implement advanced deep learning approaches for cyber-attack detection in power converters within islanded microgrids. By leveraging deep learning models, the objective is to enhance the security and reliability of microgrid operations by accurately identifying and mitigating potential cyber threats such as Denial of Service (DOS) and False Data Injection Attacks (FDI). The project will focus on integrating sophisticated neural network architectures with comprehensive datasets to build a robust cyber-attack detection framework. Ultimately, this endeavor seeks to fortify islanded microgrid infrastructures against evolving cyber threats, ensuring sustained energy delivery and system integrity.

1.4 Objectives of the Project

- Create a simulation model which represents UOM microgrid in MATLAB.
- Build up a mathematical model to represent the attack in power converters of microgrid.
- Generate healthy data and attack data using the implemented simulation model and the mathematical model.
- Build up a deep learning model to accurately detect and classify attack types.

1.5 Structure of the Report

The report is structured into nine chapters, starting with introductory sections and progressing through detailed technical discussions to conclusions and recommendations. Chapter 1 introduces the project, including background, motivation, and objectives. Chapter 2 reviews relevant literature, identifying research

gaps. Chapter 3 explores microgrids, focusing on islanded microgrids, their benefits, applications, and future prospects, along with the design of the microgrid model in MATLAB. Chapter 4 discusses cyber-attacks, past incidents, and classifications, along with modeling specific attack scenarios. Chapter 5 covers deep learning approaches, including data preprocessing and various neural network models for attack detection. Chapter 6 details the methodology used in the project. Chapter 7 presents the results of neural network models, comparing their performances. Chapter 8 validates the deep learning approaches and considers the project's commercialization potential. Finally, Chapter 9 provides conclusions and recommendations, followed by references. The report also includes a declaration, abstract, acknowledgements, lists of figures, tables, and abbreviations.

CHAPTER 2

2. LITERATURE REVIEW

2.1 Related Literature

In cyber security, methods for detecting and classifying attacks fall into two main categories: model-based and data-driven approaches. Model-based approaches evaluate system behavior against predefined models to ensure variables align with expected norms. It is focusing on confirming model coherence. Data-driven approaches analyze system data to identify abnormalities by exploring relationships between input and output state variables. These methods can detect threats that model-based methods might miss due to their limited system knowledge. Artificial neural networks are part of the data-driven category, using data to uncover vulnerabilities and respond to cyber threats. The literature highlights the strengths and limitations of both approaches, showing how they complement each other in enhancing cyber security.

A novel model-based cyber-attack detection method is proposed, focusing mainly on false data injection (FDI) attacks. The Harmonics State Space Matrix (H-Matrix) is used to customize the closed-loop transfer function. This approach provides model-based estimation using the grid voltage and control reference. The Space Phase Model (SPM) is then used to determine voltage residuals to detect cyberattacks in VSC. These detection methods are mostly based on voltage variations. However, this research does not address other types of Denial of Service (DOS) attacks and other power converters' controllers.[2]

The proposed system is novel in its approach to evaluating the stability and performance of individual inverters and the grid within a smart grid environment against cyberattacks. It used both total harmonic distortion (THD) and the space phasor model (SPM) to evaluate inverter performance when cyber-attacks happen. The system design also includes stability-based and metric-based criteria for an effective evaluation. Mathematical models are provided to create false data injection attack scenarios. This paper focused on FDI attacks on inverters. However, this paper does not focus on detection mechanisms, it limits the analysis to only two inverters and exclusively considers FDI attacks.[5]

The paper mainly focused on a cyber-resilient control strategy for microgrids (MGs) when a system is attacked by hybrid false data injection and denial-of-service attacks. islanded systems with distributed generators (DGs) and energy storage systems (ESSs) are designed. It proposed an approach that uses adaptive methods to handle bounded FDI attacks on secondary controllers at the same time managing the DoS attacks on communication links. dwell time and Lyapunov stability theories are used to prove the stability of the control method. Tested in a 13-bus MG system with real-time simulation, the method effectively maintains frequency restoration, power sharing, and energy balance despite hybrid cyber-attacks. The model represents a mathematical system for FDI and DOS which attack controllers in battery systems and distributed generators. The paper assumed that attackers have limited energy and that FDI attacks are bounded, and it is happening within specific time intervals for voltage source

converters (VSC) and current source inverters (CSI). However, this model also does not focus on cyber-attack detection and classification. [11]

This paper focused on the vulnerabilities of distributed secondary control methods in AC microgrids against cyber-attacks. It evaluates three control methods-distributed consensus, average distributed control, and distributed finite-time control three types of attacks: false data injection (FDI), denial-of-service (DoS), and controller hijacking. Using MATLAB simulations, the study assesses each method's resilience. Findings reveal that the average control method performs well against DoS and FDI attacks. controller hijacking causes significant damage. The paper focuses on comparing the effectiveness of these control methods in safeguarding against various cyber threats in microgrids, but it mainly focuses on the performance of secondary control against cyber-attacks. [7]

This paper focuses on detecting False Data Injection (FDI) attacks using a combination of machine learning and deep learning approaches. It highlights that the Convolutional Neural Network (CNN) model achieves high accuracy in detecting FDI attacks with score. They used the Micro PMU data to detect the attack. The study is specifically designed for DC-to-DC boost converters and DC-to-AC converter controllers. It does not address Denial-of-Service (DoS) attacks or battery inverter systems, and it limits its scope to detection rather than classification. [3]

The paper examines Denial-of-Service (DoS) attacks on microgrids. it is focusing on their impact on service-provider-edge routers. It introduces consensus-based secondary frequency controllers with dynamic P−f droop controllers to enhance microgrid tolerance to DoS attacks. A state-space model of the microgrid is designed to analyze stability under DoS attacks. It uses free-weighted matrices and Linear Matrix Inequality (LMI) tools to analyze the stability. The effectiveness of this approach is validated through simulations and experiments. a potential gap is extending the results to data-driven communication and studying other cyber-attacks (False Data Injection attacks).[14]

2.2 Research Gap

Previous papers typically focus on a single power converter, one type of attack, and only detection and stability performance. In contrast, our project examines three types of power converters and two types of attacks (FDI and DOS), incorporating both detection and classification using deep learning. This broader approach enhances the understanding and capability for detecting and classifying cyberattacks in islanded microgrid systems, providing a more comprehensive and robust solution.

CHAPTER 3

3. MICROGRID

A microgrid is a small collection of electrical sources and loads that can run independently of the larger, traditional centralized grid when necessary due to environmental or financial reasons. Microgrids are often connected to and synchronized with the main grid. In recent years, the global energy sector has undergone an overhaul toward decentralization due to the critical need for resilience, sustainability, and energy access. Microgrids, which are localized and self-sufficient electricity distribution networks, have emerged as a game-changing answer to these issues.

It functions both independently and in tandem with the main electrical grid, effectively being a scaled-down replica of the bigger system. Various energy sources, such as solar panels, wind turbines, generators, batteries, and other energy storage devices, can be incorporated into microgrids. We use the Moratuwa University microgrid as a model for our project and use the university-selected area as a load. The sources of power for our microgrid include solar panels and batteries.

3.1 Islanded Microgrid

An islanded microgrid refers to a microgrid that is operating independently from the main grid. This means it is "islanded" or isolated from the traditional grid infrastructure. A remote village uses solar panels and a wind turbine with battery storage to supply its power needs. It is not connected to the main grid due to geographical constraints and operates entirely independently. Although the UOM microgrid is grid-connected, we are using it as a reference to model an islanded microgrid. Our project also assesses the feasibility and value of operating a microgrid in islanded mode.[20]

3.2 Benefits of Islanded Microgrid

Microgrids offer a compelling answer to the global energy crisis by reducing transmission and distribution losses, promoting renewable energy sources, cutting carbon emissions, reducing costs, and minimizing large land use.
microgrid is a traditional Power Systems. In traditional centralized power systems, electricity is generated at large power plants and then transmitted over long distances to consumers. This process incurs energy losses at each stage, especially over long transmission lines. By generating power locally by microgrid, it significantly reduces the distance electricity needs to travel, thereby minimizing energy losses during transmission and distribution.

Microgrids are well-suited to incorporate various renewable energy sources like solar panels, wind turbines, and biomass. This integration is often more manageable at a smaller, local scale. By promoting the use of renewable energy, microgrids help reduce reliance on fossil fuels, contributing to a more sustainable energy system. As

microgrids utilize more renewable energy sources, they emit fewer greenhouse gases compared to traditional power systems that rely heavily on fossil fuels. Reduced transmission losses also mean that less energy is wasted, leading to a more efficient and environmentally friendly energy supply.

Microgrids typically require less land than large, centralized power plants and extensive transmission networks. They can be implemented in smaller, distributed sites such as rooftops or community spaces, making better use of available land and reducing the environmental impact of large-scale infrastructure. While microgrids provide these substantial benefits, their reliance on centralized cloud environments for monitoring and control introduces a vulnerability to cyber-attacks. The interconnected nature of these systems means that a successful cyber-attack could disrupt the microgrid's operations, leading to potential outages, data breaches, or even damage to the physical infrastructure. Therefore, ensuring stable cybersecurity measures is crucial to protect these advanced energy systems.

3.3 Applications of Islanded Microgrid

3.3.1 remote and rural areas

Providing power to remote villages, islands, and rural areas that are not connected to the main grid is difficult due to the necessity for long-distance transmission lines. Increasing the standard of living by powering homes, schools, clinics, and businesses.

3.3.2 Military base and defense operations

Microgrids are crucial to military operations as they provide military bases and installations with a secure and independent power source. Their ability to operate autonomously enhances mission-critical activities and mitigates risks associated with power outages. Additionally, microgrids contribute to resilience planning and serve as testbeds for advanced energy technologies in military applications.

3.3.3 Telecommunications

Supplying power to remote communication towers and data centers, ensuring uninterrupted connectivity. Supporting communication infrastructure during disasters for better coordination and recovery efforts.

3.3.4 Agricultural Applications

In remote farming areas, microgrids play a critical role by powering irrigation systems, greenhouses, and processing facilities. These systems ensure consistent electricity supply, crucial for maintaining crop yields and farm operations in areas where grid connectivity may be unreliable or nonexistent.

Microgrids also support sustainable agricultural practices by leveraging renewable energy sources. By integrating solar, wind, or biomass energy into farm operations, microgrids help reduce reliance on fossil fuels and mitigate environmental impacts. This sustainable approach not only enhances energy efficiency but also contributes to long-term environmental stewardship in agricultural communities.

3.3.5 Mining Operations

Providing power in distant places where grid connections are not viable. Providing continuous and reliable power for important mining operations.

3.3.6 Islands and Coastal Communities

For isolated islands and coastal areas without access to the mainland grid, microgrids offer a sustainable and dependable power solution. By generating electricity locally from renewable sources like solar and wind, microgrids ensure continuous energy supply, enhancing resilience against disruptions.

Microgrids also play a crucial role in supporting tourism by powering resorts and facilities, thereby improving visitor experiences. Reliable electricity enables hospitality services to operate smoothly, ensuring comfort and convenience for tourists exploring these scenic destinations.

3.4 Future of microgrids

While microgrids offer significant benefits, their widespread adoption faces barriers such as standardization issues, integration challenges, regulatory hurdles, and cost constraints. However, ongoing research, technological advancements, and regulatory support are steadily addressing these obstacles.

Microgrids are poised for a promising future, supported by increasing investments and partnerships driving innovation. Enhancements in performance and flexibility are expected through advancements in control systems, energy storage technologies, and digitalization. Furthermore, emerging technologies like peer-to-peer energy trading platforms and blockchain could revolutionize microgrid operations, facilitating more efficient energy transactions and promoting energy democratization.

Microgrids are transforming the energy sector by providing reliable, sustainable, and localized electricity solutions. Microgrids enable a more decentralized and sustainable energy future by incorporating renewable energy sources, increasing grid reliability, and optimizing energy use. Microgrids can lead to sustainable and community-focused energy systems, while also addressing global challenges such as climate change and energy availability.

3.5 Our microgrid model designed in MATLAB

3.5.1 microgrid model design showing load in UOM

Our microgrid design draws inspiration from the UOM microgrid, utilizing its load profiles for various facilities.

Our design includes:

- A new administration building requiring 100 kW of power.
- A canteen with a power demand of 50 kW.
- The Sumanadasa building, which has a load requirement of 200 kW.

These specific load considerations are essential for optimizing the microgrid's generation, storage, and distribution capabilities. By modeling these requirements, our goal is to develop an efficient and sustainable energy system that meets the specific needs of each facility within our microgrid framework, ensuring reliable and resilient power supply for operational continuity.

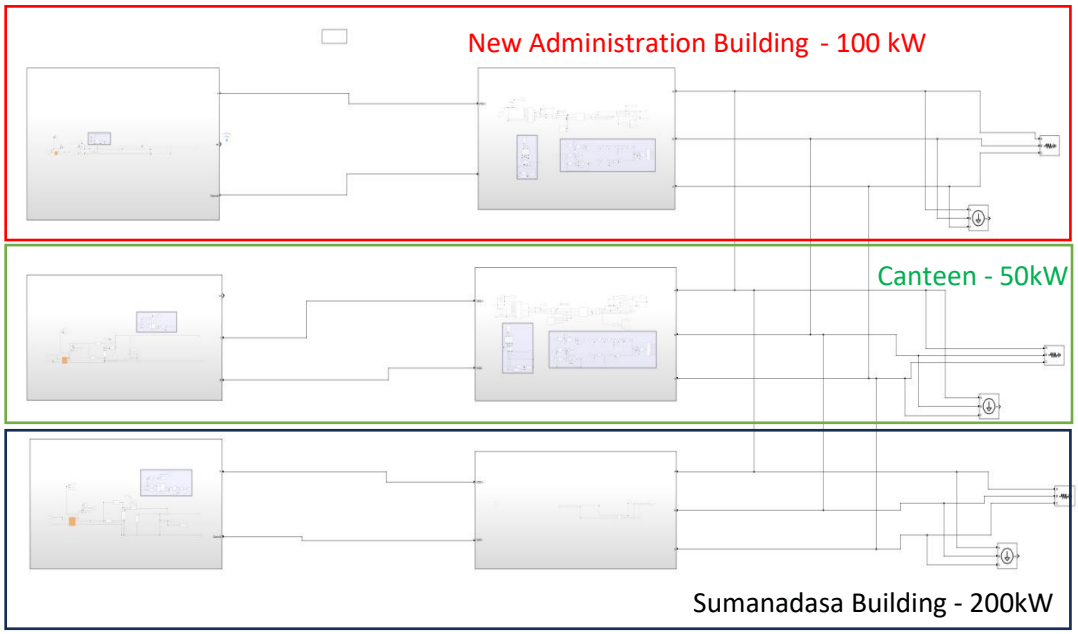


Figure 1: Microgrid model design showing load in UOM

3.5.2 microgrid model design showing power converters

We can divide this design into two sections:

- In one section, we incorporate a solar inverter with a DC to DC boost converter.

- In the other section, we integrate a battery inverter with a DC to AC converter.

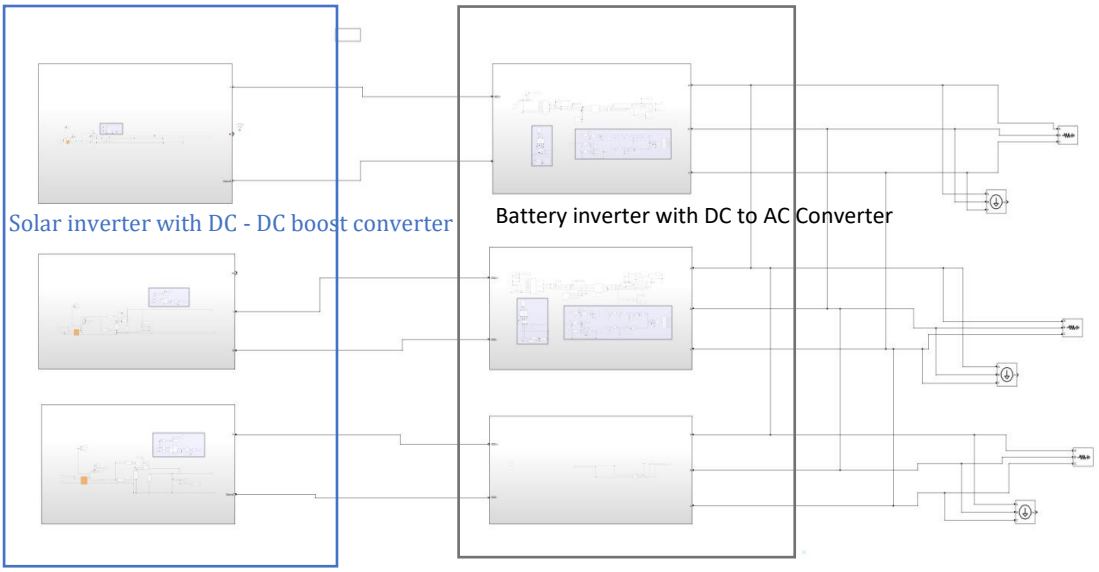


Figure 2: Microgrid model design showing power converters

Our project focuses on exploring cyber-attacks targeting power converters, specifically solar inverters, battery inverters, and DC to DC Boost Converters.

3.5.2.1 Solar inverters

The Solar PV inverters play a pivotal role in converting DC power generated by Solar PV modules into AC power for use within the system. Each inverter is equipped with multiple DC inputs to accommodate individual DC strings, optimizing the system's overall efficiency through maximum power point tracking. These DC strings are strategically configured to group solar modules of similar roof orientation and inclination, ensuring optimal performance. Detailed string layouts and PVsyst reports are included in the project documentation to facilitate precise planning and installation.

Each PV inverter features a central AC output connected to PV AC combiner panels, which then distribute power to individual buildings. Communication between the inverters and the DHYBRID Universal Power Platform Interfaces is established via RS485 communication lines, enabling comprehensive monitoring and control at the DHYBRID SCADA level. The DHYBRID UPP regulates PV inverter operations by dynamically sending power setpoints for active and reactive power. This coordination ensures efficient energy utilization, particularly in islanded mode scenarios where careful management may be required to prevent battery overcharging.

3.5.2.2 DC to DC Boost Converters

A DC-to-DC boost converter is a type of power electronic device that efficiently increases the input DC voltage to a higher output DC voltage level. This conversion process is crucial in applications where the input voltage from a power source, such as a battery or solar panel, is lower than the required voltage for the load or subsequent power conversion stages. The boost converter operates by switching an internal semiconductor switch (typically a transistor) at high frequencies, storing energy in an inductor during the switch-on period, and releasing it to the load during the switch-off period. This technology enables the efficient management and utilization of DC power across various electronic and energy systems, contributing to improved overall system performance and energy efficiency.

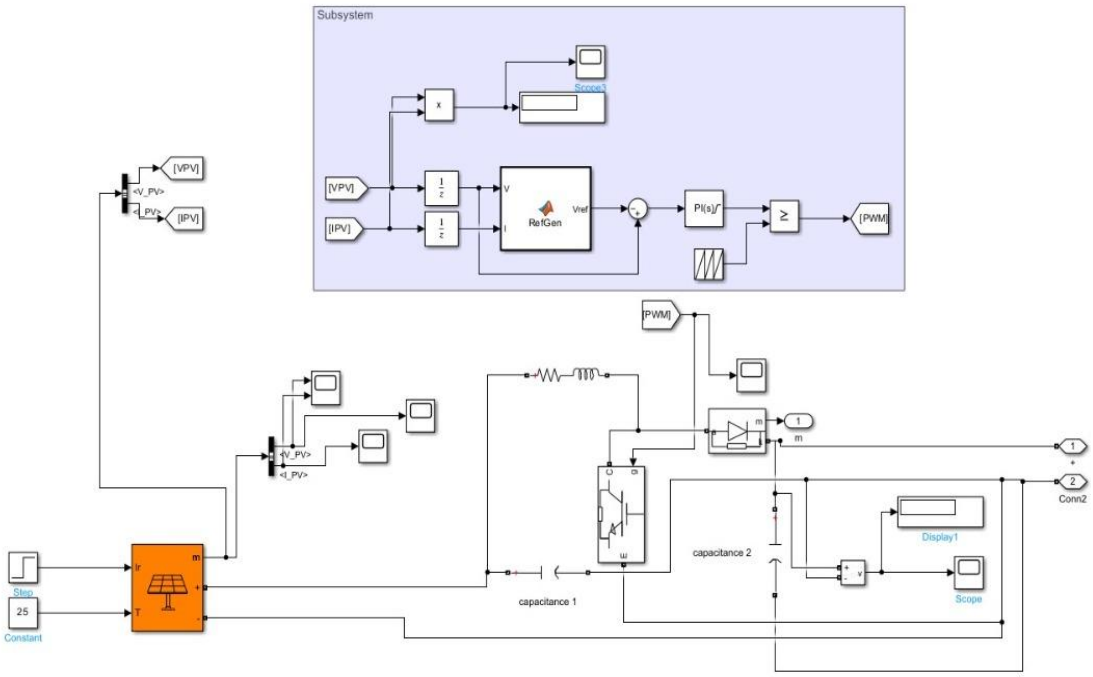


Figure 3: Solar inverter with DC to DC boost converter

3.5.2.3 Battery Inverters

The battery inverter, also known as the Power Conversion System (PCS), serves a crucial role in islanded microgrids by converting DC power from batteries into AC power and vice versa. It operates bidirectionally across four quadrants to provide both active and reactive power. In grid-connected mode, the battery inverter synchronizes with the public grid, maintaining voltage and frequency as dictated by grid conditions. In off-grid mode, it functions akin to a virtual synchronous generator, controlling AC voltage and frequency through droop control. During high load periods, the inverter lowers output frequency, while increased photovoltaic (PV) production raises it when charging batteries. Reactive power management adjusts voltage levels accordingly.

Communication with the DHYBRID Universal Power Platform Main Controller occurs via Ethernet, facilitating real-time monitoring and control. The system displays battery inverter data at the DHYBRID SCADA level and regulates grid-connected operation through dynamic power setpoints for active and reactive power. This integration ensures adaptive power output, optimizing battery charging and discharging operations based on grid demand and renewable energy availability.

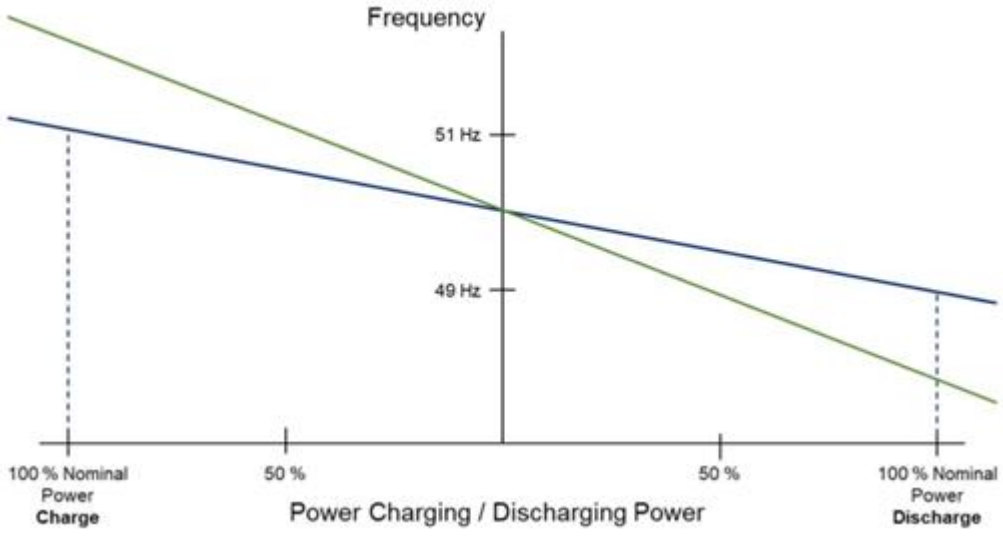


Figure 4: Power charging / Discharging power

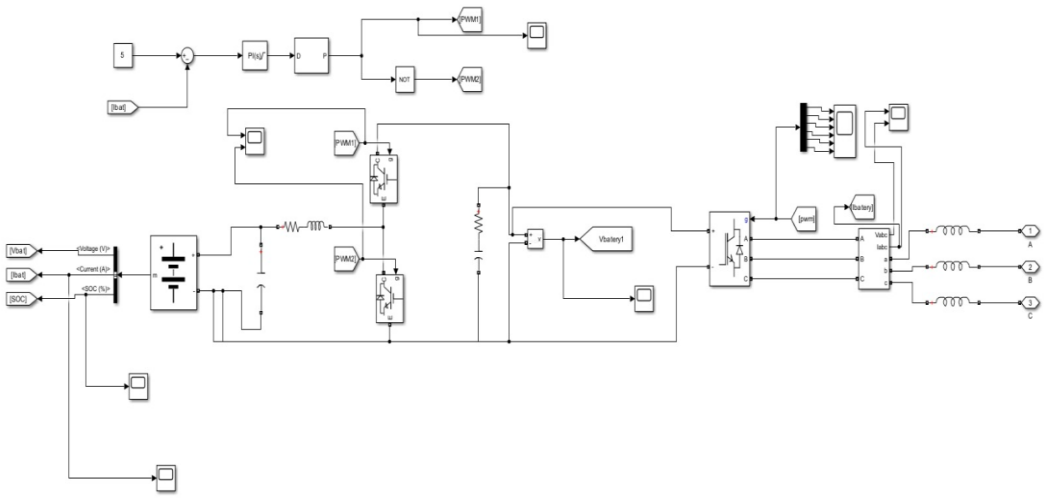


Figure 5: Designed Battery inverter in MATLAB

3.5.2.4 DC to AC converter

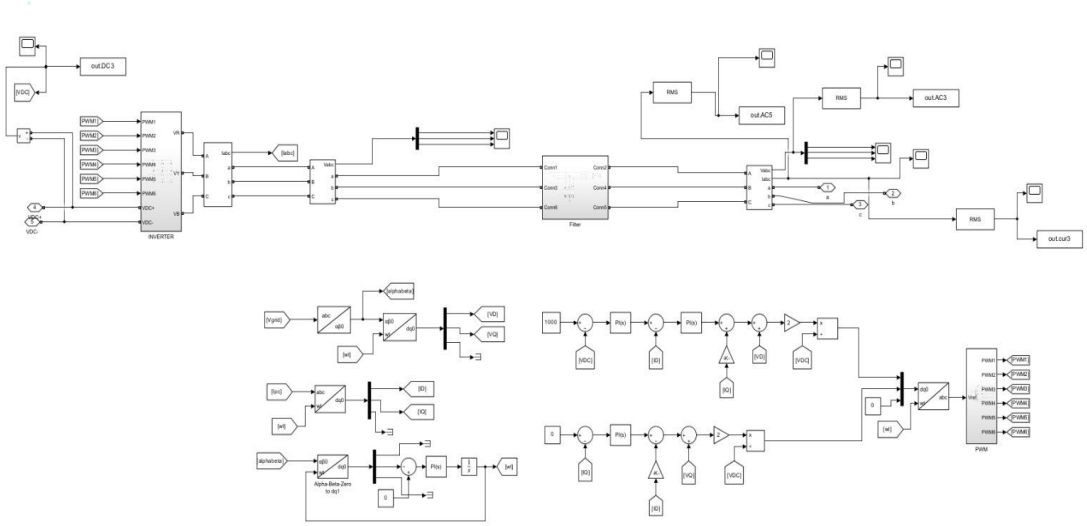


Figure 6: DC to AC Converter

CHAPTER 4

4. CYBER-ATTACKS

Cyber-attacks are deliberate activities taken by people or groups to compromise or disrupt computer systems, networks, or digital devices. These assaults seek to obtain unauthorized access to sensitive information, modify data, disrupt operations, or harm digital infrastructure. Malware infections, phishing attempts, denial-of-service (DoS) attacks, ransomware, and the exploitation of software or hardware vulnerabilities are all possible types of cyber-attacks. They pose enormous risks to enterprises, governments, organizations, and individuals, jeopardizing financial stability, privacy, and operational continuity. Effective cybersecurity measures are critical for reducing the dangers presented by cyber-attacks and protecting digital assets and information.

