Department of Electrical Engineering and Computer Science
Siegen University
Chair of Interconnected Automation Systems
Prof. Dr.-Ing. Oliver Wallscheid

Master Thesis

Development of a Reinforcement learning-based Controller using Short-Term PV Forecasts for Grid Voltage Stability

by

Hardik Zalavadiya
Student ID: 1635018

Supervisor: Henning Schlachter
Thesis Nr.: MA_002_Zalavadiya Hardik
Filing Date: March 6, 2025

Declaration of Authorship

I declare that I have authored this thesis independently, that I have not used other than the declared sources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. I am aware that the use of AI-supported software tools is a (permissible) aid, the use and scope of which must be indicated. This work was not previously presented to another examination board and has not been published.

Siegen, 07.03.2025 (Date) Hardik Zalavadiya (Signature)

Acknowledgments

First and foremost, I want to extend my heartfelt thanks to Mr. Henning Schlachter, my supervisor at DLR - Institute for Networked Energy Systems. Henning, you have been much more than just a supervisor—you have been a mentor, a guide, and at times, my much-needed voice of reason. Your patience, constant encouragement, and technical expertise made a huge difference in shaping this thesis. From the very first day, you helped me settle into a new city and navigate the complexities of my research, and for that, I am truly grateful. When I was frustrated with my final code and couldn't get the expected results, you not only helped me fix the issue but also calmed me down and gave me the motivation to keep going.

I would also like to sincerely thank Mr. Holger Behrends, my group leader at DLR, for giving me the opportunity to work in such an excellent research environment. Your guidance and leadership helped create a space where I could learn and grow.

A special thank you to Professor Dr.-Ing. Oliver Wallscheid from Siegen University for his academic insights and support throughout this thesis. His expertise provided valuable direction for my research.

Most importantly, I want to thank my parents. Your unconditional love and support were my backbone during this entire journey. There were times when things felt overwhelming, but knowing that you were always there for me gave me the strength to push forward.

This thesis has been a journey filled with challenges, learning, and personal growth, and I couldn't have done it without the support of all these incredible individuals. Thank you all for being part of this chapter in my life.

Siegen, March 6, 2025
Hardik Zalavadiya

Abstract

Ensuring voltage stability in power grids with high photovoltaic (PV) penetration is a critical challenge due to the intermittent nature of solar energy. This thesis presents the development of a reinforcement learning (RL)-based voltage control framework that leverages short-term PV forecasts to enhance grid stability. The research focuses on integrating machine learning-driven voltage forecasting with RL-based decision-making, executed within a co-simulation framework that connects MATLAB-based grid simulations with Python-based control logic using Mosaik.

A machine learning model was developed to predict grid voltage fluctuations based on forecasted PV power generation and load demand. The goal was to provide an accurate forecast of future voltage states to inform the RL-based controller. The co-simulation framework was established to enable real-time data exchange between MATLAB and Python, allowing the RL algorithm to make informed voltage control decisions. However, despite the structured approach, several technical challenges hindered the successful execution of the RL-based control system. Synchronization mismatches between MATLAB and Python, inconsistencies in data exchange, and computational constraints due to multi-threading and queue-based communication prevented the RL controller from receiving complete state observations, thereby rendering training infeasible. These challenges highlight the complexity of implementing real-time reinforcement learning for voltage stability control.

The findings from this research underscore the necessity of improved synchronization mechanisms, buffering techniques, and refined RL training methodologies to better integrate and train RL-based voltage control within the co-simulation platform. Future research directions include the implementation of enhanced synchronization strategies, offline pre-training of RL models using historical data, and the incorporation of additional predictive features into the voltage forecasting model. Furthermore, alternative control strategies, such as hybrid AI-based controllers that integrate rule-based decision-making with reinforcement learning, could improve reliability and performance.

This research contributes to the growing body of work on AI-driven grid stability solutions by demonstrating the challenges and potential improvements in reinforcement learning-based voltage control within the co-simulation framework. Addressing these challenges through enhanced co-simulation frameworks and refined control methodologies will be crucial for improving our co-simulation approach, enabling seamless transitions between software-based and hardware-based setups for voltage control algorithm development.

Contents

1 Introduction

1.1 Motivation

The shift towards renewable energy sources has become a global imperative, driven by the urgent need to mitigate climate change and reduce dependence on finite fossil fuels. Renewable energy resources, such as photovoltaic (PV) systems, wind turbines, and hydropower, offer sustainable alternatives that significantly lower greenhouse gas emissions and reduce environmental impacts associated with traditional energy generation [1]. This transition not only supports environmental sustainability but also enhances energy security by diversifying energy supply sources and reducing geopolitical risks linked to fossil fuel dependency [2].

Integrating renewable energy into the power grid, however, poses numerous technical challenges due to the variable and intermittent nature of sources like wind and PV. Unlike conventional power plants, which can adjust output based on demand, renewable energy availability depends on factors outside human control, such as weather and time of day. This variability can destabilize the grid, making voltage regulation more challenging [3].

PV systems, which convert sunlight directly into electricity, are among the fastest-growing sources of renewable energy. Their appeal lies in their scalability—from small rooftop installations to large utility-scale farms—their minimal operating costs, and their zero direct emissions. However, high PV penetration can lead to rapid voltage fluctuations, especially under changing weather conditions, which may destabilize the grid and cause power quality issues or blackouts [3]. Additionally, the intermittent nature of PV and wind energy results in unpredictable power flow variations, further complicating grid stability [1].

To address these challenges, battery storage systems are increasingly paired with PV installations. Battery storage can mitigate the variability of PV power by storing excess energy during peak production times and discharging it during periods of low production or high demand. This capability not only helps to stabilize the grid but also enhances the efficiency of energy use, ensuring that the generated renewable energy is not wasted but used in an optimized manner [1].

Battery storage also plays a critical role in enhancing grid resilience, providing backup power during outages and helping to balance supply and demand. By smoothing out the supply of renewable energy, batteries help to maintain a stable voltage level within the grid, which is crucial for the operational reliability of all connected devices and systems [4].

Thus, the integration of renewable energy, particularly PV systems, into power grids is a critical step towards achieving energy sustainability and reducing carbon footprints. However, the challenges posed by their integration, particularly in terms of grid voltage stability, require innovative solutions such as advanced battery storage systems and sophisticated grid management technologies. As the penetration of renewable energy continues to grow, these technologies will be pivotal in ensuring the reliability and stability of the power grid.

While renewable energy sources provide sustainable power, their ability to contribute to voltage regulation differs from conventional power plants. Traditional generators, such as those powered by coal or natural gas, can actively regulate voltage by adjusting their reactive and active power output. However, PV systems can participate in voltage regulation through power electronics-based solutions such as Q(U) control strategies, smart inverters, and coordinated energy storage integration [4].

One crucial yet often overlooked aspect of voltage stability management is the role of short-term forecasting in optimizing control strategies. Traditional voltage regulation methods, such as tap-changing transformers and capacitor banks, lack the flexibility to respond quickly to rapid fluctuations in renewable generation. This limitation necessitates more adaptive, data-driven approaches. Reinforcement learning (RL), when combined with predictive PV forecasting, provides a proactive solution by learning from historical patterns and anticipating future grid conditions. The novelty of this research lies in integrating short-term PV forecasts with RL to optimize voltage stability control, rather than simply applying RL reactively.

1.2 Statement of the problem

The rapid expansion of PV and wind energy has revolutionized global energy production. However, integrating these sources into existing grids presents challenges in maintaining voltage stability and grid reliability. The inherent variability of renewable energy sources results in fluctuating power outputs, directly impacting voltage levels [1].

For instance, PV output varies widely throughout the day and is subject to abrupt changes due to cloud cover or other atmospheric conditions. These fluctuations can lead to sudden spikes or drops in voltage levels, potentially destabilizing the grid if not managed correctly. Similarly, wind energy output can change rapidly with wind speed variations, adding another layer of complexity to grid management [2].

These dynamics pose a unique challenge for voltage stability, defined as the ability of a power system to maintain steady voltages at all buses in the system after being subjected to a disturbance. Voltage instability can lead to various operational issues, including inefficient power distribution, increased losses, risk of equipment damage, and even widespread power outages [5].

Traditional voltage control mechanisms, such as tap-changing transformers and capacitor banks, are often insufficient to handle the rapid and random fluctuations introduced by high levels of renewable integration. These devices were designed for relatively stable power flows, making them less effective in modern decentralized energy systems [3].

Additionally, most existing grid infrastructure and control strategies are based on centralized models that are increasingly inadequate in a landscape dominated by distributed generation sources. The centralized control systems struggle to effectively manage the dispersed and variable energy inputs from numerous small-scale renewable installations. This disconnect highlights the need for developing new control strategies that are flexible and responsive enough to manage the dynamics of renewable energy within the grid effectively [6].

A critical gap in current grid management tools is the lack of predictive capability. Traditional methods often rely on real-time measurements, reacting to voltage deviations rather than anticipating them. This reactive approach results in delayed responses and suboptimal voltage control. The increasing complexity of grid dynamics due to renewable integration underscores the importance of predictive analytics and intelligent control strategies [7].

RL has emerged as a promising adaptive control technique for power systems, offering the ability to learn optimal control policies from past experiences. However, its application in voltage stability remains limited, primarily because most RL-based methods function reactively rather than proactively. Furthermore, RL's "black-box" nature raises concerns about transparency, reliability, and trustworthiness among grid operators [8]. This highlights the need to integrate short-term PV forecasts into RL-based voltage control, enabling controllers to make informed, predictive decisions rather than solely reacting to disturbances.

Thus, the primary problem addressed in this thesis is: How can short-term PV forecasts enhance RL-based voltage control strategies to ensure stable and reliable power grid operation in high PV penetration scenarios? This study proposes an RL-based control framework that incorporates short-term PV forecasting to improve grid voltage stability, reduce fluctuations, and optimize power flow management.

1.3 Research questions

The central question driving this thesis is:

"How can short-term PV forecasts enhance grid voltage stability through a RL-based power management system for a PV/battery system?"

This question reflects the need to integrate advanced analytical and control techniques to manage the dynamic and unpredictable nature of renewable energy sources effectively.

To comprehensively address the main research question, several sub-questions have been formulated:

1. How can PV forecast data be leveraged to accurately predict future voltage behavior?
2. To what extent can the proposed approach enhance voltage stability, such as reducing tolerance band crossing events or voltage volatility, compared to current reactive power-based methods (e.g., Q(U)) [9]?
3. How much does the algorithm enhance PV power utilization while ensuring grid voltage stability and meeting load demand, while adhering to optimized battery characteristics?
4. What safety and reliability considerations are essential in designing a RL-based power management approach for PV/battery systems?

1.4 Thesis objectives

The primary objective of this study is to develop a RL-based control system that enhances grid voltage stability by leveraging short-term PV forecasts. To achieve this, a predictive model is designed to forecast voltage fluctuations based on PV power generation and load demand, enabling proactive decision-making in power management. This forecasting model serves as the foundation for the RL controller, which optimizes voltage control strategies in real-time by learning from historical and predicted grid conditions.

To validate the effectiveness of the RL-based control approach, the system is implemented within a co-simulation framework that integrates a MATLAB-based grid simulation with a Python-based RL controller using the Mosaik platform [8]. The use of a co-simulation platform was a requirement of the overarching research project at the DLR Institute of Networked Energy Systems in which this thesis work was embedded, as it enables seamless integration between different simulation environments and ensures the feasibility of real-time control decision-making. The co-simulation allows for a realistic representation of grid dynamics and facilitates direct interaction between the simulated power system and the artificial intelligence (AI)-driven control mechanism. This setup not only ensures a robust testing environment but also serves as a stepping stone for future deployment

in hardware-based control systems, as the long-term objective is to transition from a simulation-based validation to real-world implementation.

The study further aims to compare the RL-based control system with traditional voltage regulation methods to assess improvements in grid stability, operational efficiency, and response time. By benchmarking the proposed approach against conventional techniques, the research evaluates whether RL offers a more effective alternative for managing voltage fluctuations in power grids with high PV penetration.

Although RL-based methods have shown potential in complex decision-making tasks, their black-box nature raises concerns among grid operators regarding transparency, interpretability, and reliability. Unlike traditional control methods, which follow deterministic rules, RL-based controllers adapt dynamically based on learned experiences, making it challenging to predict their behavior in all grid scenarios. This study addresses these concerns by evaluating the robustness, safety, and explainability of the proposed RL approach, ensuring it can be seamlessly integrated into existing grid infrastructures without introducing instability or risks.

The findings from this study will contribute to advancing AI-driven grid management strategies, demonstrating how RL, when combined with short-term PV forecasting, can enhance voltage stability in a transparent and operator-trustworthy manner, paving the way for future intelligent control solutions in renewable energy-based power networks.

By addressing these questions and objectives, the research aims to demonstrate the potential for advanced machine learning techniques, specifically RL, to enhance the management of power grids in the context of increasing renewable energy integration. Additionally, this study highlights the benefits of incorporating short-term PV forecasts in RL-based control strategies to improve voltage stability and grid reliability. The ultimate vision is to improve grid stability, optimize the use of renewable resources, and ensure a reliable power supply in the face of growing energy demand and the transition to sustainable energy sources.

1.5 Structure of the thesis

The structured approach ensures a thorough examination of the research questions and provides a comprehensive understanding of the methodologies, results, and implications of this study.

The remainder of the thesis is organized as follows:

Chapter 2 provides the theoretical background and literature review. It explores fundamental concepts of voltage stability, existing grid management strategies, and the role of RL in power system control. This chapter reviews current methodologies, identifies limitations in traditional approaches, and highlights the need for integrating forecasting methods into RL-based control strategies.

Chapter 3 details the methodology, describing data collection and preprocessing techniques, particularly the use of unprocessed Eye2Sky data. It further elaborates on the development of the voltage forecasting model, RL implementation, and the co-simulation framework that integrates MATLAB-based grid simulations with Python-based control algorithms.

Chapter 4 presents the results and discussion, analyzing the outcomes of the voltage forecasting model and evaluating the performance of the co-simulation framework. Challenges in data synchronization and RL implementation are discussed, along with potential solutions and future refinements to enhance real-time decision-making capabilities.

