Reinforcement Learning-based False Data Injection Attack on Smart Distribution Systems

A thesis submitted in partial fulfillment of the requirements for
the award of the degree of

B.Tech.

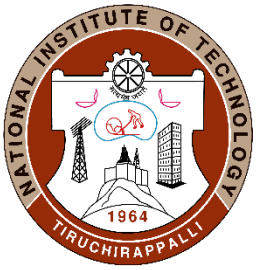
in

Electrical and Electronics Engineering

By

Akshaiy Thinakaran (107121006)

Atharva Rathi (107121016)



DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI-620015

MAY 2025

BONAFIDE CERTIFICATE

This is to certify that the project titled **"Reinforcement Learning-based False Data Injection Attack on Smart Distribution Systems"** is a bonafide record of the work done by

Akshaiy Thinakaran (107121006)
Atharva Rathi (107121016)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electrical and Electronics Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI,** during the year 2025.

Dr. Dipanshu Naware
Project Guide

Head of the Department

Project Viva-voce held on _____

Internal Examiner **External Examiner**

ABSTRACT

Smart distribution systems are witnessing increased deployment due to their improved grid efficiency and reliability. These systems rely heavily on accurate meter readings for real-time monitoring and control, but the growing dependence on communication networks makes them vulnerable to cyber-physical threats such as false data injection attacks (FDIAs). Moreover, the need for data-driven technologies such as artificial intelligence (AI), machine learning (ML), and deep learning (DL) in various sectors has been rising for over a decade, specifically in the energy sector. To address this challenge, this work proposes a reinforcement learning based framework for FDIA detection and mitigation. In the proposed approach, there is an agent that acts as an adversary, learning to simulate and discover stealthy FDIA strategies, while the other agent serves as a defender, tasked with detecting and localising these attacks. This continuous adversarial training enables the defender agent to adapt dynamically to new and previously unseen FDIA patterns. Simulation results show that the RL-based defender provides better accuracy and reliability than traditional offline detection methods. The overall findings demonstrate the effectiveness of the proposed framework in building resilient smart distribution systems and highlight the importance of adaptive, data-driven defence strategies in securing modern power grids.

Keywords: Cybersecurity, False data injection attack (FDIA), Reinforcement learning (RL), Smart distribution system.

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to the following individuals for their invaluable guidance and support throughout this course. Without their encouragement and assistance, this project and the outcomes achieved would not have been possible.

We are profoundly thankful to our project guide **Dr. Dipanshu Naware**, Assistant Professor, Department of Electrical and Electronics Engineering, for his constant support, insightful guidance, and encouragement during this project. His knowledge, patience, and positive demeanour have been a continuous source of motivation and inspiration.

We are grateful to **Dr. M.P. Selvan**, Professor, Department of Electrical and Electronics Engineering, and **Dr. (Mrs.) S. Mageshwari,** Assistant Professor, Department of Electrical and Electronics Engineering, our internal reviewers, for their valuable insights and advice provided during the review sessions.

We extend our heartfelt thanks to **Dr. Sishaj P Simon**, Head of the Department, Department of Electrical and Electronics Engineering, for providing us with the necessary facilities and a conducive environment to carry out this project work.

We would also like to express our sincere appreciation to **Dr. G. Aghila**, Director, NIT Tiruchirappalli, for facilitating the infrastructure and resources essential for the successful completion of this project.

Our gratitude also goes to all the faculty and staff members of the Department of Electrical and Electronics Engineering, our parents, and friends, whose unwavering support and encouragement have been instrumental throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Result

ABBREVIATIONS

AI	Artificial intelligence
ANN	Artificial Neural Networks
CV	Computer Vision
DNN	Deep Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
FDIA	False Data Injection Attack
GCN	Graph Convolutional Network
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IOT	Internet of Things
LSTM	Long Short-Term Memory
DSO	Distribution System Operator
ML	Machine Learning
PPO	Proximal Policy Optimization
RDN	Radial Distribution Network
RL	Reinforcement Learning
RMSE	Root Mean Square Error
SCADA	Supervisory Control and Data Acquisition
MBE	Mean Bias Error

CHAPTER 1

INTRODUCTION

This chapter offers a fundamental synopsis of the internship project, covering the theoretical underpinnings, main goals, and particular contributions achieved. It lays the groundwork for comprehending the development objectives and problem environment.

1.1. OBJECTIVES

The primary objectives of this project are as follows:

- **To Implement Forecasting Models for Day-Ahead Load Prediction:** Incorporate an LSTM-based forecasting model to predict day-ahead load demand values in the system. The model will provide baseline load predictions, which will serve as input to both the attacker and defender agents, ensuring that attack strategies are evaluated using realistic, context-aware power demand forecasts.
- **To design an FDIA Attack Model:** Develop a reinforcement learning-based attack agent that can generate stealthy and effective false data injection attacks on a smart distribution system.
- **To Develop a Robust FDIA Detection Framework:** Develop a detection mechanism that can effectively identify false data and adapt to previously unseen attack strategies.
- **Integration of Attacker and Defender**: Develop a framework where the attacker and defender agents learn and improve their strategies, allowing both agents to improve their behaviour in response to the actions of the other.

1.2. CONTRIBUTIONS

This thesis offers several significant advancements in the field of cybersecurity in smart grids:`

- **Development of an LSTM-Based Load Forecasting Model**: A Long Short-Term Memory (LSTM) based load forecasting model was developed to predict electricity demand a day in advance. These forecasts serve as crucial input for both the attacker and defender agents within the system.
- **Development of an RL-Based Attacker for FDIA Attack Generation: An** RL-based attacker was developed to generate False Data Injection Attacks (FDIA). This attacker employs reinforcement learning that allows it to grow smarter and improve its attack strategies over time.
- **Development of a Defender Agent:** An RL-based defender agent was designed to detect manipulated false data while continuously evolving to the attacker's strategies, allowing it to improve its detection capabilities.

1.3. ORGANISATION OF THESIS

The thesis is organised as follows:

- Chapter 1 presents Introduction, Objectives, Contributions and the organisation of the thesis.
- Chapter 2 presents the Literature Review carried out and utilised to help formulate the problem statement and complete the project.
- Chapter 3 presents the theoretical background for the given project. It includes the fundamentals of smart grids, cybersecurity and machine learning.
- Chapter 4 describes the actual work completed during the project. This includes the system design, development and testing of the RL agent, and system integration.
- Chapter 5 discusses the various algorithms developed and the results obtained from their implementation.
- Chapter 6 concludes the thesis by summarising key outcomes and proposing future work for the work carried out.

After establishing the project's goals and scope, the next chapter explores the foundational principles on which the entire project was built.

CHAPTER 2
LITERATURE REVIEW

2.1. Reinforcement Learning for FDIA Modelling

Recent developments indicate **Reinforcement Learning (RL)** as a powerful tool for both attack synthesis and defence. RL agents learn optimal strategies via direct interaction with the environment, enabling real-time adaptability in cyber-physical systems **[1].** However, most existing works implement a **single-agent RL model**, which fails to capture the strategic dynamics between an attacker and a defender **[6].**

2.2. Role of Load Forecasting in FDIA Defence

Accurate load forecasting is essential for both attack planning and anomaly detection in smart grids. Advanced **deep learning models**, particularly **Long Short-Term Memory (LSTM)** networks with attention mechanisms, enable precise day-ahead load prediction based on historical and environmental data. These predictions are critical for attackers to blend malicious signals with expected load trends and for defenders to identify discrepancies **[2], [4].** The inclusion of such forecasts into FDIA simulations enhances realism and improves model performance.

2.3 Adversarial Multi-Agent Frameworks

To overcome these limitations, **adversarial RL frameworks** have emerged, where attacker and defender agents are co-trained in a shared environment. The attacker aims to maximise disruption while avoiding detection, and the defender continuously adapts to emerging strategies. Such multi-agent approaches improve defence robustness by modelling realistic adversarial behaviours and have shown promise in smart grid intrusion detection scenarios **[3], [6].**

2.4. Cyber-Physical Vulnerabilities in Smart Grids

As modern power systems evolve into smart grids through digitisation and integration of IoT devices, communication networks, and automation, they face increased exposure to cyber-physical threats. Among these, **False Data Injection Attacks (FDIAs)** pose a significant risk due to their potential to manipulate sensor readings or control signals, leading to grid instability, blackouts, and economic loss. Key entry points like Advanced Metering Infrastructure (AMI) and Supervisory Control and Data Acquisition (SCADA) systems expand the attack surface, demanding robust and adaptive defence mechanisms **[5].**

2.5. Traditional FDIA Detection Methods and Limitations

Historically, FDIA detection relied on residual-based techniques and **supervised learning models**, which compare predicted and measured system states. While effective against known intrusion patterns, these models cannot generalise to stealthy or evolving attacks. Some studies explored **constrained optimisation** and **adversarial training**, but these approaches typically require prior system knowledge and are not suited for real-time deployment **[5], [6].**

2.6 Research Gaps and Motivation

Despite significant advancements, key challenges remain:

- Traditional models are inadequate for detecting **stealthy FDIA patterns**.
- **Single-agent RL frameworks** lack the realism of adversarial dynamics.
- **Load forecasting techniques** are often developed independently and not integrated into FDIA detection systems.

This thesis addresses these gaps by proposing a **unified framework** that integrates **multi-agent adversarial RL** and **deep learning-based load forecasting** to enhance the resilience of smart distribution systems under cyber-physical threats.

CHAPTER 3

THEORETICAL BACKGROUND

3.1 Smart Grid

Smart grids are the modern form of traditional power systems. Unlike conventional grids that rely on one-way power flow from centralised generation units, smart grids provide two-way communication between utilities and consumers by using technologies such as smart meters, automated substations, and responsive distribution management systems. Smart grids improve operational efficiency, grid reliability, and environmental sustainability as they can adapt to changing conditions in both supply and demand.

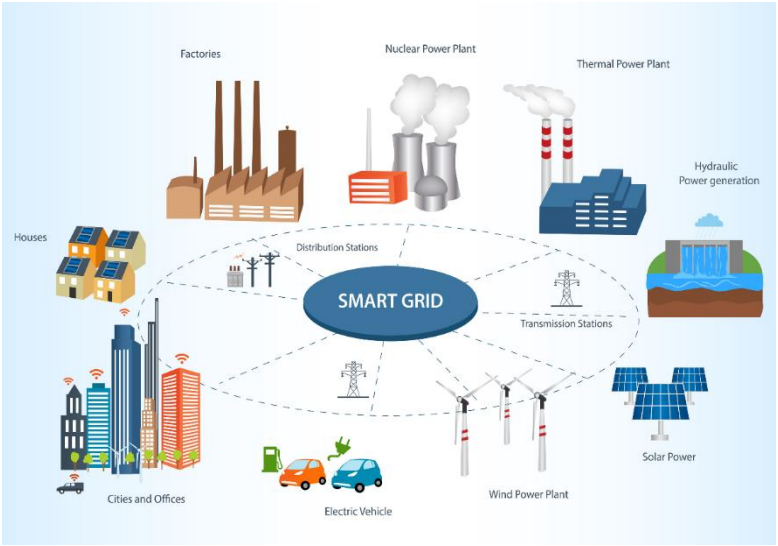


Fig 3.1. Smart Grid

3.1.1. Communication Infrastructure in Smart Grids

The communication infrastructure forms the most important layer of the smart grid as it enables real-time data exchange among distributed components such as smart meters, sensors, substations, controllers, and centralised control centres. This infrastructure plays an important role in enabling automated metering, demand response, distributed energy resource (DER) coordination, outage detection, and

remote asset management. Smart grid communication relies on a variety of standardised protocols to ensure reliable and secure data transfer.