4.1 Cyber-attacks in microgrid

Cyber-attacks are a significant threat across many areas, especially those handling highly confidential information. Microgrids, which are increasingly used in such sensitive areas to provide reliable electricity, are not exempt from these threats. Cyber-attacks on microgrids have unique characteristics, as they can specifically target power systems, potentially disrupting energy generation, distribution, and storage. Safeguarding microgrids against these attacks is crucial to ensure the security and reliability of the power supply in these critical environments.

Despite advantages of microgrid, the centralization of monitoring and control systems in a cloud environment heightens the risk of cyber-attacks, presenting a significant challenge that must be addressed.

4.2 past incidents of cyber-attacks in power system

In recent decades, numerous cyber-attacks have targeted the energy sector, resulting in diverse and significant impacts at various levels. For instance, the Mumbai power outage investigation revealed that Trojan horses were used to hack into the city's electrical infrastructure, disrupting the power supply. This incident highlighted the vulnerability of urban energy systems to cyber threats and the potential for widespread disruption in densely populated areas.

In 2015, a wide blackout in Kyiv, Ukraine, occurred for several hours due to a cyber-attack employing the Darkside malware. This attack demonstrated the potential for cyber threats to cause extensive damage to national infrastructure, leaving large populations without power and highlighting the need for international cooperation and advanced cybersecurity defenses. Similarly, on April 11, 2021, the Natanz nuclear facility in Iran was attacked, with the insertion of the Stuxnet virus causing extensive damage. This incident underscored the severe consequences that cyber-attacks can have on critical infrastructure and national security.

Additionally, on May 7, 2021, a ransomware cyber-attack on oil resources halted the operation of oil pipelines across nearly 17 states in the USA. This attack affected three major distribution companies and more than 225,000 customers, demonstrating the far-reaching economic and logistical impacts of cyber-attacks on energy supply chains. These incidents underscore the critical need for robust cybersecurity measures in the energy sector to protect against the growing threat of cyber-attacks and ensure the reliability and security of essential services

4.3 cyber-attacks classification in microgrid.

Cyber-attacks on microgrid data flow can be classified into three attacks. They are attacks compromising availability, integrity, and confidentiality.

4.3.1 Attacks on Data Integrity

Any attack that undermines data integrity alters the information transmitted within the cyber system. These attacks can corrupt measurements or command signals within the communication network, leading to malfunctions in the microgrid and affecting its control systems. This includes the regulation of frequency and voltage, power and energy management, islanding detection, and resynchronization processes. A notable example of an attack compromising data integrity is the False Data Injection (FDI) attack. FDI attacks are among the most challenging threats for microgrids, and their impacts on modern power grids can be severe and unacceptable.[4]

FDI attack Mathematical model

$$Y_F(t) = \alpha * Y_O(t) + \beta$$

- $\alpha \neq 0$, $\beta = 0$, this attack manipulates the real measurement by α
- $\alpha = 0$, $\beta \neq 0$, this attack replaces the real measurement by β
- $\alpha = 1$, $\beta = 0$, there is no attack in controller.

From this model we classified FDI attack in our project as 3 cases.

1. FDI Case1 - $\alpha \neq 1$, $\beta > 0$
2. FDI Case2 - $\alpha \neq 1$, $\beta = 0$
3. FDI Case3 – $\alpha = 1$, $\beta > 0$

4.3.2 Attacks on Data Availability

The cybersecurity system must ensure timely and accessible data, which is essential for controlling power electronics converters in smart microgrids, particularly in islanded mode and during transient events. Attacks aimed at obstructing or delaying data communications are known as attacks on data availability. Examples of such attacks include denial of service (DoS) and distributed denial of service (DDoS), both of which can significantly impact the reliability and performance of microgrid operations.[8]

DOS attack mathematical model

$$Y_F(t) = \alpha * Y_O(t) + \beta$$

- $\alpha = 0$, $\beta = 0$, this attack denied service.

Attackers can employ DoS attacks targeting the communication links while FDI attack targeting controller.

4.3.3 Attacks on Data Confidentiality

Data confidentiality ensures that information is safeguarded against access and understanding by unauthorized entities. Cyber-attacks that breach confidentiality enable hackers to eavesdrop on the communication network, gaining access to sensitive information about customers and microgrid operation and control strategies. While these attacks may not immediately disrupt microgrid operations, they pose significant privacy and security risks.[17]

In our project, we employ False Data Injection (FDI) attacks in three distinct cases, along with Denial of Service (DoS) attacks, which has more impact on microgrid and attacker commonly willing to use these types.

4.4 Overall Cyber-attack model

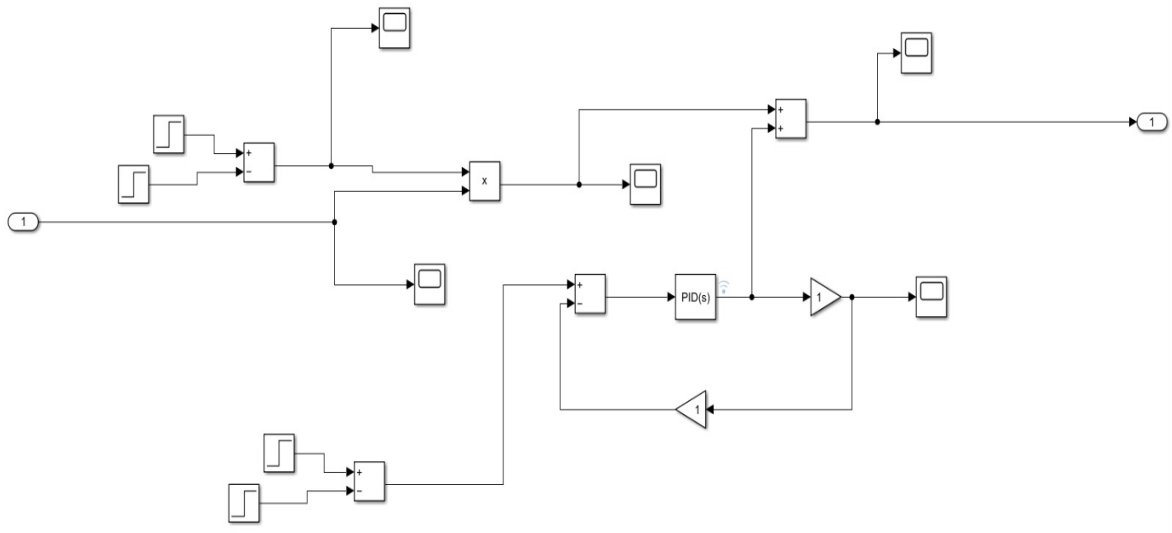


Figure 7: Cyber-attack model

In our project, we investigate cyber-attacks on power converters, focusing on identifying potential vulnerability areas within power converters. This includes exploring the possibilities and implications of cyber-attacks on these critical components. Attack area can be multiple choice in the power converters. It can be satisfied by matrix equation.

$$Y = \alpha X + \beta$$

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

4.4.1 solar inverter with DC-to-DC Boost Converter.

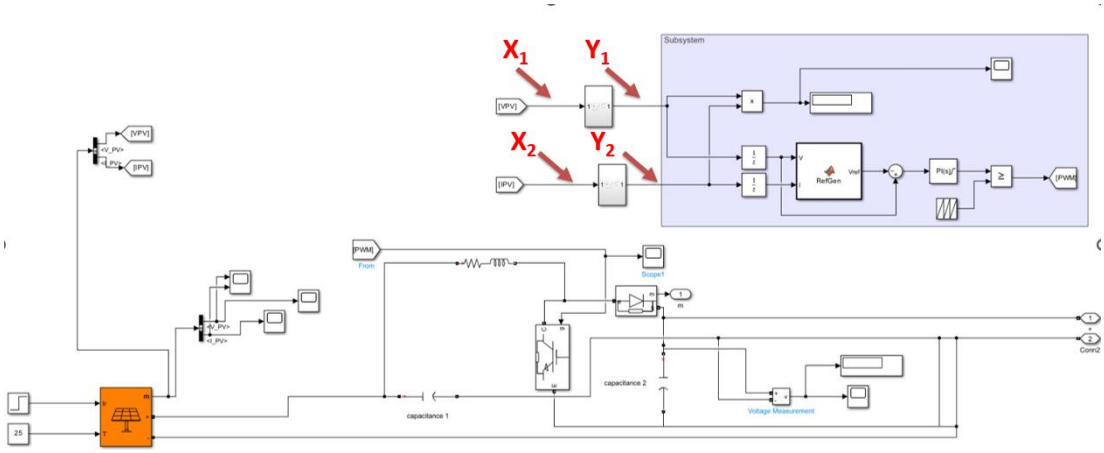


Figure 8: Solar inverter with DC - DC boost converter with Attack

We can vary various numerical values in matrix for alpha (a_{11}, a_{22}) and beta (b_1, b_2). In our project, we aim to achieve maximum accuracy in detecting and classifying cyber-attacks on power converters while maintaining minimal threshold values. These threshold values are determined based on the strategies and objectives of potential cyber attackers, as well as their anticipated impact on the system.

4.4.1.1 FDI Case1

For this case, we have $\alpha \neq 1$ and $\beta > 0$. To achieve the maximum accuracy of our model, we set the minimum values for α and β, specifically $\alpha = 2$ and $\beta = 200$. This allows us to observe how the attack situation differs from the healthy situation, where "healthy situation" refers to the absence of any attacks in the system.

We consider that in all our scenarios, the cyber-attack occurs within 2 to 4 seconds. We observe how it impacts the system and how the values differ before, during, and after the attack.

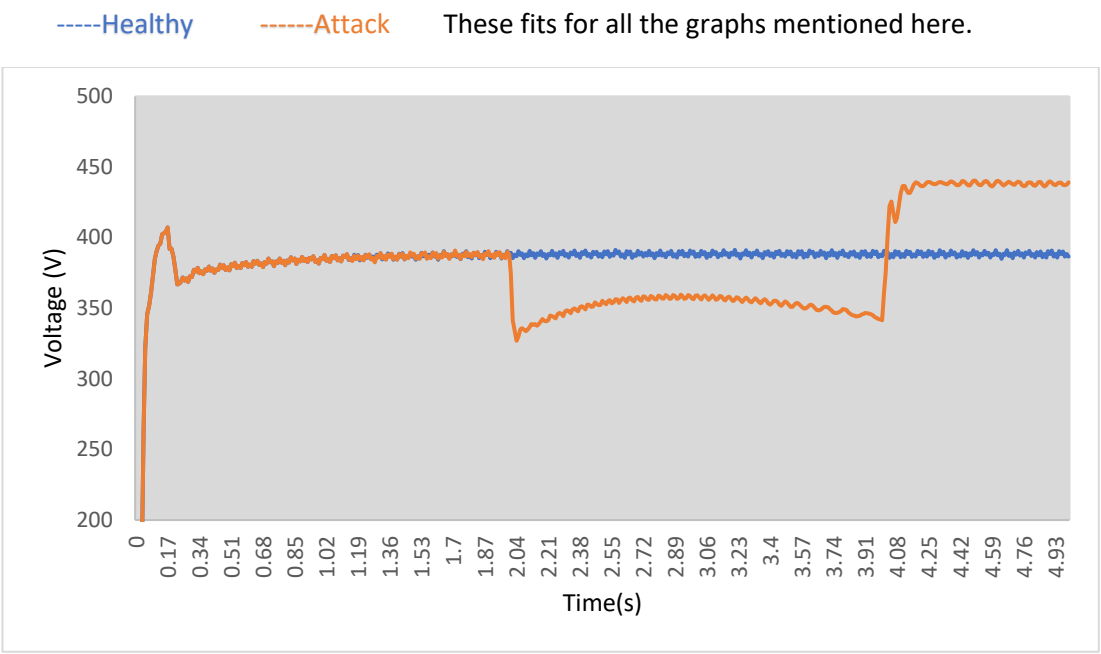


Figure 9: AC Voltage -Solar inverter- α=2 β=200

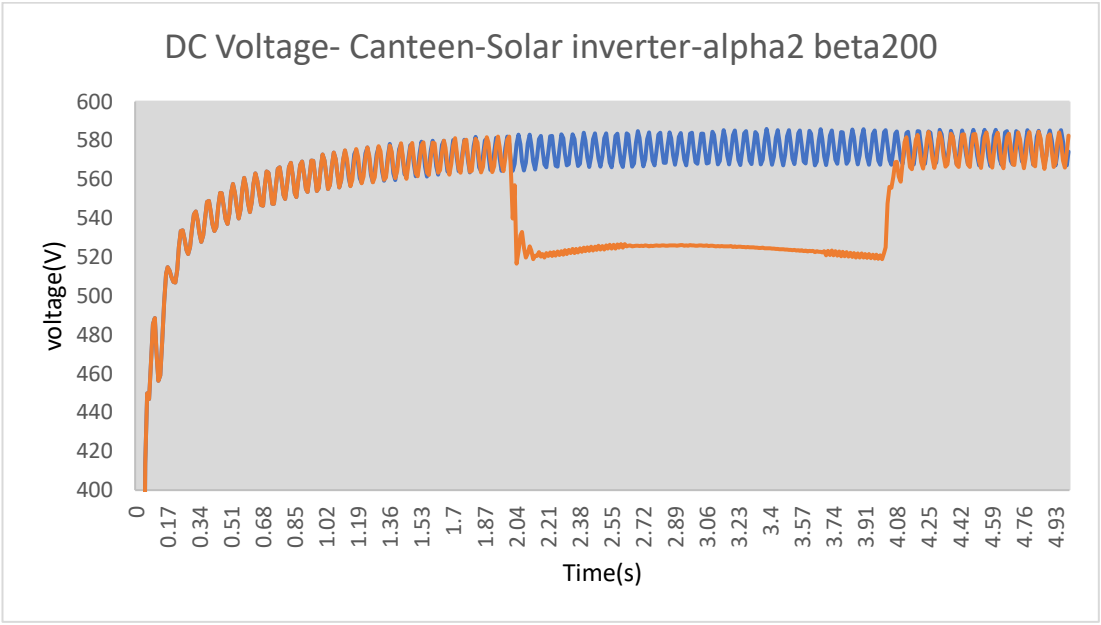


Figure 10: DC Voltage- Canteen-Solar inverter- α=2 β=200

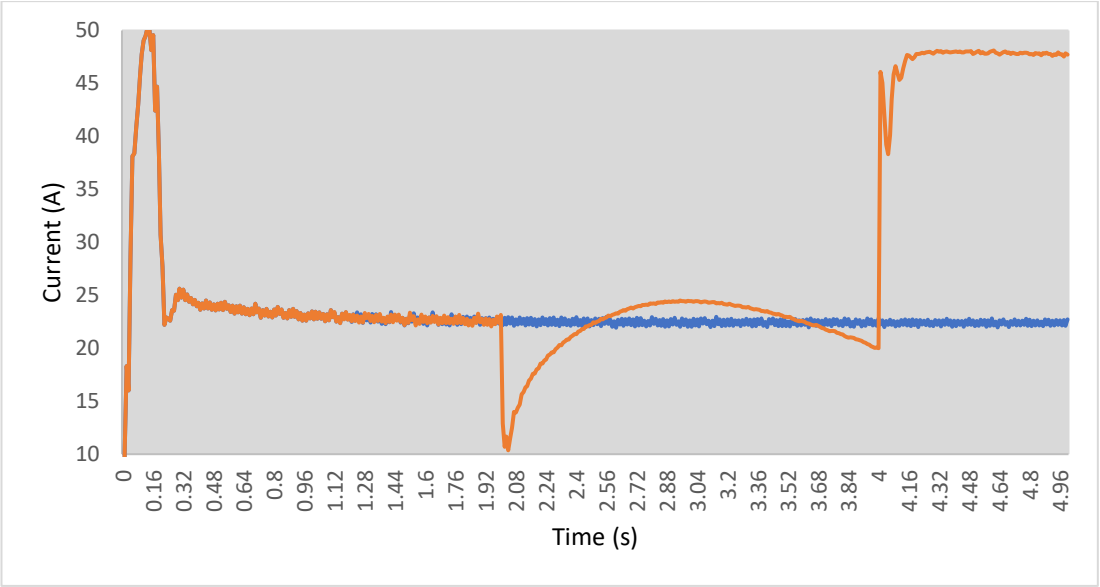


Figure 11: Current-Sumanadasa building-Solar inverter- α=2 β=200

When we increase the values of α and β to 5 and 500 respectively, we observe a significant difference. This indicates that the impact of this cyber-attack on the microgrid is substantially higher.

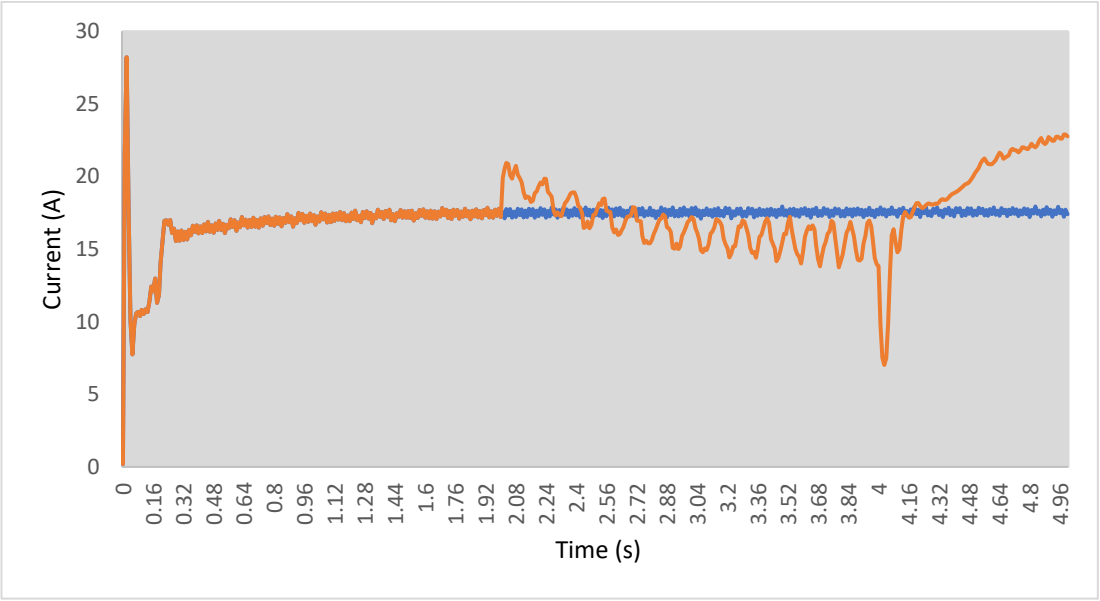


Figure 12: Current - New Admin Building- α=5 β=500

4.4.1.2 FDI Case2

For this case, we have α > 1 and β = 0. To achieve the maximum accuracy of our model, we set the minimum value for α, specifically α = 2 and β = 0.

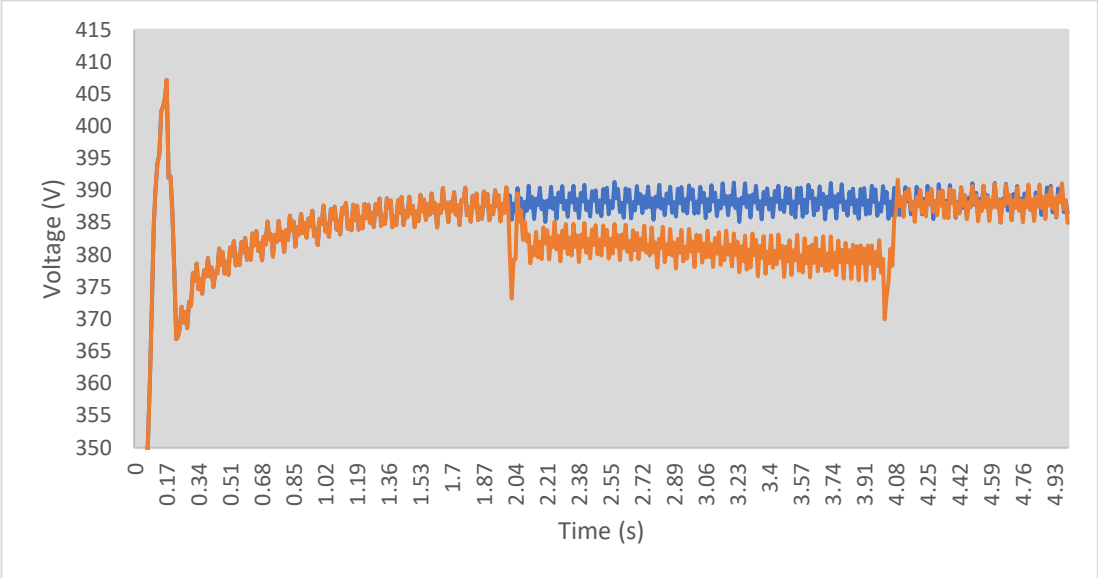


Figure 13: AC Voltage -solar inverter- α=2 β=0

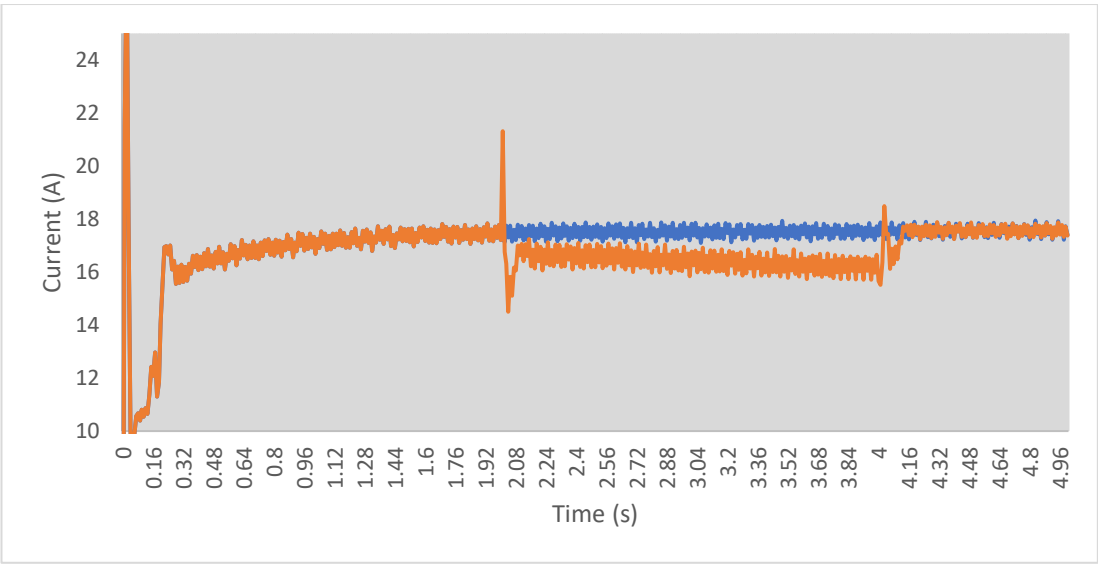


Figure 14: AC Current - Canteen - Solar inverter- α=2 β=0

But for minimum value of α as 2 we can't get significant change in DC voltage. So we increase value for α step by step and found the acceptable attack deviation from healthy situation which can make impact on microgrid system, that is 5 for α.

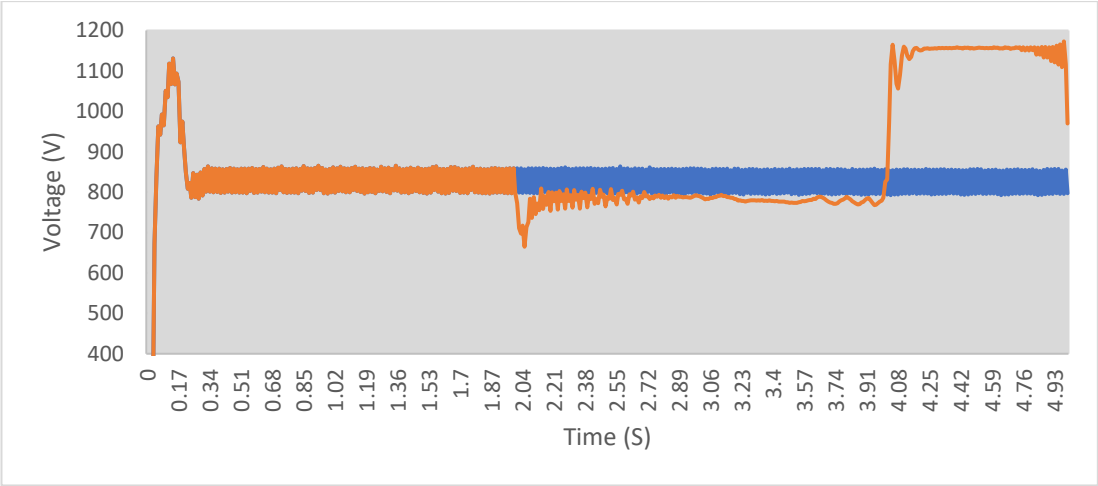


Figure 15: DC Voltage-Sumanadasa building- α=5 β=0

4.4.1.3 FDI Case3

For this case, we have α = 1 and β > 0. To achieve the maximum accuracy of our model, we set the minimum values for β, specifically β = 200.
In this case.

DC Voltage output of each load is variable.