Chapter 5 concludes the thesis by summarizing key findings, highlighting the contributions of the research, and outlining directions for future work. The identified challenges in RL-based grid control emphasize the need for improved synchronization techniques, offline RL training, and further refinements in predictive modeling.

Appendices: This appendix contains high-level pseudocode descriptions for the core components of the system developed in this thesis. Specifically, it provides pseudocode for the co-simulation platform, the queue management system, the RL-based voltage controller, and the voltage prediction model, offering an overview of the logic and data flow in each subsystem.

Having established the motivation, problem statement, and research objectives in this chapter, the next chapter provides the theoretical background and a review of existing approaches to voltage stability, RL, and PV forecasting in grid management.

2 Fundamentals and state-of-the-art

The integration of renewable energy sources, particularly photovoltaic and wind power, has significantly transformed modern power grids. These energy sources contribute to sustainability by reducing greenhouse gas emissions and minimizing dependence on fossil fuels. However, their inherent variability introduces significant challenges in maintaining grid stability, particularly in terms of voltage fluctuations and frequency deviations. Unlike conventional power plants, which can adjust their generation in response to load variations, PV and wind power output are highly dependent on environmental factors such as solar irradiance and wind speed. This intermittency complicates power balance and requires new approaches for grid management.

Chapter 1 introduced the broad challenges of integrating renewable energy into power systems, highlighting concerns such as voltage instability, grid reliability, and the limitations of conventional control mechanisms. This chapter builds upon that discussion by exploring the fundamental principles behind voltage stability and analyzing existing solutions used to mitigate instability issues in modern grids.

The chapter begins by discussing the challenges of renewable energy integration, focusing on how PV intermittency affects voltage regulation in distribution networks and why traditional grid management methods struggle to accommodate these fluctuations. It then examines current approaches to voltage stability, including both conventional solutions—such as capacitor banks, transformer tap changers, and synchronous compensators—and more advanced techniques like static var compensators (SVCs), flexible AC transmission systems (FACTS), and smart inverters.

Following this, the chapter introduces recent advancements in grid management technologies, including both hardware-based solutions—such as advanced inverters and energy storage systems—and software-driven innovations, including real-time monitoring, predictive analytics, energy management systems (EMS), and distributed energy resource management systems (DERMS). These technologies enhance grid flexibility and play a crucial role in maintaining stability under high renewable energy penetration.

The role of artificial intelligence (AI) in modern grid control is then explored, with an emphasis on the growing use of machine learning (ML) and RL in grid optimization. AI-driven techniques are increasingly being investigated for applications in voltage control, demand-side management, energy storage optimization, and predictive forecasting. This discussion leads to a detailed analysis of RL, its fundamental principles, and its potential to enhance voltage stability in PV-integrated power grids.

By outlining these theoretical foundations and technological advancements, this chapter establishes the groundwork for the RL-based voltage control approach developed in later sections of this thesis. Additionally, it highlights how short-term PV forecasting can be integrated with AI-driven grid management, demonstrating the potential of data-driven optimization strategies to enhance grid stability and operational efficiency.

2.1 Challenges of renewable energy integration

The integration of renewable energy sources, such as PV and wind, has significantly transformed modern energy systems. While these sources contribute to sustainability by reducing carbon emissions and reliance on fossil fuels, their inherent variability presents challenges in maintaining grid stability and reliability [10]. Unlike conventional power plants, which provide a stable and controllable power output, renewable energy generation is highly dependent on environmental conditions. PV output fluctuates throughout the day due to variations in solar irradiance caused by cloud cover, seasonal changes, and the natural day-night cycle. Similarly, wind power is influenced by unpredictable changes in wind speed and direction, leading to substantial variations in power generation [10].

A key concern in renewable energy integration is its impact on grid stability. Traditional grids are designed to manage stable, predictable loads and rely on controllable generation sources to balance supply and demand. The introduction of intermittent renewable energy disrupts this balance, leading to voltage fluctuations, frequency instability, and power quality issues [11]. Renewable energy sources do not inherently provide voltage and frequency support as conventional synchronous generators do, making it necessary to deploy additional stabilization mechanisms [11].

To address these challenges, several strategies have been developed. One common approach is the reinforcement of grid infrastructure through expanded transmission capacity and improved interconnections, allowing power flows to be adjusted dynamically based on generation patterns [12]. Energy storage systems, particularly battery technologies, have emerged as critical solutions for mitigating the variability of renewable sources. These systems store excess energy during periods of high production and discharge it when generation is low, thereby enhancing grid flexibility and stability [12]. Additionally, demand-side management techniques, such as dynamic pricing and load-shifting programs, enable consumers to adjust their electricity usage in response to grid conditions, reducing stress on the system during peak variability [13].

Advances in forecasting technologies have also played a key role in improving grid stability amid increasing renewable penetration. By utilizing sophisticated weather prediction models and real-time monitoring, grid operators can anticipate fluctuations in solar and wind generation and take proactive measures to maintain system stability [13]. However, these forecasting methods are typically applied on a larger time scale, such as day-ahead planning, rather than for real-time voltage control. Since this thesis focuses on short-term PV forecasting, it aims to complement these longer-term forecasting techniques with real-time decision-making strategies.

Despite these advancements, integrating large-scale renewable energy remains a complex challenge that requires further improvements in storage technologies, smart grid solutions, and AI-driven grid management techniques. Addressing these challenges is crucial for ensuring a stable, reliable, and sustainable power supply as the world transitions toward a renewable energy-dominant future.

2.2 Current approaches to voltage stability

As renewable energy penetration increases, ensuring voltage stability has become a central challenge in modern power grids. Voltage stability refers to the ability of a power system to maintain acceptable voltage levels under normal operating conditions and after being subjected to disturbances. Conventional voltage control strategies rely on mechanical devices such as on-load tap changers, voltage regulators, and shunt capacitors, which adjust transformer ratios or inject reactive power to regulate voltage levels [5]. However, these traditional approaches face significant challenges in addressing the dynamic fluctuations introduced by high levels of renewable energy generation.

One primary limitation of conventional voltage regulation techniques is their slow response time. Mechanical control mechanisms, including tap-changing transformers, were designed for stable and predictable power flows, making them inadequate for handling the rapid and unpredictable fluctuations characteristic of PV and wind power [14]. Since renewable energy sources exhibit rapid variations in generation, traditional voltage control devices often fail to react in real time, leading to temporary periods of instability before corrective actions can take effect.

Another major challenge arises from the transition toward decentralized power generation. Traditional grid management techniques were originally designed for centralized power systems, where large power plants supplied energy in a hierarchical, top-down manner. In contrast, modern power grids are shifting toward decentralized and distributed generation, where renewable energy sources are installed closer to the load. This transition complicates voltage control, as existing methods struggle to adapt to the bidirectional power flows introduced by distributed energy resources (DERs) [14].

To address these limitations, advanced power electronics solutions have been developed. Technologies such as SVCs and FACTS provide faster and more flexible reactive power compensation, enabling real-time voltage control [15]. Unlike traditional capacitor banks,

which rely on mechanical switching, FACTS devices utilize power electronics to dynamically regulate voltage levels in response to real-time grid conditions, significantly improving adaptability in grids with high renewable penetration.

In addition to hardware-based solutions, modern smart grid technologies leverage real-time monitoring, predictive analytics, and adaptive control algorithms to enhance voltage stability. The deployment of DERMS allows utilities to coordinate DERs such as PV systems, battery storage, and demand-side management solutions to dynamically adjust voltage levels based on grid conditions [16].

Although progress has been made, achieving long-term voltage stability in grids with high renewable energy penetration remains an ongoing challenge. The increasing shift toward AI-driven control strategies, such as machine learning and RL-based voltage regulation, offers promising solutions for overcoming these challenges. By incorporating real-time data processing and predictive modeling, AI-based approaches have the potential to enhance voltage stability and grid resilience in a way that traditional control mechanisms cannot. The following sections explore how machine learning and RL offer promising solutions for further improving voltage control in renewable-integrated power systems.

2.3 Advancements in grid management technologies

As renewable energy penetration increases, grid management technologies have evolved to enhance stability, flexibility, and efficiency [17]. These advancements span both hardware-based solutions, such as power electronic converters and energy storage systems, and software-driven innovations, including real-time monitoring, EMS, and DERMS. Together, these technologies play a crucial role in maintaining reliable grid operation in an environment with high renewable energy penetration.

One of the most significant hardware advancements is the development of smart inverters and power electronic converters [18], which enable PV and wind energy systems to provide voltage and frequency regulation. Unlike conventional synchronous generators, renewable energy sources do not inherently provide inertia or reactive power support. Modern grid-connected inverters utilize advanced control strategies, such as Q(U) control (adjusting reactive power based on voltage deviations) and $\cos\phi$ control (adjusting power factor in response to grid conditions), to stabilize voltage levels dynamically [19].

In addition to hardware solutions, software-based advancements have significantly improved grid flexibility. EMS optimize energy consumption by forecasting demand, adjusting load schedules, and coordinating distributed generation assets. By integrating real-time data analytics, EMS enhances energy efficiency while minimizing grid disturbances [16]. Similarly, DERMS provide centralized coordination of distributed energy resources, ensuring that PV systems, battery storage, and controllable loads operate in a way that supports overall grid stability [13].

Moreover, the deployment of Advanced Metering Infrastructure (AMI) allows utilities to monitor voltage conditions in real time, enabling automated grid adjustments and demand-side response mechanisms [20]. By leveraging predictive analytics, these technologies enhance the grid's ability to respond dynamically to fluctuations in renewable energy generation, ensuring voltage stability and efficient power distribution.

Notwithstanding these achievements, challenges remain in achieving full grid stability under high renewable penetration. The next section explores how machine learning and RL are being developed as data-driven solutions to further enhance voltage control and grid reliability.

2.4 Application of machine learning in grid management

The integration of machine learning (ML) in grid management has revolutionized traditional power system operations, enabling more efficient, data-driven decision-making processes. With the increasing penetration of renewable energy sources, the complexity of power system dynamics has grown significantly, making conventional rule-based control strategies less effective. Machine learning techniques provide an alternative approach by leveraging historical and real-time data to optimize grid stability, forecast energy demand, and automate voltage regulation.

Several machine learning methodologies have been applied to grid management, each offering unique advantages based on the specific challenges they address. Supervised learning is one of the most commonly used approaches, where models are trained on historical data to predict future grid behavior [21]. This method is widely employed for load forecasting, renewable energy output prediction, and fault detection in power networks.

By analyzing past trends and real-time measurements, supervised learning algorithms can enhance the accuracy of demand forecasts and improve the integration of intermittent renewable energy sources. Another key technique, unsupervised learning, is used for anomaly detection and clustering in grid operations. This approach is particularly valuable for identifying abnormal power consumption patterns, detecting cyber threats, and segmenting consumers based on their energy usage behavior [22].

Beyond predictive analytics, a major research focus has been the application of RL for real-time grid control. Unlike traditional optimization techniques, RL enables grid management systems to learn and improve over time through trial and error. In this framework, an RL agent interacts with the grid environment, receives feedback in the form of rewards or penalties, and continuously refines its decision-making policy to optimize grid performance [23]. This capability makes RL particularly well-suited for dynamic voltage control, optimal energy dispatch, and demand response strategies in smart grids. More-

over, RL algorithms can be integrated with DERMS to autonomously coordinate power generation, storage, and consumption in response to grid fluctuations [23].

One of the most promising applications of machine learning in grid management is predictive maintenance, where ML models analyze real-time sensor data to detect early signs of equipment failures. By identifying potential faults before they occur, predictive maintenance reduces downtime, lowers operational costs, and enhances the reliability of power grid infrastructure [24]. Furthermore, ML-driven optimization techniques have been applied to energy market operations, where algorithms predict electricity prices, optimize bidding strategies, and balance supply-demand dynamics to ensure grid efficiency [25].

Even with these improvements, challenges remain in deploying machine learning solutions in real-world grid management. The availability and quality of training data, computational complexity, and integration with existing grid infrastructure are key barriers that must be addressed to fully harness the potential of AI-driven grid control. Moreover, trust and safety considerations play a crucial role in AI adoption for critical infrastructure. Ensuring model interpretability, robustness against cyber threats, and compliance with regulatory frameworks are essential for the widespread deployment of AI-based voltage control strategies [26]. Addressing these concerns through transparent AI models, robust validation techniques, and regulatory-compliant safety mechanisms will be critical in gaining industry and stakeholder confidence. However, ongoing research and advancements in machine learning continue to push the boundaries of smart grid technologies, paving the way for more resilient, adaptive, and autonomous power systems [27].

Machine learning's capability to enhance predictive accuracy and operational efficiency paves the way for more advanced discussions on the specific machine learning models, particularly RL, and their applications in subsequent sections of the thesis.

2.5 Reinforcement learning for energy systems

RL is a machine learning approach where an agent interacts with an environment, taking actions based on observed states to maximize a cumulative reward. The agent continuously learns through trial and error, refining its policy over time. The fundamental components of an RL framework include the state space, which represents the environment's current condition, the action space, which defines possible decisions the agent can take, and the reward function, which provides feedback to guide the learning process. The Figure 2.1 illustrates the general framework of RL.

In energy systems, the RL agent typically optimizes control actions such as adjusting power output, regulating voltage levels, or scheduling energy storage usage to enhance grid stability. The iterative learning process enables RL models to adapt dynamically to real-time grid conditions, optimizing power flow and minimizing disruptions [28]. There are several types of RL algorithms, including Q-learning, Deep Q-Networks (DQN), and Proximal Policy Optimization (PPO), each with unique strengths suited to different as-

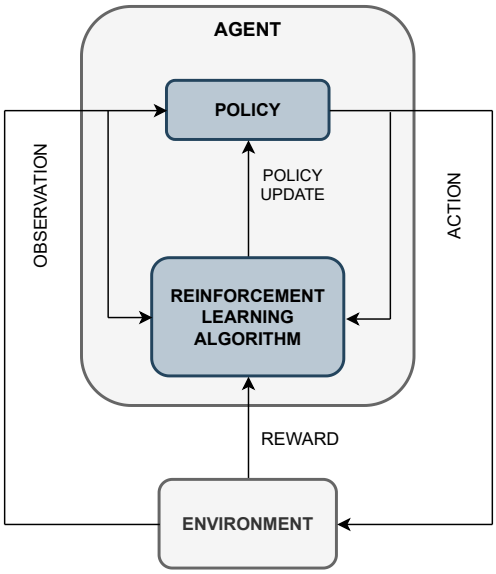

Fig. 2.1: General representation of RL

pects of grid management. For instance, DQN can handle high-dimensional state spaces, making it useful for complex grid environments [29].