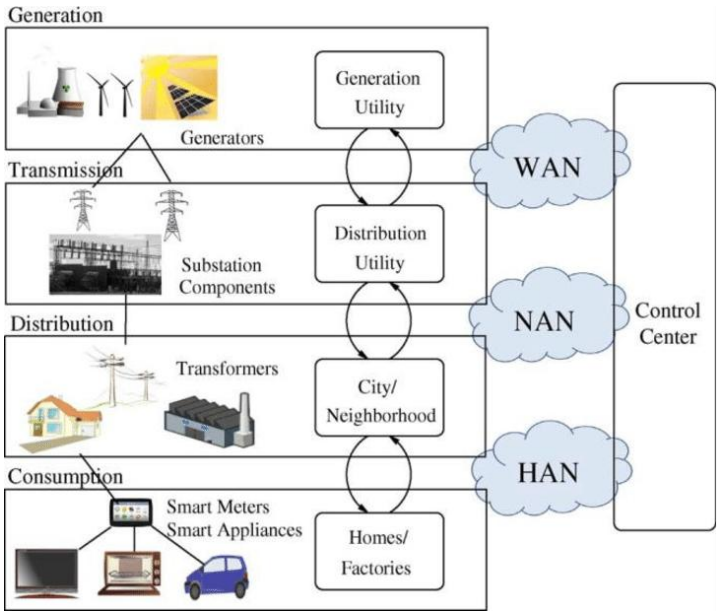


Fig.3.2. Communication Infrastructure in Smart Grids

Some of the commonly used protocols include:

- **IEC 61850:** Used in substation automation for communication between protection devices.
- **DLMS/COSEM:** Used in smart metering for data exchange between meters and data concentrators.
- **Modbus and DNP3:** Used in SCADA systems and for communication between control centres and field devices.
- **MQTT and CoAP:** Lightweight protocols used in IoT-enabled smart grid components for efficient messaging.

Smart grid communication is made up of several network layers. These include:

i. Home Area Network (HAN)
ii. Neighbourhood/Field Area Network (NAN/FAN)
iii. Local Area Network (LAN)
iv. Wide Area Network (WAN).

3.1.1.1 Home Area Network (HAN)

The HAN is deployed within individual residences or commercial buildings. It interconnects consumer-side devices such as smart appliances, energy monitors, electric vehicle (EV) chargers, and smart meters. The HAN enables end-users to actively monitor and manage their energy consumption patterns. Short-range wireless communication protocols like ZigBee, Z-Wave, Bluetooth Low Energy (BLE), and Wi-Fi are commonly used in HANs due to their low power consumption and ease of integration.

3.1.1.2. Neighbourhood/Field Area Network (NAN/FAN)

The Neighbourhood/Field Area Network functions as a bridging layer that collects information from several Home Area Networks or intelligent metering systems. This middle tier enables communication links between residential or commercial properties and utility company equipment, including data gathering units, operational field hardware, and nearby power distribution centres. Technologies commonly deployed in this network segment include radio frequency mesh systems, mobile telecommunications.

3.1.1.3. Local Area Network (LAN)

LAN is used within substations and utility control facilities to interconnect operational devices like Intelligent Electronic Devices (IEDs), Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), and Supervisory Control and Data Acquisition (SCADA) systems. These networks are typically implemented using Ethernet and fibre-optic technologies and rely on standardised communication protocols such as IEC 61850, DNP3, and Modbus, providing low-latency, high-speed communication to enable real-time monitoring and control.

3.1.1.4 Wide Area Network (WAN)

Wide-area frameworks connect utility command centres with power stations, distributed energy producers that are in different distant geographical areas. These networks handle important operations that need high data capacity and strict timing

requirements. These are implemented using optical fibre, microwave transmission, space-based communication systems, and specialised utility communication infrastructure.

3.1.1.5 Power Line Communication (PLC)

Power Line Communication utilises existing electrical distribution networks to transmit data signals without requiring additional communication cables. This is economical and used in remote areas where installing fibres or wireless systems is not economical.

There are two main variations:

i) Narrowband systems handle slower data applications like meter reading and load adjustment.

ii) Broadband implementations supporting faster tasks, including remote updates and visual monitoring systems.

3.2. NIST Smart Grid Architecture

To ensure standardised and interoperable communication across the smart grid, the National Institute of Standards and Technology (NIST) proposed a Smart Grid Conceptual Model, which organises the grid into seven functional domains:

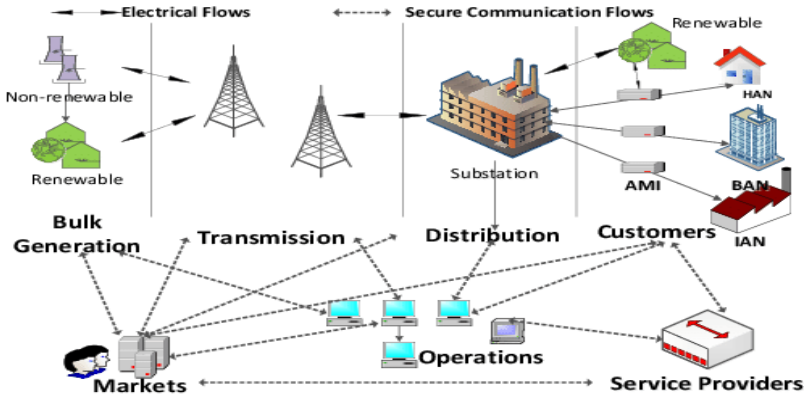


Fig. 3.3 NIST Architecture of Smart Grid

- Bulk Generation
- Transmission
- Distribution
- Customer
- Markets
- Service Providers
- Operations

These domains interact through secure communication pathways, facilitating seamless data exchange and coordinated decision-making across the grid ecosystem. For instance, the Operations domain leverages WAN-based telemetry from the Transmission and Distribution domains, real-time status updates from substations via LAN, and consumer-level behaviour data from HAN/NAN layers to maintain grid stability and reliability.

3.3. Cybersecurity Considerations

While the smart grid is better and more efficient, it also introduces new cybersecurity challenges. Inadequately secured communication links can be exploited by eavesdropping, spoofing, or injecting false data. To mitigate these risks, cybersecurity mechanisms must be equipped across all network layers.

3.3.1 Cybersecurity

Cybersecurity protects systems, networks, and data from unauthorised access, misuse or disruption. Modern cybersecurity usually has multiple layers of defence, including encryption, firewalls, access control and prevents various threats like malware, phishing, ransomware, and denial-of-service (DoS) attacks.

3.3.2 Cybersecurity in Power Systems

As power systems become increasingly digitised and interconnected, cybersecurity has emerged as a need of the hour to protect power systems against various threats, as mentioned below.

Types of Cybersecurity Threats:

1. **Data Breaches**

   A data breach occurs when unauthorised individuals gain access to sensitive system information.

2. **Denial-of-Service (DoS) Attack**

   A denial-of-service (DoS) attack floods a system with excessive requests or traffic, overwhelming its resources and rendering critical services unavailable, blocking communication between control centres and field devices, delaying responses to faults or emergency events.

3. **False Data Injection Attacks (FDIAs)**

   False Data Injection Attacks involve manipulating sensor or meter data to mislead power systems without triggering alarms. These attacks are particularly dangerous because they can be carefully crafted to appear legitimate, potentially causing incorrect load balancing, unstable operations, or unnoticed grid disruptions and can even cause blackouts.

3.4. Overview of Machine Learning and Deep Learning

There has been a growing advancement in the fields of Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL). These technologies make intelligent systems learn from data and make autonomous decisions.

3.4.1 Artificial Intelligence (AI):

This refers to creating systems that mimic human functions such as reasoning, learning, and problem-solving.

3.4.2. Machine Learning:

ML is a subfield of AI that focuses on developing algorithms that allow systems to learn from data without being explicitly programmed and is commonly used in tasks such as classification, prediction, and clustering. Machine learning algorithms can be broadly categorised into three types:

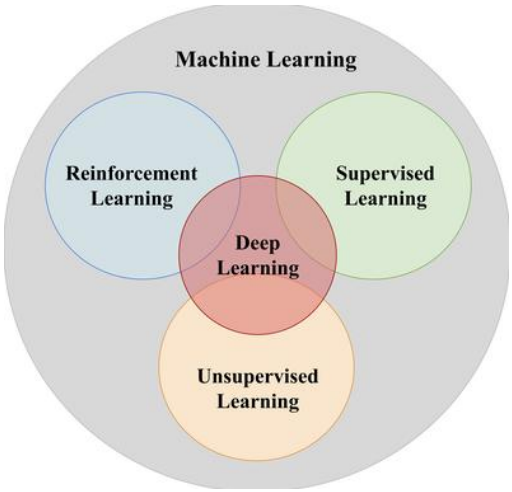


Fig 3.4 Classification of Machine learning

- **Supervised Learning:**

In supervised learning, the model is trained on a labelled dataset, where each input is mapped to the correct output, enabling it to predict the output for unseen data. Supervised learning is commonly used in tasks like classification and regression.

- **Unsupervised Learning:**

Unsupervised learning involves training the model on unlabelled data. The goal is to identify underlying patterns, structures, or relationships within the data. This type of learning is often used for clustering or dimensionality reduction.

- **Reinforcement Learning (RL):**

  In Reinforcement learning, an agent learns to make decisions by interacting with its environment. The agent receives feedback in the form of rewards or penalties based on the actions it takes, and the objective is to maximise cumulative reward over time. RL is commonly applied in areas like robotics, autonomous driving.

3.4.3 Deep Learning

Deep learning is a part of machine learning that utilises artificial neural networks with multiple layers to extract high-level features from raw data. These are capable of learning intricate patterns and representations directly from the data, without the need for explicit feature engineering. Deep learning has gained success in various research domains due to its ability to handle large-scale datasets.

Use in Research:

- Image Recognition and Computer Vision: Deep learning has improved image recognition tasks, achieving performance in tasks such as object detection, image classification.
- Natural Language Processing (NLP): Deep learning models, particularly recurrent neural networks (RNNs) and transformers, have significantly advanced in natural language processing tasks such as language translation, sentiment analysis, text generation, and language understanding, powering virtual assistants, chatbots.
- Speech Recognition and Audio Processing: Deep learning techniques, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been highly successful in speech recognition and audio processing tasks.
- Deep learning is also used in climate science to analyse satellite and sensor data for weather forecasting and climate modelling.

3.4.4 Long Short-Term Memory (LSTM)

- LSTM is a special kind of neural network that works well with sequences of data, especially when the important information is spread out over time. It's an improved version of the older RNN model, which often had trouble remembering things from earlier in a long sequence, mainly because of problems like gradients becoming too small or too large during training.

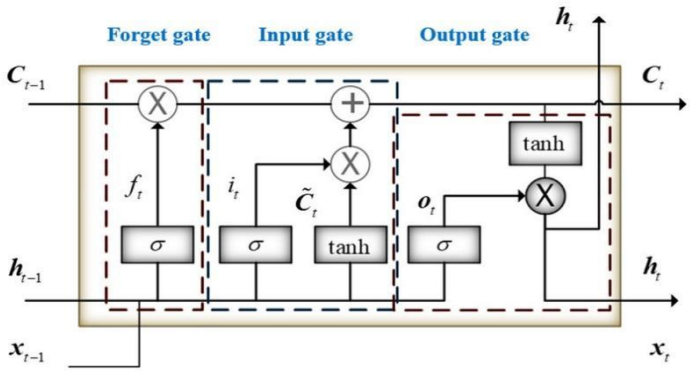


Fig.3.5 LSTM Cell Architecture

- An LSTM network is made up of a series of recurrent units, each known as an LSTM cell. Each cell stores two types of information: the cell state (Ct) as long-term memory and the hidden state (ht) as short-term output for the next time step.
- The LSTM cell architecture contains three basic gating mechanisms: the forget gate, input gate, and output gate, which control the flow of information into and out of the cell.

Forget Gate (f_t): Determines which parts of the previous cell state C_{t-1} should be discarded. It is computed as:

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \qquad (3.1)$$

Where σ denotes the sigmoid function, x_t is the input at time t, h_{t-1} is the previous hidden state, and W_f, b_f are learnable parameters.