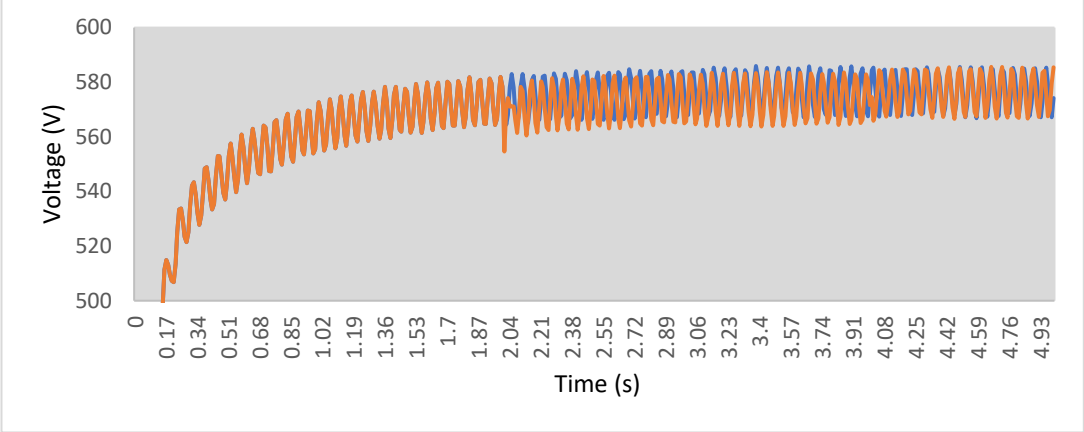


Figure 16: DC Voltage - Canteen- α=1 β=200

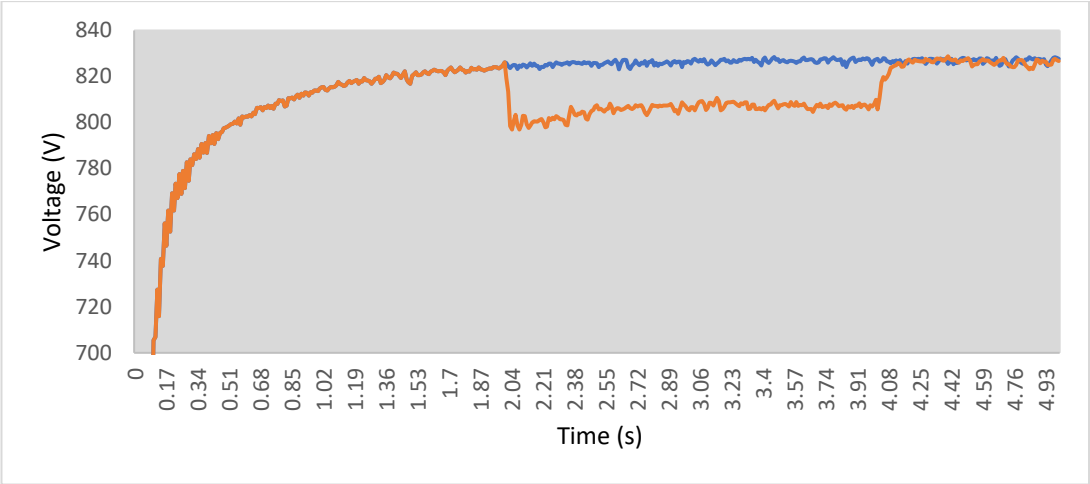


Figure 17: DC Voltage-New Admin Building- α=1 β=200

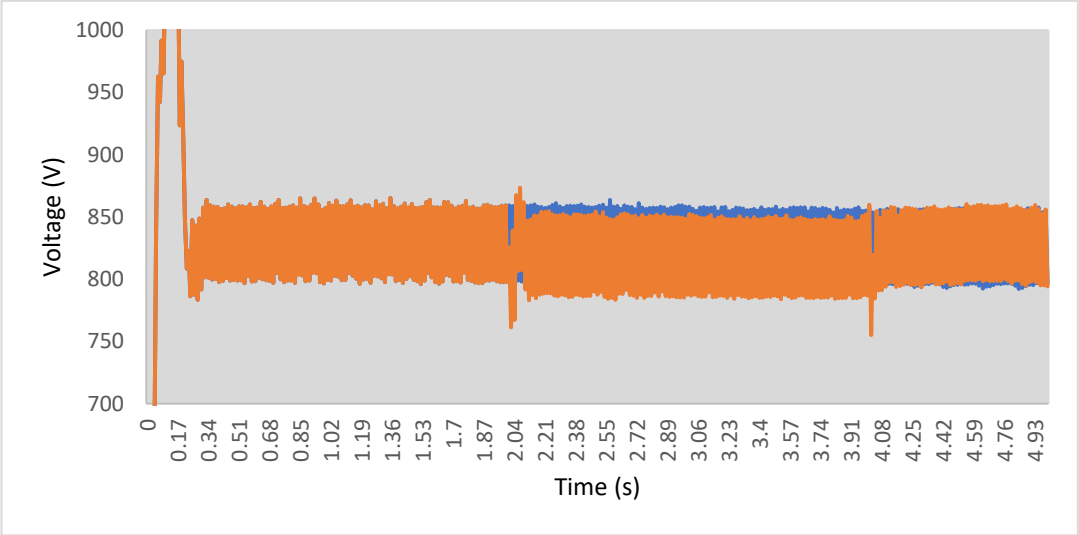


Figure 18: DC Voltage-Sumanadasa Building- α=1 β=100

AC Voltage comparing for increasing values of beta,

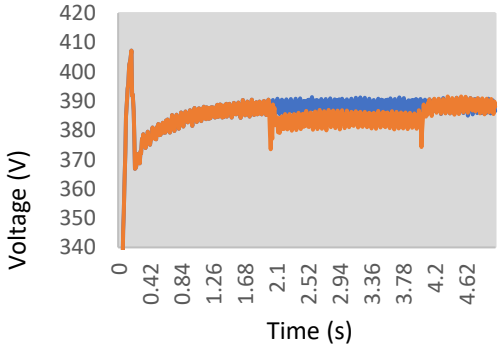


Figure 20: AC Voltage- α=1 β=200

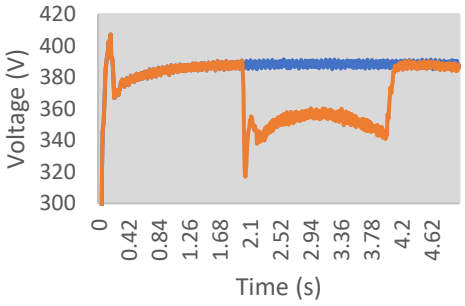
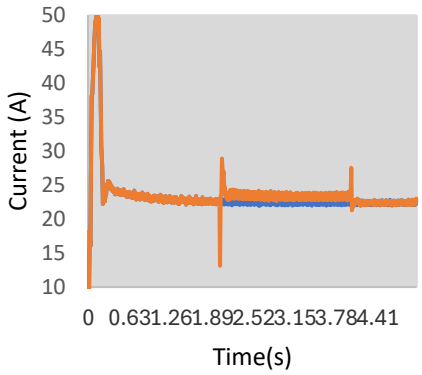


Figure 19: AC Voltage- α=1 β=500

Current at Sumanadasa Building Comparing for increasing values of beta,



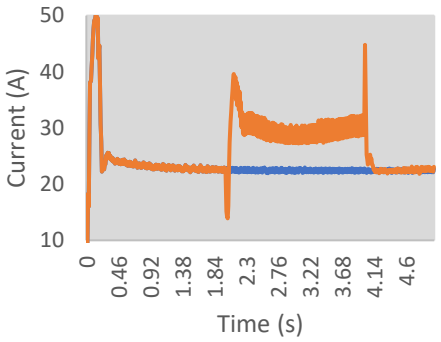


Figure 21: Current- α=1, β=200

Figure 22: Current- α=1, β=500

In comparing the four graphs for these two scenarios, we can clearly see that increasing the value of β does not change the pattern, but it does increase the deviation. This observation holds true for all scenarios. However, for various load conditions, the pattern will change for DC voltage and current, while the AC voltage remains the same.[1]

4.4.1.4 DOS Attack

For this case, we have α = 0 and β = 0 which denied the service in the system.

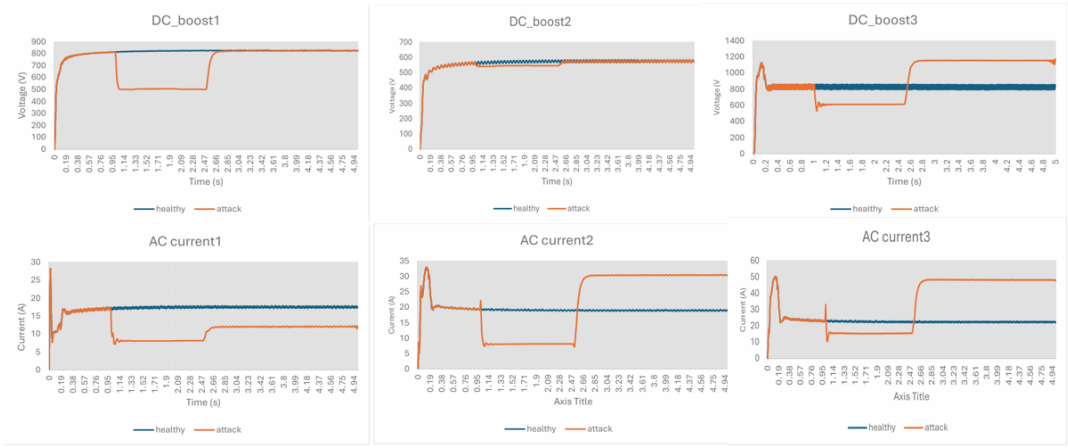


Figure 23: DoS attack of DC Voltage and AC current

DC boost1, AC current 1 refers the DC voltage and Current in the New Admin building respectively. Same as 2 means Canteen load and 3 means Sumanadasa Building load.

AC Voltage

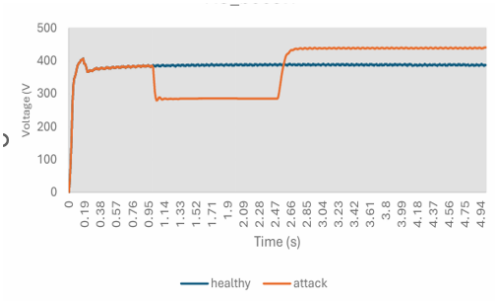


Figure 24: AC Voltage - DoS attack - Solar inverter

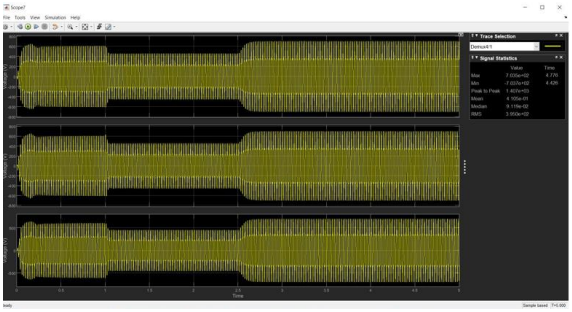


Figure 25: 3 phase AC Voltage in MATLAB Simulation

4.4.2 Battery Inverter with Attack

The battery inverter primarily functions as a storage device, so a cyber-attack on a battery inverter does not typically cause significant variations. Therefore, if an attacker chooses to target the battery inverter, they would need to use higher values for α and β compared to those required for attacking the solar inverter.

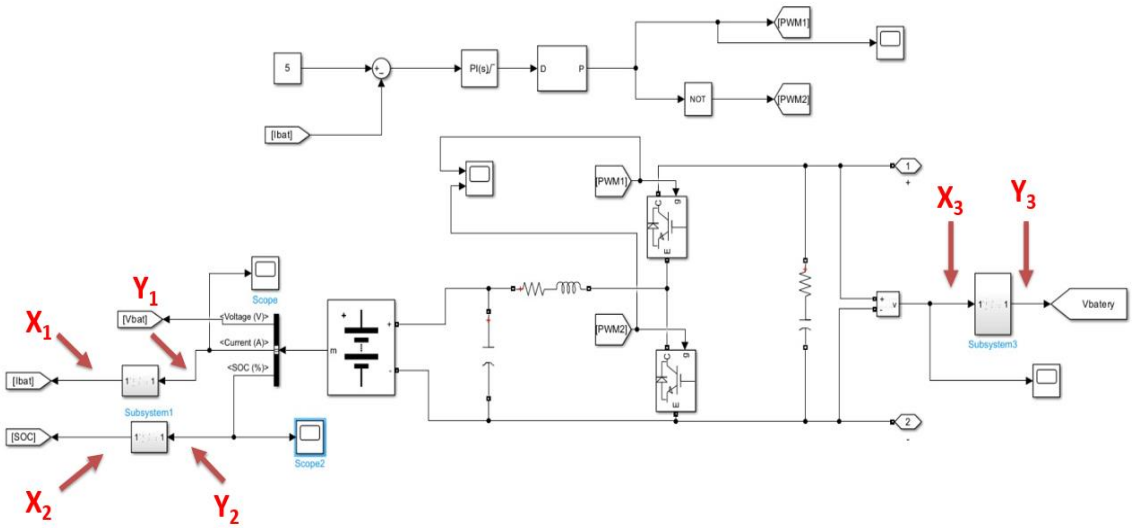


Figure 26: Battery inverter with attack

4.4.2.1 FDI Case1

For this case, we have $\alpha \neq 1$ and $\beta > 0$. To achieve the maximum accuracy of our model, we set the minimum values for α and β, specifically $\alpha = 50$ and $\beta = 1000$.

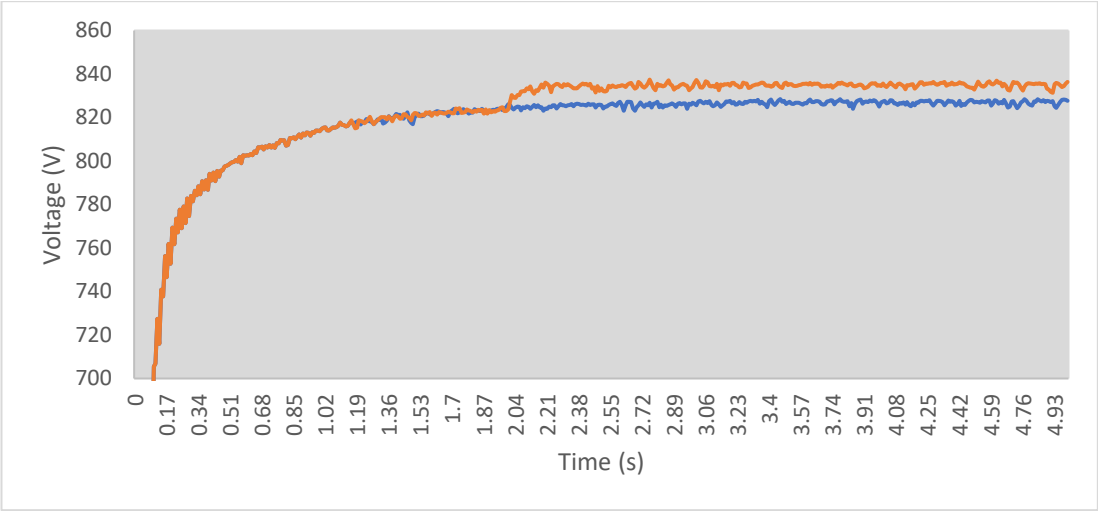


Figure 27: DC Voltage-New Admin Building- α=50 β=1000

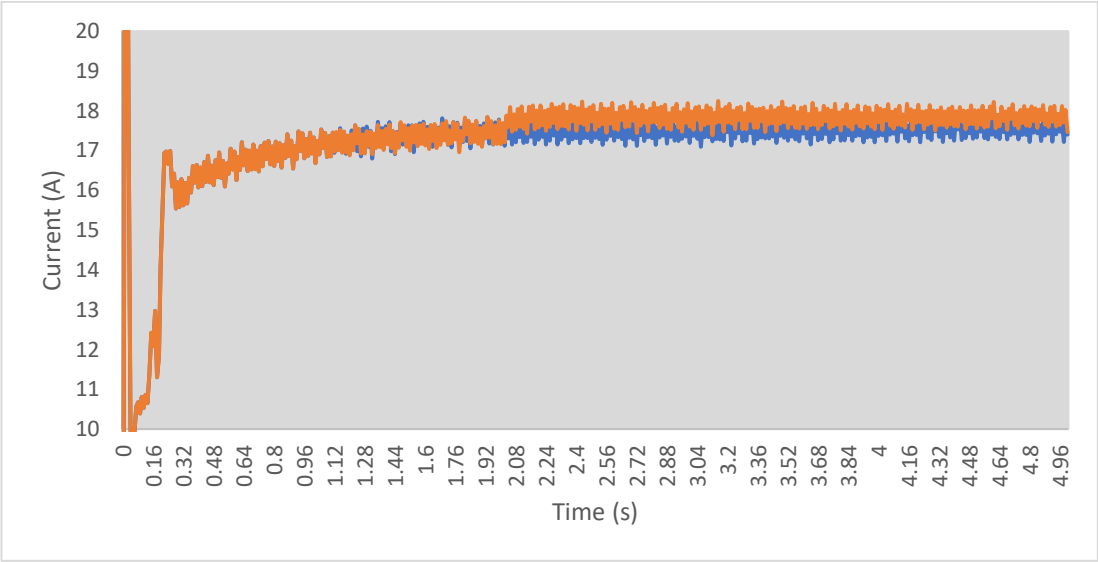


Figure 28: Current - New Admin Building – α=50 β=1000

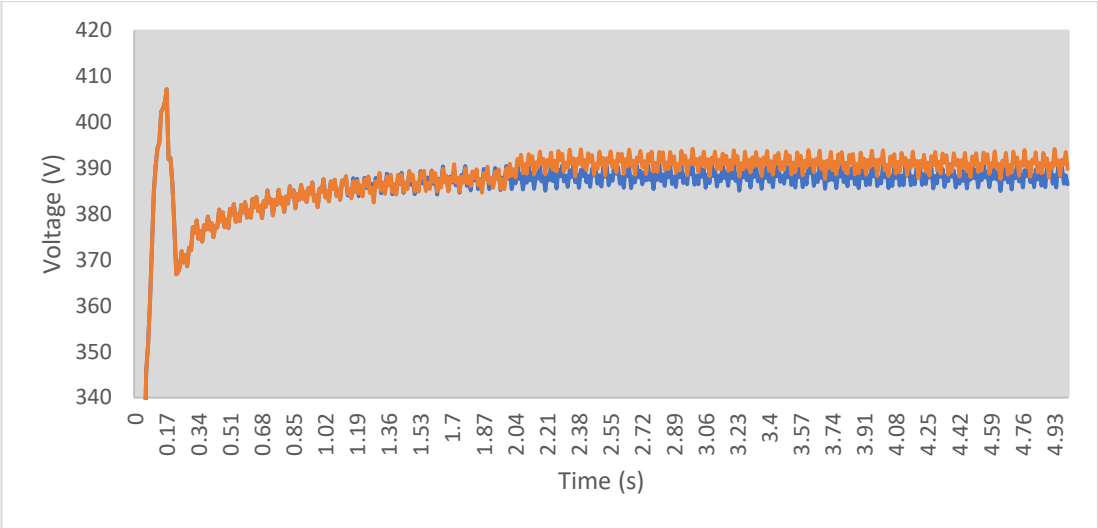


Figure 29: AC Voltage-FDI Case1 – α=50 β=1000

Here we can observe how AC voltage, Current and DC voltage change for specific load, New Admin building for FDI Case 1 cyber-attack.

4.4.2.2 FDI Case2

For this case, we have $\alpha > 1$ and $\beta = 0$. To achieve the maximum accuracy of our model, we set the minimum value for α, specifically $\alpha = 50$ and $\beta = 0$

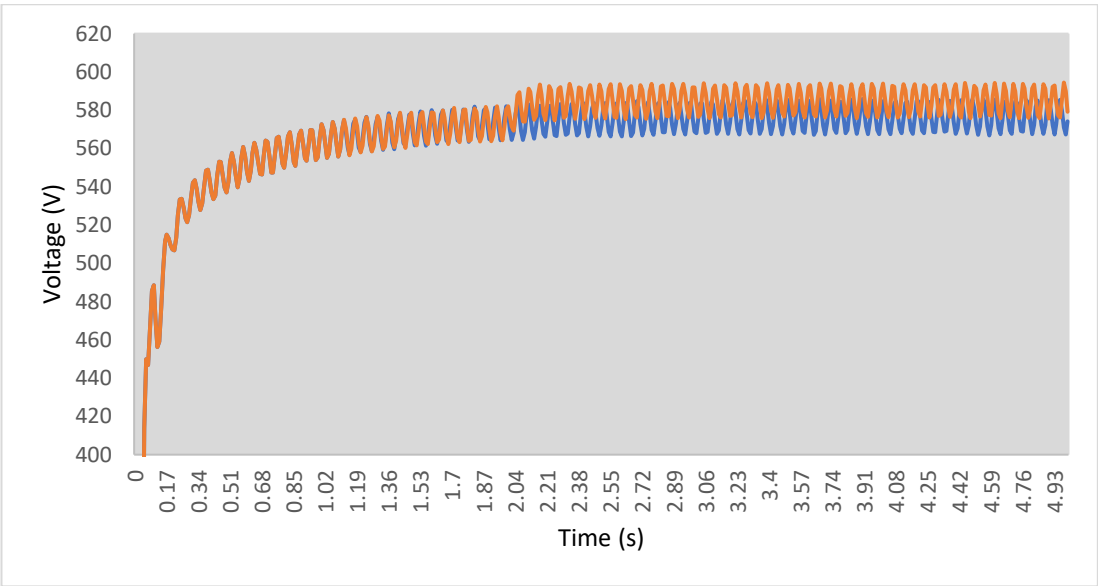


Figure 30: DC Voltage -Canteen – α=50 β=0

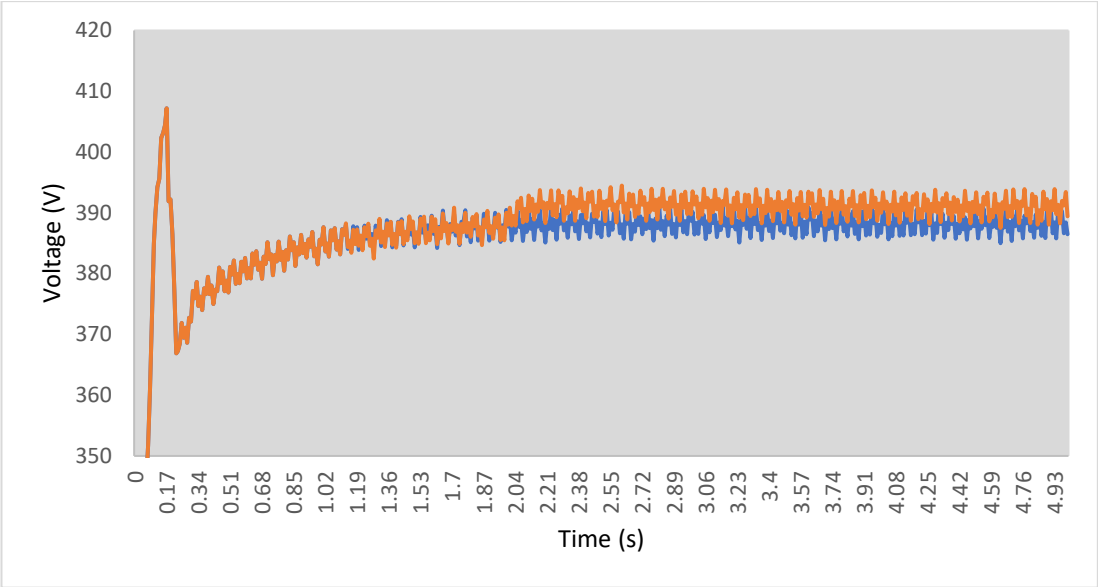


Figure 31: AC Voltage-FDI Case2 – α=50 β=0

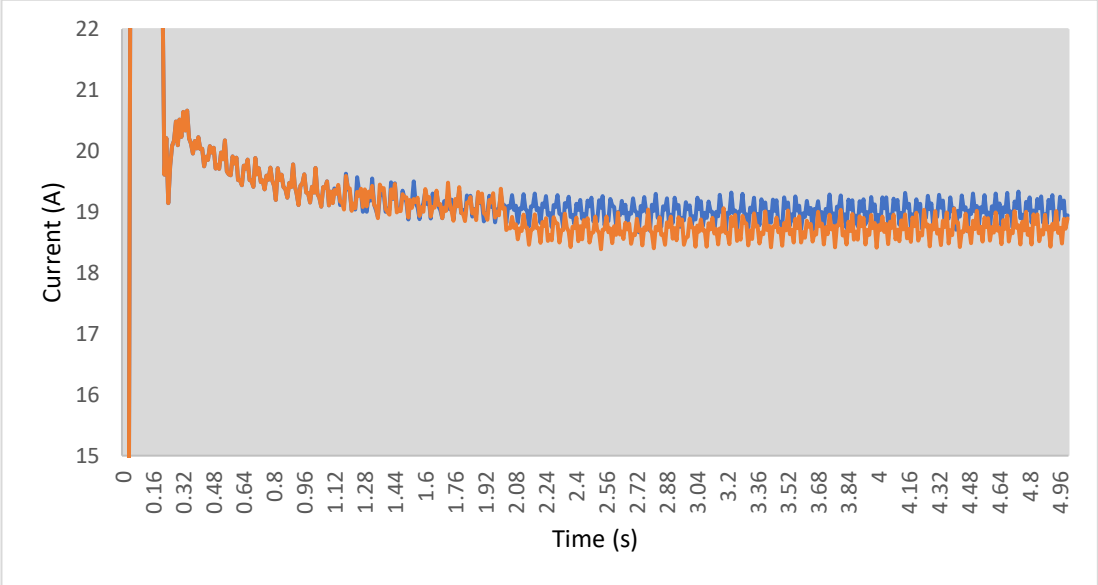


Figure 32: Current – FDI Case 2 – α=50 β=0

Here, we can observe how AC voltage, current, and DC voltage change for a specific load, such as a canteen, during an FDI Case 2 cyber-attack.

4.4.2.3 FDI Case3

In this scenario, we set $\alpha = 1$ and $\beta > 0$. To optimize our model's accuracy, we use the smallest value for α, which is $\alpha = 1$, and set β to 1000.