One of the primary research directions for RL in energy systems is grid stabilization, where RL-based controllers are being investigated for their ability to adjust voltage regulation settings, manage reactive power compensation, and optimize energy dispatch to ensure stable grid operations. By continuously evaluating system performance and adjusting control parameters, RL has the potential to enable more adaptive and autonomous grid management.

Another area of exploration is load balancing, where RL algorithms are studied for predicting energy demand patterns and optimally allocating power generation resources. In proposed smart grid frameworks, RL-based demand response mechanisms could incentivize consumers to shift their electricity usage based on real-time grid conditions, reducing peak load stress and improving overall system efficiency [30].

Furthermore, RL is being actively researched for energy storage management, focusing on optimizing when and how battery systems charge or discharge to balance supply and demand fluctuations. While simulations and pilot projects have demonstrated promising results, large-scale deployment in real-world grid operations remains an area of ongoing investigation. This approach could significantly enhance the efficiency of renewable energy utilization by mitigating the impact of solar and wind variability.

RL is also being explored for optimal energy trading, where power producers and consumers interact in decentralized energy markets. By learning pricing trends, grid con-

straints, and market dynamics, RL agents can make informed decisions on when to buy, sell, or store electricity, maximizing economic benefits while maintaining grid reliability [23]. In addition, RL techniques have been applied to distributed energy resource (DER) coordination, ensuring that PV systems, batteries, and other distributed generators operate in a coordinated manner to enhance voltage stability and minimize energy losses [10].

Despite its advantages, RL-based grid management faces several challenges, including high computational requirements, long training times, and the need for large-scale simulation environments to effectively train the models. Additionally, the real-world deployment of RL-based controllers requires seamless integration with existing grid infrastructure, regulatory compliance, and robustness against cyber threats. Ongoing research continues to refine RL frameworks, aiming to enhance their scalability and real-time decision-making capabilities for modern energy systems [31]. As RL methodologies advance, their role in optimizing grid stability, energy efficiency, and autonomous decision-making is expected to grow, making them a crucial tool for future smart grid management [32].

Despite its advancements, RL still faces challenges in practical implementation. The next section explores key research gaps and potential areas for further development in RL-based grid management.

2.6 Gaps in current research

Despite substantial advancements in grid management technologies and machine learning applications, there remain several significant gaps in the current research that need addressing to optimize the integration and management of renewable energy sources in smart grids. This section elaborates on these gaps, particularly focusing on the integration of predictive PV data with RL, real-time voltage stability, and the validation of battery management in co-simulation frameworks.

Limited Integration of PV Forecast Data with RL Algorithms: While RL has garnered increasing interest in grid management, a significant gap remains in how short-term PV forecasts are integrated into RL algorithms. Most current research focuses either on RL-based voltage control or on PV forecasting methods in isolation, rather than examining how predictive PV data could enhance RL-based decision-making. Integrating real-time PV forecasts into RL-based controllers has the potential to significantly improve grid stability by enabling proactive rather than reactive control actions. However, existing studies fall short of exploring the full potential of combining these technologies, creating a gap in the development of adaptive and predictive control strategies for power grids [31]. To bridge this gap, further research is required to develop hybrid methodologies that seamlessly integrate forecasting models with RL-based grid management techniques. Such an approach would ensure a more data-driven, anticipatory framework for voltage stability control.

Insufficient Focus on Voltage Stability in Real-Time Scenarios: Although RL has been widely studied for its potential in grid control, its application in real-time voltage stability remains underexplored. Many existing studies focus on long-term optimization and offline learning, but few evaluate how RL performs in real-time grid operation when faced with sudden fluctuations in load and generation [33]. Voltage stability challenges, such as tolerance band crossing events and rapid voltage variations, require immediate corrective actions, yet the effectiveness of RL-based approaches in addressing these real-time events is still not well-documented. Traditional control methods, such as PID controllers and centralized voltage regulation strategies, have been extensively validated in real-world applications, whereas RL-based voltage control lacks empirical validation in practical, real-time settings [33]. To close this research gap, future studies must assess how RL algorithms adapt to real-time constraints, ensuring that control decisions are not only accurate but also computationally feasible for real-world deployment.

Lack of Simulation Validation for Battery Management Optimization: While battery storage optimization plays a crucial role in stabilizing power grids with high PV penetration, there is limited research on the use of co-simulation frameworks to validate RL-based battery management strategies. Many existing studies explore theoretical models of battery storage control, but few validate these approaches through realistic simulations that integrate battery operation, RL-based decision-making, and dynamic grid conditions [34]. A significant challenge in battery management optimization is balancing short-term energy demand with long-term storage efficiency, yet most research does not address how RL policies are tested in co-simulated environments before deployment. Without proper validation, RL-based battery management models may lack reliability, making their real-world implementation difficult. To overcome this issue, future studies should focus on developing comprehensive co-simulation platforms that allow realistic testing of RL-based battery optimization strategies, ensuring they can effectively regulate energy storage, discharge cycles, and grid stability in practical scenarios.

These research gaps highlight critical areas requiring further exploration to enhance RL-based grid control, voltage stability, and battery management optimization. Addressing these issues will contribute to more robust, data-driven power management strategies, improving renewable energy integration and real-time grid stability in the future.

2.7 Conclusions and future directions

The literature review has provided a comprehensive overview of current advances and ongoing challenges in integrating renewable energy sources into power grids, with a particular focus on the potential of machine learning, especially RL, to enhance grid management. This section concludes the literature review by summarizing key insights and outlining future directions that can further advance this field, directly informing the research methodology and experiments that will be detailed in the subsequent chapters of this thesis.

Significant technological advancements have been made in both hardware and software domains. Hardware improvements include energy storage solutions and advanced inverters, while software innovations involve DERMS and sophisticated monitoring tools. These developments contribute to greater flexibility and stability in grid operations, enabling better integration of renewable energy sources. Machine learning, particularly RL, has emerged as a transformative approach for real-time, adaptive grid management. RL's ability to optimize decisions in complex and dynamic environments makes it especially suitable for grids with high renewable energy variability. However, despite these improvements, several critical research gaps remain. The literature identifies three key issues: limited integration of short-term PV forecast data with RL algorithms, insufficient focus on real-time voltage stability, and a lack of comprehensive validation for RL strategies in managing battery storage and balancing grid demands.

Addressing these gaps presents clear future research opportunities. First, integrating real-time PV forecasts with RL-based grid management strategies could significantly enhance predictive accuracy and operational efficiency. By incorporating short-term forecasting, RL-based controllers can make proactive adjustments, reducing the impact of renewable energy fluctuations on grid stability. Second, there is a pressing need to empirically test and evaluate RL algorithms in real-world grid conditions. Current research largely focuses on offline simulations, while real-world implementation requires robust testing environments and pilot projects that assess RL's capability to maintain voltage stability and adapt to dynamic grid states. Additionally, regulatory and economic considerations play a crucial role in RL deployment. Future research should explore how policy frameworks can be adapted to support RL-based grid management and evaluate the economic feasibility of RL-driven automation in utility operations.

Another crucial research direction is the enhancement of grid security and resilience. As smart grids increasingly depend on digital technologies and automated decision-making, cybersecurity threats and system vulnerabilities become critical concerns. Future studies should focus on developing secure RL-based control mechanisms that protect against cyber-attacks and operational failures, ensuring reliable grid management.

The insights gained from this literature review are pivotal for both academic research and practical applications. They lay the foundation for the experimental work detailed in the following chapters, aiming to address the identified gaps through innovative RL applications. Moreover, this review underscores the importance of an interdisciplinary approach, combining engineering, data science, and policy analysis to tackle the complex challenges associated with renewable energy integration into smart grids.

In summary, this section not only concludes the literature review but also sets the stage for the methodological and experimental chapters that follow. The upcoming research will build upon these findings, leveraging the potential of RL to advance state-of-the-art grid management solutions, ultimately enhancing the stability, efficiency, and sustainability of power systems with high renewable energy integration.

3 Methodology

According to the first chapter, the objective of this work was to develop a RL approach as a decentralized controller for a PV-battery system, utilizing short-term PV forecasts to enhance grid voltage stability. The RL controller is designed to make proactive decisions to optimize battery usage and maintain grid stability.

A key component of this research is the integration into a co-simulation (cosim) framework, enabling seamless switching between software (SW) and hardware (HW) environments and facilitating future Eye2Sky (see Section 3.3 for details) forecast data integration. This setup ensures compatibility across platforms and bridges theoretical simulations with real-world deployment.

To illustrate the system architecture, a conceptual diagram is provided in Figure 3.1, showcasing the RL-based PV-battery controller, the role of PV forecasts, and the MATLAB-Python co-simulation interaction.

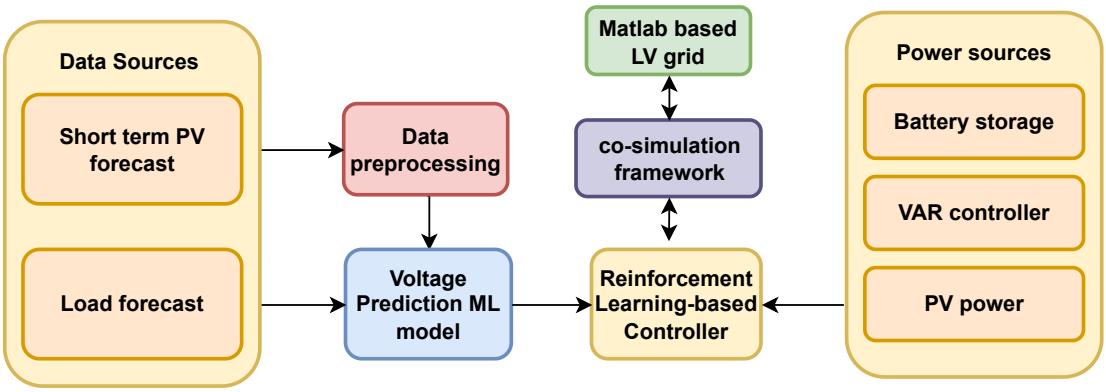

Fig. 3.1: Conceptual diagram of system

The methodology follows these key steps:

1. Grid modelling and simulation: A PV-integrated grid model is developed and simulated under varying conditions.

2. Data collection and preprocessing: PV forecast data from Eye2Sky is gathered and cleaned.
3. Developing the forecasting and labeling model: This model predicts short-term voltage fluctuations and labels stability states.
4. Implementing the RL framework: State-space, action-space, and reward functions are defined, and the RL agent is trained using algorithms like PPO and DDPG.
5. Integration into the co-simulation framework: Real-time MATLAB-Python interaction is established via Mosaik to validate RL-based control performance in a dynamic grid environment.

Each section in this chapter expands on these steps in detail.

3.1 Scope, boundaries, and limitations

This study focuses on developing a RL-based control system to enhance grid voltage stability using short-term PV forecasts. The methodology includes designing a voltage prediction model and integrating it within a co-simulation framework that connects MATLAB and Python. While the proposed approach aims to improve power grid stability, several constraints define the scope and applicability of the research.

Technologically, the study assumes the availability of advanced metering infrastructure (AMI) and sufficient communication capabilities to enable real-time data transfer and decision-making. The proposed RL controller is tested within a co-simulation framework, meaning it does not extend to real-world hardware implementation or field testing. The findings are intended for application at the distribution network level, where voltage stability issues are more prominent due to the increased integration of distributed energy resources. Transmission-level voltage stability concerns are beyond the scope of this research.

One of the primary limitations of this study lies in its reliance on short-term PV forecasts for optimizing power flow and voltage regulation. While advanced forecasting techniques improve accuracy, errors in PV predictions or sudden atmospheric changes can introduce uncertainty in the RL model's decision-making process. Additionally, the RL approach learns optimal actions based on simulated grid behavior, but in real-world implementations, unexpected constraints or external factors could affect its adaptability. Computational constraints and synchronization delays in the MATLAB-Python co-simulation setup may also introduce slight response lags, impacting the real-time efficiency of voltage regulation.

The study further assumes that battery storage systems are available and can be used to support voltage regulation. However, real-world battery degradation, efficiency losses, and operational constraints are not explicitly modelled. The research also does not incorporate dynamic regulatory changes, such as evolving grid codes or policy restrictions, which

could influence the deployment of RL-based control strategies. These boundaries and limitations define the operational context of the study, ensuring a focused exploration of RL for voltage stability while acknowledging areas that may require further investigation in future work.

Research assumptions

To develop a structured and feasible methodology, several key assumptions are made regarding the availability of data, grid stability, and forecasting accuracy. First, the study assumes that all required data, including PV generation forecasts, voltage measurements, and grid parameters, are available in real time, consistently accurate, and free from transmission delays or significant errors. The simulations operate under the assumption that the grid remains structurally stable, without experiencing major failures such as transformer malfunctions or large-scale blackouts, which would require emergency interventions beyond voltage regulation.

While regulatory and operational frameworks influence power grid management, this study does not explicitly incorporate regulatory constraints into the RL development. Instead, regulatory considerations are treated as a boundary condition rather than a key assumption. The study assumes that the overall operational environment remains stable throughout the study period, allowing for a consistent evaluation of the proposed control strategy.

A critical assumption in the voltage prediction model is that it receives perfect load forecasts as input. This simplification removes uncertainties related to load variations, ensuring that the focus remains on the impact of PV generation and RL-based decision-making. In reality, load forecasting errors could affect voltage stability, but for this study, the assumption of a perfect forecast allows for a controlled assessment of the model's effectiveness. Furthermore, the nominal voltage reference is calculated based on the mean voltage from the supply terminal, ensuring consistency in evaluating voltage deviations.