Input Gate, i_t and Candidate State $\tilde{C}_t$: These work together to update the cell state with new data. The input gate determines how much of the candidate state ($\tilde{C}_t$) is contributed to the cell state:

$$i_t = \sigma(W_i \cdot [h_t - 1, x_t] + b_i) \quad (3.2)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_t - 1, x_t] + b_c) \quad (3.3)$$

Cell State Update: The new cell state is computed by combining the retained old state and the new candidate information:

$$C = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (3.4)$$

Output Gate (o_t): This gate decides which parts of the updated cell state should be output. It is given by:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)] \quad (3.5)$$

The hidden state is then updated as:

$$h_t = o_t * \tanh(C_t) \quad (3.6)$$

This architecture enables the LSTM to selectively retain or forget information over long sequences, allowing it to effectively model temporal dependencies. The use of gating mechanisms and the separation of the memory cell from the output pathway are what make LSTM networks particularly suited for complex time-dependent tasks such as natural language processing, time-series forecasting, and anomaly detection in sequential data.

CHAPTER 4

PROPOSED METHODOLOGY

This chapter models the interaction between attackers and defenders as an adversarial game, where both parties continuously adapt to each other's strategies. This section details the components of our framework and their interactions.

4.1 Overall Workflow and System Description

The proposed framework is built around a smart distribution system that leverages a reinforcement learning based approach to simulate FDIA scenarios and develop real-time defence mechanisms. The setup is based on a 5-bus radial distribution network, which includes one Distribution System Operator (DSO) bus and four load buses, all operating at a nominal voltage of 230 kV.

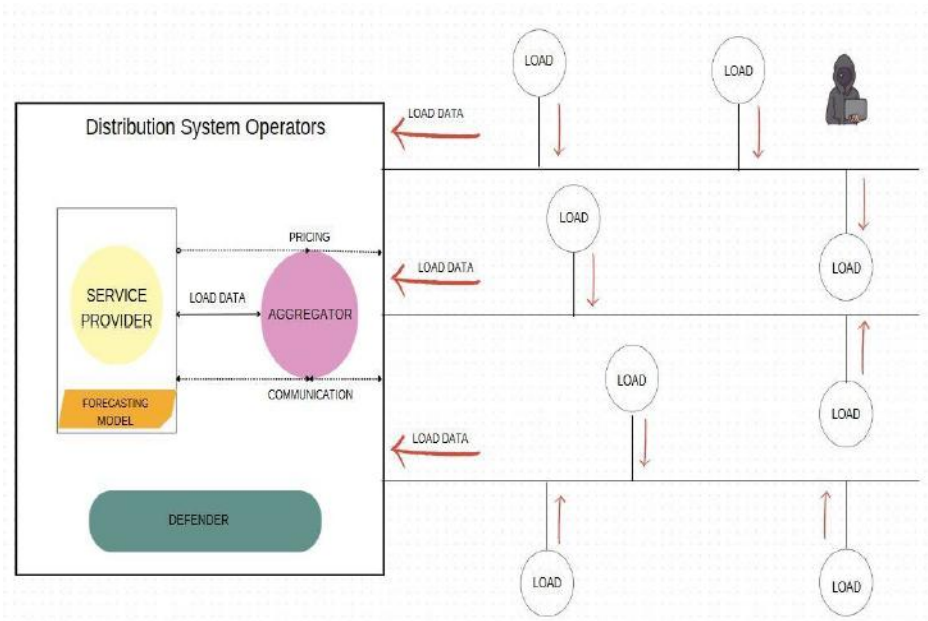


Fig. 4.1 Proposed Smart Distribution Network.

The proposed system is a **5-bus radial distribution network** modelled and simulated using the **Panda Power** library in Python, which is widely used for power system analysis and automation. The system includes one **Distribution System Operator (DSO) bus**, acting as the slack or reference bus, and four additional **load buses** (Buses 1 to 5), all operating at a nominal voltage of **230 kV**.

Buses:

- DSO Bus: Slack bus (reference bus) with a voltage magnitude of 1.00 per unit (p.u.).
- Buses 1, 2, 3, 4, and 5: each with a nominal voltage of 230 kV.

Parameters:

1. Nominal Voltage: 230 kV for both slack and load buses.
2. Resistance: 0.02 ohms/km
3. Reactance: 0.06 ohms/km
4. Capacitance: 0 nano-farads/km
5. Length: 1 km
6. Max Current Capacity: 0.5 kA
7. Power Factor: 0.5 (lagging), used to calculate reactive power components of the loads.

The system integrates three primary components:

1. An **LSTM-based forecasting model** that predicts day-ahead load demand values.
2. An **attacker agent** trained using **Proximal Policy Optimisation (PPO)** that injects false data into the Reactive Power of buses in the distribution system.
3. A **defender agent** that attempts to detect such false data injections.

The attacker and defender agents continuously interact with the simulation environment, learning and updating their policy over time based on rewards and penalties. This creates a realistic threat environment that reflects how smart grids could respond to real-world cyber-physical attacks.

4.2 Forecasting Studies

The forecasting model is a critical component in this framework, as it provides the baseline load values that guide both attacker and defender agents. An LSTM-based deep learning model was developed to compute the day-ahead load.

4.2.1 Dataset and Features

To train the LSTM model with the attention mechanism, a complete dataset was prepared that included past load values and other related features that affect electricity demand. The aim was to give the model all the important information it needs to learn and make accurate predictions.

i. Data Description

The main input to the model is the active power load (Load_w), which is the real power used at each bus in the system. The dataset includes hourly readings, and each prediction uses the past 168 hours (one full week) of data. This helps the model learn daily and weekly patterns in electricity usage. The data comes from the **PJM Dataset** from **Kaggle**.

ii. Features Used:

Three types of features were used in training:

1. **Historical Load Features:**
   - **i.** Load values from the past 168 hours.
2. **Time-Based Features: Hour of the day** (0 to 23) – to learn daily usage patterns.
   - **i.** **Day of the week** (0 to 6) – to understand weekday and weekend differences.
   - **ii.** **Week number** – to help the model learn seasonal changes.
   - **iii.** **Weekend indicator** – a value showing if the day is a weekend or not.
3. **Environmental Features:** Temperature (°C)Wind speed (m/s)Humidity (%)
   - **i.** Solar insolation (W/m²)

Weather-related features were matched with the load data based on the same time. These features are important because the weather can strongly affect how much electricity people use.

iii. Preprocessing Steps:

Before training, the data went through several steps:

i. All features were **scaled** using Min-Max normalisation, so they have similar value ranges.

ii. **Missing values** and **outliers** were removed to clean the data.

iii. The data was then divided into input and target sequences, where each input includes 168 hours of data, and the target is the next hour's load value.

4.2.2 LSTM Architecture with Attention Mechanism

The model processes a sequence of the past 168-time steps (seven days of hourly data) to forecast the upcoming load. The architecture includes layers to capture temporal dependencies, followed by an attention layer that weighs each time step's relevance. This helps the model prioritise more influential periods (e.g., peak hours or sudden demand changes) when generating predictions.

4.2.3 Model Performance

The forecasting model was evaluated using standard metrics to evaluate their performance and error margin, as given below:

i. **Mean Absolute Error (MAE):** Average of the absolute differences between predicted and actual values.
ii. **Root Mean Squared Error (RMSE):** The Square root of the average of squared differences between predicted and actual values.
iii. **Mean Bias Error (MBE):** Average of the signed differences between predicted and actual values, indicating bias.
iv. **R^2 Score:** Measures how well predictions approximate actual values, with 1 indicating perfect prediction.

4.3 Attack Vector

To simulate FDIAs, various measurable parameters in the distribution system were considered as potential attack vectors, including:

1. **P**: Active Power
2. **Q**: Reactive Power
3. **V**: Voltage Magnitude
4. **δ (delta)**: Voltage Angle

Among these, **active power (P)** was selected as the primary variable to manipulate. In a distribution system, changing active power values can affect other dependent parameters such as voltage magnitude, reactive power, and phase angles. This makes false data injection in P not only effective in destabilising the system but also stealthier, as it introduces indirect manipulations in multiple measurements without directly attacking each, making it a good choice for an attack.

4.4 Attacker and Defender Training Using PPO

The attacker and defender are both RL-based agents and were both trained using the Proximal Policy Optimisation (PPO) algorithm, a widely used reinforcement learning method known for its stability and efficiency in continuous action spaces.

PPO operates on the principle of policy gradient methods but improves them by constraining the updates to prevent drastic policy shifts. This is done using a clipped surrogate objective function, which limits how far the new policy can deviate from the old one in each update step. This ensures stable and reliable learning, especially in dynamic multi-agent environments.

In this framework, the **attacker agent** receives a reward for injecting data that remains undetected while significantly altering the system state, encouraging stealthy and impactful attacks. The **defender agent** receives a reward for successfully detecting anomalies without raising false alarms, promoting both sensitivity and precision. The PPO algorithm allows both agents to continuously adapt to each other's evolving strategies, resulting in increasingly refined and intelligent false data injection and detection techniques.

4.5 Formulation of Attack Scenarios: Attacker Design and Strategy

The attacker is a reinforcement learning-based agent trained using PPO to manipulate the predicted active power (P) values. Its actions are guided by two primary constraints: remaining below the defender's detection threshold and preserving power balance before and after the attack.

The attacker agent is designed to operate under **limited system knowledge**, with no knowledge of the grid topology. The agent aims to inject false data into the predicted active power values, while remaining undetected. The attacker is trained using the **Proximal Policy Optimisation (PPO)** algorithm, which enables it to learn the best policy through trial-and-error interaction within a simulated environment.

4.5.1 State Space

The state observed by the attacker at each decision step consists of the predicted active power values from the forecasting module, represented as a vector:

$$P_{pred} = [P_1, P_2, ..., P_n] \tag{4.1}$$

Where n denotes the number of load buses in the system.

4.5.2 Action Space

The agent's actions are defined as continuous percentage adjustments to each predicted P value:

$$\Delta P = [\delta_1, \delta_2, ..., \delta_n], \quad \delta_i \in [-0.01, 0.01] \tag{4.2}$$

The final manipulated values become:

$$Ppred = Ppred + Ppred \cdot \Delta P \tag{4.3}$$

This bounded action space (±1%) enforces subtle manipulation, allowing the attacker to remain stealthy while influencing system behaviour.

4.5.3 Power Conservation Constraint

To avoid detection through energy balance checks, the total system active power of all 5 buses must remain the same before and after manipulation:

To fulfil this constraint, we distribute the total change among all 5 buses such that any increase in one value is offset by a corresponding decrease elsewhere to avoid a total change in the aggregator.

$$\sum_{i=1}^{n} P_i^{attack} = \sum_{i=1}^{n} P_i^{pred} \quad (4.4)$$

4.5.4 Detection Threshold

To model the defender's detection capability, any change that exceeds a deviation threshold is flagged as a potential attack:

$$\left| (P_i^{attack} - P_i^{pred}) / P_i^{pred} \right| > 0.15 \Rightarrow \text{FDIA flagged} \quad (4.5)$$

The 15% threshold was chosen to account for disturbances that may occur in real-world systems, such as hardware malfunctions or minor fluctuations in data. It represents a tolerance, beyond which operators or automated systems would likely detect significant irregularities.

4.6 FORMULATION OF ATTACK SCENARIO

Flow of PPO-based Attacker Agent

The attack scenario is framed around the reinforcement learning-based attacker agent, which operates using the **Proximal Policy Optimisation (PPO)** algorithm to subtly manipulate the predicted active power values in the smart distribution system.

The flow of the PPO-based attacker agent is outlined as follows:

1. **Initialise PPO Agent**: The attacker agent is initialised with the predicted P values obtained from the load forecasting model, serving as the starting point for its actions.

2. **Take Observations**: At each time step, the agent observes the current state of the system, which consists of the active power values for each load bus.

3. **Choose Actions**: Based on the observed state, the agent selects continuous actions that represent small, subtle changes in the P values. These changes are typically within a ±1% range to ensure minimal disturbance.

4. **Apply Changes**: The selected changes to the P values are applied to the system, resulting in manipulated active power values P attack.

5. **Detection Check**: After applying the changes, the system checks for any anomalies, and if the change in any P value exceeds the predefined detection threshold (e.g., 15%), the attack is flagged.