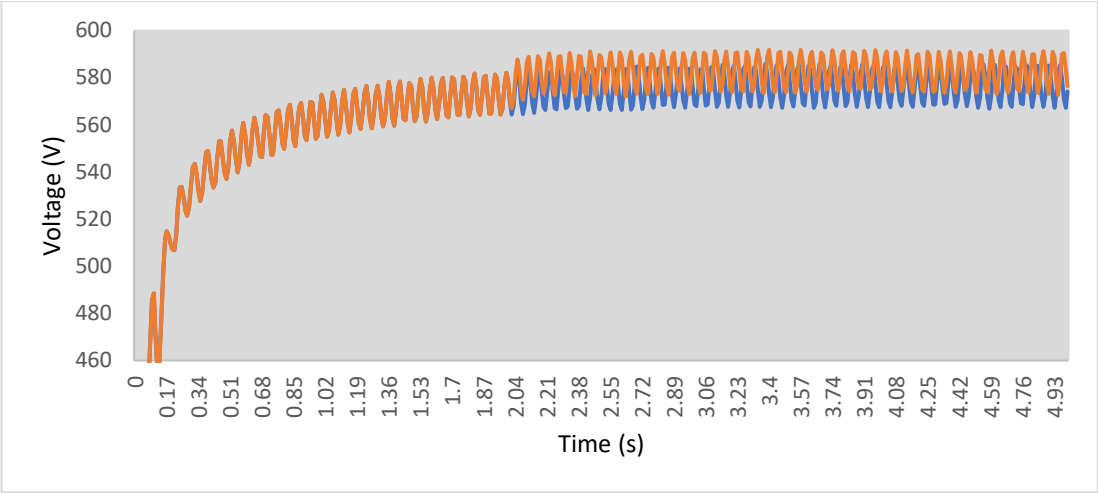


Figure 33: DC Voltage - Canteen -FDI Case3 – α=1 β=1000

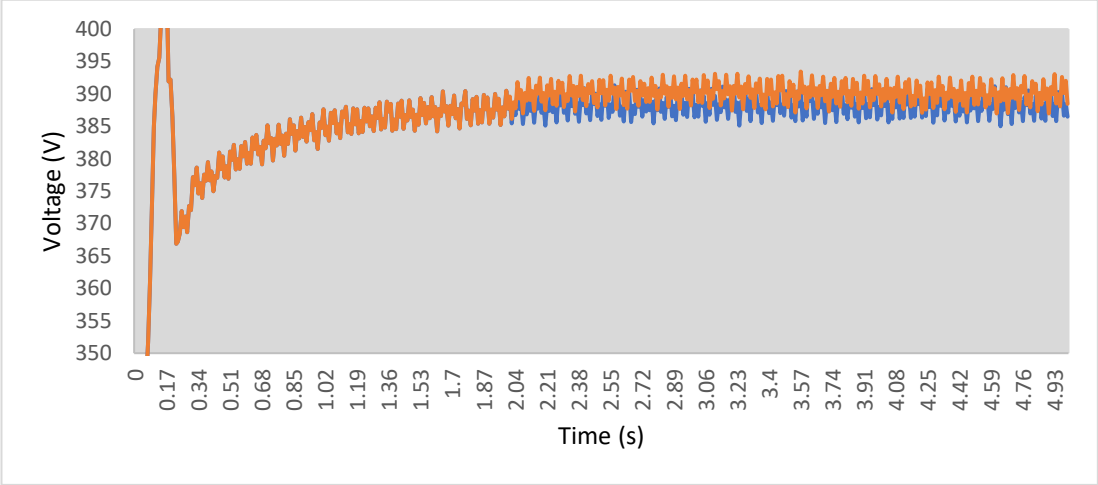


Figure 34: AC Voltage-FDI Case3 – α=1 β=1000

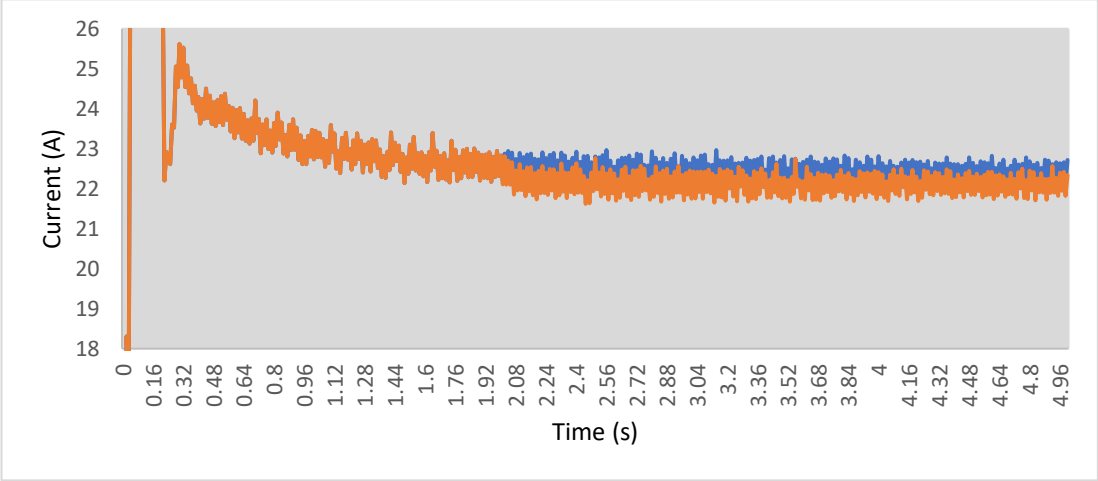


Figure 35: Current - Sumanadasa Building - FDI Case3 – α=1 β=1000

4.4.2.4 DOS attack

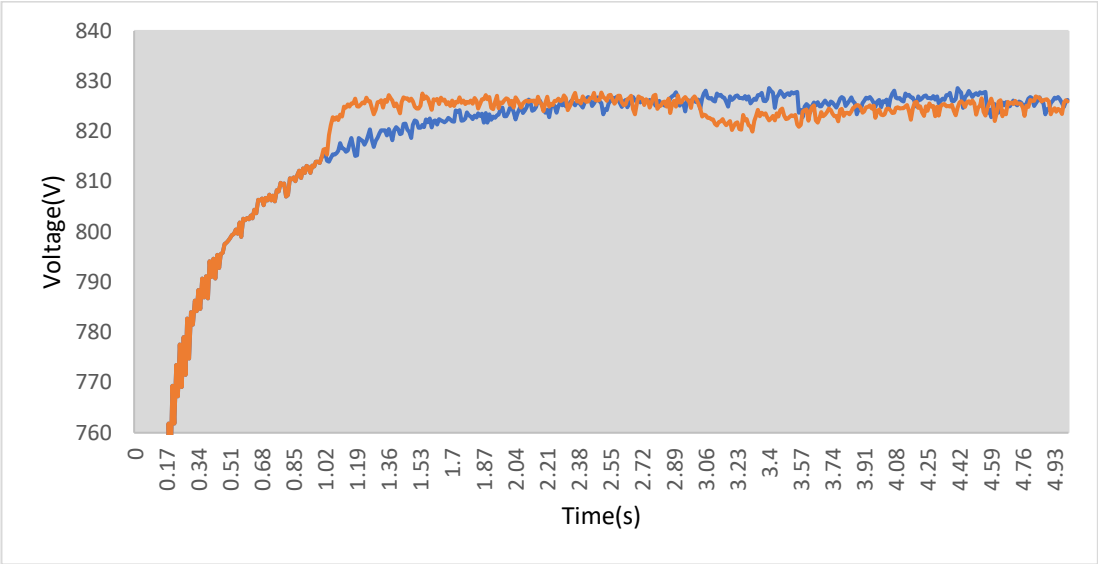


Figure 36: DC Voltage-New Admin Building-DOS

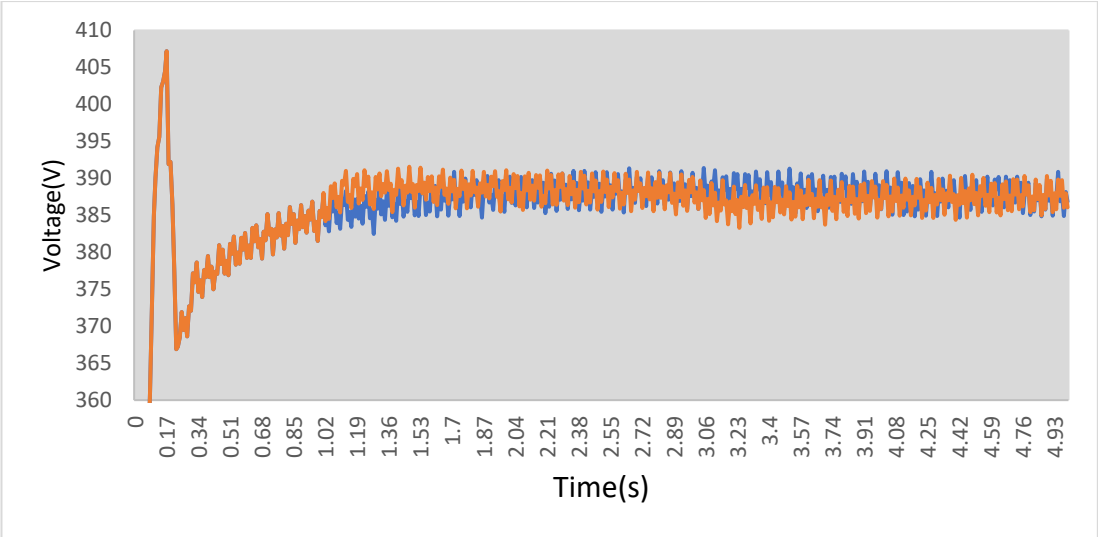


Figure 37: AC Voltage-DOS Attack

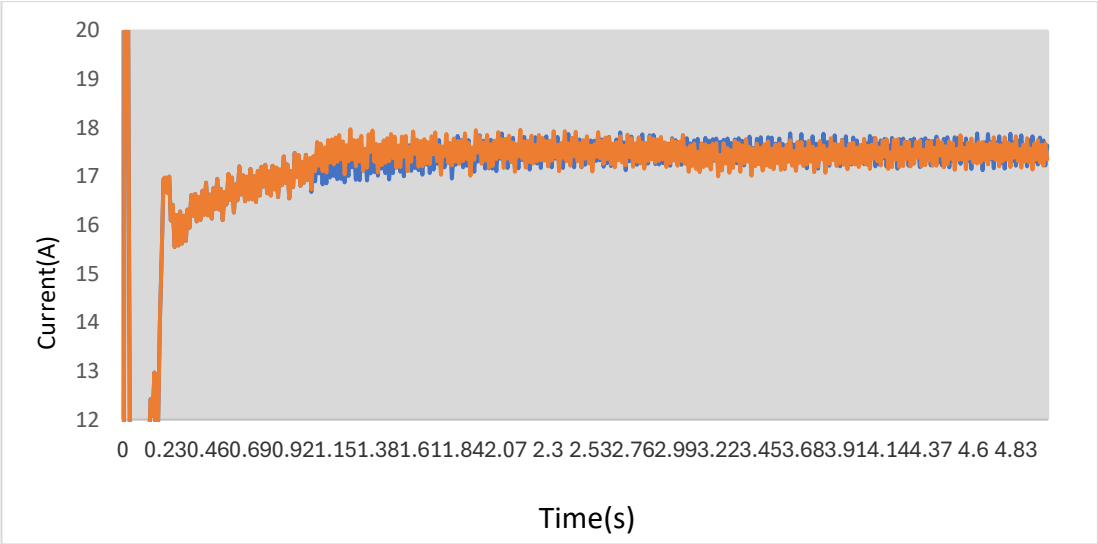


Figure 38: DOS-Current-New Admin Building

4.4.3 DC - AC converter with Attack

DC to AC converters are highly impacted by these cyber attacks, even for very small values. Some changes in their behavior are very complex and difficult to understand why they occur in that manner.[6]

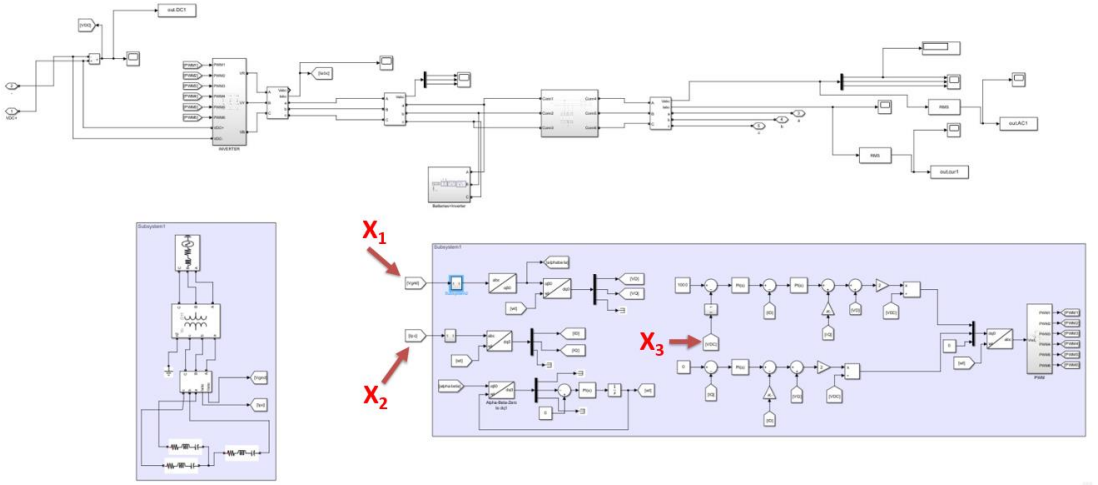


Figure 39: DC - AC converter with Attack

4.4.3.1 FDI Case1

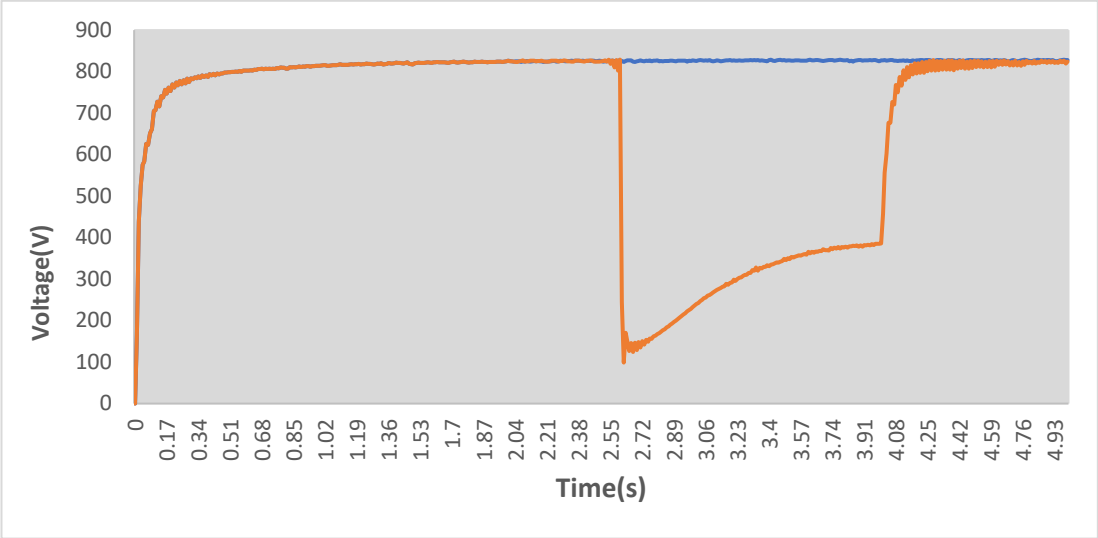


Figure 40: DC Voltage - New Admin Building – α=2 β=20

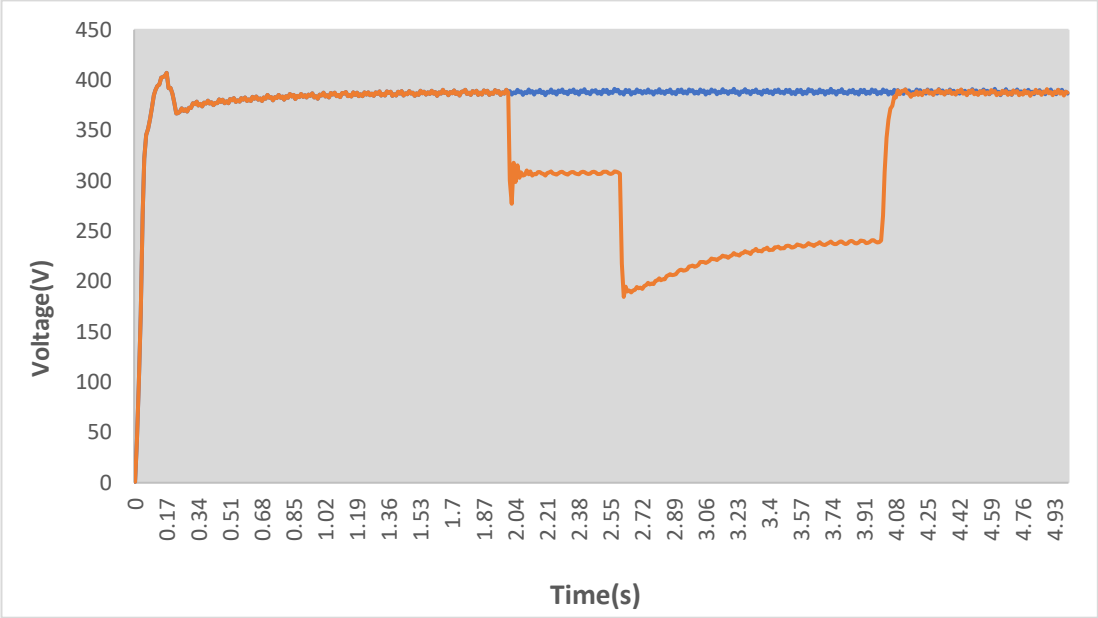


Figure 41: AC Voltage – α=2 β=20

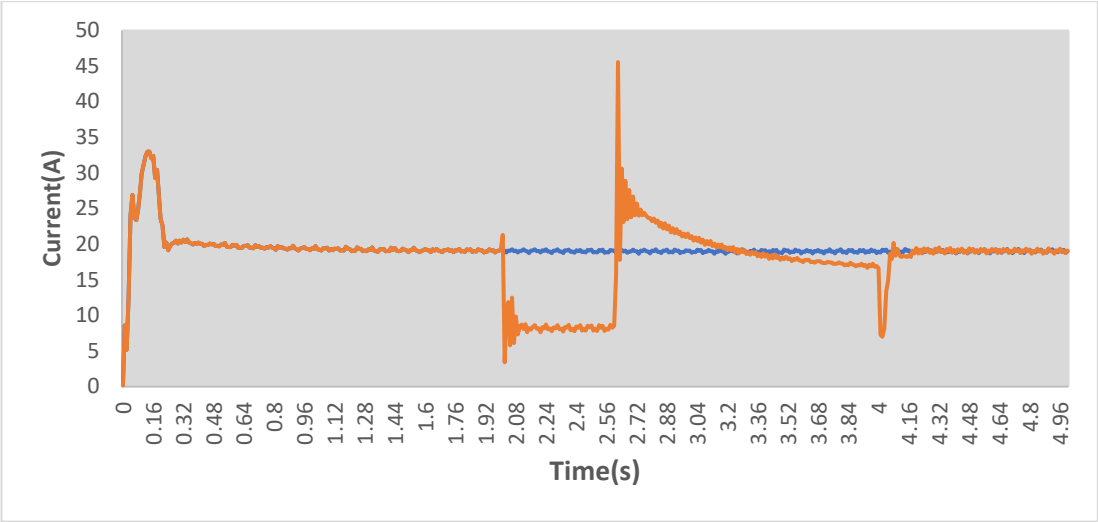


Figure 42: Current - Canteen – α=2 β=20

4.4.3.2 FDI Case2

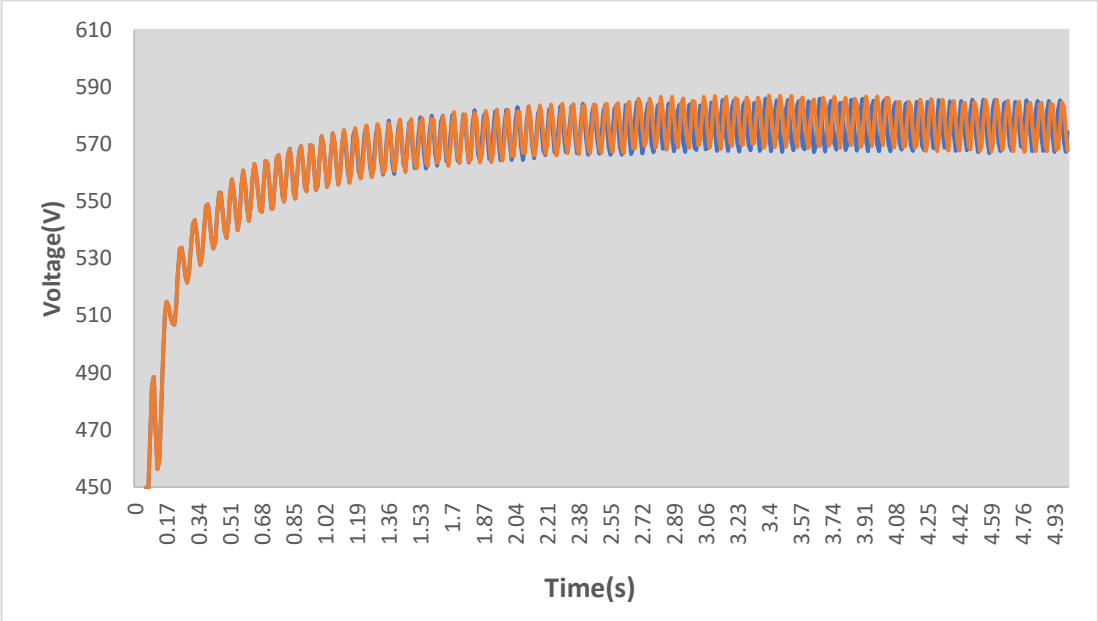


Figure 43: DC Voltage - Canteen α=2 β=0

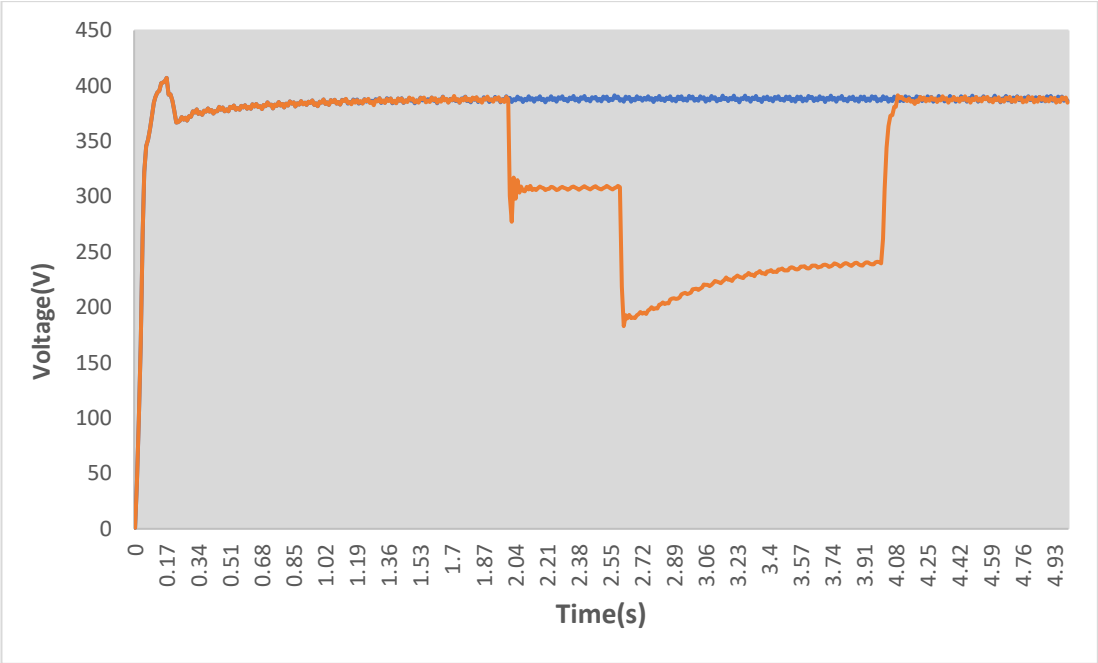


Figure 44: AC Voltage – α=2 β=0

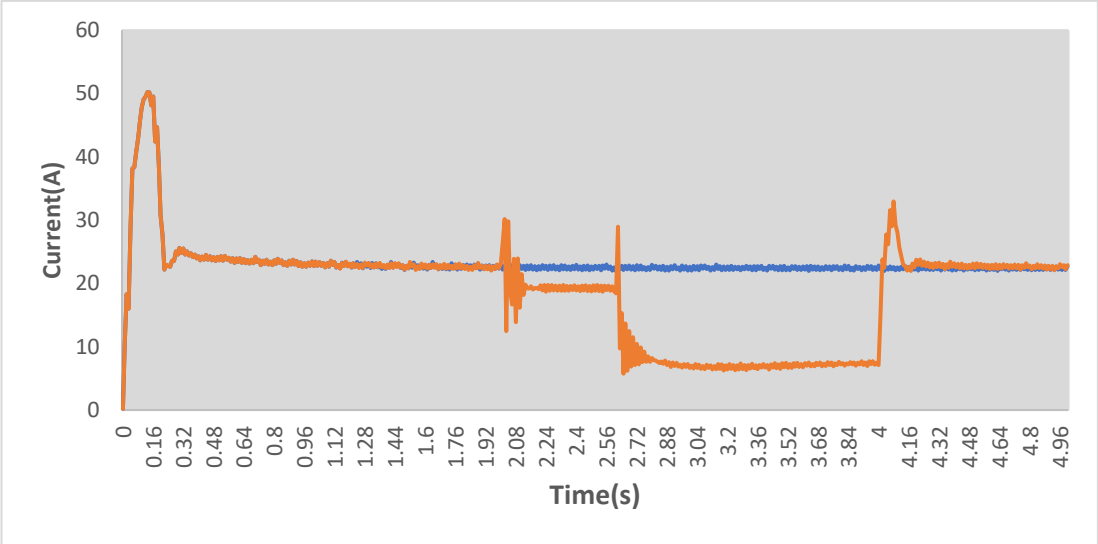


Figure 45: Current - Sumanadasa Building – α=2 β=0

4.4.3.3 FDI Case3

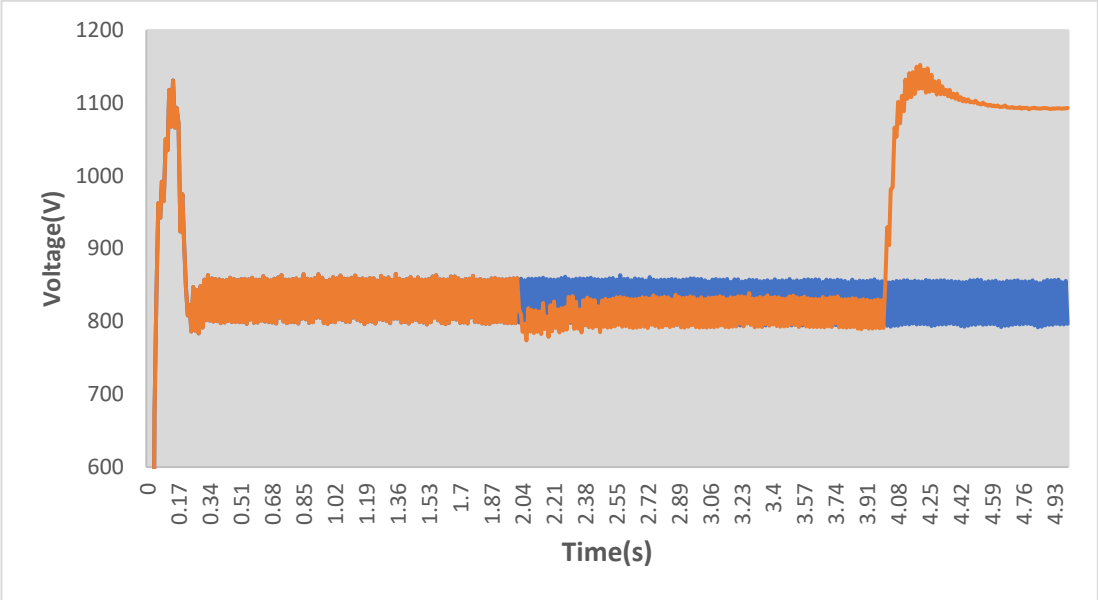


Figure 46: DC Voltage-Sumanadasa Building – α=2 β=20

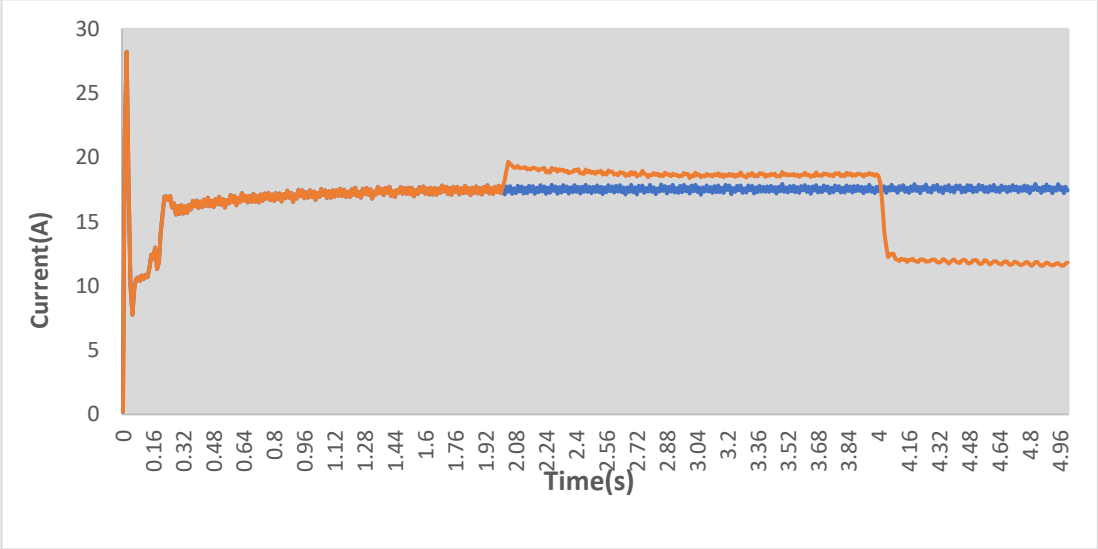


Figure 47: Current - New Admin Building – α=2 β=20

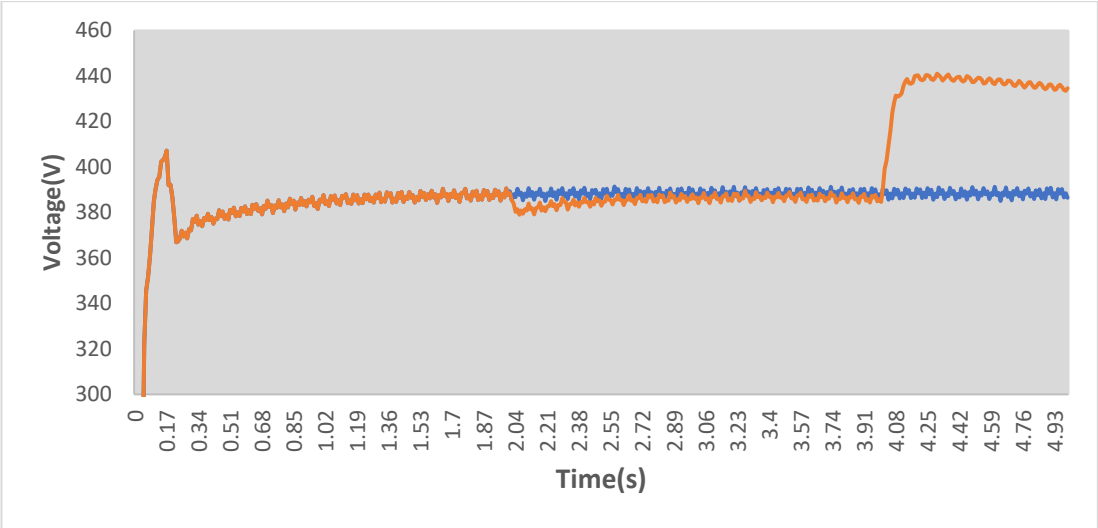


Figure 48: AC Voltage – α=2 β=20

4.4.3.4 DOS Attack

AC Voltage

Same for New Admin Building – 100kW, Canteen 50kW and Sumanadasa Building 200kW.