The study is also constrained to short-term voltage prediction, with a forecasting horizon limited to minutes. Long-term variations in solar generation, seasonal effects, and battery degradation are not considered within this time-frame. This assumption aligns with the research objective of developing a real-time adaptive control strategy rather than focusing on long-term grid planning. These assumptions provide a necessary framework for structuring the study and ensuring that the methodology remains manageable within the constraints of simulation-based validation.

3.2 Grid modelling and simulation

The grid model used in this study is a digital representation of a test grid implemented in the DLR_NESTEC lab [35], designed to analyze grid behavior and RL-based control strategies in a simulated environment before transitioning to hardware implementation.

The model consists of three load terminals and one supply terminal, with each terminal featuring a solar power generation feeding point. Corrective power is fed through the feeding point at every load terminal, allowing the system to be tested under various voltage stability conditions. The entire grid is implemented in MATLAB/Simulink (version R2020b), where voltage levels, power flow, and system behavior are evaluated under different operational scenarios.

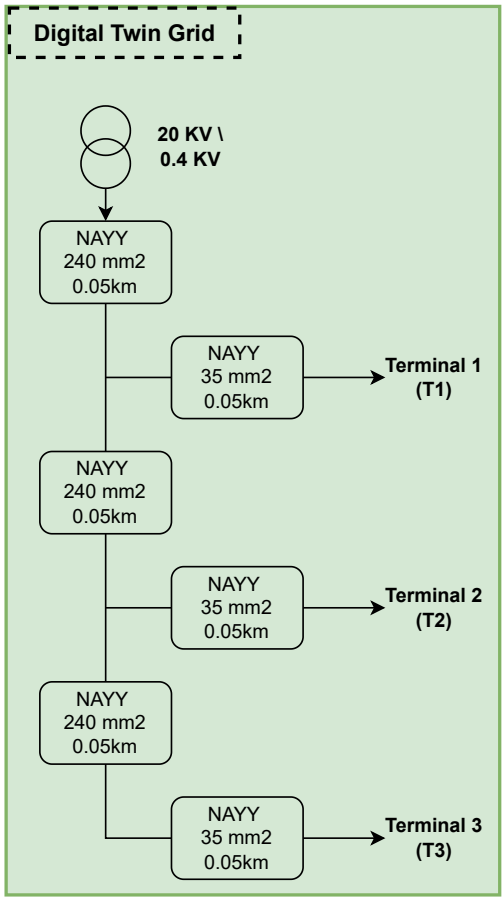

Fig. 3.2: Single line diagram of DLR_NESTEC lab's test grid

The simulation setup includes essential components such as load and supply terminals, each equipped with solar power feeding points, ensuring a distribution grid with decentralized renewable energy generation. The voltage stability mechanisms integrated into the model allow for the observation of deviations and the application of corrective strategies such as reactive power compensation. The single line diagram of the grid, as shown in Figure 3.2, provides a visual representation of the system components and power flow.

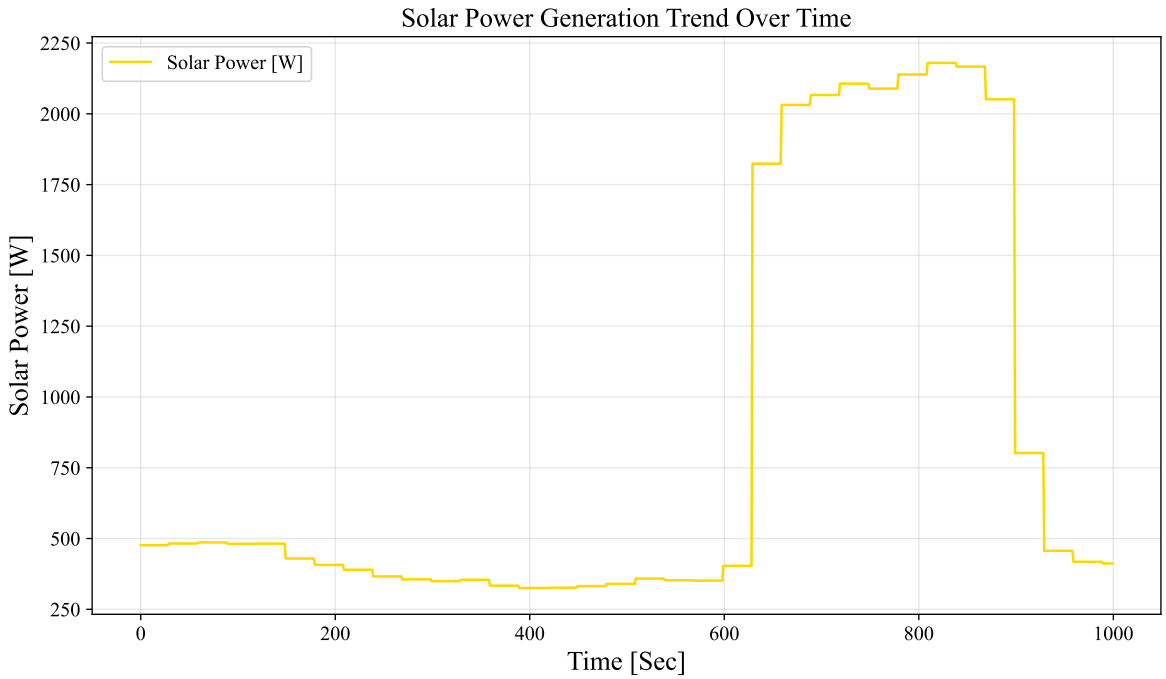

Fig. 3.3: Solar power treand over time

To extract grid behavior under various conditions and generate datasets for training the voltage forecasting model, multiple test scenarios are simulated. These include full load conditions with solar power feeding at all terminals, where peak energy demand is simulated, as well as no load conditions with no solar power, which provide a baseline for grid behavior. Additional scenarios explore no load with solar power feeding at all terminals, evaluating the impact of energy injection without active demand. Other cases include two terminals loaded with solar power while the third remains at no load, assessing partial load-sharing effects, and one terminal loaded with solar power while the others remain at no load, studying voltage stability under isolated load conditions. The solar power trend graph (Figure 3.3) illustrates how solar power generation varies over time in different scenarios, while the voltage trend plots (Figure 3.4) highlight voltage deviations in a scenario where partial load is present at some terminals.

The grid modelling and simulation phase serves as the foundation for the next steps in this research. The simulated test scenarios generate essential datasets required for data collection and preprocessing, which is described in the following section.

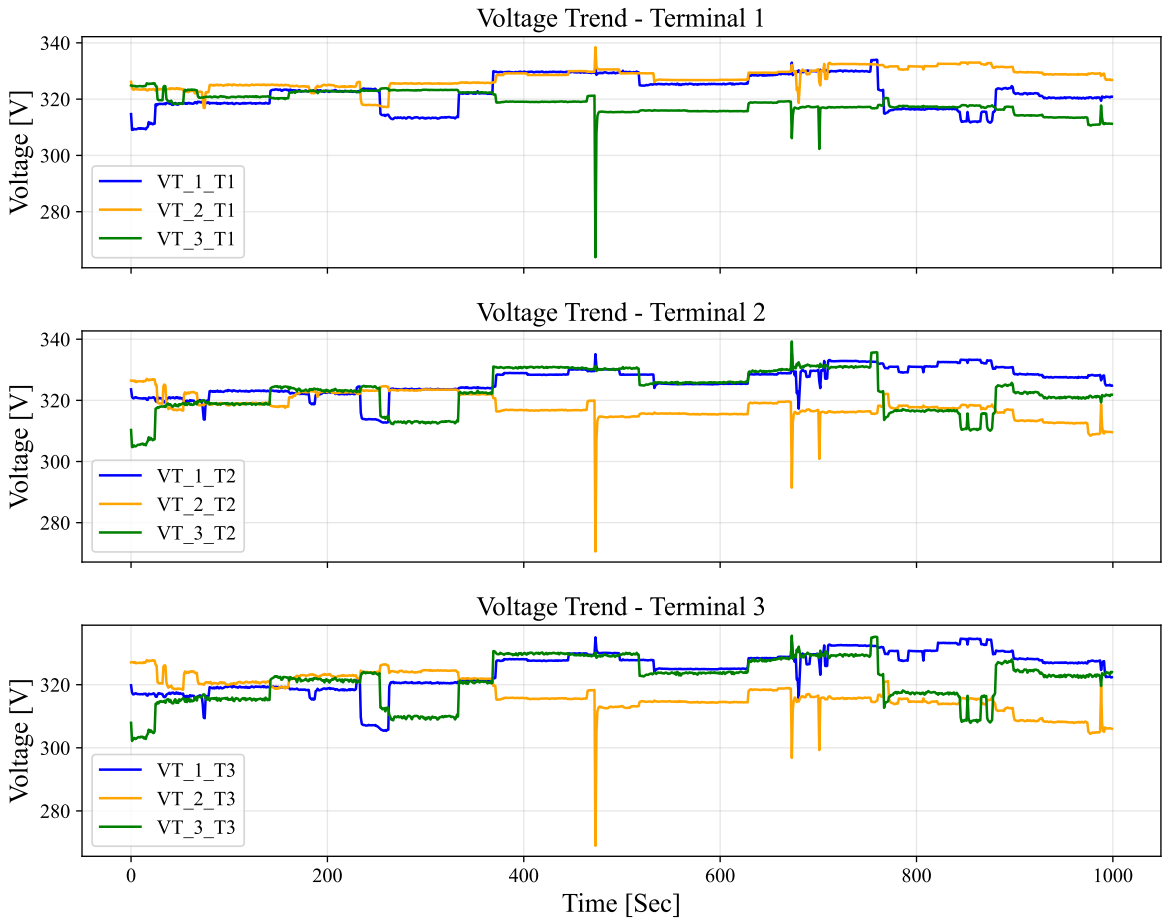

Fig. 3.4: Voltage trend over time

3.3 Data collection

In order to develop both the voltage forecasting and control algorithms, the primary data source for this research is the DLR Eye2Sky cloud camera system [36], which provides high-resolution sky images to estimate short-term solar irradiance. The Eye2Sky system enables real-time irradiance forecasting by analyzing cloud movement patterns, making it an essential component for predicting PV power fluctuations. This solar irradiance data forms the basis for the development of the voltage forecast model, ensuring that the RL-based controller can make informed decisions based on expected PV generation.

The current data collection method involves manual retrieval and local storage of historical data from the Eye2Sky department. This approach serves as the foundation for preliminary analyses and model development. However, to improve data accessibility and real-time integration, a future data collection method is planned. This will involve setting up an automated data transfer via a co-simulation platform between the Eye2Sky system

and the DLR_NESTEC lab. This enhancement will ensure real-time data availability, facilitating seamless integration into ongoing grid management simulations. Additionally, as part of this research, code has been developed to automate the data transfer process, streamlining the integration of real-time data into the co-simulation environment.

In addition to irradiance data, household load profiles were collected to represent real-world energy demand variations. These profiles are sourced from [37], providing realistic consumption patterns that influence grid voltage stability. This dataset ensures that the model accounts for fluctuations in household power demand, improving the robustness of the RL-based control system. Furthermore, these load profiles enable testing of the control performance under realistic operating conditions, ensuring the RL model adapts effectively to dynamic energy consumption patterns.

Furthermore, grid voltage data from the simulated environment was gathered to analyze system responses under different test scenarios. This data captures how voltage levels vary across different load conditions and is used to train and validate the forecasting model. Together, these datasets form the foundation for accurate voltage prediction and effective RL-based control implementation.

3.4 Data preprocessing

Data preprocessing is a crucial step in ensuring the quality and reliability of the dataset before using it for predictive modelling and RL applications. Since the collected dataset from the Eye2Sky system [36] contains missing values, outliers, and inconsistencies, a systematic preprocessing pipeline is applied to clean and transform the raw data into a structured format. This step ensures that the dataset is free of anomalies, making it suitable for training machine learning models and conducting accurate simulations.

The dataset used in this research primarily consists of solar irradiance values measured at a high temporal resolution of 30 seconds, which are essential for predicting the voltage profile of the grid and optimizing RL-based power management strategies. The data is utilized to generate short-term predictions of 30 minutes ahead with a resolution of 1 minute. However, due to environmental conditions, sensor errors, or data transmission failures, the dataset includes missing values and extreme outliers that can affect the accuracy of the predictive models. To address these issues, the preprocessing pipeline follows a structured approach consisting of handling missing values, detecting and treating outliers, and converting solar irradiance into power output to ensure the dataset is prepared for reliable forecasting and control.

A significant challenge in handling real-world solar irradiance data is the presence of missing or NaN (Not a Number) values, which can arise due to sensor failures, data transmission delays, or periods of complete cloud cover that obscure sunlight measurements. Identifying and correcting these missing values is crucial for maintaining the continuity of the dataset and ensuring that time-series forecasting models function correctly. In this research, all missing and NaN values are replaced with zero, a common practice in so-

lar energy forecasting where missing values typically represent zero solar exposure (such as during nighttime or heavy cloud cover). This approach maintains dataset continuity without introducing artificial gaps, allowing the forecasting model to learn from realistic solar generation patterns while ensuring robustness in voltage prediction.

Beyond missing values, outlier detection and treatment are essential for maintaining data integrity. Outliers in the dataset can emerge due to sudden spikes or drops in solar irradiance caused by atmospheric disturbances, sensor malfunctions, or incorrect readings. These anomalies can significantly distort the performance of predictive models, making it necessary to identify and correct them before model training. To detect outliers, a time-series decomposition method [38]. is applied, which separates the dataset into trend, seasonal, and residual components. The trend component captures long-term variations in solar irradiance, the seasonal component accounts for periodic fluctuations such as day-night cycles, and the residual component isolates random fluctuations and anomalies, which are considered potential outliers. By analyzing the residual component, extreme deviations from expected values are flagged as outliers.