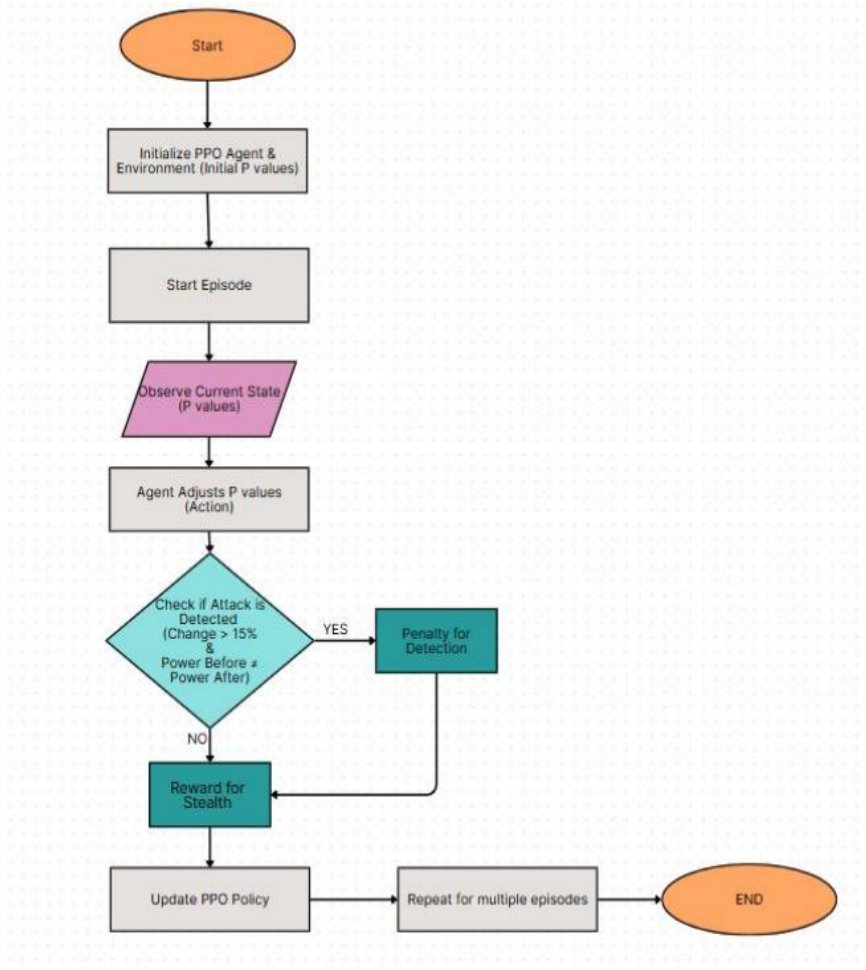


Fig.4.2 Flow of Attacker Agent

6. **Assign Rewards/Penalties**: The agent receives feedback in the form of rewards or penalties based on whether its actions remain undetected or are successfully flagged by the defender. If the attack is undetected, the agent receives a positive reward, else gets a penalty.

7. **Update PPO Policy**: The PPO algorithm updates the agent's policy based on the feedback, adjusting its strategy to improve its future performance.

8. **Repeat for Multiple Episodes**: The process is repeated over multiple episodes, with the agent continuously refining its strategy to make more effective and stealthy manipulations in the system**.**

4.7 Reward Function Formulation

The reward function for the PPO-based attacker agent is designed to manipulate the system's predicted active power values while ensuring power conservation and stealth. The total reward is defined as the sum of the following components:

$$Total\ reward = Manipulation\ Reward + Stealth\ Reward + Detection\ penalty + Power\ conservation\ penalty$$

4.7.1 Manipulation Reward

This component rewards the attacker agent for successfully altering the predicted P values in a manner that deviates from the baseline without being detected. The manipulation reward encourages effective power manipulation, fostering the agent's ability to subtly influence the system.

4.7.2 Stealth Reward

The stealth reward incentivises the attacker to remain below the detection thresholds set by the defender. The agent is rewarded for minimising the magnitude of its changes to the P values, thereby avoiding triggering the detection system.

4.7.3 Detection Penalty

A negative reward is assigned if the attack is detected by the defender. This penalty discourages large, detectable alterations to the P values and forces the agent to adopt more cautious, subtle strategies to evade detection.

4.7.4 Power Conservation Penalty

This penalty ensures that the attacker maintains the constraint of power conservation. The agent incurs a penalty if the total system power balance is violated by its changes to the P values, thus encouraging the attacker to make adjustments that do not disrupt the overall system integrity.

4.8 Defender Agent

The defender agent is designed as an adaptive detection system that operates in parallel with the attacker. Its primary objective is to identify and localise manipulated data in real-time while minimising the occurrence of false positives. The key elements of the defender's training and learning process are outlined as follows:

4.8.1 Training Objective

The defender agent’s goal is to accurately detect the manipulated Power values by using only observable system state features. It must differentiate between normal system fluctuations and actual false data injections, thereby maintaining high detection accuracy without becoming overwhelmed by false alarms.

4.8.2 Learning Strategy

The agent receives positive rewards for correctly identifying manipulated Power values, while it incurs penalties for false positives (incorrectly flagging legitimate data as manipulated) or missed attacks (failing to identify actual manipulations). This feedback mechanism enables the agent to refine its detection capabilities over time. This enables the defender to evolve its strategy while continually adapting to increasingly stealthy attacks.

4.9 Simulation Framework for FDIA Attack and Defence

This simulation environment creates a system that replicates cyber-physical interactions in a smart distribution network, creating false data injection attack (FDIA) strategies and corresponding defence mechanisms.

4.9.1 Day-Ahead Load Forecasting

The process begins with an LSTM-based forecasting model that predicts the day-ahead active power demand at each bus. The model uses historical load data, time-based inputs, and environmental parameters to generate realistic baseline values.

4.9.2 Attacker Manipulation

These forecasted values are then manipulated by the attacker agent, which is trained using the Proximal Policy Optimisation (PPO) reinforcement learning algorithm. The attacker modifies the predicted active power (P) values remaining below detection thresholds and maintains total power constraints.

4.9.3 Data Transmission to Defender

The defender observes the altered active power values and uses its learned policy to determine whether an attack has taken place or not.

4.9.4 Defender Detection

Based on its training, the defender flags the data as manipulated and acts. The agent is rewarded for correctly identifying attacks and penalised for false or missed detections, helping in training itself.

4.9.5 Feedback and Policy Updates

After each episode, both attacker and defender agents receive feedback in the form of rewards or penalties. This feedback loop allows each agent to adjust its policy in response to the other's strategies, enabling a co-adaptive learning environment.

4.9.6 Monitoring and Analysis

Throughout the simulation, key performance indicators such as detection accuracy, power imbalance, and stealth efficiency are continuously recorded.

This data provides a detailed basis for evaluating the effectiveness of both the attacker and defender and supports further improvements to the system.

4.10 Tools and Technologies

Our FDIA simulation and detection framework relied on several open-source tools that provided the necessary capabilities for power system modelling and machine learning implementation.

4.10.1 Core Technologies

We used Python as our main programming language, leveraging its scientific ecosystem for both power system calculations and machine learning integration. For power system modelling, Panda Power enabled us to build and analyse a 5-bus radial distribution network, calculating essential parameters like reactive power, voltage magnitude, and angle measurements.

4.10.2 Machine Learning Framework

The defender agent was implemented using PyTorch due to its flexibility in designing custom neural architectures. This allowed us to create a classification model that identifies potential false data injection attacks by analysing measurement deviations. Our simulation environment was built on OpenAI Gym, providing a standardised interface for the attacker-defender interaction scenarios involving Reinforcement Learning.

4.10.3 Supporting Libraries

We utilised NumPy for efficient numerical operations and data manipulation, particularly when handling load value vectors and computing statistical features. Visualization of training progress and attack metrics was accomplished with Matplotlib. Python's built-in random and collections libraries supported our data sampling approach and experience replay buffer implementation, which proved essential for the defender's continuous learning process.

CHAPTER 5

ALGORITHMS AND RESULTS

5.1 Algorithm for Load Forecasting Model

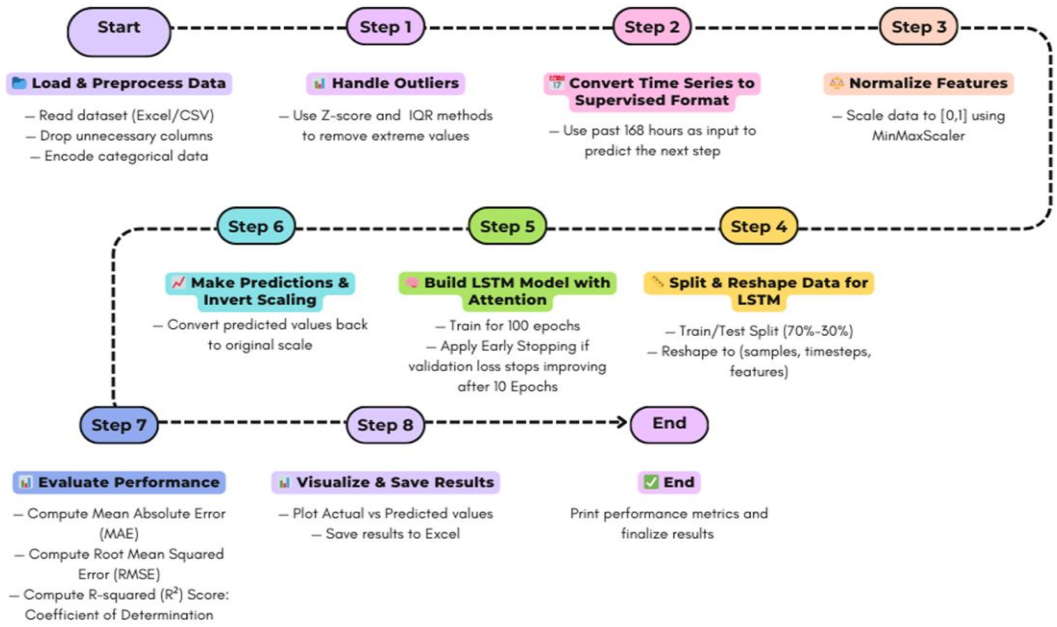


Fig. 5.1: Flowchart of LSTM-based Load Forecasting Model

Results: After 80 Epochs of Training

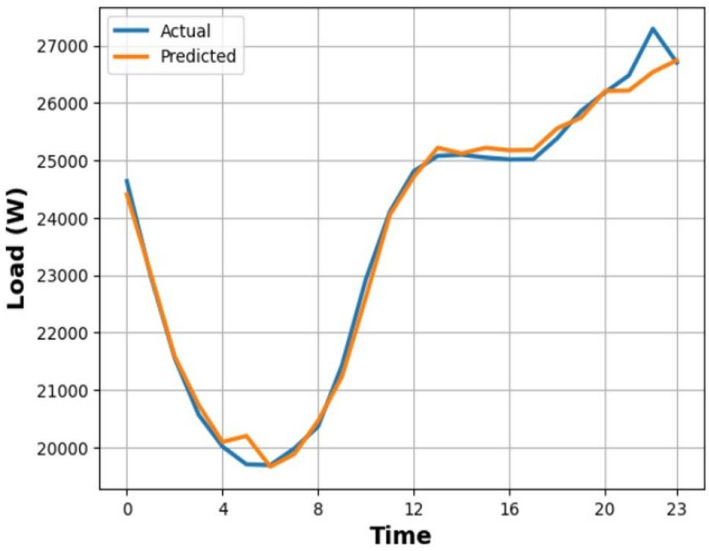
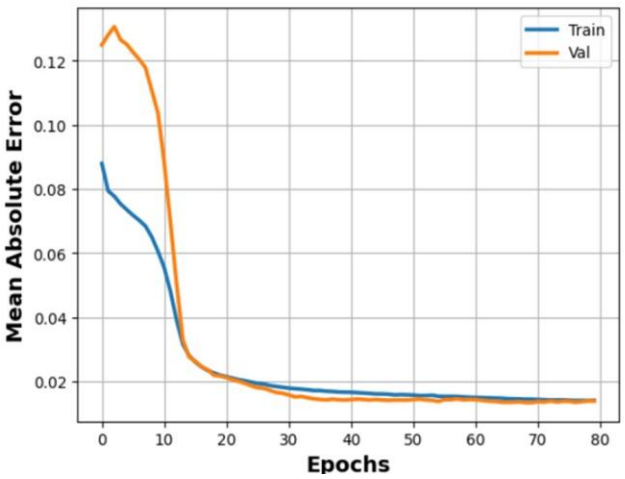


Fig. 5.2: Actual Load vs Predicted Load

The Graph shows the actual load and predicted load by our forecasting model for each time step based on the previous 168 hours of data.