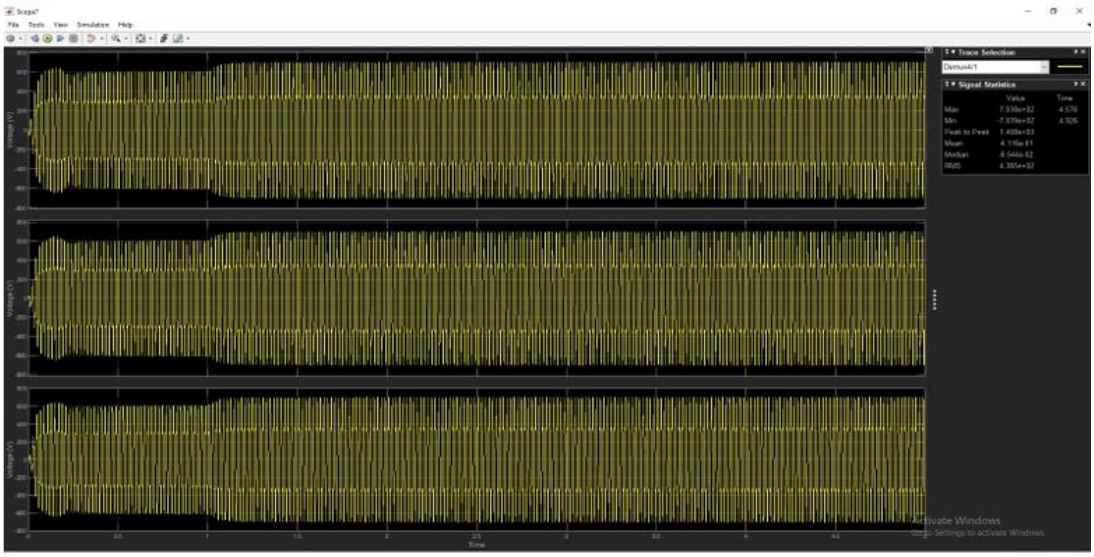


Figure 49: AC Voltage - DOS Attack

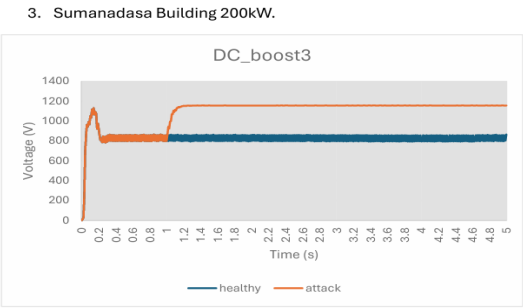


Figure 50: AC Voltage - Sumanadasa Building - DOS Attack

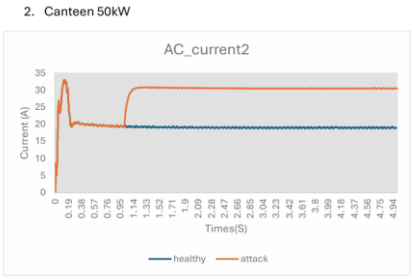


Figure 51: AC Voltage - Canteen - DOS Attack

We generated numerous graphs depicting various scenarios and used them as data to train our cyber-attack detection and classification model. Through these graphs, we identified high-impact attacks on each type of power converter. Our analysis led us to conclude that the impact of a Denial of Service (DoS) attack on power converters is relatively low compared to the three cases of False Data Injection (FDI) attacks. Among the FDI cases, FDI Case 1 demonstrated the highest impact across all scenarios studied.

Furthermore, our findings indicate that DC to AC Converter are particularly vulnerable to cyber-attacks compared to other types of converters. They exhibit significant susceptibility due to their integration into grid systems and reliance on digital communication protocols. In contrast, battery inverters were found to be the least vulnerable among the converters studied, likely due to their primary function as energy storage devices rather than active power generators. These insights underscore the importance of robust cybersecurity measures, especially for DC to AC converter, to mitigate potential risks and ensure the stability and reliability of microgrid systems.

CHAPTER 5

5. DEEP LEARNING APPROACHES

5.1 Data preprocessing

The basic step is data analysis to understand how the relationships between variables and data are distributed. Data cleaning focuses on handling missing values and detecting and removing outliers. Data transformation involves normalization, removing duplicates, encoding categories, and discretization. Feature engineering is used to select relevant features and create new features from existing ones to identify patterns. Data integration involves combining data from different sources and resolving inconsistencies, while data reduction aims to simplify the dataset.

5.1.1 Data Processing method-1

Directly applying raw data, with one row and nine features, as input to the model often results in lower accuracy for detection and classification tasks. This method struggles to extract patterns from the features effectively.

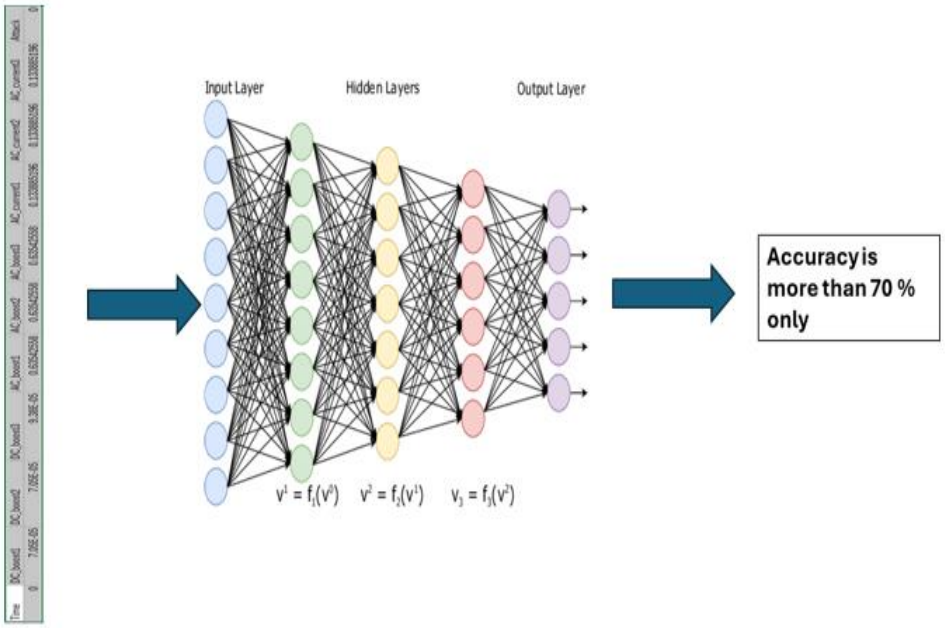


Figure 52: Data processing method-1-input

5.1.2 Data Processing method 2

In data processing, each window is assigned a label based on the maximum frequency of outputs within those 10 rows. 10 rows and 9 features are combined into a matrix. To address this, the ANN and LSTM models were designed to accept two-dimensional

input with a window size of 10 rows and 9 features. The process involves sliding the window from the 1st to the 10th row, then from the 2nd to the 11th row, and so on.

Time	DC_boost1	DC_boost2	DC_boost3	AC_boost1	AC_boost2	AC_boost3	AC_current1	AC_current2	AC_current3
0	9.80791E-05	9.80718E-05	0.000130445	0.935059927	0.935059927	0.935059927	0.197018706	0.197018706	0.197018706
0.01	163.6704165	91.05547667	10.02488968	40.78091439	40.78091439	40.78091439	21.54171784	8.593428917	10.41718597
0.02	438.4813686	131.9864806	32.46981821	98.47267173	98.47267173	98.47267173	28.20778877	5.177901588	18.31019977
0.03	525.0839349	172.9461938	273.9427976	166.993167	166.993167	166.993167	17.8978589	12.40869972	16.00369848
0.04	575.839912	311.5925752	671.7222352	266.2694284	266.2694284	266.2694284	10.05489736	24.00936172	28.63025865
0.05	583.3240456	413.8296568	837.1495137	322.0435971	322.0435971	322.0435971	7.739594238	26.90029981	38.14420622
0.06	625.5348195	449.7962958	962.5981327	345.6438168	345.6438168	345.6438168	9.85254458	23.75494722	38.37930333
0.07	623.1025832	446.8903293	942.336994	351.6534226	351.6534226	351.6534226	10.56350443	23.37336698	40.85042734
0.08	651.4338549	465.9682244	991.754425	361.0116559	361.0116559	361.0116559	10.67291176	24.89428927	42.96546134
0.09	661.1357881	485.4588519	965.2558665	372.7055637	372.7055637	372.7055637	10.39439109	27.41109175	45.79654787
0.1	705.4527963	488.499029	1050.430395	384.9104471	384.9104471	384.9104471	10.80834795	29.93258174	47.74139038
0.11	706.8903856	471.1405505	1034.855511	390.0041927	390.0041927	390.0041927	10.53569882	31.15459915	49.05763632
0.12	727.488061	456.2070054	1118.06739	394.198167	394.198167	394.198167	10.86166497	32.36491456	49.48527706

Figure 53: Change of windows

This feature extraction method enhances the model's efficiency.

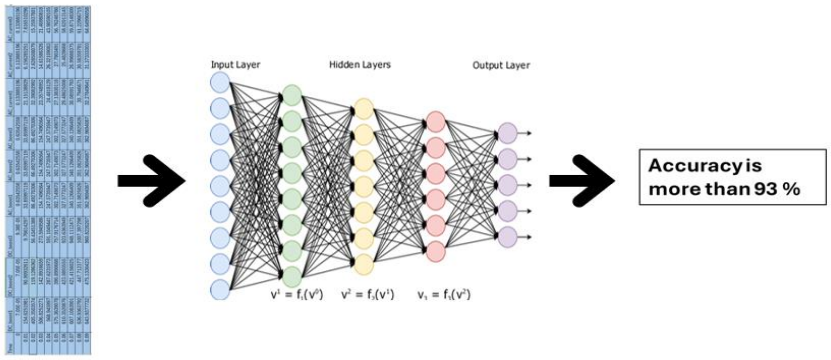


Figure 54: Data processing method 2- input

5.2 Correlation matrix

The correlation matrix reveals the strength and direction of the linear relationship between the input features and binary attack classification. A positive correlation indicates a direct relationship, where an increase in the input feature corresponds to a higher likelihood of an attack. Conversely, a negative correlation suggests an inverse relationship. Certain input features exhibited strong positive or negative correlations with attack classification, indicating significant discriminatory information. Leveraging these correlations helps the ANN model effectively differentiate between the presence or absence of an attack.

Additionally, the correlation matrix identifies potential redundancies or high interdependencies among input features, crucial for feature selection and

dimensionality reduction. This enhances the performance and efficiency of the attack classification ANN model. By checking the correlation coefficients, identify the features with the strongest influence on attack outcomes. This understanding aids in optimizing and refining the binary attack classification system

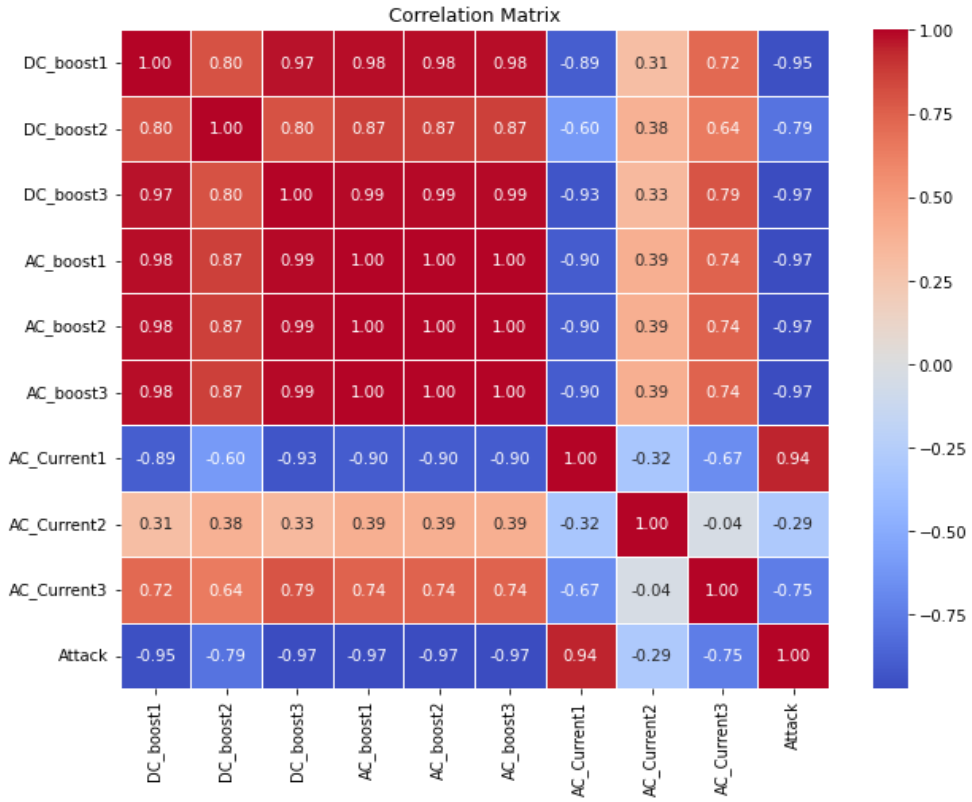


Figure 55: Correlation matrix of binary classification

DC_boost1 - DC to DC boost converter voltage output system-1
DC_boost2 - DC to DC boost converter voltage output system-2
DC_boost3 - DC to DC boost converter voltage output system-3
AC_boost1 - DC to AC converter voltage output system-1
AC_boost2 - DC to AC converter voltage output system-2
AC_boost3 - DC to AC converter voltage output system-3
AC_current1- DC to AC converter current output system-1
AC_current2- DC to AC converter current output system-2
AC_current3- DC to AC converter voltage output system3

5.3 ANN for binary classification (FNN -Attack detection)

5.3.1 First ANN model for which 1D data as input

In the starting stage of the project, started to work simple design of Ann which is used for detection. It accepts 1D input at one time. Here we used the Feedforward Network (FNN). It is a general model of ANN that is used for basic regression and classification. We used grid search to find the best parameters for the number of neurons in your

model. Grid search is a method for systematically working through multiple combinations of parameter values to find the best setup.

The model architecture consists of three hidden layers with 128, 64, and 32 neurons, respectively, each layer activated by ReLU functions. The model also includes an input layer and an output layer, which is activated by a sigmoid function to give binary classification results.

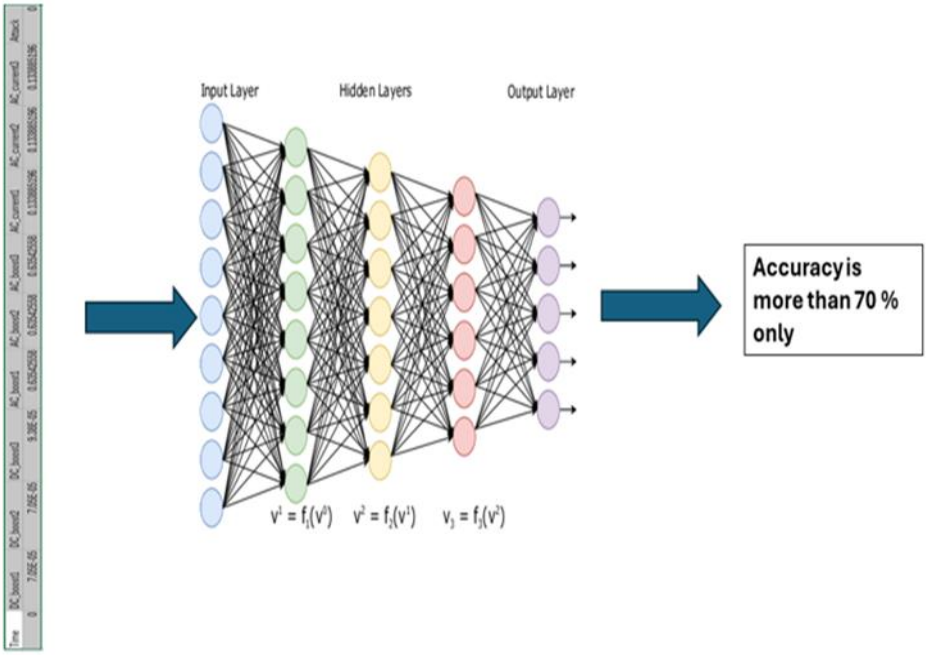


Figure 56: Data processing method-1 -input

5.3.2 Second ANN model for which 2D data is input for detection

In this project, we used a 2D model that takes a (10,9) matrix as input. The label for the matrix is the highest value from the 10 rows. This way of processing data is good at handling sudden changes because it picks the highest Attack label. The accuracy of this model is better than the previous one because it handles transients well. By focusing on the highest value in each row, it filters out noise and sudden changes, giving more accurate results. This makes the model better at detecting and classifying attacks, improving our data processing.

Our model features three hidden layers activated by ReLU. The input layer accepts 2D input, and the output layer uses the sigmoid activation function. We are optimizing hyperparameters to enhance performance, ensuring the model effectively processes data and delivers accurate results.

Implementing a hyperparameter tuning process for an ANN model using a grid search technique. The hyperparameters being considered are the number of units in four hidden layers (units) and the dropout rate (dropout rate).

units_1: [128, 64]
units_2: [64, 32]
units_3: [32, 16]
'dropout_rate: [0.2, 0.5]

These hyperparameters are defined in a dictionary called param_grid, which specifies the different values each hyperparameter can take. The ParameterGrid function from sci-kit-learn is then used to generate all possible combinations of these hyperparameter values.

Our model has three hidden layers with 128, 64, and 32 neurons, respectively, activated by ReLU. The input layer accepts 2D input, and the output layer uses the sigmoid activation function. Using grid search, we determined the optimal configuration, including dropout rates, to enhance performance.

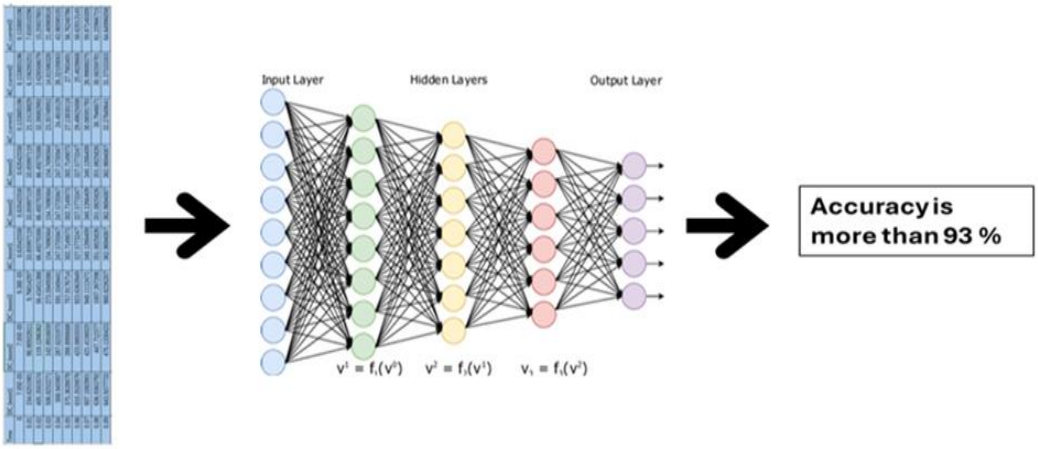


Figure 57: Second ANN model

5.4 LSTM model for Attack detection of 2D input

The LSTM is a type of RNN that is used to remember information. these usual RNNs can struggle with long-term dependencies due to issues by vanishing gradients. But overcoming these issues, unique structures with memory cells and gates are used by LSTM. Cells and gates help LSTMs manage and memorize important information.

LSTMs use three key gates; they use to manage information flow. There are the input gate, the forget gate, and the output gate.

- Input gate - decides what new information is added to the memory cell. it is controlled by a sigmoid function. It outputs values between 0 and 1. (A value of 0 means "ignore,1 means "consider fully.")
- Forget gate- determines which information to discard from the memory cell. It also uses a sigmoid function to adjust the retention rate.
- The output gate - controls what information is output from the LSTM, influencing the final output based on the current cell state.

We train these gates using backpropagation through time, enabling the LSTM to selectively keep or discard information. This gating mechanism allows the LSTM to capture and manage long-term dependencies effectively in sequential data.

The LSTM model consists of four LSTM layers, each with 50 units of neutrons, which process sequential data by maintaining 50-dimensional hidden states. After each LSTM layer, a dropout rate of 0.2 is applied to reduce overfitting. The final Dense layer with a sigmoid activation outputs a binary classification.

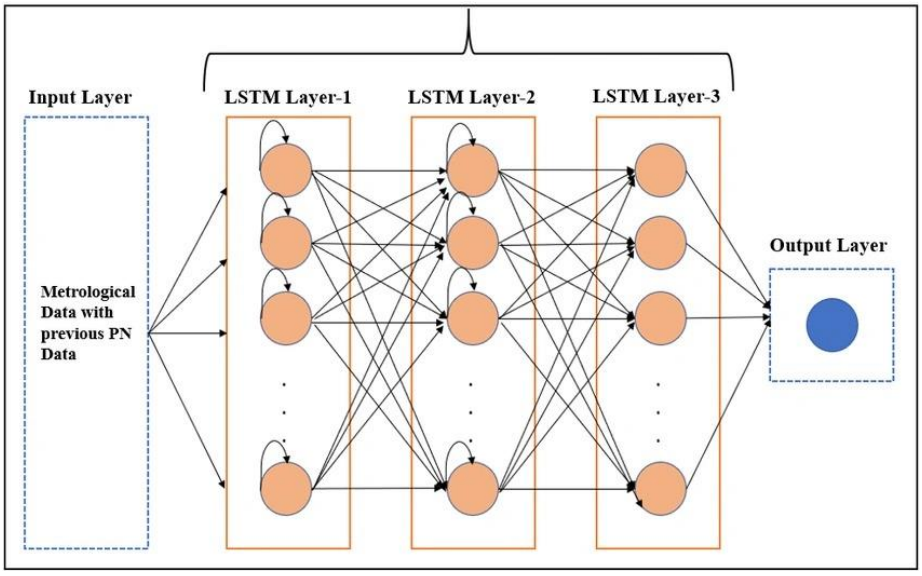


Figure 58: Sample LSTM model

5.5 Classification of FDI and DOS attack by ANN model (FNN)

These classifications mainly focused on two types of attacks. There are FDI and DOS. here we used the ANN model for classification. Its main motive is to deploy the model with sensors continuously monitor the outputs and find the type of attack. Ann's model is simple and less complex, so it is going faster than other models. Additionally, the vanishing gradient effect is comparatively low.

This ANN model consists of four hidden layers that are activated by the RELU function. There is also an output layer that provides classification output and contains

three neutrons that are activated by the SoftMax function. input layer can accept the 2D input matrix as input. output orderly gives the 1D matrix output.

Outputs and their representations
[1,0,0] - DOS
[0,1,0] - FDI
[0, 0, 1] - No Attack

Grid search algorithm to identify the number of neutrons in each hidden layer. It gives the best parameters for each data set. want to find the best set of parameters that give the highest performance for most of the inputs.

Based on the results of the grid search, 128, 64,64, and 32 are the number of neutrons that are contained by hidden layers in order from the input layer. It gives higher performance compared to other sets of parameters

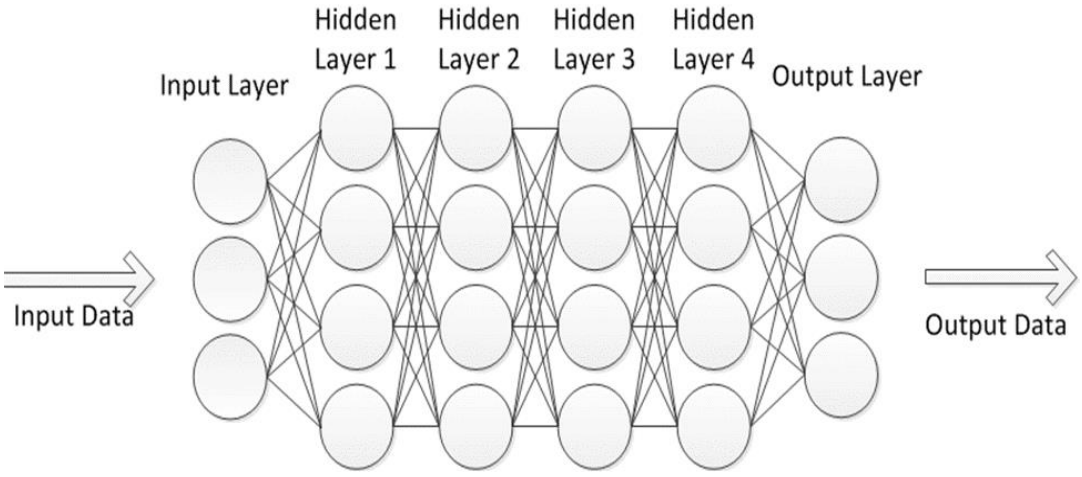
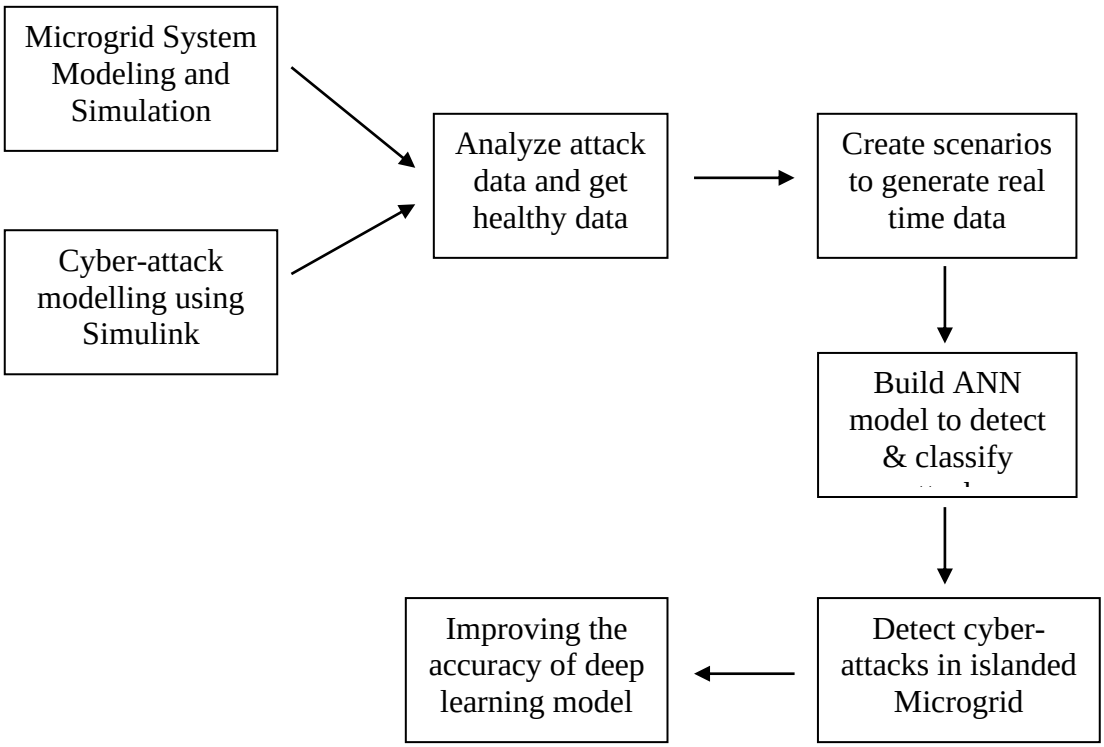


Figure 59: ANN model for classification

CHAPTER 6

6. METHODOLOGY

The methodology for this project includes several key steps aimed at developing and implementing a robust cyber-attack detection system for power converters within AC islanded microgrids using deep learning approaches.