Once outliers are detected, they are replaced with averaged adjacent values to preserve the natural trends of the time series. Specifically, each detected outlier is replaced with the average of the two nearest valid data points before and after it, ensuring that the correction is based on realistic values. To further validate the replacement, a Z-score method is applied, which measures how many standard deviations a data point deviates from the mean. Any corrected value with a Z-score greater than 3 (indicating an extreme deviation) is further refined using additional smoothing techniques. This combined approach — leveraging time-series decomposition and statistical validation — ensures that the dataset remains both accurate and representative while minimizing the impact of extreme fluctuations [39].

Following the handling of missing values and outliers, the cleaned solar irradiance data is converted into power output for use in voltage stability analysis and RL-based control. The conversion follows the standard PV power output equation [40], which relates irradiance levels to electrical power generation. This step is critical for integrating the dataset into the grid modelling framework, as the RL controller requires power values rather than raw irradiance data to make informed voltage regulation decisions. The conversion formula used in this thesis is as follows:

$$P_{PV}\,[kW] = I_{POA}\left[\frac{kW}{m^2}\right] \times A_{PV}[m^2] \times \frac{\eta_{mod}[\%]}{100} \times f_{Loss} \tag{3.1}$$

with:

- $I_{POA}\left[\frac{kW}{m^2}\right]$, irradiance on the module's POA,
- $A_{PV}[m^2]$, the PV array module area,

- $\eta_{mod}[\%]$, the efficiency of the PV modules,
- f_{Loss}, a factor accounting for additional deviations (e.g., deviations due to electrical, optical, or thermal losses).

These preprocessing steps ensure that the dataset is free from inconsistencies, allowing it to be used effectively in machine learning-based voltage forecasting models. The overall data preprocessing workflow is illustrated in Figure 3.5, which summarizes the key steps applied to clean, transform, and prepare the dataset for predictive modelling and RL-based voltage control.

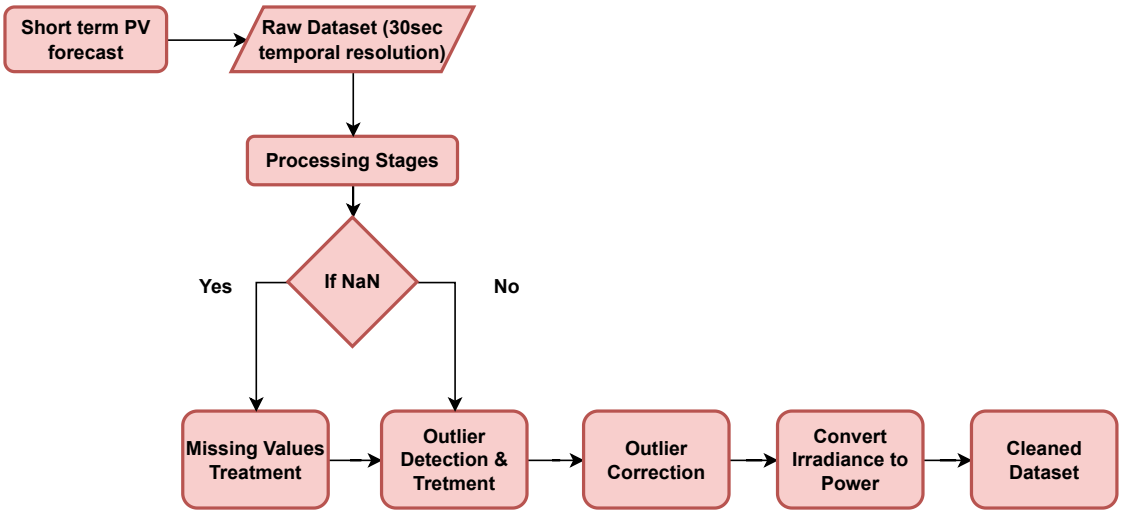

Fig. 3.5: Data preprocessing pipeline

3.5 Development of forecasting models

The development of the forecasting model is a critical step in this research, as accurate voltage profile predictions provide additional insights for the RL-based control mechanism. While RL can function without explicit forecasts, incorporating predictive models enhances the understanding of the RL decision-making process and potentially improves performance by providing advance information about expected system behavior. The forecasting model is designed to predict short-term voltage fluctuations based on anticipated solar power generation and load demand variations at different terminals. By leveraging machine learning techniques, specifically the Random Forest algorithm, this model supports the RL-based control strategy by enabling more proactive decision-making for grid voltage stability. By leveraging machine learning, specifically the Random Forest algorithm, this model ensures that the RL-based control strategy can take proactive decisions to enhance grid voltage stability.

In power systems, voltage stability is significantly influenced by the fluctuations in solar power generation and dynamic load variations. An accurate forecasting model allows the RL-based controller to anticipate future voltage deviations and take corrective actions in advance.

This research develops a data-driven voltage forecasting model, which:

- Predicts voltage profiles at three load terminals across all three phases.
- Uses PV power forecasts from Eye2Sky and historical voltage and power data as input.
- Provides real-time predictions to inform RL-based control decisions.

The forecasting model is not a standalone system, but rather an integral component of the overall control strategy, feeding predicted voltage profiles into the RL control mechanism. This next section presents the structured approach followed in developing the forecasting model, including dataset preparation, model selection, training, hyperparameter tuning, evaluation, and its integration into the RL-based control system. The general structure of the voltage forecasting model is illustrated in Figure 3.6, showing how input features, model predictions, and RL integration are connected within the control system.

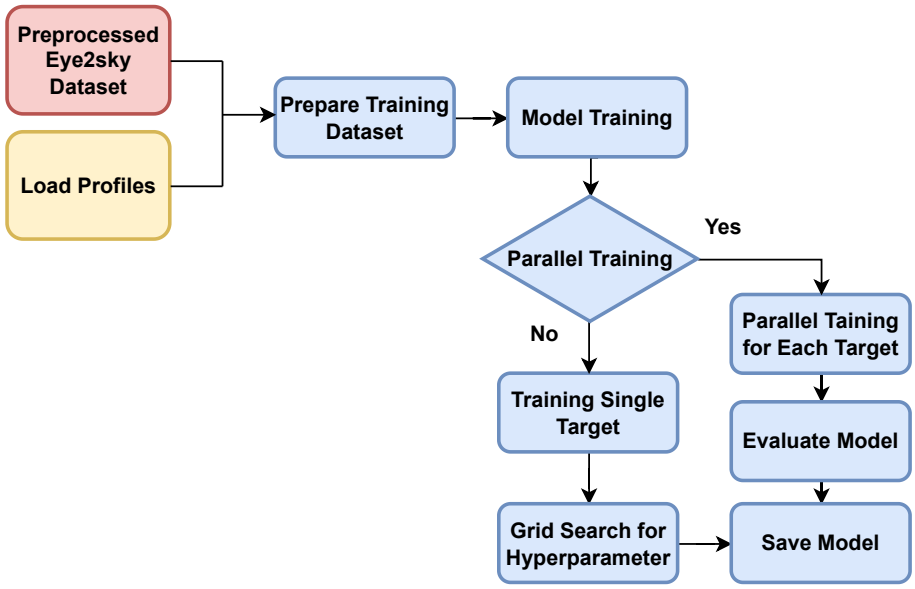

Fig. 3.6: Voltage prediction model pipeline

3.5.1 Dataset preparation for training the forecasting model

To develop an accurate voltage forecasting model, a structured dataset was generated by running multiple test scenarios on the simulated grid model. These scenarios covered a

diverse range of operating conditions, ensuring that the model could learn to predict voltage fluctuations under various system states. The training dataset was carefully curated to include relevant power system parameters, allowing the forecasting model to generalize effectively to unseen data.

The dataset consists of multiple input features (independent variables) and corresponding target outputs (dependent variables). The input features include power values at all three load terminals, specifically active and reactive power values for P_{L1}, Q_{L1}, P_{L2}, Q_{L2}, P_{L3}, and Q_{L3}, as well as power values at all three solar terminals, represented by P_{S1}, Q_{S1}, P_{S2}, Q_{S2}, P_{S3}, and Q_{S3}. These features capture the key factors influencing voltage fluctuations within the system.

The target variables in the dataset are the voltage values at each load terminal across all three phases, denoted as V_{T1}, V_{T2}, and V_{T3}. These voltage values are measured in volts (V) with a time resolution of one second, ensuring that the forecasting model captures short-term voltage variations. The values represent RMS (Root Mean Square) voltage measurements, which provide a reliable indication of the voltage profile within the grid. Each data entry represents a single time step, where the forecasting model is trained to predict future voltage profiles based on historical and real-time input features. The structured nature of the dataset allows the model to recognize patterns in power generation, load variations, and grid dynamics, improving its predictive accuracy.

By leveraging this well-structured dataset, the voltage forecasting model is designed to support real-time grid stability management. The predicted voltage profiles serve as a critical input for the RL-based control system, enabling proactive decision-making and improved voltage regulation across the network.

3.5.2 Model selection and training

The first approach for forecasting voltage profiles was based on neural network architectures, including Multilayer Perceptron (MLP) and Recurrent Neural Networks (RNNs). These models were initially selected due to their ability to capture complex temporal patterns in voltage variations. However, the neural network models encountered several challenges, leading to suboptimal performance. The primary issues included high sensitivity to hyperparameters, which made training difficult and required extensive tuning. Additionally, the models exhibited overfitting, meaning they performed well on training data but failed to generalize effectively to unseen scenarios. Furthermore, they demonstrated poor adaptability to different grid conditions, making them unreliable for practical voltage forecasting. Due to these limitations, an alternative machine learning model was explored to achieve better accuracy and robustness.

To address these limitations, the Random Forest algorithm was selected as the preferred forecasting model. Random Forest (RF) is an ensemble learning method that constructs multiple decision trees and averages their predictions to improve accuracy and reduce overfitting. Unlike deep learning models, which require extensive computational resources and careful hyperparameter tuning, Random Forest is computationally efficient, robust,

and interpretable. It provides insights into feature importance, making it easier to analyze the key factors affecting voltage stability. By training the model on a diverse dataset that includes various grid operating conditions, the forecasting model achieves greater generalizability and reliability across different system states. A schematic of the Random Forest structure used in this research is illustrated in Figure 3.7, showing how multiple decision trees contribute to the final voltage prediction.

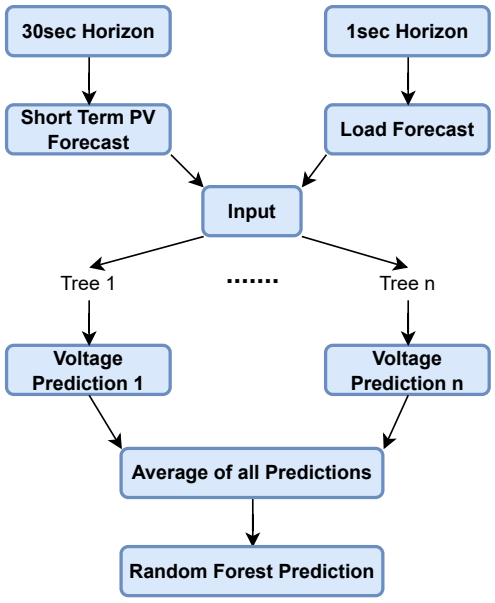

Fig. 3.7: Random Forest structure showing multiple decision trees used for voltage forecasting

To optimize the performance of the Random Forest model, a Grid Search approach was used to systematically explore different hyperparameter configurations. The key hyperparameters fine-tuned included:

- **Number of trees** (`n_estimators`): Determines the number of decision trees in the ensemble.
- **Maximum depth** (`max_depth`): Restricts tree depth to prevent overfitting while capturing sufficient detail in voltage trends.
- **Minimum samples per split** (`min_samples_split`): Controls how many samples are required to split a node, impacting model complexity.
- **Minimum samples per leaf** (`min_samples_leaf`): Ensures that no overly small branches are created, reducing the risk of overfitting.

After extensive experimentation, the best hyperparameter combination was selected, ensuring the highest forecasting accuracy while maintaining computational efficiency. The final Random Forest model was then trained and validated to ensure its suitability for real-time voltage forecasting.

3.5.3 Model training and evaluation

The effectiveness of a forecasting model is determined by how accurately it can predict future voltage profiles under various grid conditions. To ensure that the Random Forest models generalize well, a structured training and evaluation process was implemented. This section details the training process, data splitting techniques, hyperparameter tuning, and the evaluation metrics used to assess model performance.

3.5.3.1 Training process

The training phase of the forecasting models involved the following structured steps to ensure optimal accuracy and generalization across various grid conditions.

Step 1: Data Splitting: To ensure that the models could effectively learn from past data while being evaluated on unseen scenarios, the dataset was split into training and testing sets using an 80/20 ratio. This meant that 80% of the dataset was allocated for training, allowing the models to learn voltage patterns, while the remaining 20% was reserved for testing, enabling an unbiased assessment of the model's ability to generalize. To maintain the integrity of time-series forecasting, the temporal order of the data was strictly preserved, ensuring that historical data was exclusively used to predict future voltage values. This precaution was necessary to prevent data leakage and ensure that the model was learning real-world patterns rather than artificially induced correlations.

Step 2: Feature Engineering: Once the dataset was split, the next step was to select and refine the input features for training the model. The dataset contained multiple features, including power values from all load terminals (active and reactive power), power values from all solar terminals (active and reactive power), and historical voltage data. The target output variables were the future voltage values (V_{T1}, V_{T2}, V_{T3}) for all three phases at each load terminal. This study assumes that real-time data from all nodes is available via smart meters, enabling comprehensive feature selection and accurate forecasting. In scenarios where smart meter data is missing or delayed, alternative imputation techniques or predictive estimations could be required to maintain model reliability. Feature selection was automated using the feature importance rankings from the Random Forest algorithm, ensuring that only the most influential predictors were retained. This step helped reduce computational complexity, improve interpretability, and enhance prediction accuracy by eliminating irrelevant or redundant features that could introduce noise into the model.