Our Forecasting model predicts loads quite accurately and close to the actual load demand at every timestep.



The Graph shows the mean absolute error of our forecasting while training and while in evaluation across 80 epochs.

We can notice how the error significantly reduces steeply after 15 epochs of training and evaluation.

Fig. 5.3 Mean Absolute Error while Training vs Evaluation

Evaluation Scores of the Model:

- **MAE_lf:** 0.0138
- **RMSE_lf:** 0.0184
- **MBE_lf:** -0.0024
- **R2 Score_lf:** 0.9852

Features Used for Prediction:

1. **Load_W** (Load demand in watts)
2. **Time-based Features** (Hour, Day, Week)
3. **Weather Parameters** (Temperature, Wind Speed, Humidity, Solar Insolation)

5.2 Algorithm with RL Attacker and No Defender

The algorithm simulates a False Data Injection Attack (FDIA) in a 5-Bus Distribution System using a reinforcement learning (RL) agent trained with the Proximal Policy Optimisation (PPO) algorithm. The objective is to train an attacker to modify the predicted active power (P) values at selected buses in a way that the attack does not violate the power system constraints defined by us.

5.2.1 Step-by-Step Flow:

i. Environment Initialisation:

- The environment is initialised with predicted active power values (P) for a set of buses.
- Specific buses are chosen as targets for attack.

ii. Agent Observation and Action:

- The agent observes the current P values at the targeted buses.

- It outputs a vector of small percentage changes (within ±1%) to be applied to these values.

iii. Executing the Attack:

- The proposed percentage changes are applied to the targeted P values.
- A redistribution step ensures the total power remains conserved across the system.
- All values are clipped to remain physically valid (e.g., non-negative power).

iv. Reward Calculation: A custom reward function is evaluated based on stealthiness, manipulation success, and adherence to physical constraints.

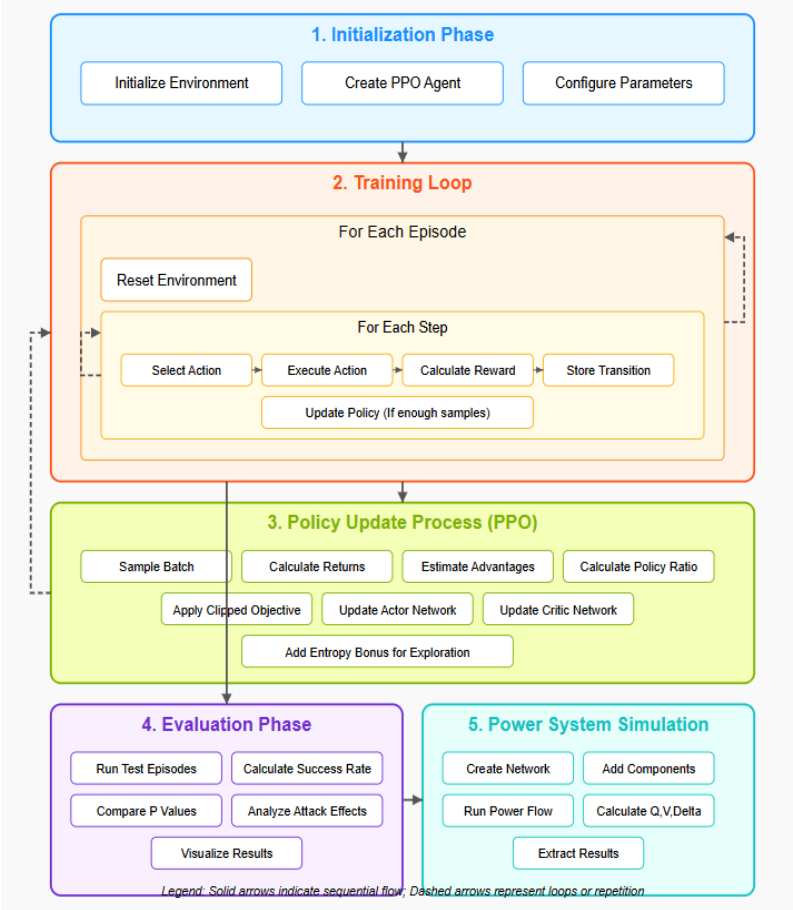


Fig. 5.4 Flowchart for False Data Injection Attack Algorithm

v. Policy Update (PPO):

- Transitions (state, action, reward, next state) are stored.
- The PPO algorithm uses these transitions to update the actor-critic policy.

vi. Training and Evaluation:

- The agent is trained over multiple episodes.
- Performance metrics such as reward and attack success rate are tracked and visualised.

5.2.2 Attacker Design and Behaviour

The attacker is modelled as a reinforcement learning agent:

- It targets specific P values (e.g., from a state estimator) to manipulate.
- It applies small, continuous perturbations that aim to go undetected.
- It respects the total power conservation by redistributing any changes proportionally.
- It learns to exploit weak spots in the prediction or control layers without triggering anomaly detectors.

The attacker is constrained to act within narrow action bounds (±1%), forcing it to discover subtle and efficient strategies.

5.2.3 Reward Function Design

The reward function is critical for guiding the agent toward effective yet stealthy attacks. It is composed of the following elements:

i. Stealth Reward: Provides a positive reward if the manipulated values are within 10% of the maximum allowed deviation (20%).
ii. Detection Penalty: Applies a penalty if any manipulation exceeds the 20% deviation threshold and scales with the episode step number to discourage late detection.
iii. Manipulation Reward: Encourages noticeable but bounded changes to the target values.
iv. Power Conservation Penalty: Penalises deviations from total system power, promoting physically realistic attacks.
v. Success Bonus: Awards a bonus if all stealth and conservation criteria are met.

5.2.4 Evaluation Phase

After training, the performance of the attacker agent is assessed:

i. Run Test Episodes: The trained agent is evaluated on unseen scenarios.
ii. Calculate Success Rate: Measures how often the agent bypasses detection.
iii. Compare P Values: Examines how much the injected values differ from the original.
iv. Analyse Attack Effects: Assesses how the manipulations affected the system.
v. Visualise Results: Graphs and plots are generated for interpretability.

5.2.5 Results:

Case – 1: Varying Reward Function to get the best-suited constants for the rewards for 120 Episodes

Reward Function 1:

stealth_reward = 5 * np.sum(detection_closeness <= 0.1)

detection_penalty = -25 * (self.current_step / self.episode_length) *np.sum(np.maximum(detection_closeness - 1.0, 0))

manipulation_reward = 20 * np.sum(np.abs(current_targets - original_targets)) / np.sum(original_targets)

power_conservation_penalty = -50 * abs(total_current_power - self.total_power) / self.total_power

attack_success = (np.all(detection_closeness <= 1.0) and abs(total_current_power - self.total_power) / self.total_power < 0.01)

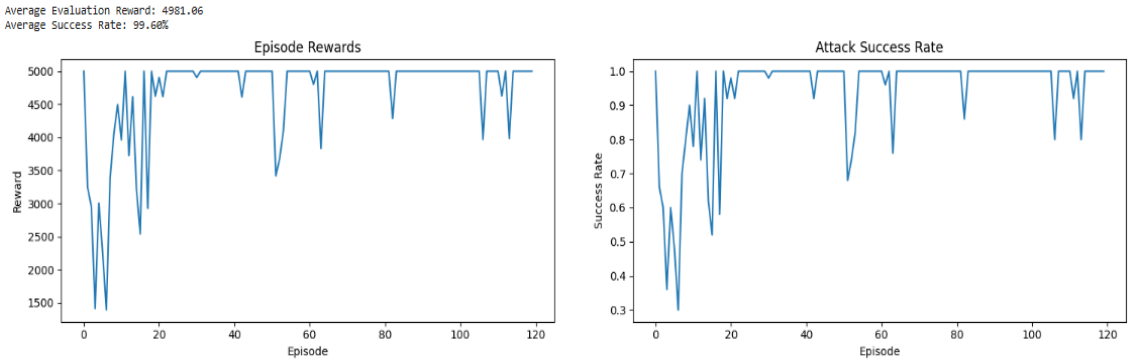


Fig. 5.5: Average Success Rate and Average Evaluation Reward for Reward Function 1

Reward Function 2:

stealth_reward = 10 * np.sum(detection_closeness <= 0.1)

detection_penalty = -20 * (self.current_step / self.episode_length) *np.sum(np.maximum(detection_closeness - 1.0, 0))

manipulation_reward = 30 * np.sum(np.abs(current_targets - original_targets)) / np.sum(original_targets)

power_conservation_penalty = -40 * abs(total_current_power - self.total_power) / self.total_power

attack_success = (np.all(detection_closeness <= 1.0) and abs(total_current_power - self.total_power) / self.total_power < 0.01)

reward = manipulation_reward + stealth_reward + detection_penalty + power_conservation_penalty

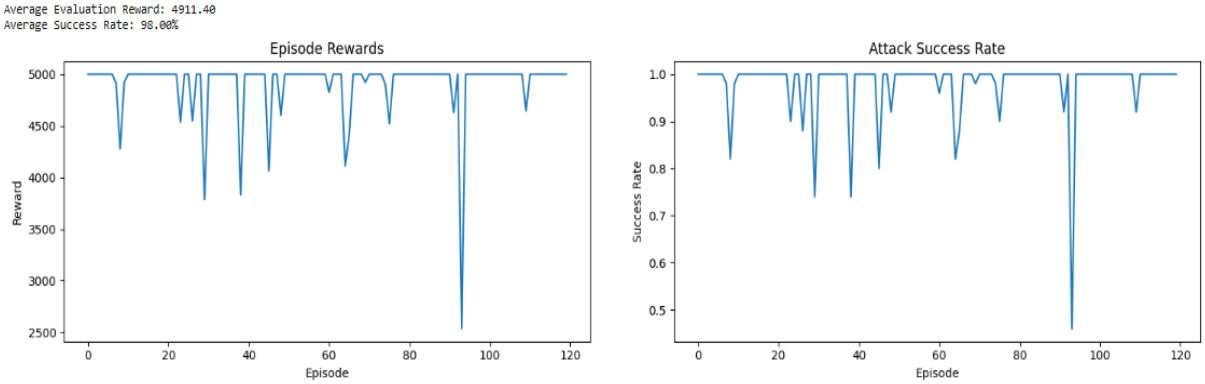


Fig. 5.6: Average Success Rate and Average Evaluation Reward for Reward Function 2

Reward Function 3:

stealth_reward = 10 * np.sum(detection_closeness <= 0.1)

detection_penalty = -20 * (self.current_step / self.episode_length) *np.sum(np.maximum(detection_closeness - 1.0, 0))

manipulation_reward = 30 * np.sum(np.abs(current_targets - original_targets)) / np.sum(original_targets)

power_conservation_penalty = -40 * abs(total_current_power - self.total_power) / self.total_power

attack_success = (np.all(detection_closeness <= 1.0) and abs(total_current_power - self.total_power) / self.total_power < 0.01)

reward = manipulation_reward + stealth_reward + detection_penalty + power_conservation_penalty

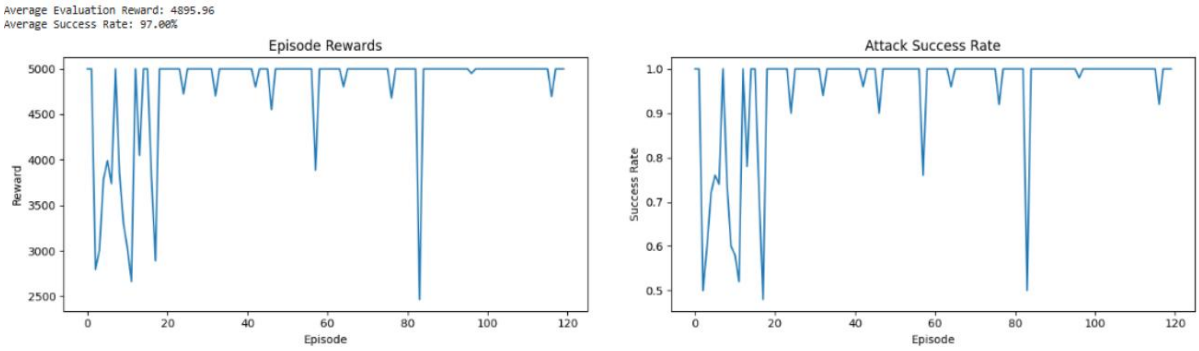


Fig 5.7: Average Success Rate and Average Evaluation Reward for Reward Function 3

Hence, we choose **Reward Function 1** as the base Reward Function for all future algorithms using RL Attacker, as it gives us the best average attack success rate of 99.6% compared to 98% of Reward Function 2 and 97% of Reward Function 3.