1. Microgrid System Modeling and Simulation: Begin by creating a detailed simulation model of the University of Microgrid (UOM) using MATLAB. This model will replicate the operational dynamics of an AC islanded microgrid, incorporating various power converters such as Voltage Source Converters (VSCs), Current Source Inverters (CSIs), DC-DC boost converters, and battery chargers. The simulation will simulate normal operation scenarios as well as potential cyber-attack scenarios.

2. Cyber Attack Modeling using Simulink: Utilize Simulink to model cyber-attacks on the power converters within the microgrid. Develop mathematical models that simulate Denial of Service (DOS) attacks and False Data Injection Attacks (FDI), among others. These models will be crucial in generating attack data that will be used to train and validate the deep learning models.

3. Analysis of Attack Data and Collection of Healthy Data: Analyze the simulated attack data generated in the previous step and collect corresponding healthy data from normal microgrid operation. This step involves preprocessing the data to ensure it is suitable for training the deep learning models.

4. Scenario Creation for Real-time Data Generation: Create diverse scenarios within the simulation environment to generate real-time data that mimics the dynamic nature of cyber-attacks and normal operations in an islanded microgrid. These scenarios will be used to train and test the deep learning models under various conditions.

5. Building Artificial Neural Network (ANN) Model: Develop and implement an Artificial Neural Network (ANN) model using deep learning frameworks such as TensorFlow. Train the ANN model on the collected dataset, optimizing its architecture to detect and classify different types of cyber-attacks on power converters within the microgrid.

6. Detection of Cyber-Attacks in Islanded Microgrid: Deploy the trained ANN model to detect cyber-attacks in real-time within the simulated microgrid environment. Evaluate the model's performance in accurately identifying and classifying DOS, FDI, and other types of attacks, ensuring its effectiveness in enhancing microgrid security.

7. Improving the Accuracy of the Deep Learning Model: Fine-tune the ANN model based on performance evaluations and feedback from initial detection results. Implement techniques such as hyperparameter tuning, regularization, and ensemble methods to enhance the model's accuracy, sensitivity, and specificity in detecting cyber threats.

This methodology integrates advanced simulation techniques, cyber-attack modeling, deep learning model development, and rigorous evaluation to achieve a comprehensive cyber-attack detection system for AC islanded microgrids. By following these steps, the project aims to contribute significant advancements in the field of microgrid security, ensuring the resilience and reliability of future energy systems against evolving cyber threats.

CHAPTER 7

7. RESULTS OF NEURAL NETWORKS

7.1 Results of First ANN (FNN) model for which 1D data as input

However, it cannot detect the attack because its accuracy is very low due to overfitting, and it cannot identify the pattern of attack because of a lack of data preprocessing.

Parameters for evaluation	values
Accuracy	0.5860306403853677
Mean Squared error	0.4139705246145075
Binary cross entropy	2463.2673395330257

Table 1: Output of first ANN model before tunning

Overfitting occurs when the validation loss is much higher than the training loss. This means the model works well with the training data but struggles with new data, showing it has memorized the training data instead of learning general patterns.

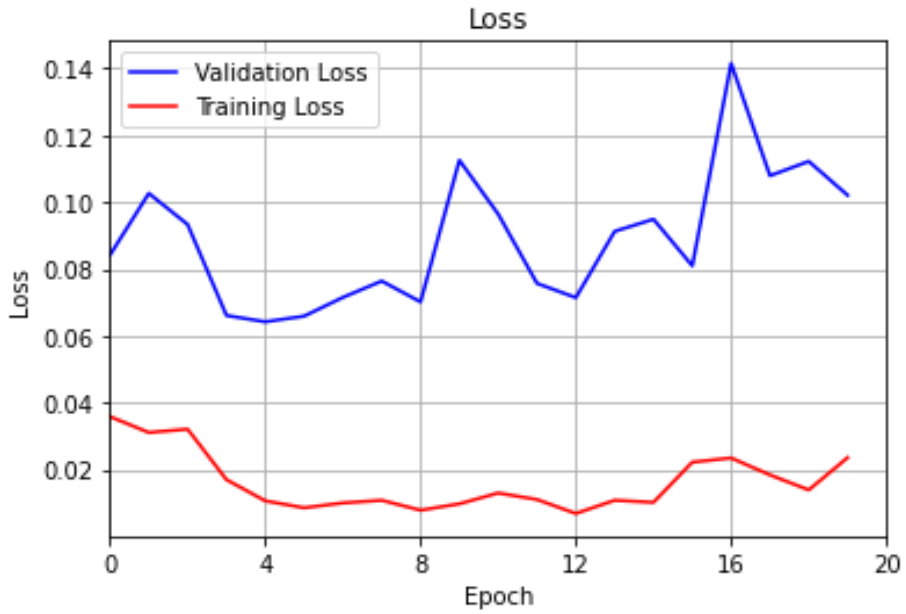


Figure 60: Loss vs epoch curve

The accuracy of 0.586 suggests that the model is not performing well in making correct predictions. In the context of overfitting, this might mean that while the model is very good at fitting the training data, it doesn't generalize well to the validation data.

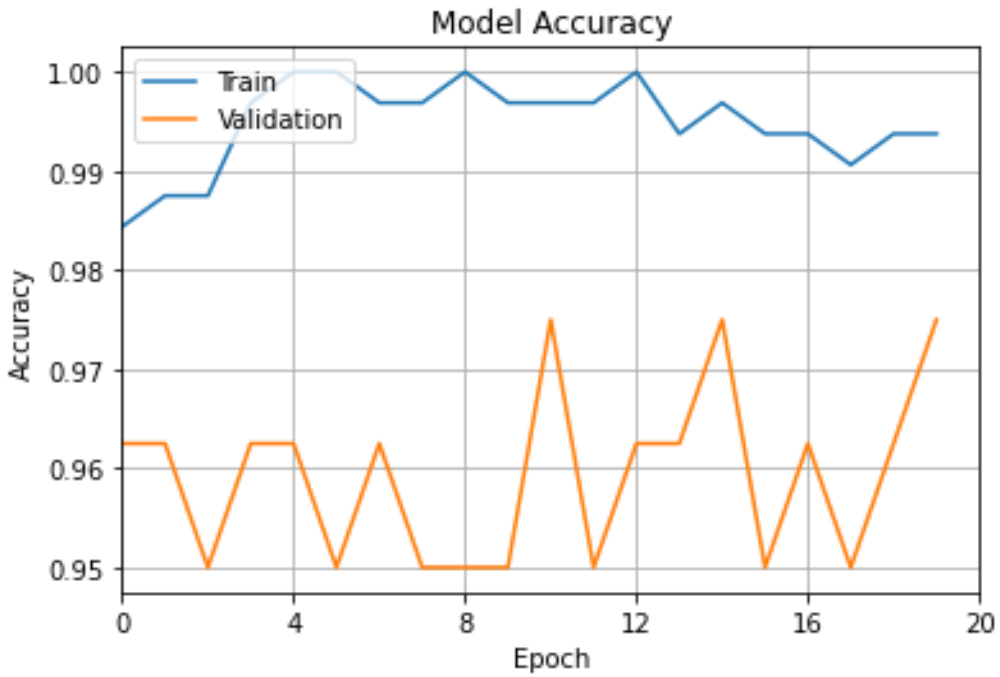


Figure 61: Accuracy vs epoch curve

A high binary cross-entropy value means the model's predictions are far from the true labels, showing that the model is having trouble learning the right patterns. This often means the model might be overfitting or learning the training data too well without generalizing to new data.

After adding dropout and testing other hyperparameters, we saw some improvement in results. However, the accuracy is still very low. This indicates that, despite the changes, the model is still struggling to perform well.

Accuracy	0.6122492514550686
Mean Squared error	0.3868210781365633
Binary cross entropy	1198.9317753389478

Table 2: Output of first ANN model after tunning

To fix generalization problems, we made changes to our ANN model. We adjusted the model’s design, improved how we process data and redesigned some parts of our

approach. These changes help the model perform better and be more accurate with new data.

7.2 Results of Second ANN(FNN) model for which 2D data is input for detection

7.2.1 The average values of the results of the model

The metrics listed are used for evaluating a binary classification model's performance comprehensively. Each metric provides unique insights into different aspects of the model's behavior.

Binary Cross entropy	0.2782528533975387	Lower values indicate better performance, as it measures the divergence between the predicted probability distribution and the true distribution.
Mean Squared Error	0.040759421559616586	Lower values indicate better performance. It assesses the accuracy of probabilistic predictions
Binary Accuracy	0.9532712267504798	Higher values indicate better performance. It's useful for balanced datasets
Precision	0.9329721497164832	Higher values indicate better performance. It's crucial when the cost of false positives is high.
Recall	0.9497611853811476	Higher values indicate better performance. It's crucial when the cost of false negatives is high.
AUC (Area Under the ROC Curve):	0.9782901273833381	Values range starts from 0 to 1. higher values represent better performance. An AUC of 1 represents a perfect model.
AUC (PRC, Area Under the Precision-Recall Curve)	9678535925017463	The value range starts from 0 to 1, with higher values representing better performance. An AUC of 1 represents a perfect model.

Table 3: Output of performance of ANN second model

Early stopping
```
monitor='val_loss',
min_delta=0.001,
patience=10,
mode='min',
restore_best_weights=True
```

The early stopping technique with the best weights restored based on the validation loss enhances model performance. It waits for 10 epochs to check the validation loss's change. If the change is not significant, it stops the training. This method stops training

when the validation performance gets worse, preventing overfitting and keeping the best weights. As a result, it provides better accuracy and generalization compared to the previous approach, significantly boosting the model's effectiveness.

7.2.2 Example1: DC to DC boost converter at FDI attack

i) Confusion matrix

A confusion matrix evaluates a model's overall performance by showing correct and incorrect predictions. It helps calculate accuracy, precision, and recall and identifies errors such as false positives and false negatives. This detailed breakdown aids in understanding and improving the model's performance.

A confusion matrix for data from a DC-to-DC boost converter under an FDI (false data injection) attack helps evaluate the model's accuracy in predicting attack instances. It shows the number of true positives, true negatives, false positives, and false negatives, providing insight into the model's performance and its ability to correctly classify attack and non-attack conditions.

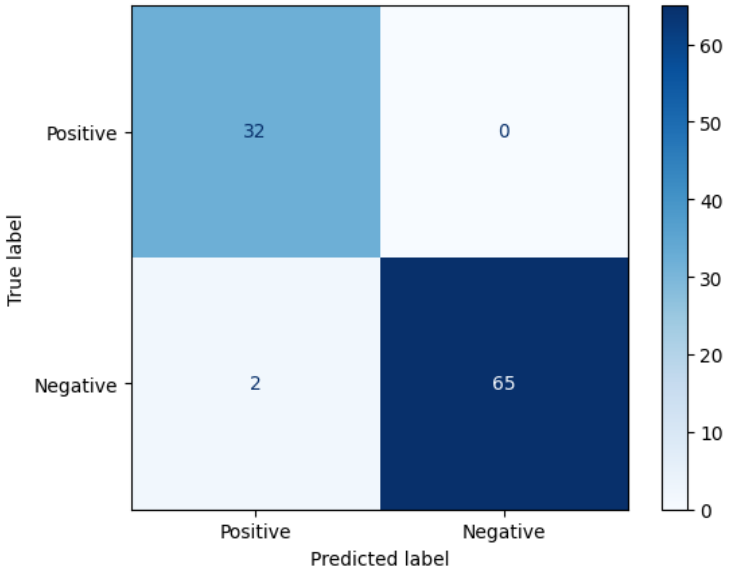


Figure 62: Confusion matrix for ANN model 2 under DC-to-DC boost converter at FDI attack for ANN model-2

Confusion matrix outputs
True Positive (TP): You predicted positive and it’s true.
True Negative (TN): You predicted negative and it’s true.
False Positive (FP) / Type I Error: You predicted positive and it’s false.
False Negative (FN) / Type II Error: You predicted negative and it’s false

ii) Accuracy vs epoch graph

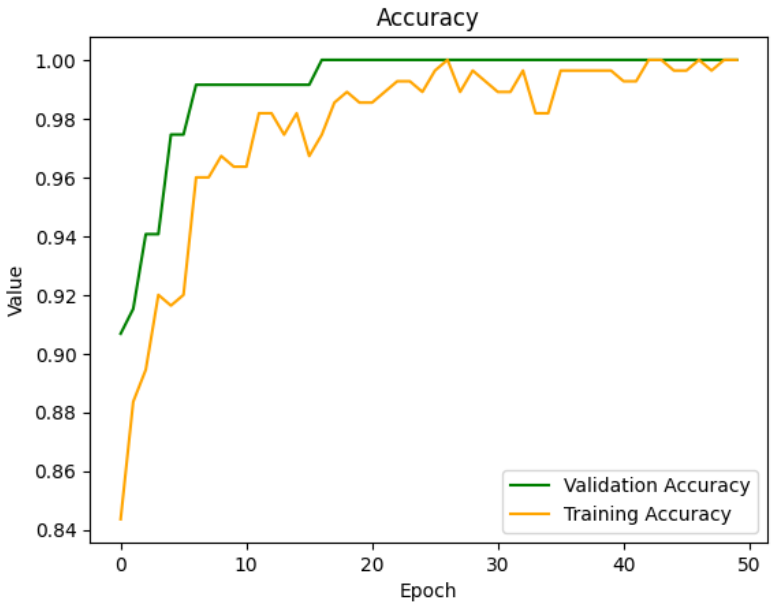


Figure 63: DC to DC boost converter under an FDI (false data injection) accuracy for ANN model 2

iii) Loss vs epoch graph

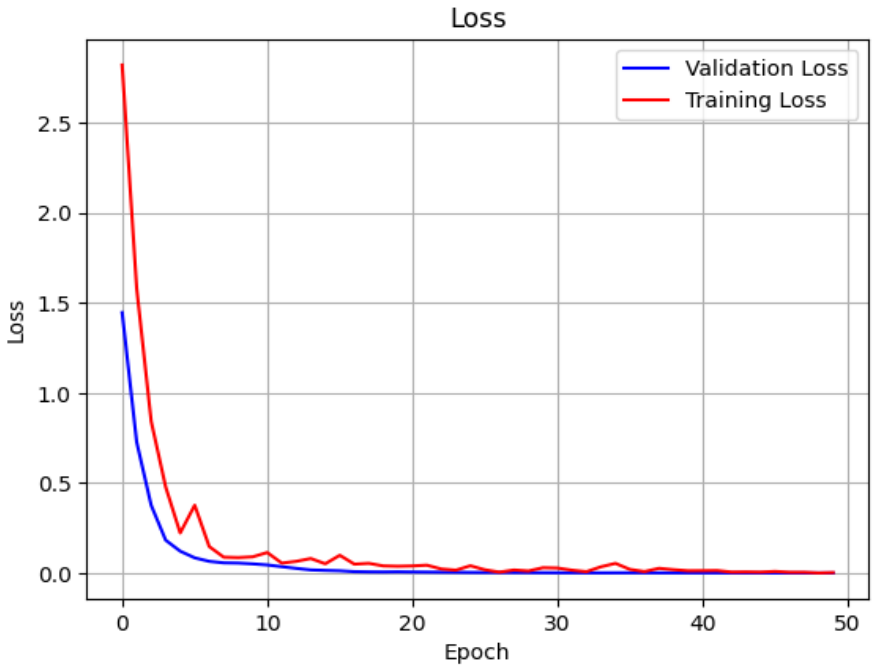


Figure 64: DC to DC boost converter under an FDI (false data injection) accuracy for ANN model 2

iv) Overall graphs vs epoch

In here below these graphs showed precision, PRC, and recall. These results represent the overall efficiency and model performance of the trained model under a DC-to-DC boost converter under an FDI (false data injection) accuracy

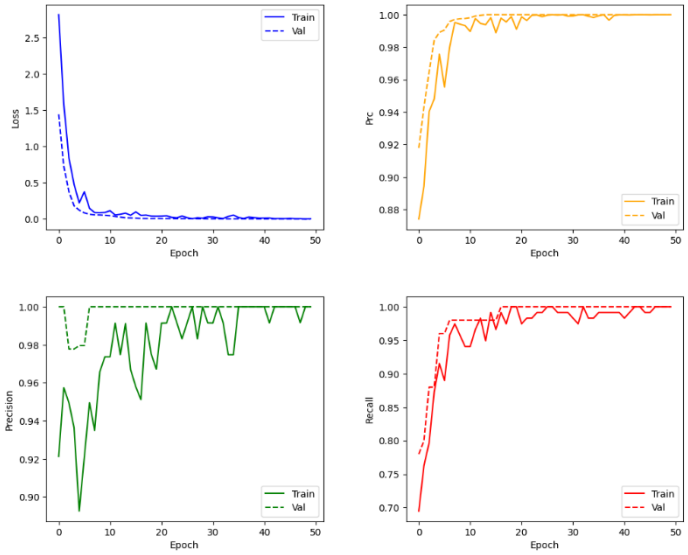


Figure 65: DC to DC boost converter under an FDI (false data injection) overall graphs for ANN model 2

7.2.3 Example 2: DC to DC boost converter under DOS and FDI attack

These data consist of multiple attacks which include FDI and DOS. It contains 13,481 data sets. It is the largest data set.

i) Confusion matrix

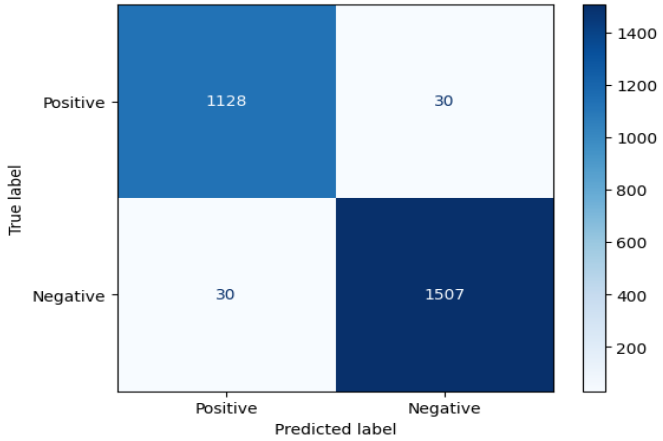


Figure 66: Confusion matrix for ANN model 2 under DC-to-DC boost converter under DOS and FDI attack for ANN model-2

ii) Accuracy vs epoch graph

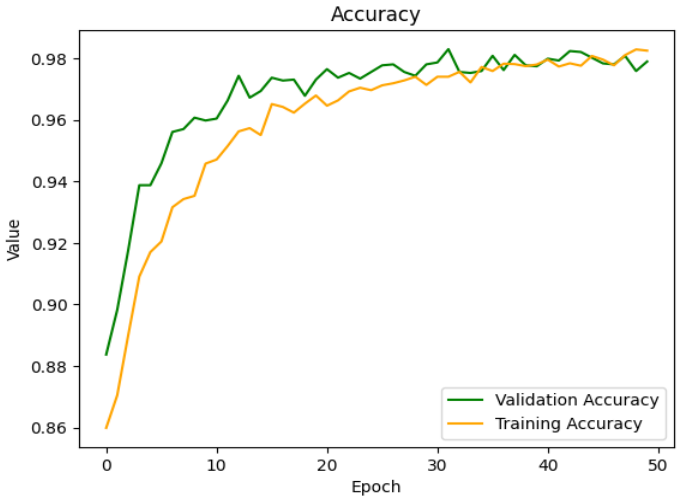


Figure 67: DC to DC boost converter under a DOS and FDI accuracy for ANN model 2

iii) Loss vs epoch graph

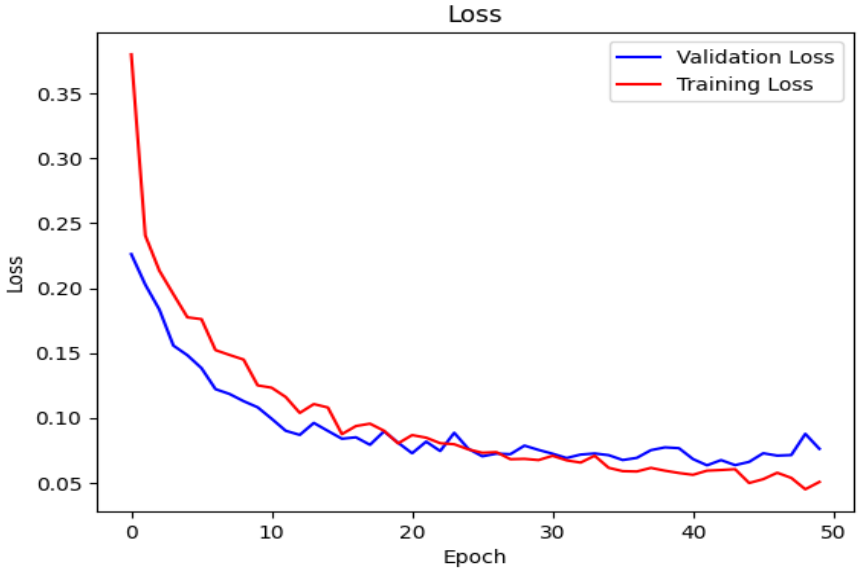


Figure 68: DC to DC boost converter under a DOS and FDI loss for ANN model 2

iv) Overall graphs

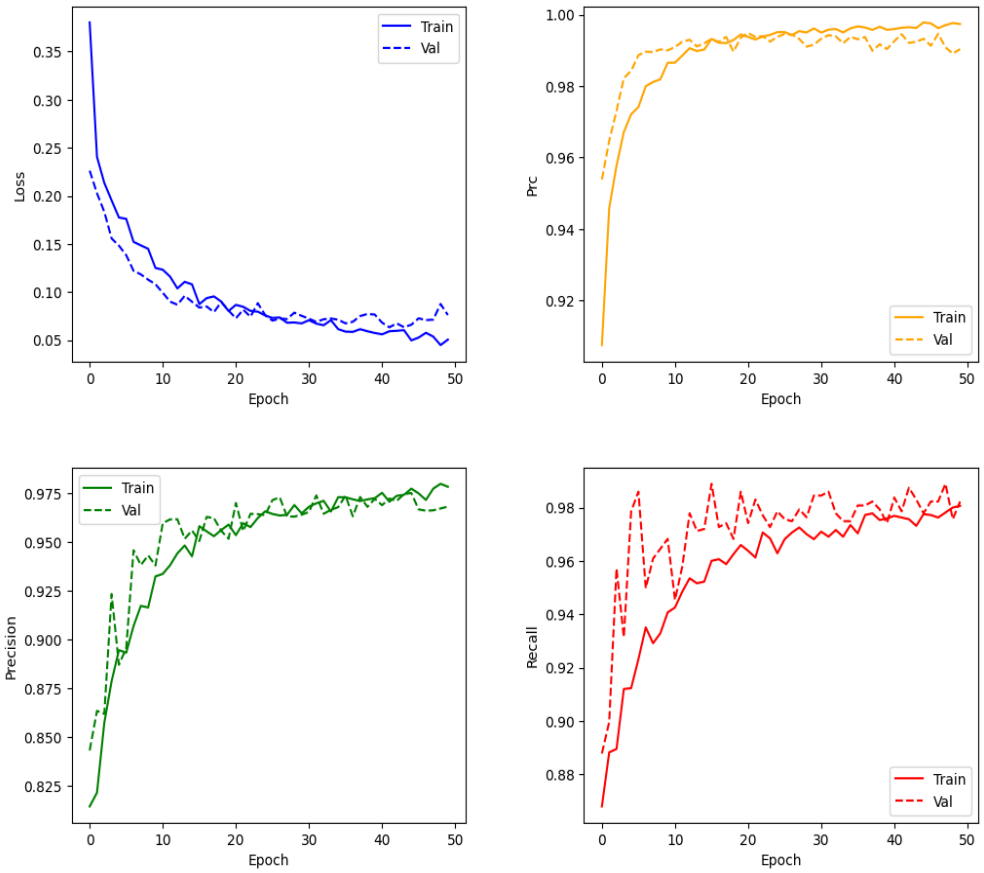


Figure 69: DC to DC boost converter under a DOS and FDI loss for ANN model-2

7.3 LSTM model for Attack detection of 2D input

7.3.1 The average performance measurements of the results

The metrics listed are used for evaluating a binary classification LSTM model's performance comprehensively. Each metric provides unique insights into different aspects of the model's behavior.

Binary Cross entropy	0.171276028606702	Lower values indicate better performance, as it measures the divergence between the predicted probability distribution and the true distribution.
Mean Squared Error	0.030489846938673345	Lower values indicate better performance. It assesses the accuracy of probabilistic predictions
Binary Accuracy	0.9660238731991161	Higher values indicate better performance. It's useful for balanced datasets
Precision	0.9901613593101501	Higher values indicate better performance. It's crucial when the cost of false positives is high.
Recall	0.930474888194691	Higher values indicate better performance. It's crucial when the cost of false negatives is high.
AUC (Area Under the ROC Curve):	0.9832164699381049	Values range starts from 0 to 1. higher values represent better performance. An AUC of 1 represents a perfect model.
AUC (PRC, Area Under the Precision-Recall Curve)	0.9834727428176187	The value range starts from 0 to 1, with higher values representing better performance. An AUC of 1 represents a perfect model.

Table 4: Output of performance of LSTM model

Early stopping is used in training the LSTM learning model to prevent overfitting. It involves monitoring the validation loss during training. It stops the training when the validation loss no longer improves for a specified number of epochs.

randomly deactivating neurons during training is used to reduce overfitting by dropout. This encourages the LSTM to learn more valuable features and prevents lying on neurons. Overall, the LSTM model generalizes better to upcoming data. dropout improves performance on unseen data by avoiding excessive fit to the training set.

7.3.2 Example1: DC to DC boost converter at FDI attack

i) Confusion matrix

A confusion matrix evaluates a model's overall performance by showing correct and incorrect predictions. It helps calculate accuracy, precision, and recall and identifies errors such as false positives and false negatives. This detailed breakdown aids in understanding and improving the model's performance.

A confusion matrix for data from a DC-to-DC boost converter under an FDI (false data injection) attack helps evaluate the model's accuracy in predicting attack instances. It shows the number of true positives, true negatives, false positives, and false negatives, providing insight into the model's performance and its ability to correctly classify attack and non-attack conditions.