Step 3: Model Training: With the structured dataset prepared, the Random Forest models were trained separately for each of the three load terminals and across all three phases, resulting in a total of nine models. This terminal-phase-specific approach was chosen to account for the individual voltage dynamics observed at each terminal and across different phases. Given that each load terminal experiences unique variations in solar power generation, demand fluctuations, and grid interactions, a single model for all terminals would risk overgeneralization, reducing its predictive accuracy. Similarly, voltage fluctuations differ between phases due to imbalanced loads, asymmetric network configurations, and phase-specific reactive power flow. Training separate models for each phase ensures that phase-specific behaviors are captured accurately, allowing for more precise and localized voltage forecasting. The optimized hyperparameters, obtained through grid search, were applied to ensure that each model was tuned for maximum accuracy. This step involved adjusting parameters such as the number of trees (`n_estimators`), maximum tree depth (`max_depth`), minimum samples per split (`min_samples_split`), and minimum samples per leaf (`min_samples_leaf`) to balance model complexity and performance. By training separate models for each phase and terminal, the forecasting approach maintains a high level of granularity, capturing localized effects more effectively than a single, generalized model would. This method significantly improves the model's ability to predict short-term voltage fluctuations and enhances its applicability within the RL-based control system.

Step 4: Avoiding Overfitting: To prevent the models from memorizing patterns instead of generalizing, cross-validation was performed using K-Fold Cross-Validation (K=5). The dataset was divided into five subsets, and each subset was used for validation while the others were used for training. This iterative approach ensured that the models learned from diverse scenarios and could adapt to different grid conditions. Additionally, regularization parameters such as `min_samples_split` and `min_samples_leaf` were fine-tuned to prevent excessive model complexity, ensuring that the trees were neither too deep nor too shallow.

Step 5: Parallel Training Optimization: Given that nine separate Random Forest models were being trained simultaneously, the process was optimized using parallel computing techniques, which significantly reduced the overall training time. Instead of training each model sequentially, multiple models were trained in parallel, making use of multi-core processing capabilities to enhance efficiency. Additionally, memory management strategies were applied to ensure that the large datasets could be handled without excessive computational overhead.

Step 6: Model Validation: After training, the models were tested on the previously reserved 20% test set, ensuring that their performance was evaluated on unseen data. This allowed for an objective assessment of how well the models could generalize to new voltage conditions. Various evaluation metrics were used to measure the accuracy and reliability of the predictions, which are discussed in the next section.

By implementing a well-defined training process, the Random Forest-based forecasting models were systematically optimized to predict future voltage profiles with high accuracy while ensuring robust generalization across diverse grid conditions.

3.5.4 Integration with RL-based control

The forecasting model serves as an integral component of the RL-based control strategy, allowing the RL controller to anticipate voltage fluctuations and take proactive measures instead of relying solely on real-time observations. This section details the workflow of integration, data exchange between the forecasting model and RL agent, implementation strategies, and challenges encountered in embedding the forecasting model within the RL control system.

3.5.4.1 Role of the forecasting model in RL control

The forecasting model plays a critical role in the RL-based voltage control system, providing future voltage profile predictions that enhance decision-making capabilities. Instead of relying solely on real-time voltage observations, the RL controller leverages these predictions to anticipate voltage fluctuations before they occur, enabling proactive grid management rather than reactive responses.

The forecasting model utilizes predicted solar power availability from Eye2Sky forecasts and predicted load demand at each terminal as input data. By processing these key influencing factors, it generates forecasts of voltage variations across different phases and terminals. These voltage predictions are then incorporated into the RL controller's state space, allowing the agent to analyze and account for future grid conditions when selecting control actions.

By integrating voltage forecasting into the RL-based control framework, the system gains the ability to adjust power management strategies before voltage deviations exceed predefined stability limits. This results in a more stable and efficient grid operation, as the controller can preemptively adjust solar inverters, manage power flows, and optimize battery storage to mitigate fluctuations. The combination of data-driven forecasting and RL enhances the system's ability to maintain grid stability and voltage regulation under varying operating conditions.

3.5.4.2 Workflow of integration

The integration of the Random Forest-based forecasting model into the RL framework follows a structured, data-driven approach that enables seamless interaction between the two components. The workflow can be divided into three key stages: input data preparation, voltage forecasting, and RL-based decision-making.

Stage 1: Input Data Preparation: At each control step, the RL controller requires accurate predictions of future voltage profiles to optimize its decision-making process. To

achieve this, the forecasting model receives structured input data, including predicted solar power availability (from Eye2Sky forecasts), and predicted load demand at each terminal. These inputs represent the key factors influencing voltage fluctuations in the grid. The data is then formatted into structured feature vectors, ensuring that it can be efficiently processed by the trained Random Forest model. The structured nature of the dataset allows the forecasting model to analyze patterns in power generation and demand dynamics, leading to more accurate voltage predictions over the forecast horizon.

Stage 2: Voltage Forecasting: Once the input data is prepared, it is fed into the trained Random Forest model, which generates future voltage predictions for each phase at every load terminal. Since voltage control decisions must account for phase-specific variations, nine separate forecasting models—one for each phase at each terminal—are executed simultaneously. The models process the input features and output the predicted voltage profiles (V_{T1}, V_{T2}, V_{T3}) for all three load terminals over the next prediction horizon at one-second intervals. These voltage forecasts provide critical insight into potential fluctuations, allowing the RL controller to anticipate voltage deviations before they occur. The forecasting model's ability to generate real-time predictions ensures that the RL-based control system can proactively regulate voltage, rather than relying solely on immediate, real-time observations.

Stage 3: RL-Based Decision Making: With the predicted voltage profiles available, the RL agent integrates these forecasts into its decision-making process. The forecasted voltage values are incorporated into the state space of the RL environment, enabling the agent to analyze future grid conditions instead of merely reacting to present values. The policy evaluation step follows, where the RL agent evaluates multiple possible control actions and assesses their expected impact on future voltage stability. Using the learned RL policy, the agent selects the optimal action that minimizes voltage deviations while optimizing power flow and battery usage.

After selecting the best action, the computed control signals are applied to the MATLAB-based grid model. These control actions include adjusting solar inverter setpoints to regulate voltage fluctuations, redistributing power flows to balance supply and demand, and activating battery storage or other corrective measures to mitigate voltage deviations. By incorporating forecasted voltage values into the RL control strategy, the integration significantly enhances the agent's ability to preemptively adjust power flow, improving grid stability and operational efficiency.

Through this coordinated workflow, the voltage forecasting model and RL-based control mechanism work together to ensure that data-driven predictions support intelligent grid management decisions. By leveraging forecasted voltage fluctuations, the RL controller can apply preventive strategies, reducing reliance on delayed reactive corrections and thereby improving voltage stability in PV-integrated grids.

3.6 Co-simulation framework setup

To enable real-time interaction between the MATLAB-based grid model and the Python-based RL controller, a Mosaik-based co-simulation framework was implemented. This setup allows seamless data exchange between both environments, enabling the RL agent to receive real-time grid state observations from MATLAB and send control actions back for execution. The integration of Mosaik provides a modular and adaptable co-simulation environment, supporting the study of RL-based voltage control strategies under realistic grid operating conditions.

This section presents a detailed explanation of the co-simulation setup, covering its architecture, data exchange mechanisms, synchronization strategies, and existing challenges.

3.6.1 Overview of co-simulation

The implementation of a Mosaik-based co-simulation framework is a fundamental requirement in this research, serving as the bridge between MATLAB (for power system simulation) and Python (for RL-based control). This integration is necessary due to the complexity of grid dynamics and the need for real-time adaptive decision-making, which requires both high-fidelity grid simulations and advanced machine learning models. MATLAB's Simulink environment is well-suited for modelling detailed power system behaviors, including voltage stability, power flows, and transient responses. However, since RL requires a robust AI framework, Python is used for its extensive machine learning libraries, such as TensorFlow, PyTorch, and Stable-Baselines [41]. The co-simulation framework ensures that the RL agent can interact with the simulated grid environment, observe system conditions, and apply adaptive voltage control strategies in real time.

The need for such an integration arises from several critical factors. First, modern power grids are highly dynamic, and their behavior cannot be accurately modelled using predefined rule-based control approaches. Instead, RL offers the ability to continuously learn and adapt to grid conditions, requiring a real-time interaction loop between MATLAB and Python. Additionally, grid simulations must operate under real-world constraints, including voltage fluctuations, load variations, and renewable energy intermittency, necessitating an AI-driven control system that can respond dynamically. The co-simulation framework is also essential for scalability and modularity, allowing future extensions where additional models — such as weather forecasting, battery storage management, and demand-side response strategies — can be integrated seamlessly.

The Mosaik-based co-simulation framework plays a crucial role in validating the RL-based voltage control system under realistic conditions. By enabling direct interaction between the MATLAB-based grid simulation and the Python-based RL controller, the framework supports comprehensive testing before potential real-world deployment. Through real-time system monitoring and adaptive learning-based control, this setup allows the RL controller to interact dynamically with the grid model, ensuring flexibility in decision-

making rather than relying on predefined control rules. Additionally, by providing a modular and extensible architecture, this framework ensures that future improvements can be easily incorporated.

The structure of the co-simulation framework used in this work follows a methodology developed at the DLR Institute of Networked Energy Systems [42], which is further explained in the next section. This approach ensures that the simulation aligns with established research methodologies, facilitating consistent and validated testing of AI-based grid control strategies.

3.6.2 Architecture of the co-simulation environment of this work

The Mosaik-based co-simulation framework is designed with a structured architecture that ensures seamless interaction between MATLAB (for power system simulation) and Python (for RL-based control). The framework consists of three key components: simulators, web services, and protocol converters, as visualized in Figure 3.8. Each of these components plays a crucial role in ensuring smooth data exchange and real-time decision-making within the co-simulation setup.

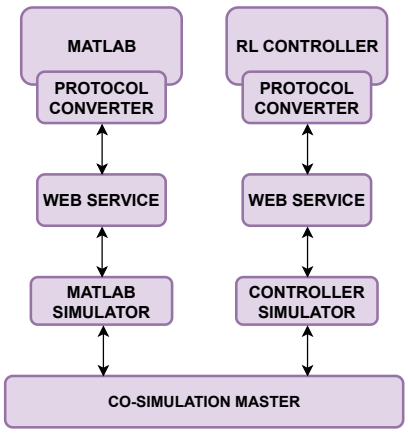

Fig. 3.8: Architecture of co-simulation

Simulators The simulators represent the two core environments used for grid modelling and RL-based voltage control. The MATLAB Grid Simulator is responsible for simulating the three-terminal power system, capturing voltage stability, power flow, and system behavior under different operational conditions. The voltage values at all load terminals and the supply terminal are computed within the Simulink model at each time step, providing essential data for the RL controller to make informed decisions. The MATLAB simulation runs on Simulink, ensuring high-fidelity modelling of power system dynamics. On the other hand, the Python-Based RL Controller processes the voltage data from

MATLAB, using it as an observation for training the RL model. The RL controller continuously learns from the received voltage states and generates optimal control actions to stabilize voltage levels. These control signals are then transmitted back to MATLAB for execution, enabling a closed-loop control mechanism.

Web services The web service in this co-simulation framework serves two primary roles: facilitating communication between MATLAB and the MATLAB-based simulator, and enabling interaction between the RL controller and the controller simulator. The first web service establishes a connection between MATLAB and the grid simulator running in Simulink. It ensures that voltage observations from the simulated power system are continuously transmitted to the RL controller while simultaneously receiving and applying control actions from the RL model in real-time. However, in addition to this, a second web service is used to link the RL controller with the controller simulator. This setup allows the RL agent to compute and send optimal control actions to the controller simulator, which then executes these actions in the power system simulation. To achieve this, the web service implements a REST API-based communication system using HTTP and WebSockets, ensuring a reliable and low-latency connection between the simulators. The communication process is structured in a way that MATLAB continuously sends updated voltage states to the RL agent, which in turn processes the data and determines the most effective control actions. These actions are then transmitted back to MATLAB, where they are applied in the next simulation step. The web services support asynchronous data flow, meaning that both MATLAB and Python can operate independently without causing delays in simulation execution. This asynchronous nature enables efficient processing by allowing MATLAB to continue its power system calculations while the RL controller evaluates optimal control policies in parallel.

Protocol converters The protocol converter in this co-simulation framework is responsible for translating protocol. It provides methods to control the execution of the MATLAB simulation in different operational modes, including continuous execution, fixed-time step execution, or indefinite looping, depending on the simulation requirements.

By integrating these components, the co-simulation framework enables real-time, data-driven interaction between the MATLAB-based power grid simulation and the Python-based RL model.

3.6.3 Synchronization and data exchange

The synchronization between MATLAB-based grid simulation and Python-based RL control system is a critical aspect of the Mosaik-based co-simulation framework. Since the RL agent relies on real-time grid state observations to make informed decisions, maintaining a consistent time step and ensuring reliable data transfer between the simulators is essential.

Mosaik provides two synchronization approaches:

1. Event-based synchronization, where data exchange is triggered by specific simulation events.

2. Time-based synchronization, where all simulators operate at a fixed time step.

For this research, a time-based approach is chosen to ensure predictable update intervals. Instead of relying on event-driven execution, which can introduce variable latencies, a time-based strategy allows Mosaik to manage data exchange at fixed time steps. This ensures that simulators progress in a controlled manner, reducing inconsistencies in simulation timing and facilitating structured interactions between the RL agent and the grid model. By using time steps to drive the simulation, the framework maintains coordination between the MATLAB-based power system model and the Python-based RL controller, ensuring that control actions align with the expected system state at each step. However, while this approach improves predictability, inherent communication delays and computational processing times can still introduce synchronization mismatches, which are considered in the system evaluation.

Time synchronization approach

In this study, time-based synchronization is implemented using fixed time steps within the Mosaik framework to facilitate the smooth execution of data exchange between MATLAB and Python. The simulation process follows a defined workflow where MATLAB Simulink runs the grid model and computes the voltage states at each terminal. At each predefined time step, Mosaik facilitates data transmission through the protocol converter and web service, ensuring that updated grid states reach the controller simulator in Python. Within the Python environment, the grid state is processed, and the RL controller determines the optimal control action to regulate voltage stability. The computed control action is then relayed back through the protocol converter and web service before being applied to the MATLAB grid simulation in the subsequent step.