5.3 Algorithm with RL Attacker and DL Defender

5.3.1 System Components:

The developed system consists of three primary components:

i. A simulated power system environment
ii. A reinforcement learning-based attacker
iii. A neural network-based defender

5.3.2 Algorithm Flow:

i. Initialisation:

- A power system environment is initialised with nominal active power (P) values.
- The attacker is set up using PPO with actor-critic architecture.
- The defender is instantiated as a multi-layer neural network trained for binary classification.

ii. Training Loop:

For each episode:

1) The environment resets to initial conditions.
2) At each timestep:
    - The attacker selects an action to modify P values.
    - The environment simulates the resulting system state (Q, V, δ).
    - The defender analyses system measurements to detect anomalies.
    - A reward is calculated based on the attack's impact and stealth.
    - The attacker's policy is updated using PPO.
    - The defender is trained on new data samples.
    - The training continues until convergence or performance criteria are met.

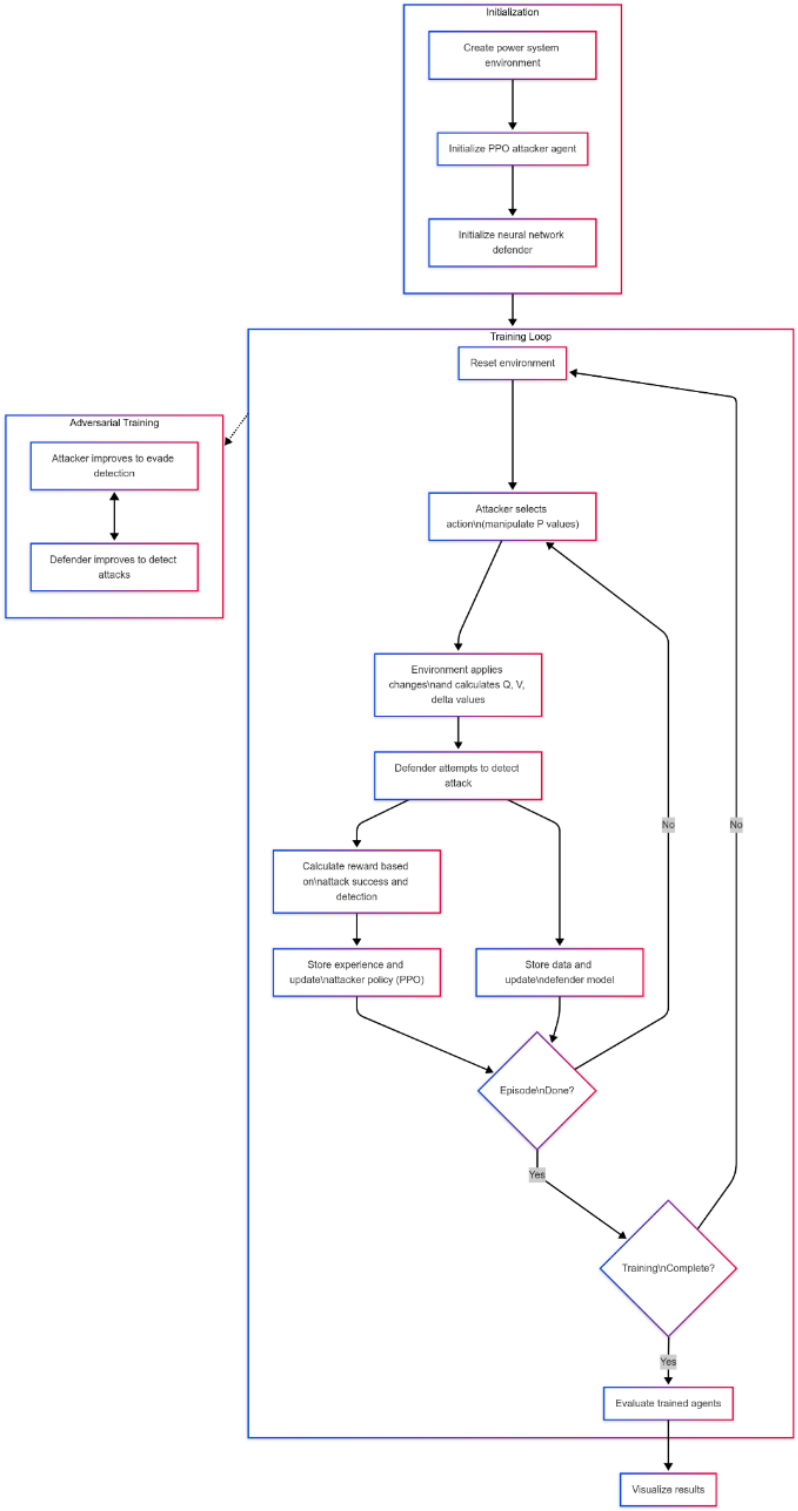


Fig. 5.8 Flowchart for RL Attacker and DL Defender Algorithm

iii. Evaluation Phase

- The trained attacker-defender pair is evaluated under various conditions.

- Key metrics like detection accuracy, attack success rate, and system impact are analysed.
- Visualisations illustrate how system parameters are affected and how learning progresses.

5.3.3 Attacker Design

i. Objectives

The attacker uses a Proximal Policy Optimisation (PPO) algorithm to learn how to manipulate power measurements in the system while avoiding detection.

The attacker's goal is to:

- Significantly alter the active power values (P),
- Preserve the overall power balance,
- Avoid triggering the defender’s detection system.

ii. Components

- Actor Network: Outputs actions to modify P values.
- Critic Network: Estimates the value of the current state.
- Action Space: Small percentage changes to P values (−1% to 1%).
- Observation Space: Current P values at target buses.

iii. Attacker Reward Function

The attacker’s behaviour is guided by a custom-designed reward function that balances stealth, impact, and realism. Key components include:

- Stealth Reward: Rewards subtle attacks that stay within 10% of the allowed deviation from original values.
- Detection Penalty: Penalises manipulations exceeding a 15% threshold, especially if detected later in the episode.
- Manipulation Reward: Encourages larger deviations in P values to maximise system disruption.
- Power Conservation Penalty: Enforces the constraint of total power conservation, preventing obvious detection.

- Defender Penalty: Imposes heavy penalties if the attack is detected by the defender.
- Success Bonus: Grants a high reward if the attack goes undetected while satisfying power and stealth constraints.

5.3.4 Defender Design

i. Objectives

- The defender uses a neural network to identify potential attacks by analysing patterns in power system measurements.
- Defender's goal is to accurately detect when power measurements have been manipulated.

ii. Components

- Neural Network: A multi-layer perceptron that processes power system measurements
- Features: Uses both raw measurements and differences from expected values
- Detection Threshold: Probability threshold for flagging an attack (0.5 here)
- Training Buffer: Stores samples for ongoing learning

iii. The defender analyses:

- Active power (P) values
- Reactive power (Q) values
- Voltage magnitude (V) values
- Voltage angle (delta) values
- Differences between the attacked and expected values
- System-wide metrics (sums, means, standard deviations)

5.3.5 Adversarial Training Process

The attacker and defender train concurrently, constantly adapting to each other's evolving strategies:

i. The attacker modifies P values to evade detection while maximising system disruption.
ii. The defender observes all measurements to flag anomalies and updates its model via supervised learning.
iii. The environment provides rewards to the attacker based on stealth and impact.

5.3.6 System Components Summary

i. Environment: Simulates power flow responses using Panda Power; integrates attacker and defender.
ii. PPO Attacker: Uses actor-critic models, experience replay, and entropy regularisation.
iii. Neural Network Defender: Learns from labelled data, applies threshold-based detection, and uses binary cross-entropy loss.
iv. Visualisation Tools: Track attacker rewards, detection rates, and system parameter changes (P, Q, V, δ) across training.

5.3.7 Results:

Training 100 Episodes of 50 Steps each and Evaluating for 10 Episodes

```
Episode 20, Avg Reward: -9947.02, Success Rate: 11.10%
Episode 40, Avg Reward: -9312.12, Success Rate: 11.00%
Episode 60, Avg Reward: -8566.03, Success Rate: 14.00%
Episode 80, Avg Reward: -8660.29, Success Rate: 15.30%
Episode 100, Avg Reward: -10495.78, Success Rate: 8.70%

Evaluating agent...
Episode 1, Reward: -7587.89, Success Rate: 20.00%
Episode 2, Reward: -12252.43, Success Rate: 4.00%
Episode 3, Reward: -10389.69, Success Rate: 6.00%
Episode 4, Reward: -8502.84, Success Rate: 16.00%
Episode 5, Reward: -9534.53, Success Rate: 6.00%
Episode 6, Reward: -8157.42, Success Rate: 14.00%
Episode 7, Reward: -9605.93, Success Rate: 4.00%
Episode 8, Reward: -8922.51, Success Rate: 16.00%
Episode 9, Reward: -9702.57, Success Rate: 14.00%
Episode 10, Reward: -9239.32, Success Rate: 10.00%

Average Evaluation Reward: -9389.51
Average Success Rate: 11.00%
```