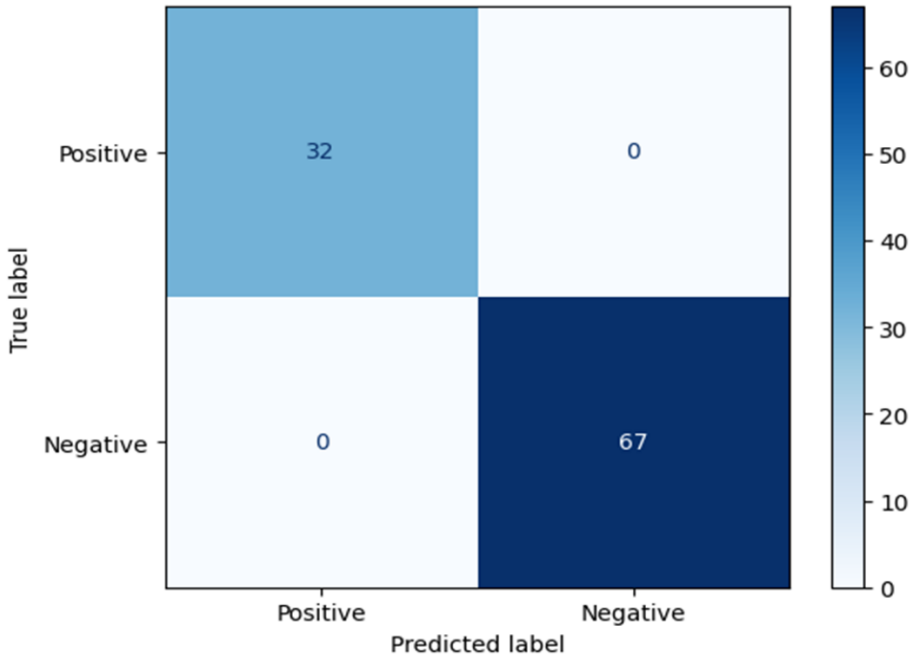


Figure 70: Confusion matrix for under DC-to-DC boost converter at FDI attack for LSTM

True Positive (TP): You predicted positive and it’s true.
True Negative (TN): You predicted negative and it’s true.
False Positive (FP) / Type I Error: You predicted positive and it’s false.
False Negative (FN) / Type II Error: You predicted negative and it’s false

ii) Accuracy vs epoch graph

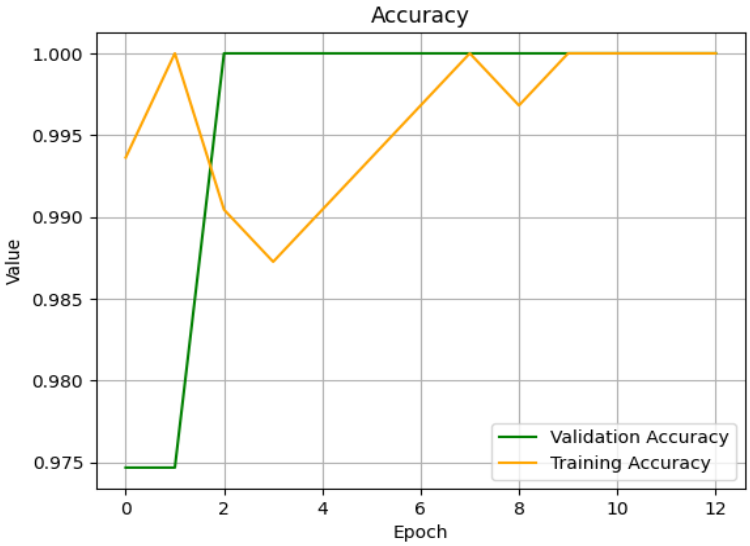


Figure 71: DC to DC boost converter under an FDI (false data injection) accuracy for LSTM

iii) Loss vs epoch graph

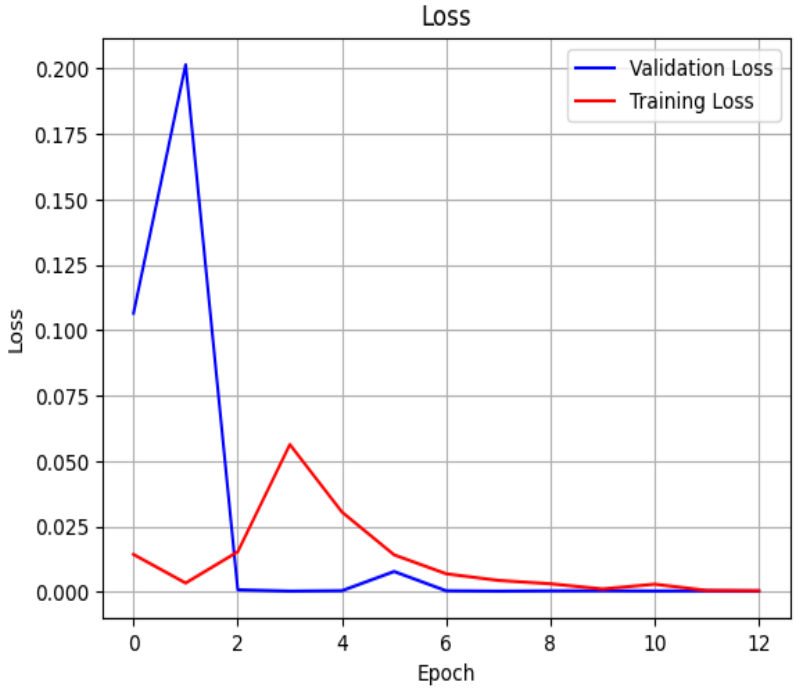


Figure 72: DC to DC boost converter under an FDI (false data injection) accuracy for LSTM

iv) Overall other graphs vs epoch graph

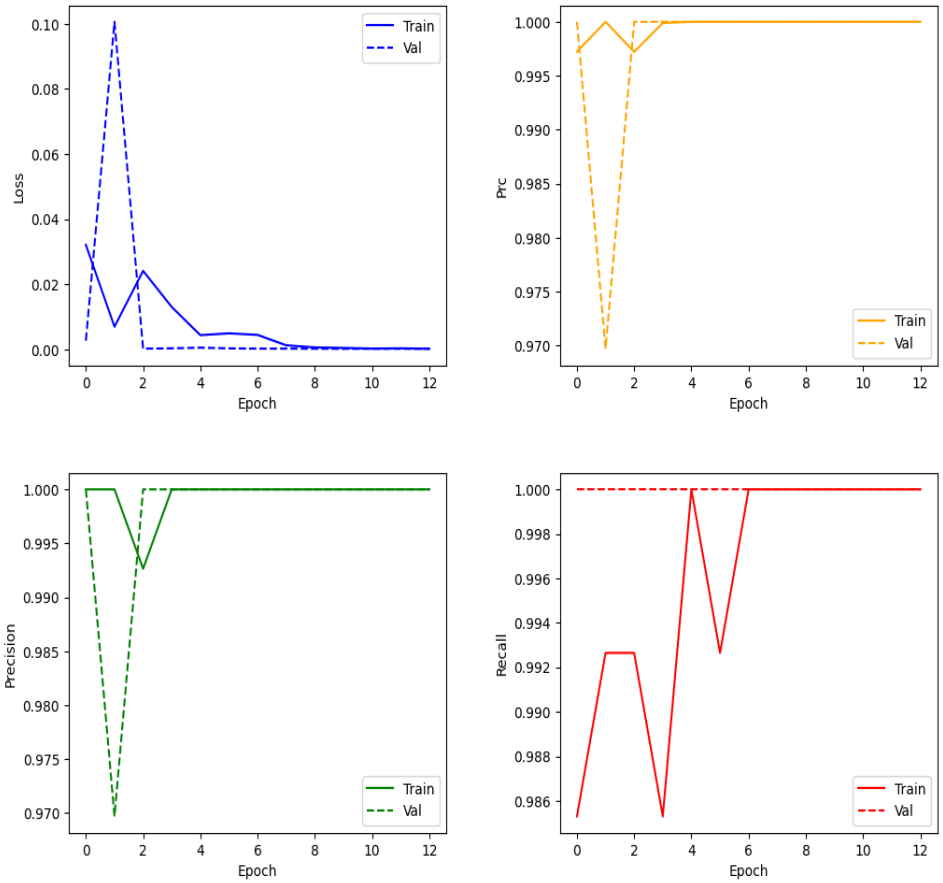


Figure 73: DC to DC boost converter under an FDI (false data injection) overall graphs for LSTM

7.3.3 Example 2: DC to DC boost converter under DOS and FDI attack

These data consist of multiple attacks which include FDI and DOS. It contains 13,481 data sets. It is the largest data set.

i) Confusion matrix

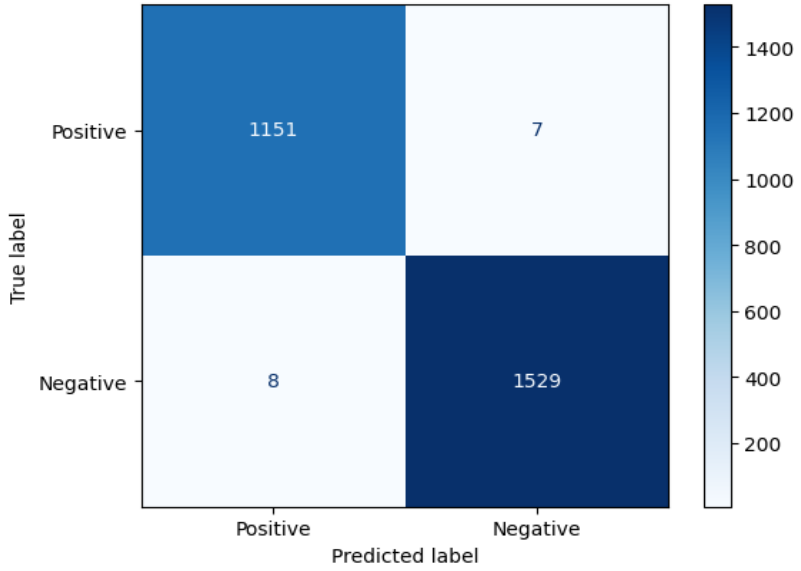


Figure 74: Confusion matrix for under DC-to-DC boost converter under DOS and FDI attack for LSTM model

ii) Accuracy vs epoch graph

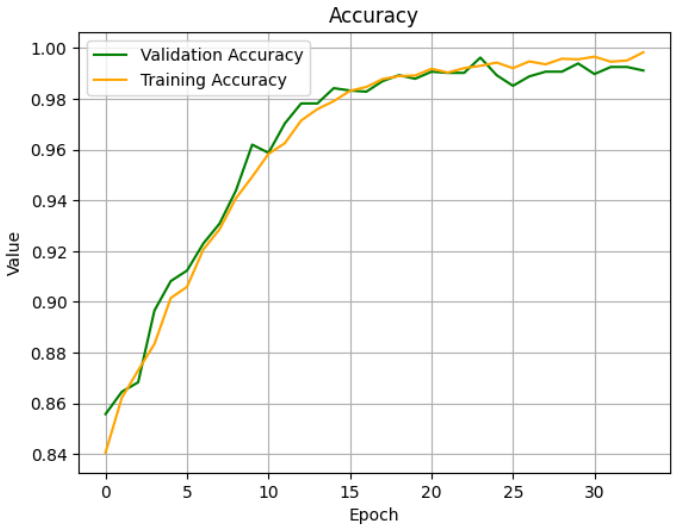


Figure 75: DC to DC boost converter under a DOS and FDI accuracy for LSTM model

iii) Loss vs epoch graph

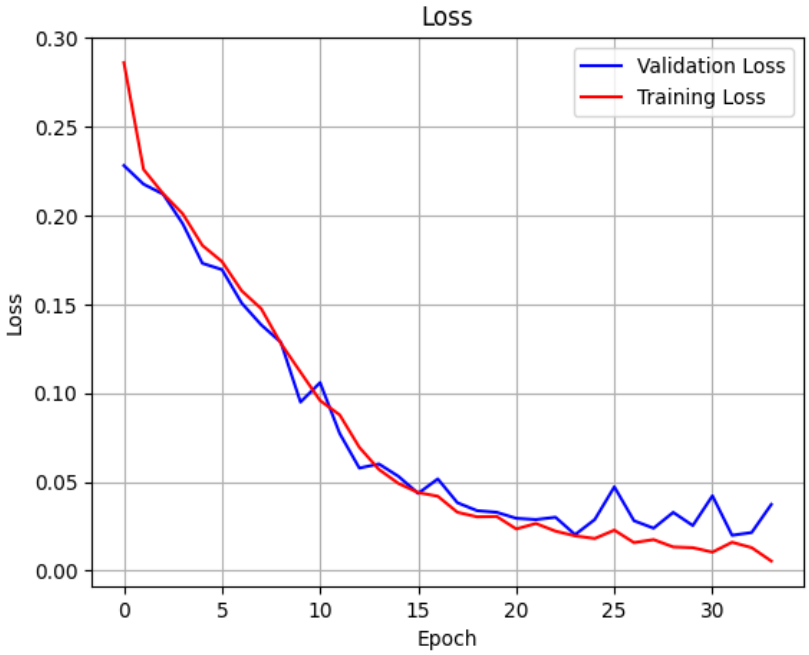


Figure 76: DC to DC boost converter under a DOS and FDI loss for LSTM model

iv) Overall graphs

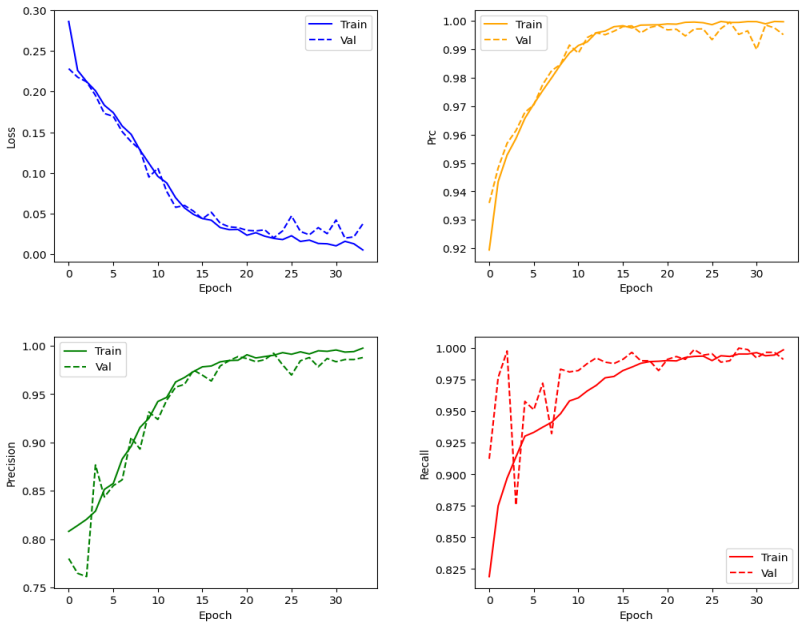


Figure 77: DC to DC boost converter under a DOS and FDI loss for LSTM model

7.4 Classification of FDI and DOS attack by ANN model

7.4.1 The average performance measurements of the results

Categorical Cross entropy, Mean Squared Error, Mean Absolute Error, Categorical Accuracy, Precision, Recall, AUC, and PRC AUC are the parameters monitored to find the performance of the model. Based on this we did a tuning of the model to improve the performance of the classification.

Categorical cross-entropy	0.37810681282112807	Lower values indicate better performance, as it measures the divergence between the predicted probability distribution and the true distribution.
Mean Squared Error	0.045587739754969686	Lower values indicate better performance. It assesses the accuracy of probabilistic predictions
Mean Absolute Error	0.08357285286407344	Higher values indicate better performance. It's useful for balanced datasets
Accuracy	0.9014673943702991	Higher values indicate better performance. It's crucial when the cost of false positives is high.
Precision	0.9014555009511801	Higher values indicate better performance. It's crucial when the cost of false negatives is high.
Recall	0.9010788935881394	Values range starts from 0 to 1. higher values represent better performance. An AUC of 1 represents a perfect model.
AUC (Area Under the ROC Curve):	0.9627486650760357	The value range starts from 0 to 1, with higher values representing better performance. An AUC of 1 represents a perfect model.
AUC (PRC, Area Under the Precision-Recall Curve)	0.9305217839204348	Lower values indicate better performance, as it measures the divergence between the predicted probability distribution and the true distribution.
F1_score	0.924886943388293	**0**: Indicates the worst performance **1**: Indicates the best performance the harmonic mean of precision and recall, balancing two metrics to provide a single measure of the model's performance.

Table 5: Output of performance of classification model

7.4.2 Example: DC to DC boost converter under DOS and FDI attack (largest data set)

It consists of 13 481 data sets which include continuous FDI and DOS attacks we can see the overall performance of the model

i) Confusion matrix

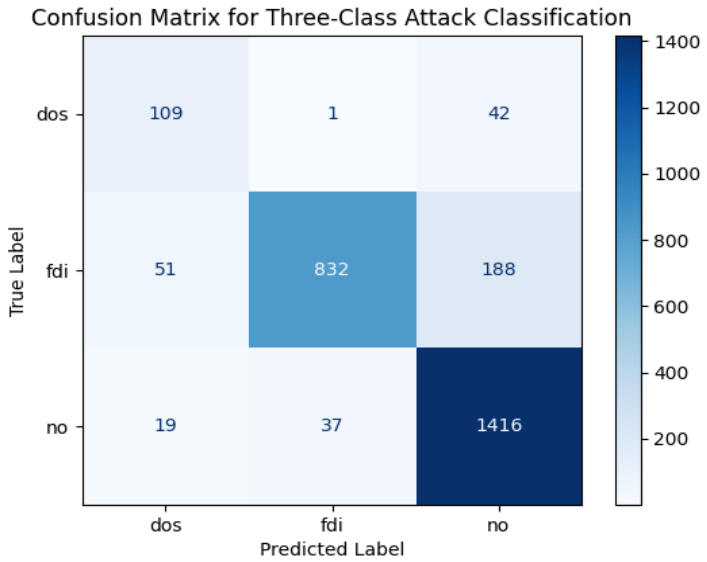


Figure 78: Confusion matrix for under DC-to-DC boost converter under DOS and FDI attack for LSTM model

ii) Accuracy vs epoch graph

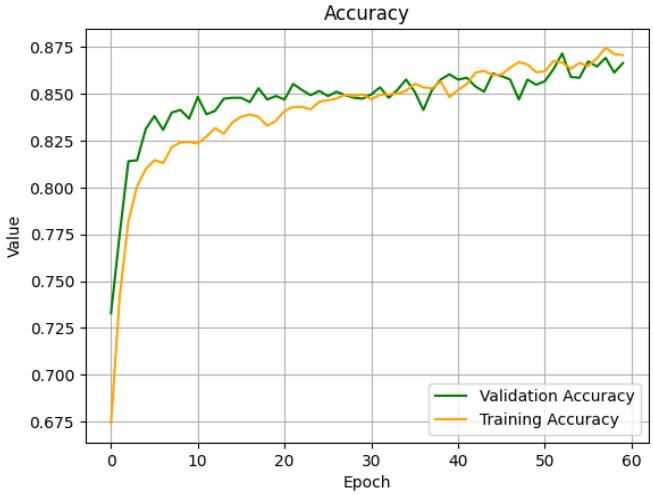


Figure 79: DC to DC boost converter under a DOS and FDI accuracy for classification

iii) Loss vs epoch graph

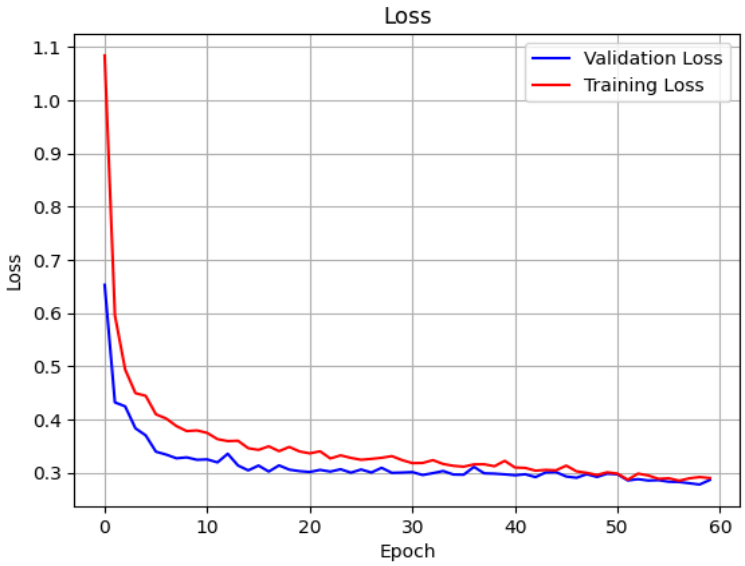


Figure 80: DC to DC boost converter under a DOS and FDI loss for classification

iv) Overall graphs

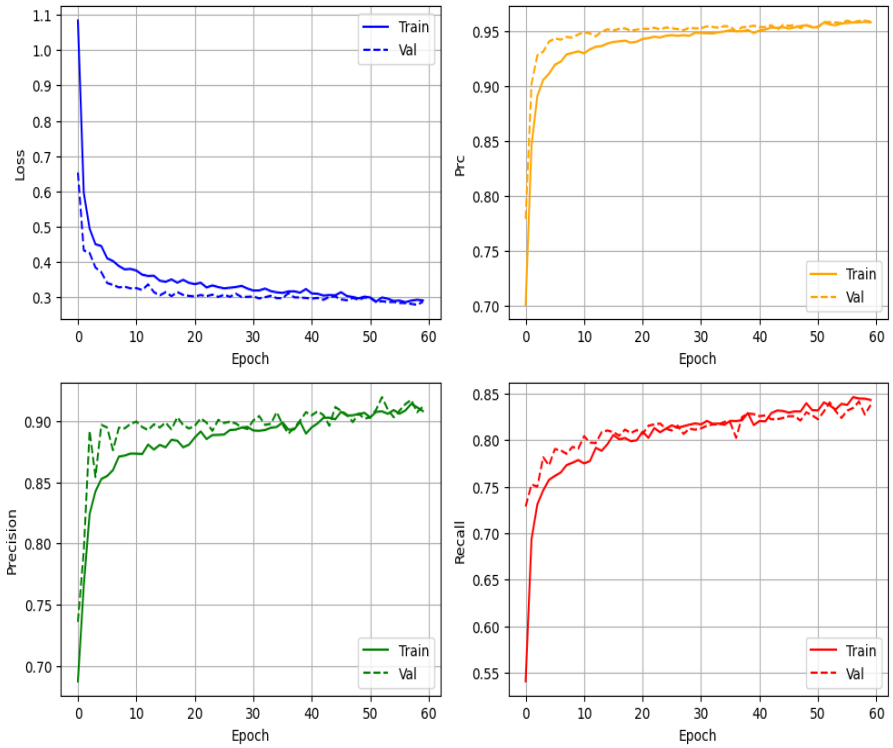


Figure 81: DC to DC boost converter under a DOS and FDI loss for classification model

7.5 Comparison between LSTM and Ann detection model 2

Parameters	ANN detection model 2	LSTM model for detection
Binary Cross entropy	0.2782528533975387	0.171276028606702
Mean Squared Error	0.040759421559616586	0.030489846938673345
Binary Accuracy	0.9532712267504798	0.9660238731991161
Precision	0.9329721497164832	0.9901613593101501
Recall	0.9497611853811476	0.930474888194691
AUC (Area Under the ROC Curve):	0.9782901273833381	0.9832164699381049
AUC (PRC, Area Under the Precision-Recall Curve)	0.9678535925017463	0.9834727428176187

Table 6: Output of performance of comparison of performance

Comparison of average values of parameters which we used to evaluate model performance LSTM model gives higher performance, but the complexity of the LSTM model is high. So, it needs more time to predict the output based on the input data.

So, we prefer the ANN model-2 which has less complexity. accuracy, and other parameters slightly different from the LSTM model for less time needed for detection. LSTM gives high accuracy with high precision so we can prefer when the accuracy of prediction is highly important with the condition of controllers have highly robust change but so it can tolerate the time needed for LSTM prediction.

CHAPTER 8

8. VALIDATION

8.1 Validation for Deep Learning Approaches

The deep learning approaches in this project, specifically ANN and LSTM models, are validated through rigorous data preprocessing and model optimization. Data preprocessing includes handling missing values, normalizing data, and encoding categories, ensuring the input data is clean and standardized. Feature engineering and integration further enhance the model's ability to recognize patterns, leading to more accurate attack detection. The use of correlation matrices helps identify significant features, reducing dimensionality and improving model efficiency.

Model validation also involves extensive hyperparameter tuning and grid search techniques to optimize neural network architectures. The ANN model's design incorporates multiple hidden layers activated by ReLU functions, while the LSTM model's unique memory cell and gating mechanisms address long-term dependencies effectively. By focusing on hyperparameter tuning and early stopping techniques, the project ensures robust model performance, reducing overfitting and enhancing generalization to new data. This comprehensive validation process ensures that deep learning models are well-suited for detecting cyber-attacks in islanded microgrids, improving system reliability and security.

8.2 Possibility of commercialization of our project

The commercialization of our project "Cyber-attack detection of power converters in islanded microgrids using deep learning approaches" has great promise. Our technology addresses the important requirement for increased security in microgrids, providing a scalable way to protect electrical infrastructure. It utilizes powerful deep learning algorithms to ensure real-time detection and mitigation of cyber threats, hence improving reliability and resilience. This breakthrough has the potential to attract energy industry stakeholders, resulting in partnerships, investments, and widespread adoption of smart grid applications.

CHAPTER 9

9. CONCLUSIONS AND RECOMMENDATIONS

This project has developed a unified mathematical model for two major cyber-attack types, Denial of Service (DoS) and False Data Injection (FDI), within microgrid environments, aiming to enhance readiness in mitigating cyber threats to microgrid infrastructures. Detailed simulations in MATLAB enabled the creation of a power converter model for boost conversion and an attack model, pivotal for realistic scenario testing and validation, thus generating datasets crucial for training our cyber-attack detection system.

Our primary goal was to implement a robust cyber-attack detection model using deep learning methodologies, specifically Artificial Neural Networks (ANNs). The ANN model-1 is a basic model that shows low efficiency, achieving only about 70% accuracy in detection due to its use of 1D data, which limits its pattern recognition capability. The ANN model-2 is a higher-performance model, delivering an accuracy of 0.9532712267504798 in detection. This improvement is due to its enhanced data processing method. The LSTM model is more complex and demonstrates very high efficacy, with an accuracy of 0.9660238731991161. It outperforms the previous two models in detection due to its advanced architecture. For classification tasks, the ANN model achieves an accuracy of 0.9014673943702991 in distinguishing between FDI and DOS attacks. Despite its simplicity, it is a high-quality model.

Moving forward, further research and practical implementation should focus on continuously refining simulation models to encompass a broader range of cyber-attack scenarios and operational conditions, integrating more sophisticated attack behaviors. Additionally, extending deep learning models for real-time microgrid monitoring can enhance data collection, analysis, and response to anomalies. Integration with existing cybersecurity frameworks should be explored to ensure broader application and interoperability across microgrid configurations. Extensive validation, including field tests, is necessary to verify the detection system's efficacy under real-world conditions. Promoting awareness of cybersecurity risks among microgrid stakeholders and fostering collaboration to collectively address emerging cyber threats are critical steps in fortifying microgrid infrastructures against cyber-attacks. This project lays a foundational step towards safeguarding microgrid operations amid evolving cybersecurity challenges

REFERENCES

[1] Gupta, K., Sahoo, S., Panigrahi, B. K., Blaabjerg, F., & Popovski, P. (2021). On the Assessment of Cyber Risks and Attack Surfaces in a Real-Time Co-Simulation Cybersecurity Testbed for Inverter-Based Microgrids. *Energies*, *14*(16), 4941. doi: 10.3390/en14164941

[2] J. Zhang, J. Ye and L. Guo, "Model-based Cyber-attack Detection for Voltage Source Converters in Island Microgrids," *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, Vancouver, BC, Canada, 2021, pp. 1413-1418, doi: 10.1109/ECCE47101.2021.9595899.