Ideally, the execution process should maintain consistent data exchange at fixed intervals, enabling the RL controller to make adaptive control decisions while adhering to operational timing constraints. However, synchronization mismatches between MATLAB and Python lead to delays in data transmission and misaligned simulation steps. These timing discrepancies prevent the RL controller from receiving the latest grid state information in a timely manner, impacting its ability to implement control actions effectively. To address these issues, adjustments were made to the Mosaik scheduling mechanism, including refining time-step coordination and optimizing the data exchange process. Despite these efforts, residual timing inconsistencies remain, highlighting the need for further improvements in synchronizing execution timing between both simulators. Further details on the encountered challenges, attempted solutions, and proposed improvements are discussed in Section 3.6.4.

Data transmission process

To enable real-time decision-making, bi-directional data exchange between MATLAB and Python is required. The data transmission process follows a structured approach:

Data sent from MATLAB to Python (State Observations): The RL-based control system relies on real-time grid observations sent from MATLAB to Python through a REST API. These observations include voltage measurements from all three load terminals, capturing the dynamic voltage conditions at different nodes of the grid. Additionally, the system transmits supply voltage readings from the main power source, providing a reference voltage that helps the RL agent understand the overall grid stability. To maintain consistency, the data follows a predefined format.

Data sent from Python to MATLAB (Control Actions): Once the RL agent processes the received grid state and determines the optimal control action, it sends the computed power adjustments back to MATLAB. These control actions include adjusting solar inverter setpoints, regulating active and reactive power flows at each terminal, and implementing corrective measures to stabilize voltage levels. This bidirectional data exchange allows the RL controller to operate within a closed-loop control environment, continuously receiving real-time feedback from MATLAB and dynamically refining its voltage control strategy.

Multithreading & queue-based data handling

Since data exchange between MATLAB and Python occurs at every simulation step, an efficient communication mechanism is essential to prevent delays and ensure real-time execution. A key challenge arose due to the interaction between Mosaik's cyclic execution and the RL learning process in Python. Specifically, calling the *learn()* method during Mosaik's cycle execution caused delays and interrupted the simulation flow. To achieve this, a multithreading and queue-based approach is implemented, allowing the MATLAB simulation to progress without being blocked by the Python-based RL controller. The structure of this queue and multithreading approach is illustrated in Figure 3.9. The RL learning agent receives future voltage predictions, power source states, and current grid voltage from the voltage queue. It computes a raw control action, which is refined through a hierarchical control strategy to ensure operational constraints are met. The final control action is then stored in the power queue and sent to MATLAB via the co-simulation framework, where it is applied to update the grid state. This closed-loop process allows the RL agent to continuously optimize voltage control based on real-time and predictive inputs.

The multithreading approach allows the RL agent to execute its learning process asynchronously, running in parallel without blocking Mosaik's real-time execution cycle. By handling RL training in a separate thread, the simulation can continue processing state updates and control actions without being stalled by the computational overhead of reinforcement learning. This enables MATLAB to proceed with grid simulation without waiting for Python to complete data processing. Meanwhile, the RL agent in Python

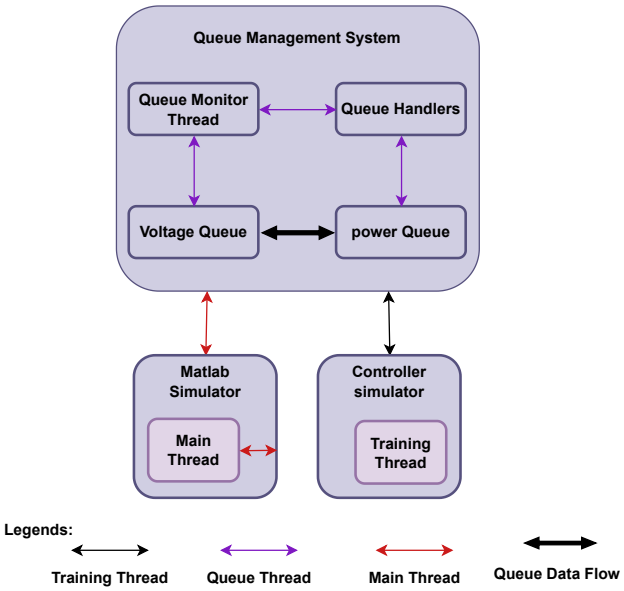

Fig. 3.9: Multithreaded Queue System Architecture

continuously receives grid observations, processes them, and prepares the next control action in parallel. This separation of tasks prevents bottlenecks, ensures uninterrupted simulation flow, and enhances the overall responsiveness of the system.

Additionally, queue-based data handling is implemented to facilitate data exchange between different threads, ensuring smooth communication without blocking execution. Incoming observations from MATLAB are first stored in a queue, allowing the RL agent to retrieve them asynchronously without disrupting the Mosaik simulation cycle. Similarly, computed control actions from Python are buffered before being sent back to MATLAB, ensuring they are processed in the correct order and reducing potential timing inconsistencies.

By implementing this multithreading and queue-based approach, the co-simulation framework enables asynchronous data exchange between MATLAB and Python without blocking the Mosaik execution cycle. The use of separate threads allows the RL agent to process incoming state observations and compute control actions in parallel, preventing execution bottlenecks. Queue-based handling ensures that data is reliably passed between different threads, maintaining the order of state updates and control actions. However, despite these improvements, challenges remain in achieving precise synchronization between MATLAB and Python due to timing mismatches and execution delays. However, further refinements are necessary to completely eliminate delays and ensure perfectly synchronized control actions.

3.6.4 Challenges in synchronization and data transmission

Despite the efficient design, the current setup faces some challenges that impact real-time performance:

1. Time Synchronization Issue: Due to the separate execution of the main simulation thread and the RL thread, occasional time mismatches occur, leading to delays in data processing. This misalignment results in the RL controller sometimes receiving outdated grid observations, which in turn affects its decision-making accuracy, as control actions may be based on previous rather than current grid states.
2. Data Loss Issue: Occasionally, some voltage readings are lost during transmission due to synchronization mismatches between the Python-based RL agent and the MATLAB grid simulation. Additionally, the queue-based data handling mechanism contributes to this issue. This data loss directly impacts the RL controller's ability to accurately respond to voltage fluctuations, as missing or incomplete grid state information can result in suboptimal control actions and reduced voltage stability. Addressing this issue requires improved queue management strategies, such as implementing buffer mechanisms or time-stamped data retrieval, to prevent unintended data loss and ensure a more reliable exchange of grid observations.

These challenges require further optimization, which will be explored in Section 3.7.

3.7 Implementation of RL algorithms

The RL controller is developed to optimize grid voltage stability by adjusting power distribution and solar inverter operations. The RL-based control framework interacts with the Mosaik-based co-simulation, where MATLAB simulates the power grid while Python executes the RL-based decision-making process. The RL algorithms are implemented using Stable-Baselines3 [41], a widely used RL library that provides optimized implementations of various deep RL algorithms. This study primarily employs Proximal Policy Optimization (PPO), with Deep Deterministic Policy Gradient (DDPG) as an alternative approach. This section details the algorithm selection, reward function design, training process, evaluation, and integration with the co-simulation framework, developing a comprehensive and effective RL-based voltage control strategy.

3.7.1 Overview of RL-based control

The RL-based controller is designed to tackle critical grid voltage stability challenges by leveraging data-driven decision-making to optimize power distribution and reactive power compensation. The primary objective is to maintain voltage within the operational tolerance band, preemptively mitigate anticipated voltage instability, and optimize resource allocation by adjusting active and reactive power flows to balance terminal loads effectively. Unlike conventional control methods such as Q(U) control [43] and battery protection mechanisms [44], which operate based on predefined rules, RL introduces an

adaptive learning approach, allowing the system to refine its voltage regulation strategies based on continuous interaction with the grid environment variables for the RL agent include voltage measurements from supply and terminal voltages, predicted voltage profiles generated from forecasting models, load profiles indicating future power demand, solar power generation data, and battery and VAR (Volt-Ampere Reactive) status. These inputs collectively define the state space, providing the RL agent with a comprehensive view of the grid's dynamic behavior.

To achieve effective voltage control, the RL agent requires a well-defined action space that represents the available control options. The action space varies depending on the control approach: in a centralized RL-based control strategy, where a single agent manages the entire grid, the action space is 18-dimensional, corresponding to nine active power (P) and nine reactive power (Q) control actions distributed across different phases and terminals. Conversely, in a decentralized control approach, where each terminal has an independent controller, the action space is reduced to six dimensions, consisting of three active power and three reactive power adjustments per terminal. This study focuses on developing a decentralized controller, which simplifies training, reduces computational complexity, and enhances system safety by localizing control decisions. By decentralizing control, each terminal can respond independently to voltage fluctuations, reducing the risk of large-scale grid instability due to a single-point failure. However, in future research, a centralized RL controller will be developed to explore the benefits and trade-offs of a global optimization approach, particularly in terms of coordination efficiency and overall grid stability.

To effectively control the grid, the RL agent processes state observations that provide real-time insights into grid conditions. These observations include voltage measurements from supply and terminal nodes, predicted voltage profiles derived from machine learning forecasting models, load profiles representing future power demand, solar power generation data, and battery storage status, including charge levels and reactive power settings. By continuously analyzing this information, the RL agent learns to make adaptive, data-driven control decisions that improve grid voltage stability, optimize power flows, and enhance the overall efficiency of the power system. The structure of the RL-based controller is illustrated in Figure 3.10, providing an overview of its key components, including state observations, action decisions, and interactions with the grid environment.

3.7.2 RL algorithm selection and configuration

The selection of an appropriate RL algorithm is crucial for ensuring effective voltage stability control within the grid environment. For this research, Proximal Policy Optimization (PPO) was chosen as the primary RL algorithm due to its ability to handle continuous action spaces, which is essential for fine-tuning active and reactive power adjustments. PPO is well-suited for power system applications because it maintains stability during training through policy clipping and multiple optimization epochs, preventing large, destabilizing updates. Additionally, PPO has been widely used in control systems due to its robust

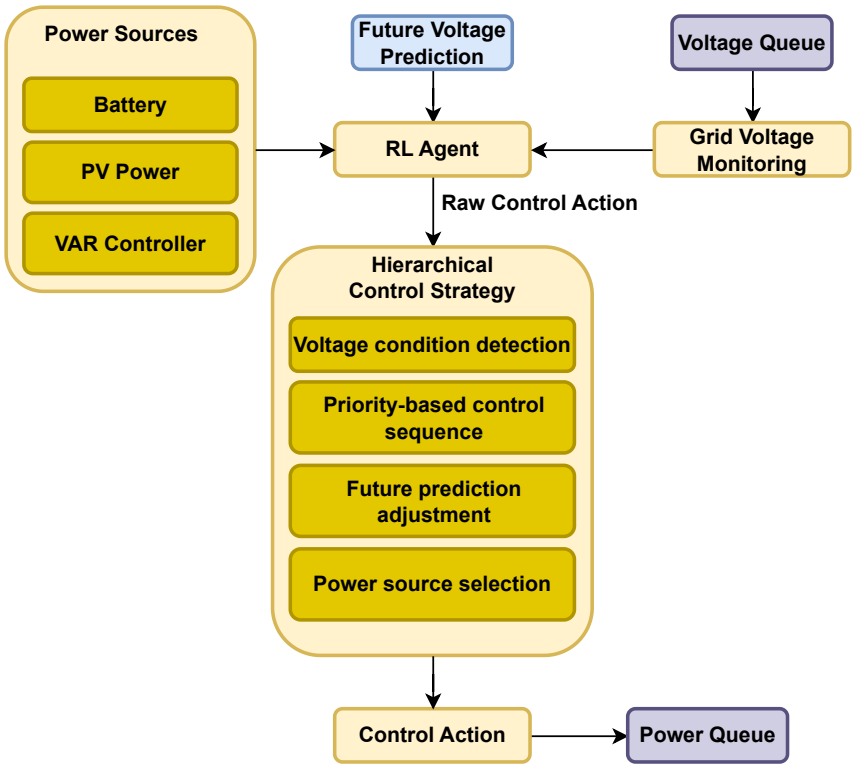

Fig. 3.10: RL controller architecture

convergence properties, making it a strong candidate for real-time grid voltage management.

As an alternative, Deep Deterministic Policy Gradient (DDPG) was also implemented, particularly for cases requiring precise, deterministic control over power adjustments. Unlike PPO, which operates as a policy-based method, DDPG is an actor-critic algorithm that leverages experience replay and target networks to stabilize learning. Its ability to handle high-dimensional, continuous state-action spaces makes it an effective approach for optimizing control actions in dynamic grid environments. By comparing both algorithms, this research evaluates which method offers better performance in terms of voltage stability, energy efficiency, and response time.

In the initial phase, the hyperparameters were chosen according to the recommendations from the Stable-Baselines3 [41] library, ensuring a reliable starting point for model training. These initial settings provided a well-balanced trade-off between learning stability and convergence speed. However, for future implementations, these parameters should be further optimized to enhance training efficiency, policy adaptability, and real-time decision-making performance.

To further improve training efficiency, several optimization techniques were applied. Adaptive learning rate scheduling was introduced, reducing the learning rate when performance improvements plateaued. An early stopping mechanism was implemented to terminate training when the policy improvements became negligible, preventing unnecessary computational overhead.

By carefully selecting and configuring these RL algorithms, the RL controller was optimized to handle real-time grid voltage stability adjustments effectively. The comparative analysis of PPO and DDPG enables the identification of the most suitable approach for dynamic power grid control, ensuring a balance between stability, adaptability, and energy efficiency.

3.7.3 Reward function design

The reward function is a critical component of RL training, as it defines the learning objectives and influences the agent's decision-making process. The designed reward function considers multiple factors to ensure that voltage stability is maintained while optimizing resource utilization.