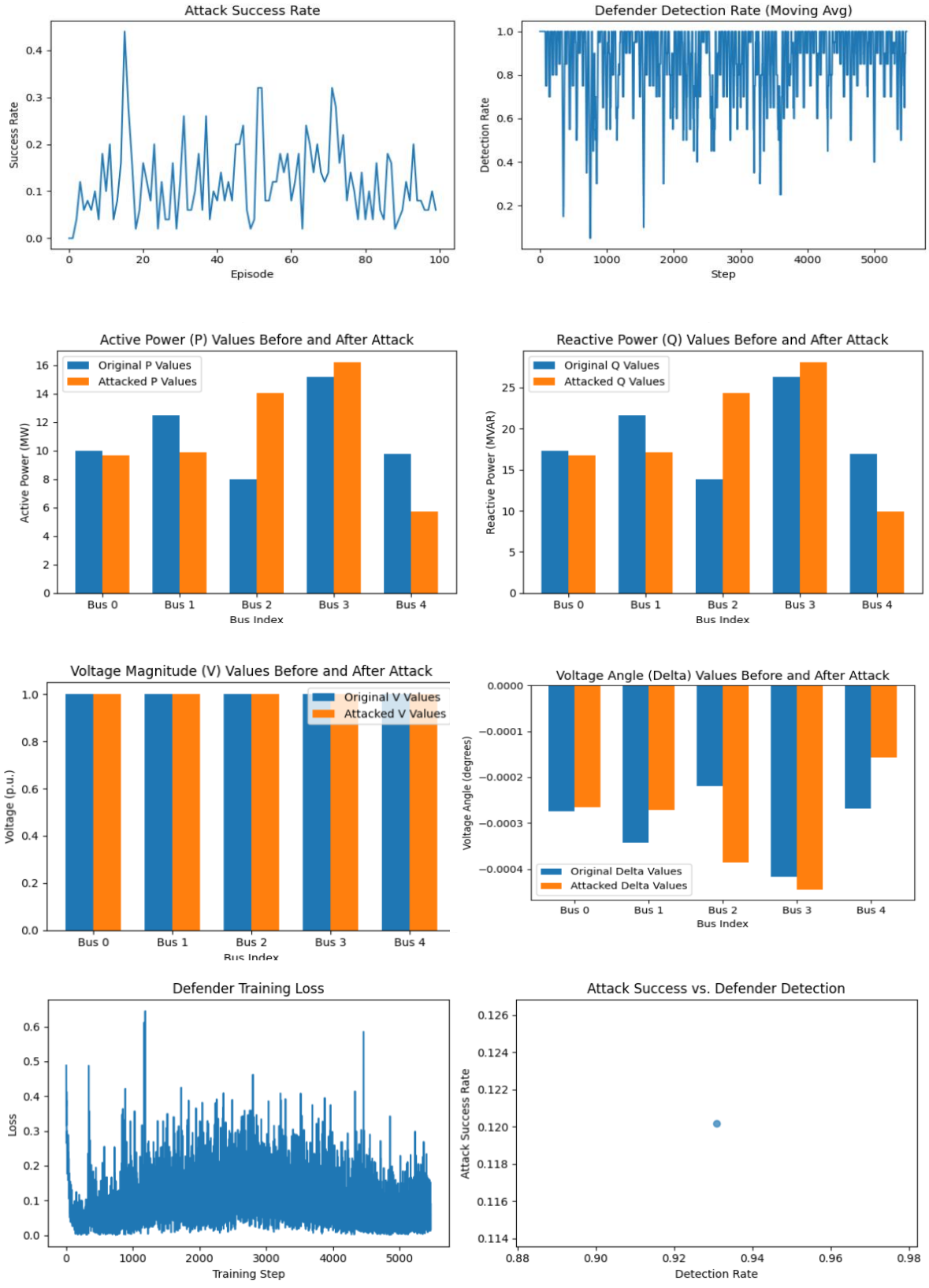


Fig 5.9: Results for RL Attacker with DL Defender Algorithm

```
Original P Values: [10.  12.5  8.  15.2  9.8]
Original Q Values: [17.32050808 21.65063509 13.85640646
26.32717228 16.97409791]
Original V Values: [0.99997657 0.99997072 0.99998126 0.99996439
0.99997704]
Original Delta Values: [-0.00027467 -0.00034334 -0.00021973 -
0.0004175  -0.00026918]
Original Total Power: 55.5

Attacked P Values: [ 9.642415   9.8665285 14.0627365 16.19926
5.729062 ]
Attacked Q Values: [16.70115277 17.08932868 24.35737413
28.0579426   9.9230266 ]
Attacked V Values: [0.99997741 0.99997689 0.99996706 0.99996205
0.99998658]
Attacked Delta Values: [-0.00026485 -0.000271   -0.00038626 -
0.00044495 -0.00015736]
Attacked Total Power: 55.5

P Value Difference: [-0.35758495 -2.63347149  6.06273651
0.99926071 -4.07093792]
Q Value Difference: [-0.61935531 -4.56130642 10.50096767
1.73077032 -7.05107131]
V Value Difference: [ 8.37714544e-07  6.16947587e-06 -
1.42032597e-05 -2.34103467e-06
  9.53689555e-06]
Delta Value Difference: [ 9.82195699e-06  7.23352830e-05 -
1.66529029e-04 -2.74479281e-05
  1.11817322e-04]
Total Power Difference: 0.0

Attack Successful: False

Defender Performance:
Attacks Detected: 4700/5500 (85.45%)
Defender Detection Threshold: 0.5


Training completed!
```

5.4 Algorithm with Reinforcement Learning Attacker and Defender

This algorithm presents the adversarial training algorithm developed for simulating cyber-physical attacks in a smart distribution system. Two reinforcement learning (RL) agents are employed: an attacker that attempts to manipulate the active power of the buses without being detected, and a defender that attempts to detect these manipulations.

5.4.1 Overall Algorithm Flow: Depicted in Fig. 5.10

The algorithm comprises a distribution system environment where:

- The attacker agent modifies active power (P) values at selected buses to achieve its objectives stealthily.
- The defender agent monitors the system and attempts to detect malicious activity based on system measurements.
- Both agents are trained simultaneously using PPO, engaging in a dynamic competition, learning through continual mutual adaptation.

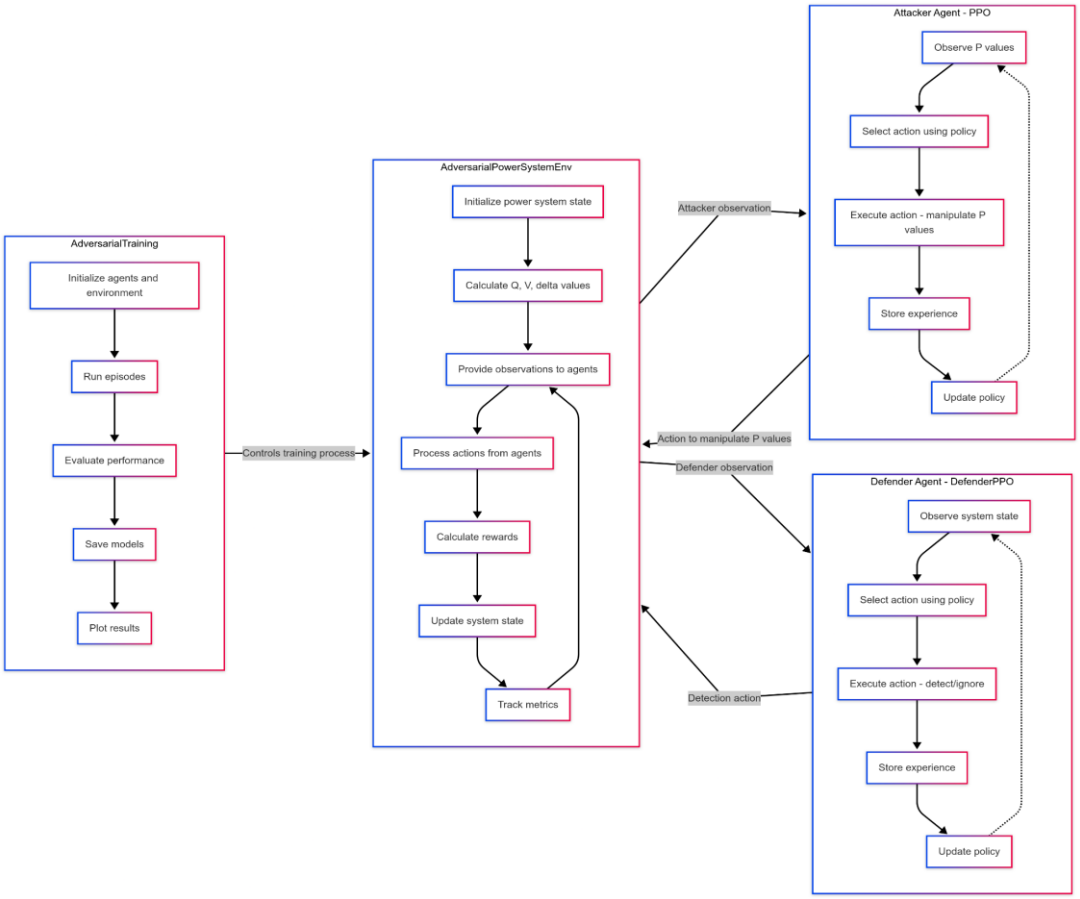


Fig 5.10: Flow of Algorithm with RL Attacker and RL Defender

5.4.2 Components:

1. Environment

The environment simulates a power grid using Panda Power and provides a realistic context for training the attacker and defender. It includes:

- State Variables: Active power (P), reactive power (Q), voltage magnitude (V), and voltage angle (δ) at different buses.

- Flow:
  - **i.** Initialisation with a stable set of original P values.
  - **ii.** Calculation of derived quantities (Q, V, δ) using power flow analysis.
  - **iii.** Observation generation for both agents.
  - **iv.** Action application and state update.
  - **v.** Reward computation for both agents.
  - **vi.** Metrics tracking (attack success, detection accuracy, false positives/negatives).

2. Attacker Agent

The attacker modifies selected P values to induce subtle yet effective system disturbances. The objective is to maximise impact while minimising detection.

- Action Space: Continuous, outputs are percentage modifications to P values.
- Observation Space: Current P values at target buses.

a) Workflow:

- **i.** Observe the current system state.
- **ii.** Select actions using the actor network.
- **iii.** Apply changes to P values in the environment.
- **iv.** Receive reward based on:
  1. Degree of manipulation.
  2. Stealthiness (avoiding significant system deviations).
  3. Power conservation.
  4. Penalty if detected.
- **v.** Store transitions in a replay buffer.
- **vi.** Periodically update policy using PPO.

b) Architecture:

- **i.** Actor Network:
  - **a)** Inputs: P values.
  - **b)** Outputs: Mean and standard deviation of a normal distribution.

c) Uses tanh for action scaling.

ii. Critic Network: Estimates the value function for advantage calculation.

iii. PPO Training:

a) Employs a clipped surrogate objective.

b) Uses Generalised Advantage Estimation (GAE).

c) Includes entropy regularisation to promote exploration.

3. Defender Agent

The defender monitors the system and classifies states as either attacked or normal. It operates in a binary action space.

- Action Space: Discrete (0: no attack, 1: attack detected).
- Observation Space: Full system state (P, Q, V, δ, and summary statistics).

a) Workflow:

i. Observe the system state.

ii. Select action via actor network.

iii. Receive reward based on:

1. True positives and true negatives.
2. Penalties for false alarms and missed detections.

iv. Store experience in a replay buffer.

v. Update policy using PPO.

b) Architecture:

i. Actor Network:

a) Outputs action probabilities (detect or not).

b) Uses ReLu activation for fast convergence.

ii. Critic Network: Same structure as the attacker's critic.

iii. PPO Training:

a) Tailored for discrete action spaces.

b) Uses a higher entropy coefficient to encourage diverse exploration strategies.

5.4.3 Adversarial Training Process

The training process is adversarial and concurrent. Both agents are optimized together, constantly adapting to each other's strategies.

Process Flow:

a) Initialise the environment and agents.

b) For each episode:

- Reset the environment.
- Run multiple time steps
- Attacker selects actions and modifies the environment.
- Defender analyses the new state and attempts detection.
- The environment provides rewards and new observations.
- Both agents store experience.

c) After each episode:

- Update attacker and defender policies using PPO.
- Track performance metrics.
- Periodically save model checkpoints and evaluate models.

5.4.4 Adversarial Dynamics

The adversarial nature of this setup introduces complexity and realism:

- Zero-Sum Interactions: Gains for the attacker imply losses for the defender and vice versa.
- Simultaneous Learning: Both agents evolve their strategies during the same episodes, resulting in a constantly shifting policy landscape.

5.4.5 Reward Structures:

Table 5.1: Reward Structure of Attacker and Defender

Agent	Positive Rewards	Negative Rewards
Attacker	Successful manipulation, stealth, and balance	Detection, power imbalance
Defender	True positives, true negatives	False positives, false negatives

5.4.6 Evaluation Metrics:

- Attack Success Rate: Frequency of undetected successful attacks.
- Detection Rate: Proportion of attacks correctly flagged.
- False Positive Rate: Frequency of benign states incorrectly flagged.
- F1 Score: Harmonic mean of precision and recall for defender.
- Confusion Matrix Stats: TP, FP, TN, FN.