[3] Q. Li, F. Li, J. Zhang, J. Ye, W. Song and A. Mantooth, "Data-driven Cyberattack Detection for Photovoltaic (PV) Systems through Analyzing Micro-PMU Data," *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*, Detroit, MI, USA, 2020, pp. 431-436, doi: 10.1109/ECCE44975.2020.9236274.

[4] A. S. Musleh, G. Chen and Z. Y. Dong, "A Survey on the Detection Algorithms for False Data Injection Attacks in Smart Grids," in *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2218-2234, May 2020, doi: 10.1109/TSG.2019.2949998.

[5] J. Zhang, J. Ye, L. Guo, F. Li and W. Song, "Vulnerability Assessments for Power-Electronics-Based Smart Grids," *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*, Detroit, MI, USA, 2020, pp. 1702-1707, doi: 10.1109/ECCE44975.2020.9235420.

[6] J. Poon, P. Jain, I. C. Konstantakopoulos, C. Spanos, S. K. Panda and S. R. Sanders, "Model-Based Fault Detection and Identification for Switching Power Converters," in *IEEE Transactions on Power Electronics*, vol. 32, no. 2, pp. 1419-1430, Feb. 2017, doi: 10.1109/TPEL.2016.2541342.

[7] Z. Shahbazi, A. Ahmadi, A. Karimi and Q. Shafiee, "Performance and Vulnerability of Distributed Secondary Control of AC Microgrids under Cyber-Attack," *2021 7th International Conference on Control, Instrumentation and Automation (ICCIA)*, Tabriz, Iran, 2021, pp. 1-6, doi: 10.1109/ICCIA52082.2021.9403548.

[8] Barua, A., & Faruque, M. a. A. (2020). Hall Spoofing: A Non-Invasive DOS attack on Grid-Tied solar inverter. *USENIX Security Symposium*, 1273–1290. doi: 10.5555/3489212.3489284.

[9] J. Liu, B. Cui, B. Chen, X. Lu, F. Qiu and S. Mazumder, "DC Microgrids Under Denial of Service Attacks: Feasibility and Stability Issues," *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*, Detroit, MI, USA, 2020, pp. 424-430, doi: 10.1109/ECCE44975.2020.9236159.

[10] Wang, Y., Zhang, M., Song, K., Li, T., & Zhang, N. (2021). An optimal DOS attack strategy disturbing the distributed economic dispatch of microgrid. *Complexity*, *2021*, 1–16. doi: 10.1155/2021/5539829

[11] Wang, Y., Deng, C., Liu, Y., & Wei, Z. (2023). A cyber-resilient control approach for islanded microgrids under hybrid attacks. *International Journal of Electrical Power & Energy Systems*, *147*, 108889. doi: 10.1016/j.ijepes.2022.108889

[12] Wang, B., Sun, Q., Wang, R., & Dong, C. (2021). Vulnerability analysis of secondary control system when microgrid suffering from sequential denial-of-

service attacks. *IET Energy Systems Integration*, *4*(2), 192–205. doi: 10.1049/esi2.12026

[14] Wang, B., Sun, Q., Han, R., & Ma, D. (2021). Consensus-based secondary frequency control under denial-of-service attacks of distributed generations for microgrids. *Journal of the Franklin Institute*, *358*(1), 114–130. doi: 10.1016/j.jfranklin.2019.01.007

[15] Anwar, A., Mahmood, A. N., & Pickering, M. (2017). Modeling and performance evaluation of stealthy false data injection attacks on smart grid in the presence of corrupted measurements. *Journal of Computer and System Sciences*, *83*(1), 58–72. doi: 10.1016/j.jcss.2016.04.005.

[16] Ahmed, M., Pathan, AS.K. False data injection attack (FDIA): an overview and new metrics for fair evaluation of its countermeasure. *Complex Adapt Syst Model* **8**, 4 (2020). doi: 10.1186/s40294-020-00070-w.

[17] Nejabatkhah, F., Li, Y. W., Liang, H., & Ahrabi, R. R. (2020). Cyber-Security of smart microgrids: A survey. *Energies*, *14*(1), 27. doi: 10.3390/en14010027

[18] Li, P., Zhang, F., Yang, Y., Ma, X., Yao, S., Yang, P., Zhao, Z., Lai, C. S., & Lai, L. L. (2022). The integrated modeling of microgrid cyber physical system based on hybrid automaton. *Frontiers in Energy Research*, *10*. doi: 10.3389/fenrg.2022.748828.

[19] S. M. Mohiuddin and J. Qi, "Attack Resilient Distributed Control for AC Microgrids with Distributed Robust State Estimation," *2021 IEEE Texas Power and Energy Conference (TPEC)*, College Station, TX, USA, 2021, pp. 1-6, doi: 10.1109/TPEC51183.2021.9384912.

[20] Y. Chen, D. Qi, H. Dong, C. Li, Z. Li and J. Zhang, "A FDI Attack-Resilient Distributed Secondary Control Strategy for Islanded Microgrids," in *IEEE Transactions on Smart Grid*, vol. 12, no. 3, pp. 1929-1938, May 2021, doi: 10.1109/TSG.2020.3047949.

APPENDICES

ANN-model-1 detection

```
import tensorflow as tf
print(tf.__version__)
```

```
2.10.0
```

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from  keras.models import load_model
```

```
from tensorflow import keras

data = pd.read_csv('Dos_healthy.csv')

data.head()

plt.plot(data['Time'],data['DC_boost1'])

plt.plot(data['Time'],data['DC_boost2'])

plt.plot(data['Time'],data['DC_boost3'])

plt.plot(data['Time'],data['AC_boost1'])

plt.plot(data['Time'],data['AC_boost2'])

data.info()

df_filled = data.fillna(data.mean())

X = df_filled.drop('Attack', axis=1)  # Adjust 'target_column' to your label column
y = df_filled['Attack']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50)

# Standardize the data (optional, but often beneficial for neural networks)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train.shape[1]
```

```
10
```

```
# Build the neural network model
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(128, activation='relu'))  # Additional hidden layer
model.add(Dense(64, activation='relu'))   # Additional hidden layer
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.load_weights('weights_file.h5')
```

```
METRICS = [
      keras.metrics.BinaryCrossentropy(name='cross entropy'),  # same as model's loss
      keras.metrics.MeanSquaredError(name='Brier score'),
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=METRICS)

# Train the model
history_model=model.fit(X_train_scaled, y_train, epochs=20, batch_size=32, validation_split=0.3)
```

Show hidden output

```
# Make predictions on the test set
y_pred = model.predict(X_test_scaled)
y_pred_binary = (y_pred > 0.5).astype(int)
```

```
8/8 [==============================] - 0s 2ms/step
```

```
accuracy = accuracy_score(y_test, y_pred_binary)

accuracy
```

```
1.0
```

```
confusion_matrix(y_test,y_pred_binary)

def plot_cm(labels, predictions, threshold=0.5):
  cm = confusion_matrix(labels, np.round(predictions))
  plt.figure(figsize=(5,5))
  sns.heatmap(cm, annot=True, fmt="d")
  plt.title('Confusion matrix ')
  plt.ylabel('Actual label')
  plt.xlabel('Predicted label')

plot_cm(y_test, y_pred)

plt.plot(history_model.history['val_loss'])
plt.title('val_loss ')
plt.ylabel('val_loss')
plt.xlabel('epoc')

plt.plot(history_model.history['val_accuracy'])
plt.ylabel('val_accuracy')
plt.xlabel('epoc')

def plot_metrics(history):
  metrics = ['loss', 'prc', 'precision', 'recall']
  colors = ['blue', 'orange', 'green', 'red']
  for n, metric in enumerate(metrics):
    name = metric.replace("_"," ").capitalize()
    plt.subplot(2,2,n+1)
    plt.plot(history.epoch, history.history[metric], color=colors[n], label='Train')
    plt.plot(history.epoch, history.history['val_'+metric],
             color=colors[n], linestyle="--", label='Val')
    plt.xlabel('Epoch')
    plt.ylabel(name)

    plt.legend()

plt.plot(history_model.epoch, history_model.history['loss'], color='blue', label='Train')

plot_metrics(history_model)


# Calculate the correlation matrix
correlation_matrix = data.corr()

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()


# Assuming 'history' contains the training history collected during model training

# Plot training & validation accuracy values
plt.plot(history_model.history['accuracy'])
plt.plot(history_model.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

ANN-model-2 detection

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os
os.chdir('/content/drive/My Drive/detection/')
```

```
import tensorflow as tf
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
```

```
from tensorflow import keras
from keras.layers import Dense, Dropout, Reshape
from collections import Counter
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from  keras.models import load_model
```

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from keras.callbacks import EarlyStopping
from sklearn.metrics import roc_curve, auc
```

```
accurac=[]
```

```
  # Drop "Time" column
  df.drop(columns=["Time"], inplace=True)

  # Define the number of rows per group
  group_size = 10

  # Create empty lists to store data matrices and labels
  data_matrices = []
  labels = []

  # Iterate over the rows in the DataFrame
  for i in range(len(df) - group_size + 1):
      # Select rows from i to i+9
      selected_rows = df.iloc[i:i+group_size, :]

      # Drop the "Attack" column
      selected_rows_no_attack = selected_rows.drop(columns=["Attack"])

      # Convert the selected rows to a numpy array
      data_matrix = selected_rows_no_attack.to_numpy()

      # Standardize the data matrix
      scaler = StandardScaler()
      data_matrix_scaled = scaler.fit_transform(data_matrix)

      # Append the scaled data matrix to data_matrices
      data_matrices.append(data_matrix_scaled)

      # Extract the mode (most common value) from the "Attack" column within each group
      mode_attack = Counter(selected_rows["Attack"]).most_common(1)[0][0]

      # Append the mode to labels
      labels.append(mode_attack)

  # Convert lists to numpy arrays
  data_matrices = np.array(data_matrices)
  labels = np.array(labels)

  # Print the data matrices and labels arrays
  print("Data Matrices:")
  print(data_matrices)
  print("Labels:")
  print(labels)
```

```
Show hidden output
```

```
  df.head()
```

```
Show hidden output
```

```
  # Define the ANN model

  def create_ann_model(input_shape):
      model = tf.keras.Sequential([
          tf.keras.layers.Flatten(input_shape=input_shape),
          tf.keras.layers.Dense(128, activation='relu'),
          tf.keras.layers.Dropout(0.2),  # Added dropout layer with 20% dropout rate
          tf.keras.layers.Dense(64, activation='relu'),
          tf.keras.layers.Dropout(0.2),  # Added dropout layer with 20% dropout rate
          tf.keras.layers.Dense(32, activation='relu'),
          tf.keras.layers.Dropout(0.2),  # Added dropout layer with 20% dropout rate
          tf.keras.layers.Dense(1, activation='sigmoid')
      ])
      return model
```

```
METRICS = [
      keras.metrics.BinaryCrossentropy(name='cross entropy'),  # same as model's loss
      keras.metrics.MeanSquaredError(name='Brier score'),
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]


# Split data into training and testing sets (you can use other methods for splitting as well)

X_train, X_test, y_train, y_test = train_test_split(data_matrices, labels, test_size=0.2, random_state=42)

# Load the pre-trained model
input_shape = X_train.shape[1:]
model = create_ann_model(X_train.shape[1:])  # Create the model with the same architecture
model.load_weights("updated_weights.h5")  # Load pre-trained weights

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=METRICS)
#early stoping
early_stopping = EarlyStopping(
    monitor='val_loss',
    min_delta=0.0001,
    patience=15,
    verbose=1,
    mode='min',
    restore_best_weights=True
)

#Train the model (assuming X_train and y_train are already defined)
history_model=model.fit(X_train, y_train, epochs=50, batch_size=20, validation_split=0.3,callbacks=[early_stopping])

# Save the updated weights
model.save_weights("updated_weights.h5")
```

Show hidden output

```
model.evaluate(X_train, y_train)
```

Show hidden output

```
def plot_metrics(history):
  metrics = ['loss', 'prc', 'precision', 'recall']
  colors = ['blue', 'orange', 'green', 'red']
  for n, metric in enumerate(metrics):
    name = metric.replace("_"," ").capitalize()
    plt.subplot(2,2,n+1)
    plt.plot(history.epoch, history.history[metric], color=colors[n], label='Train')
    plt.plot(history.epoch, history.history['val_'+metric],
             color=colors[n], linestyle="--", label='Val')
    plt.xlabel('Epoch')
    plt.ylabel(name)

    plt.legend()


def plot_metrics(history):
    metrics = ['loss', 'prc', 'precision', 'recall']
    colors = ['blue', 'orange', 'green', 'red']

    # Create a figure and axes
    fig, axs = plt.subplots(2, 2, figsize=(12, 10))  # Adjust figsize as needed

    for n, metric in enumerate(metrics):
        name = metric.replace("_", " ").capitalize()
        ax = axs[n // 2, n % 2]  # Get the correct subplot

        ax.plot(history.epoch, history.history[metric], color=colors[n], label='Train')
        ax.plot(history.epoch, history.history['val_' + metric],
                color=colors[n], linestyle="--", label='Val')
```

```
            ax.set_xlabel('Epoch')
            ax.set_ylabel(name)
            ax.legend()

        # Adjust spacing between subplots
        plt.subplots_adjust(wspace=0.3, hspace=0.3)  # Adjust wspace and hspace as needed

        plt.show()


plot_metrics(history_model)
```

Show hidden output

```
plt.plot(history_model.history['val_loss'], label='Validation Loss', color='blue')
plt.plot(history_model.history['loss'], label='Training Loss', color='red')
plt.title('Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.grid(True)
plt.legend()
plt.show()
```

Show hidden output

```
plt.plot(history_model.history['val_accuracy'], label='Validation Accuracy', color='green')
plt.plot(history_model.history['accuracy'], label='Training Accuracy', color='orange')

plt.title('Accuracy')
plt.ylabel('Value')
plt.xlabel('Epoch')
plt.grid(True)
plt.legend()
plt.show()
```

Show hidden output

```
# Generate predictions and calculate the ROC curve
y_pred_prob = model.predict(X_test).ravel()
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:0.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

LSTM model

```
from google.colab import drive
drive.mount('/content/drive')
import os
os.chdir('/content/drive/My Drive/LSTM_detection/')
```

```
Mounted at /content/drive
```

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout


import tensorflow as tf
from sklearn.model_selection import train_test_split
import pandas as pd


from tensorflow import keras
from keras.layers import Dense, Dropout, Reshape
from collections import Counter


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from  keras.models import load_model


import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from keras.callbacks import EarlyStopping
from sklearn.metrics import roc_curve, auc
```

```
# Read the CSV file into a pandas DataFrame
df = pd.read_csv("2hour_attackornot.csv")

# Drop "Time" column
df.drop(columns=["Time"], inplace=True)

# Define the number of rows per group
group_size = 10

# Create empty lists to store data matrices and labels
data_matrices = []
labels = []

# Iterate over the rows in the DataFrame
for i in range(len(df) - group_size + 1):
    # Select rows from i to i+9
    selected_rows = df.iloc[i:i+group_size, :]

    # Drop the "Attack" column
    selected_rows_no_attack = selected_rows.drop(columns=["Attack"])

    # Convert the selected rows to a numpy array
    data_matrix = selected_rows_no_attack.to_numpy()

    # Standardize the data matrix
    scaler = StandardScaler()
    data_matrix_scaled = scaler.fit_transform(data_matrix)

    # Append the scaled data matrix to data_matrices
    data_matrices.append(data_matrix_scaled)

    # Extract the mode (most common value) from the "Attack" column within each group
    mode_attack = Counter(selected_rows["Attack"]).most_common(1)[0][0]

    # Append the mode to labels
    labels.append(mode_attack)

# Convert lists to numpy arrays
data_matrices = np.array(data_matrices)
labels = np.array(labels)

# Print the data matrices and labels arrays
print("Data Matrices:")
print(data_matrices)
print("Labels:")
print(labels)
```

```
X_train, X_test, y_train, y_test = train_test_split(data_matrices, labels, test_size=0.2, random_state=42)
```

```
def create_rnn_model(input_shape):
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=input_shape))  # First LSTM layer
    model.add(Dropout(0.2))  # Dropout layer to prevent overfitting
    model.add(LSTM(50, return_sequences=True))  # Second LSTM layer
    model.add(Dropout(0.2))  # Dropout layer
    model.add(LSTM(50, return_sequences=True))  # Second LSTM layer
    model.add(Dropout(0.2))  # Dropout layer
    model.add(LSTM(50))  # Third LSTM layer
    model.add(Dense(1, activation='sigmoid'))  # Binary classification, so using sigmoid activation
    return model


METRICS = [
      keras.metrics.BinaryCrossentropy(name='cross entropy'),  # same as model's loss
      keras.metrics.MeanSquaredError(name='Brier score'),
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]


# Split data into training and testing sets (you can use other methods for splitting as well)
```



```
X_train, X_test, y_train, y_test = train_test_split(data_matrices, labels, test_size=0.2, random_state=42)

# Load the pre-trained model
input_shape = X_train.shape[1:]
model = create_rnn_model(X_train.shape[1:])  # Create the model with the same architecture
model.load_weights("updated_weights_rnn.h5")  # Load pre-trained weights

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=METRICS)
#early stoping
early_stopping = EarlyStopping(
    monitor='val_loss',
    min_delta=0.001,
    patience=10,
    verbose=1,
    mode='min',
    restore_best_weights=True
)

#Train the model (assuming X_train and y_train are already defined)
history_model=model.fit(X_train, y_train, epochs=50, batch_size=20, validation_split=0.2,callbacks=[early_stopping] )

# Save the updated weights
model.save_weights("updated_weights_rnn.h5")


results=model.evaluate(X_test, y_test)


def plot_metrics(history):
    metrics = ['loss', 'prc', 'precision', 'recall']
    colors = ['blue', 'orange', 'green', 'red']

    # Create a figure and axes
    fig, axs = plt.subplots(2, 2, figsize=(12, 10))  # Adjust figsize as needed

    for n, metric in enumerate(metrics):
        name = metric.replace("_", " ").capitalize()
        ax = axs[n // 2, n % 2]  # Get the correct subplot

        ax.plot(history.epoch, history.history[metric], color=colors[n], label='Train')
        ax.plot(history.epoch, history.history['val_' + metric],
                color=colors[n], linestyle="--", label='Val')
        ax.set_xlabel('Epoch')
        ax.set_ylabel(name)
        ax.legend()
```

```
plot_metrics(history_model)
```

```
plt.plot(history_model.history['val_loss'], label='Validation Loss', color='blue')
plt.plot(history_model.history['loss'], label='Training Loss', color='red')
plt.title('Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.grid(True)
plt.legend()
plt.show()
```

```
plt.plot(history_model.history['val_accuracy'], label='Validation Accuracy', color='green')
plt.plot(history_model.history['accuracy'], label='Training Accuracy', color='orange')

plt.title('Accuracy')
plt.ylabel('Value')
plt.xlabel('Epoch')
plt.grid(True)
plt.legend()
plt.show()
```

```
# Generate predictions and calculate the ROC curve
y_pred_prob = model.predict(X_test).ravel()
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:0.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

ANN classification -1(FNN)

```
import tensorflow as tf
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np


from tensorflow import keras
from keras.layers import Dense, Dropout, Reshape
from collections import Counter



%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from keras.callbacks import EarlyStopping
from sklearn.metrics import roc_curve, auc


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from  keras.models import load_model


cross=[]
brier=[]
mae=[]
accurac=[]
pre=[]
reca=[]
auc=[]
prc=[]
f1=[]


filtered_indices1= [i for i, value in enumerate(f1) if value >= 0.64]
f1_1 = [f1[i] for i in filtered_indices1]


# average value of performance eavaluvators
filtered_indices = [i for i, value in enumerate(accurac) if value >= 0.64]

# Filter the lists based on the filtered indices
cross1 = [cross[i] for i in filtered_indices]
brier1 = [brier[i] for i in filtered_indices]
mae1 = [mae[i] for i in filtered_indices]
accurac1 = [accurac[i] for i in filtered_indices]
pre1 = [pre[i] for i in filtered_indices]
reca1 = [reca[i] for i in filtered_indices]
auc1 = [auc[i] for i in filtered_indices]
prc1 = [prc[i] for i in filtered_indices]
#f1_1 = [f1[i] for i in filtered_indices]

# Print the updated lists with their means
print("cross:", np.mean(cross1))
print("brier:", np.mean(brier1))
print("mae:", np.mean(mae1))
print("accurac:", np.mean(accurac1))
print("pre:", np.mean(pre1))
print("reca:", np.mean(reca1))
print("auc:", np.mean(auc1))
print("prc:", np.mean(prc1))
#print("f1:", np.mean(f1_1))
```

```
df_fill = pd.read_csv("DOS_batteryinverter.csv")


# Identify numerical and non-numeric columns
numeric_cols = df_fill.select_dtypes(include='number').columns
non_numeric_cols = df_fill.select_dtypes(exclude='number').columns


df_fill.head()


means = df_fill[numeric_cols].mean()

# Fill missing values in numerical columns
df_numeric_filled = df_fill[numeric_cols].fillna(value=means)


# Extract non-numeric columns from the original DataFrame
df_non_numeric = df_fill[non_numeric_cols]

# Combine the filled numerical columns with the original non-numeric columns
df_filled = pd.concat([df_numeric_filled, df_non_numeric], axis=1)
df=df_filled


df_filled.head()



# Read the CSV file into a pandas DataFrame

# Drop "Time" column
df.drop(columns=["Time"], inplace=True)

# Define the number of rows per group
group_size = 10

# Create empty lists to store data matrices and labels
data_matrices = []
labels = []

# Iterate over the rows in the DataFrame
for i in range(len(df) - group_size + 1):
    # Select rows from i to i+9
    selected_rows = df.iloc[i:i+group_size, :]

    # Drop the "Attack" column
    selected_rows_no_attack = selected_rows.drop(columns=["Attack"])

    # Convert the selected rows to a numpy array
    data_matrix = selected_rows_no_attack.to_numpy()

    # Standardize the data matrix
    scaler = StandardScaler()
    data_matrix_scaled = scaler.fit_transform(data_matrix)

    # Append the scaled data matrix to data_matrices
    data_matrices.append(data_matrix_scaled)

    # Extract the mode (most common value) from the "Attack" column within each group
    mode_attack = Counter(selected_rows["Attack"]).most_common(1)[0][0]

    # Append the mode to labels
    labels.append(mode_attack)

# Convert lists to numpy arrays
data_matrices = np.array(data_matrices)
labels = np.array(labels)

# Print the data matrices and labels arrays
print("Data Matrices:")
print(data_matrices)
print("Labels:")
print(labels)
```

```
def create_ann_model(input_shape):
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=input_shape),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(0.3),  # Added dropout layer with rate 0.3
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.3),  # Added dropout layer with rate 0.3
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dropout(0.3),  # Added dropout layer with rate 0.3
        tf.keras.layers.Dense(3, activation='softmax')  # Output layer for multi-class classification
    ])
    return model


METRICS = [
      keras.metrics.CategoricalCrossentropy(name='cross_entropy'),  # same as model's loss
      keras.metrics.MeanSquaredError(name='Brier score'),
      keras.metrics.MeanAbsoluteError(name='mae'),
      keras.metrics.CategoricalAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'),# precision-recall curv
]


X_train, X_test, y_train, y_test = train_test_split(data_matrices, labels, test_size=0.2, random_state=42)

# Load the pre-trained model
input_shape = X_train.shape[1:]
model = create_ann_model(X_train.shape[1:])  # Create the model with the same architecture
model.load_weights("updated_weights_class.h5")  # Load pre-trained weights

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=METRICS)

# Early stopping for validation loss
early_stopping_val_loss = EarlyStopping(
    monitor='val_loss',  # Monitor the validation loss
    min_delta=0.0001,
    patience=15,
    verbose=1,
    mode='min',  # Use 'min' because you want to minimize the validation loss
    restore_best_weights=True
)

# Early stopping for validation Brier score
early_stopping_val_brier = EarlyStopping(
    monitor='val_mae',  # Monitor the validation Brier score
    min_delta=0.0001,
    patience=15,
    verbose=1,
    mode='min',  # Use 'min' because you want to minimize the Brier score
   restore_best_weights=True
)
callbacks = [early_stopping_val_loss, early_stopping_val_brier]
#Train the model (assuming X_train and y_train are already defined)
history_model=model.fit(X_train, y_train, epochs=60, batch_size=20, validation_split=0.2,callbacks=callbacks)
```

```
#Save the updated weights
model.save_weights("updated_weights_class.h5")
```

```
results = model.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 13ms/step - loss: 0.1544 - cross_entropy: 0.1544 - Brier score: 0.0275 - mae: 0.0797 - acc
```

```
cross.append(results[0])
brier.append(results[2])
mae.append(results[3])
accurac.append(results[4])
pre.append(results[5])
reca.append(results[6])
auc.append(results[7])
prc.append(results[8])
```

```
y_pred =model.predict(X_test)
```

```
4/4 [==============================] - 0s 4ms/step
```

```
def plot_metrics(history):
    metrics = ['loss', 'prc', 'precision', 'recall']
    colors = ['blue', 'orange', 'green', 'red']
    plt.figure(figsize=(10, 10))  # Increase figure size if needed
    for n, metric in enumerate(metrics):
        name = metric.replace("_"," ").capitalize()
        plt.subplot(3, 2, n+1)
        plt.plot(history.epoch, history.history[metric], color=colors[n], label='Train')
        plt.plot(history.epoch, history.history['val_' + metric], color=colors[n], linestyle="--", label='Val')
        plt.xlabel('Epoch')
        plt.grid(True)
        plt.ylabel(name)
        plt.legend()
    plt.tight_layout()  # Adjusts the padding between and around subplots
    plt.show()
```

```
plot_metrics(history_model)
```

```
plt.plot(history_model.history['val_accuracy'], label='Validation Accuracy', color='green')
plt.plot(history_model.history['accuracy'], label='Training Accuracy', color='orange')

plt.title('Accuracy')
plt.ylabel('Value')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```

```
plt.plot(history_model.history['val_loss'], label='Validation Loss', color='blue')
plt.plot(history_model.history['loss'], label='Training Loss', color='red')
plt.title('Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.grid(True)
plt.legend()
plt.show()
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Sample one-hot encoded true labels and predicted labels
y_true = np.array(y_test)  # Replace with your actual one-hot encoded true labels

y_pred = np.array(model.predict(X_test))  # Replace with your actual one-hot encoded predicted labels

# Convert one-hot encoded labels to class labels
y_true_labels = np.argmax(y_true, axis=1)
y_pred_labels = np.argmax(y_pred, axis=1)

# Compute confusion matrix
cm = confusion_matrix(y_true_labels, y_pred_labels, labels=[0,1,2])
```

```
# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['dos', 'fdi', 'no'])
disp.plot(cmap=plt.cm.Blues)
plt.grid(False)
plt.title('Confusion Matrix for Three-Class Attack Classification')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
from sklearn.metrics import f1_score

# Ensure y_true and y_pred are binary indicator matrices
y_true = np.array(y_test)  # True labels (one-hot encoded)
y_pred = np.round(model.predict(X_test))  # Convert predicted probabilities to binary predictions

# Calculate F1 score per class
f1_per_class = f1_score(y_true_labels, y_pred_labels, average=None)
print("F1 score per class:", f1_per_class)

# Calculate F1 score for the overall model
f1_overall = f1_score(y_true_labels, y_pred_labels, average='weighted')  # 'macro', 'micro', or 'weighted'
print("Overall F1 score:", f1_overall)
```

```
4/4 [==============================] - 0s 3ms/step
F1 score per class: [0.97175141 0.76190476]
Overall F1 score: 0.952674444199868
```

```
f1.append(f1_overall)
```