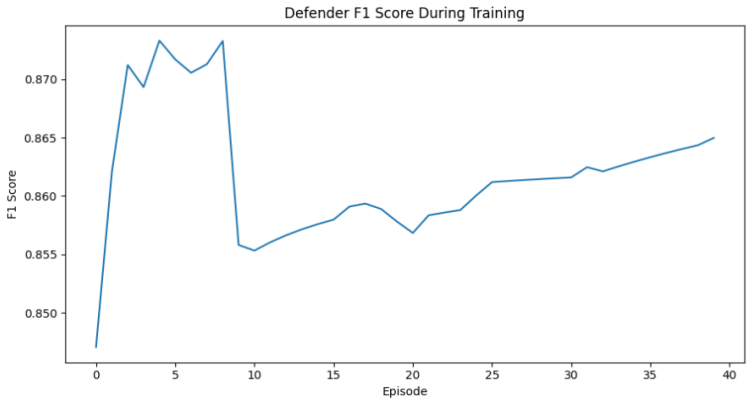
This structure enables both agents to develop sophisticated, adaptive strategies. The attacker learns to camouflage malicious activity within the normal operating noise of the system, while the defender learns to distinguish genuine disturbances from fluctuations.

5.4.7 Results:

i. Training for 40 Episodes with an Episode length of 100 Steps

```
Starting adversarial training...
Episode 10/40
Attacker Reward: -132.60, Defender Reward: -17.07
Attack Success Rate: 0.0220, Detection Rate: 0.9432
F1 Score: 0.8558
---
Episode 20/40
Attacker Reward: -208.03, Defender Reward: 117.91
Attack Success Rate: 0.0110, Detection Rate: 0.9699
F1 Score: 0.8578
---
Episode 30/40
Attacker Reward: -229.00, Defender Reward: 129.52
Attack Success Rate: 0.0073, Detection Rate: 0.9800
F1 Score: 0.8615
---
Episode 40/40
Attacker Reward: -213.54, Defender Reward: 134.71
Attack Success Rate: 0.0055, Detection Rate: 0.9850
F1 Score: 0.8650
---

Training complete!
```



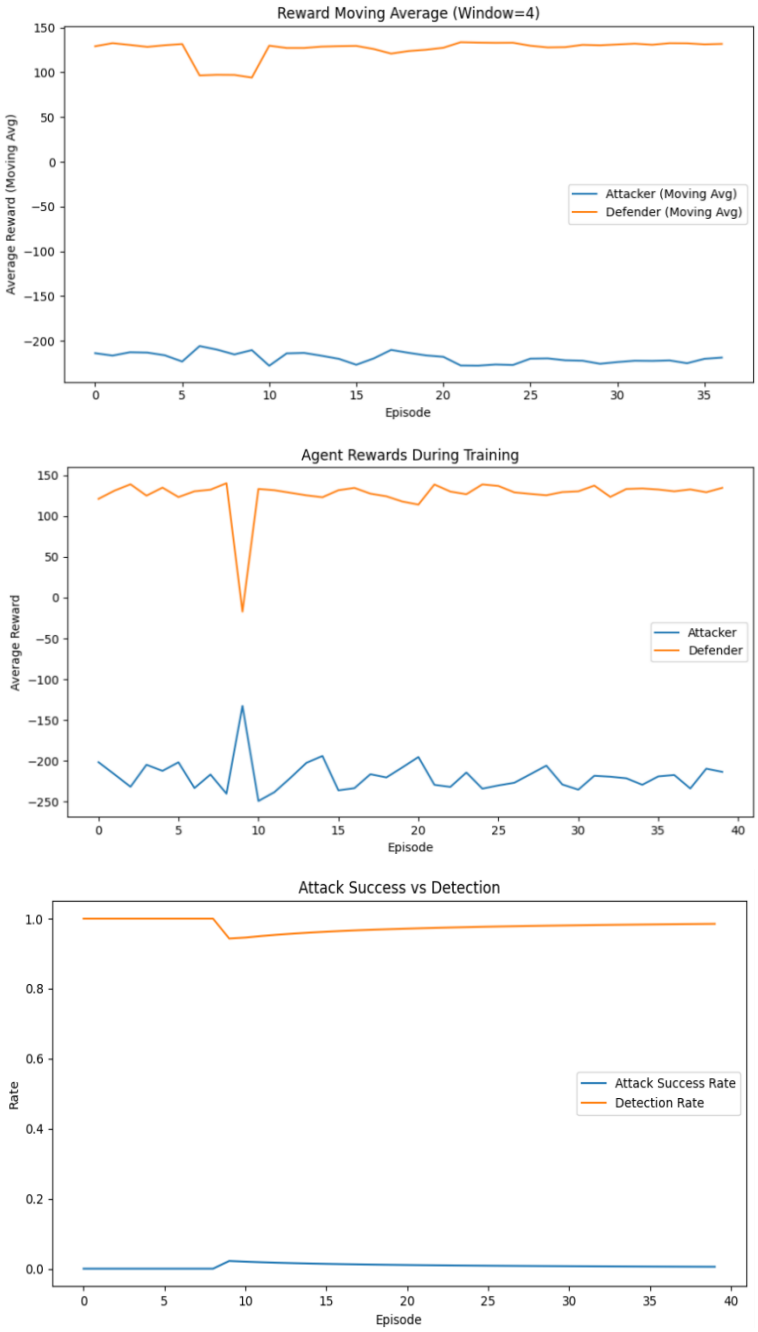


Fig 5.11: Results of RL Attacker with RL Defender Algorithm

ii. Evaluating 5 Episodes with a length of 100 Steps and a Confusion Matrix

```
Starting evaluation...

Evaluation Episode 1/5
Attack Success Rate: 0.0054
Detection Rate: 0.9854
False Positive Rate: 0.2259
F1 Score: 0.8646
```

```
Confusion Matrix:
True Positives: 1553, False Positives: 463
True Negatives: 11, False Negatives: 23
Precision: 0.7703, Recall: 0.9854
F1 Score: 0.8647, Accuracy: 0.7629

Evaluation Episode 2/5
Attack Success Rate: 0.0052
Detection Rate: 0.9858
False Positive Rate: 0.2252
F1 Score: 0.8652

Confusion Matrix:
True Positives: 1593, False Positives: 473
True Negatives: 11, False Negatives: 23
Precision: 0.7711, Recall: 0.9858
F1 Score: 0.8653, Accuracy: 0.7638

Evaluation Episode 3/5
Attack Success Rate: 0.0051
Detection Rate: 0.9861
False Positive Rate: 0.2247
F1 Score: 0.8658

Confusion Matrix:
True Positives: 1633, False Positives: 483
True Negatives: 11, False Negatives: 23
Precision: 0.7717, Recall: 0.9861
F1 Score: 0.8659, Accuracy: 0.7647

Evaluation Episode 4/5
Attack Success Rate: 0.0050
Detection Rate: 0.9864
False Positive Rate: 0.2241
F1 Score: 0.8663

Confusion Matrix:
True Positives: 1673, False Positives: 493
True Negatives: 11, False Negatives: 23
Precision: 0.7724, Recall: 0.9864
F1 Score: 0.8664, Accuracy: 0.7655

Evaluation Episode 5/5
Attack Success Rate: 0.0049
Detection Rate: 0.9867
False Positive Rate: 0.2236
F1 Score: 0.8668

Confusion Matrix:
True Positives: 1713, False Positives: 503
True Negatives: 11, False Negatives: 23
Precision: 0.7730, Recall: 0.9868
F1 Score: 0.8669, Accuracy: 0.7662

Evaluation Results:
Average Attack Success Rate: 0.0051
Average Detection Rate: 0.9861
Average F1 Score: 0.8658
Average Attacker Reward: -218.28
Average Defender Reward: 133.42
```

CHAPTER 6
OVERALL CONCLUSION AND FUTURE WORKS

6.1. Load Prediction Model

The Load Prediction Model successfully employs an LSTM-based deep learning model with attention to electricity load forecasting. The model captures temporal dependencies by leveraging 168 past time steps, while the attention mechanism enhances interpretability by focusing on the most relevant time steps. The model achieves accurate predictions, validated using MAE, RMSE, and R^2 scores. Future improvements include hyperparameter tuning, hybrid models, and incorporating external factors for even better forecasting.

6.2. RL-Based Attacker

The Attacker Model employs a Proximal Policy Optimisation (PPO)-based reinforcement learning algorithm to subtly manipulate system parameters (P values) while avoiding detection. The model successfully learns to balance effective attacks with stealth by adjusting P values within a ±1% range. The effectiveness of the model is evaluated based on how well it remains undetected, with reward functions that penalise detection.

A comprehensive environment was built to simulate the power system, with constraints ensuring power conservation. The system's state and action spaces were carefully defined, allowing the model to interact with realistic operational limits. Power flow is adjusted to ensure balance while being manipulated by the attacker, and the system behaviour was closely monitored for threshold breaches, particularly with a 15% tolerance for detection.

6.3. RL-Based vs DL-Based Defender

From the algorithms formulated and trained, we can conclude that for a Reinforcement Learning based Attacker, a Reinforcement Learning based Defender offers the highest average detection ratio of 0.9861, stopping the average attack success rate at a mere 0.0051. Followed by Deep Learning based Defender offering an

average detection rate of 0.8545, limiting the average success rate to 0.11. The worst algorithm was the one without any defender, with no detection, but due to a strong RL Attacker had the average attack success rate of 0.996. However, there is a significant difference between an RL Defender and a DL Defender against an attacker, depending on the circumstances and situations.

6.3.1 RL-Based Defender Advantages:

1. **Adaptability**: An RL-based defender can continuously adapt to new attack strategies without explicit training examples
2. **Strategic Responses**: Can learn optimal detection policies that account for attacker adaptation
3. **No labelled data requirement**: Doesn't need pre-classified examples of attacks and normal states
4. **Long-term optimisation**: Optimises for cumulative detection performance rather than immediate classification

6.3.2 DL-Based Defender Advantages:

1. **Stability**: A more stable training process with fewer hyperparameters
2. **Faster initial learning**: Can quickly learn patterns from historical data
3. **Less computational overhead**: Typically requires less computation during operation
4. **Interpretability**: Classification models can be more interpretable than policy-based models

6.3.3 Which is Better?

The optimal choice depends on your specific scenario:

RL-based defender is better when:

- The attacker is highly adaptive and constantly changing strategies
- You have limited labelled data, but can simulate many scenarios
- Detection has complex sequential dependencies
- You want the defender to anticipate future attack strategies

A DL-based defender is better when:

- You have access to substantial labelled data of attacks and normal operations
- The attack patterns are relatively stable
- Computational resources during operation are limited

6.4. Future Works

Future work involves combining load forecasting, state estimation algorithms, RL-based attackers and Defenders to simulate and analyse False Data Injection Attacks (FDIAS) on the real smart grids, assessing their impact on stability and accuracy.

The next steps also include further refining the attack model by improving its stealth capabilities and extending it to simulate attacks across larger, more complex power grids. Future work will also explore advanced detection methods and potential countermeasures, ensuring the model contributes to a more secure and stable smart grid infrastructure.

Future improvements include experimenting with more complex attack vectors and using a more advanced defence mechanism. Many advanced defence systems use a hybrid approach, combining the pattern recognition strengths of supervised learning with the strategic adaptation capabilities of RL.

REFERENCES

[1] Luo, Weifeng, and Liang Xiao. "Reinforcement learning based vulnerability analysis of data injection attack for smart grids." *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021.

[2] Naware, Dipanshu, Hira Singh Sachdev, and Saransh Chourey. "Investigating the Role of Data Preprocessing for Load Demand Forecasters in Smart Household Applications." *2024 Third International Conference on Power, Control and Computing Technologies (ICPC2T)*. IEEE, 2024.

[3] Zhang, Guihai, and Biplab Sikdar. "A novel adversarial FDI attack and defence mechanism for Smart Grid demand-response mechanisms." *IEEE Transactions on Industrial Cyber-Physical Systems* (2024).

[4] Naware, Dipanshu, and Arghya Mitra. "Data-driven Technology Applications in Planning, Demand-side Management, and Cybersecurity for Smart Household Community." *IEEE Transactions on Artificial Intelligence* (2024).

[5] Roomi, Muhammad M., et al. "Analysis of false data injection attacks against automated control for parallel generators in IEC 61850-based smart grid systems." *IEEE Systems Journal* 17.3 (2023): 4603-4614.

[6] Rahman, Moshfeka, Jun Yan, and Emmanuel Thepie Fapi. "Adversarial Artificial Intelligence in Blind False Data Injection in Smart Grid AC State Estimation." *IEEE Transactions on Industrial Informatics* (2024).