1. **Voltage Stability Reward:** The agent receives a positive reward when voltage levels remain within the defined operational range. However, if voltage deviates beyond the acceptable tolerance band, penalties are applied. The penalty magnitude increases as the deviation worsens, ensuring that the RL agent prioritizes voltage stability.
2. **Resource Optimization Reward:** The agent is encouraged to maximize the use of solar energy while minimizing unnecessary battery discharge and reactive power compensation. A small positive reward is given for effectively utilizing solar power, while penalties are applied if excessive battery energy is used or reactive power is injected beyond required levels.
3. **Predictive Control Reward:** Since the RL controller operates with the help of ML-based forecasting, it also evaluates future voltage conditions. The agent is incentivized for making proactive control adjustments that prevent predicted voltage violations. If the forecasted voltage remains stable within limits, additional rewards are given; otherwise, penalties are assigned to discourage actions that may cause future instability.

Reward shaping and balancing objectives

To ensure that the RL agent learns an effective control strategy, the reward function must be carefully designed to balance multiple objectives, preventing the agent from prioritizing one aspect at the cost of another. A weighting mechanism is used to assign appropriate scaling factors to different reward terms, ensuring that both immediate voltage corrections and long-term power efficiency are considered in decision-making. The hierarchical decision-making structure ensures that the RL agent prioritizes critical voltage stability

issues while simultaneously optimizing energy resource utilization. Additionally, normalization techniques are applied to ensure that rewards remain consistent across different operational states, preventing the RL model from being biased toward specific conditions.

Although these techniques were considered in the reward function design, the reward shaping and balancing approach was not explicitly tested during this research due to the unstable data exchange pointed out in Section 3.6.4. Future work should focus on evaluating and fine-tuning the weighting parameters, exploring how different reward structures impact learning efficiency and control performance in real-time voltage regulation scenarios.

3.7.4 Training process and model evaluation

The training process for the RL model is conducted within the Mosaik-based co-simulation framework, enabling real-time interaction between the MATLAB-based grid simulation and the Python-based RL agent. The training environment is carefully designed to replicate realistic grid scenarios, incorporating fluctuating solar power generation, varying load demands, and diverse operating conditions. The objective of training is to optimize voltage stability by dynamically adjusting power flow and reactive power compensation based on real-time observations.

The training process follows a systematic sequence to facilitate progressive learning and policy refinement. It begins with environment initialization, where the co-simulation framework is configured by defining the observation space, action space, and initial grid parameters. This step ensures that the RL agent starts with a well-defined baseline. In the exploration phase, the RL agent interacts with the environment using a trial-and-error approach, allowing it to collect experiences and gradually learn the relationship between different grid states and their corresponding control actions. This is followed by the policy learning phase, where the model refines its decision-making process based on feedback from the reward function. Over successive training iterations, the agent gradually improves its ability to regulate voltage levels effectively. As training progresses, the model reaches the convergence and optimization phase, where its policy stabilizes and consistently selects optimal control actions across different grid conditions.

To evaluate the performance of the trained RL model, multiple metrics are used to assess its effectiveness in maintaining voltage stability. One key metric is the Mean Absolute Voltage Deviation (MAVD), which quantifies how well the RL agent keeps voltage levels within the desired range. The Voltage Stability Index is also considered, evaluating the overall stability of the grid by analyzing voltage fluctuations and transient behaviors. Another critical factor is the control response time, which measures the speed at which the RL model reacts to voltage deviations and applies corrective actions. Additionally, the energy efficiency of the system is assessed by examining how effectively the RL controller manages power distribution, minimizes unnecessary battery discharge, and optimizes solar power utilization.

Through this systematic training and evaluation process, the RL model's ability to enhance voltage stability, reduce power losses, and improve overall grid performance is thoroughly tested before being integrated into the operational framework. However, challenges such as synchronization mismatches and training instabilities remain, requiring further refinements in RL policy design and co-simulation execution to improve overall reliability and robustness.

3.7.5 Integration with co-simulation framework

The integration of the RL agent with the Mosaik-based co-simulation framework ensures real-time interaction between MATLAB and Python, enabling a realistic representation of grid dynamics. Acting as a bridge between the MATLAB-based grid simulation and the Python-based RL controller, the co-simulation facilitates seamless data exchange and decision-making. This setup allows the RL agent to receive time-sensitive grid state information, ensuring accurate policy evaluation under varying grid conditions while maintaining computational feasibility.

Within this framework, data exchange between MATLAB and Python follows a structured process. The state observation transfer begins with the MATLAB-based simulator computing voltage states at each time step. Voltage readings, power demand, and PV generation data are extracted, formatted, and sent to the RL agent in Python. Once received, the RL controller processes the grid state information and determines the optimal control actions based on the trained RL model. These actions primarily involve adjusting active and reactive power at different terminals to optimize voltage stability. The computed control actions are then formatted and transmitted back to MATLAB, where they are applied to update the grid simulation.

A significant challenge in this integration process is ensuring proper synchronization between MATLAB and Python execution. Since MATLAB operates as a continuous simulation while the RL agent functions in discrete time steps, execution timing misalignments can lead to data loss, delayed responses, or inefficient RL training. If MATLAB progresses faster than Python, the RL controller may receive outdated information, leading to suboptimal control actions. On the other hand, if Python takes longer to compute control decisions, the RL model might fail to respond quickly to changing grid conditions. These synchronization mismatches can ultimately hinder the agent's ability to learn meaningful control strategies and impact the overall stability of the co-simulation environment.

3.7.6 Challenges and potential improvements

Despite the ongoing integration efforts of the RL-based controller into the co-simulation framework, several challenges remain that impact the effectiveness and efficiency of the system. One of the primary challenges is the synchronization issue between MATLAB and Python. The RL agent in Python and the grid simulation in MATLAB operate asynchronously, leading to timing mismatches. As a result, control actions may not always be executed at the correct time step, affecting real-time voltage regulation. Additionally, dur-

ing high-speed simulations, delays in data exchange between MATLAB and Python can cause data loss and delayed responses, leading to situations where the RL agent receives outdated grid information, resulting in suboptimal or delayed control actions. Another challenge is slow policy convergence, as training the RL agent requires a large number of interactions with the grid simulation. The complexity of the action space further increases training time, making it computationally intensive. Running both MATLAB-based grid simulations and Python-based RL models simultaneously demands significant computational resources, potentially introducing performance bottlenecks that limit the feasibility of real-time deployment.

To address these challenges, several potential improvements are proposed. A hybrid synchronization approach can be implemented, where MATLAB and Python communicate at fixed intervals while allowing minor asynchronous updates. Instead of completely stopping MATLAB execution, buffer-based data handling can be introduced to better align data exchange. Improving data handling and transmission efficiency is another solution, which involves using a message queue system to ensure that every voltage observation and control action is correctly transmitted without loss. Additionally, error-checking mechanisms can be implemented to detect and correct missing data in real time.

For training enhancements, adaptive RL training strategies can be introduced. Curriculum learning, where the RL agent is initially trained on simpler grid conditions before progressing to more complex scenarios, could improve learning efficiency. Similarly, transfer learning could be leveraged to fine-tune the RL model based on historical grid data, reducing the number of interactions required for training. Another critical improvement involves optimization of computational workload by utilizing parallel processing to distribute computational tasks across multiple processors. Additionally, dynamically adjusting the simulation step size can help reduce unnecessary computations without sacrificing accuracy.

Beyond these technical improvements, future research directions should explore advanced RL strategies to enhance grid voltage control. One promising approach is to develop a hybrid control strategy that combines RL with traditional rule-based controllers to improve robustness and stability. Additionally, the feasibility of multi-agent RL approaches should be investigated, where multiple RL agents control different aspects of the grid, leading to improved efficiency and decentralized decision-making. Another area for exploration is meta-learning techniques, which could enable adaptive RL models that self-tune based on real-time performance feedback, making them more resilient to varying grid conditions.

By further optimizing the synchronization mechanism and improving real-time data exchange, the co-simulation framework is expected to enhance the training and deployment of RL-based voltage control strategies. Future enhancements will focus on refining integration techniques, improving computational efficiency through parallel processing, and ensuring that the RL controller operates seamlessly within the co-simulation environment. These improvements aim to create a robust development environment for advancing RL-based voltage control strategies, ultimately achieving more reliable voltage stability in

PV-integrated power grids. By addressing these challenges, RL-based control strategies will become a practical and scalable solution for dynamic grid management.

By addressing these challenges and implementing the proposed improvements, the RL-based controller can achieve greater accuracy, reliability, and real-time applicability in smart grid environments. These refinements will play a crucial role in ensuring that RL-based voltage control strategies become a viable and scalable solution for future power grids.

This section provided a comprehensive overview of the RL-based voltage control system, covering key aspects such as algorithm selection, reward function design, training strategies, and integration with the co-simulation framework. While the developed system demonstrated potential in utilizing RL for proactive voltage control, challenges related to synchronization mismatches, RL training instability, and forecasting limitations affected its overall performance. Future improvements will focus on refining the synchronization mechanism to enhance real-time control accuracy and training efficiency, ensuring a more reliable interaction between MATLAB and Python.

This chapter outlined the development and integration of the RL-based voltage control system, detailing the processes of data collection, preprocessing, forecasting, RL algorithm selection, and co-simulation setup. It emphasized the role of voltage forecasting within the control framework and identified critical challenges, including timing inconsistencies, policy convergence issues, and limitations in real-time data exchange reliability. While the voltage forecasting model yielded promising results, its practical effectiveness in guiding RL-based control actions remained constrained due to these unresolved technical limitations. The co-simulation framework also faced data transmission and synchronization challenges, impacting real-time decision-making and the ability of the RL agent to adapt effectively to grid conditions.

Despite these challenges, the foundation established in this chapter lays the groundwork for future refinements and optimizations. Key areas requiring further development include enhancing RL training methodologies, optimizing reward function design, improving synchronization between MATLAB and Python, and refining voltage forecasting accuracy to enable more precise and reliable real-time decision-making. Additionally, exploring alternative control strategies or hybrid approaches could further improve the effectiveness of voltage stability management in PV-integrated grids.

Building on these findings, the next phase of this thesis will focus on experimental validation and system performance evaluation. Chapter 4 will examine the impact of voltage forecasting on grid stability, assess the efficiency of the co-simulation framework, and explore alternative approaches to enhance grid control strategies. These insights will contribute to a more refined and functional RL-based voltage control system, paving the way for its practical implementation in real-world PV-integrated power networks.

4 Results and discussion

This chapter presents the results obtained from the voltage forecasting model and the co-simulation framework, analyzing their effectiveness in predicting voltage profiles and facilitating real-time grid voltage control. As described in the methodology chapter, several significant challenges emerged during the development phase, particularly in training the RL-based control algorithm. These challenges hindered the ability to fully implement and evaluate the RL control strategy within the co-simulation framework. Therefore, this section primarily focuses on assessing the performance of the forecasting model, evaluating the efficiency of the co-simulation framework, and discussing the obstacles encountered in achieving real-time voltage stability control.

The chapter is structured as follows: First, the accuracy and reliability of the voltage forecasting model are examined, with key evaluation metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R^2 score. A discussion on the impact of forecasting accuracy on voltage stability and grid performance follows. Next, the performance of the co-simulation framework is analyzed, highlighting the efficiency of data exchange between MATLAB (grid simulation) and Python (control implementation), as well as synchronization issues and computational constraints. Subsequently, the challenges faced in implementing the RL-based voltage control strategy are discussed, including training instabilities, synchronization issues, and execution delays. Finally, the chapter concludes with a broader discussion of the overall system performance, implications for future grid control strategies, and the necessary refinements to enhance real-time voltage control.

By identifying the strengths, limitations, and potential improvements of each system component, this chapter provides a foundation for future refinements in achieving an effective voltage control system for smart grids.

4.1 Results of voltage forecasting model

This section presents an analysis of the input features used in the voltage forecasting model, followed by the evaluation of the model's performance. The input data plays a crucial role in determining the accuracy and reliability of the forecasting process. To understand the relationships between different features and their impact on voltage stability,

correlation heatmaps and feature dependency plots were generated. These visualizations provide insight into the dependencies between power generation, load demand, and voltage fluctuations across different terminals.

4.1.1 Input data analysis and feature selection

The forecasting model follows a multi-model structure, where each model predicts the voltage for a specific terminal and phase. The input matrix is designed to capture the influence of power demand and solar power generation across all terminals to ensure a comprehensive representation of the system's dynamics. The general structure of the input matrix $\mathbf{X}$ and the output matrix $\mathbf{Y}$ can be expressed as:

$$\mathbf{Y} = f(\mathbf{X}) \tag{4.1}$$

where:

- $\mathbf{X}$ represents the input feature matrix, containing load demand and solar power generation from all three terminals over multiple time steps.
- $\mathbf{Y}$ represents the output matrix, predicting voltage values at each terminal and phase.

The generalized input matrix can be represented as:

$$\mathbf{X} = \begin{bmatrix} PL_{P_i}(t) & PL_{Q_i}(t) & PS_{P_i}(t) & PS_{Q_i}(t) \\ PL_{P_i}(t-1) & PL_{Q_i}(t-1) & PS_{P_i}(t-1) & PS_{Q_i}(t-1) \\ \vdots & \vdots & \vdots & \vdots \\ PL_{P_i}(t-n) & PL_{Q_i}(t-n) & PS_{P_i}(t-n) & PS_{Q_i}(t-n) \end{bmatrix} \tag{4.2}$$

where:

- $i \in \{1, 2, 3\}$ represents the three terminals.
- $PL_{P_i}(t)$ and $PL_{Q_i}(t)$ are the active and reactive power demand at terminal i.
- $PS_{P_i}(t)$ and $PS_{Q_i}(t)$ are the active and reactive power generation at terminal i.
- The matrix consists of past n time steps, enabling time-series forecasting.

The generalized output matrix is:

$$\mathbf{Y} = \begin{bmatrix} VT_i^j(t) \\ VT_i^j(t+1) \\ \vdots \\ VT_i^j(t+m) \end{bmatrix} \tag{4.3}$$

where:

- $VT_i^j(t)$ is the predicted voltage at terminal i, phase j.
- $j \in \{1, 2, 3\}$ represents the three phases.
- Each forecasting model predicts voltage at one terminal-phase pair.

This approach ensures that each model is trained with a consistent set of input features, allowing for terminal-specific voltage forecasting while leveraging system-wide information on power demand and solar generation. By incorporating past time-series data, the forecasting model can effectively predict voltage behavior under different grid conditions.

The correlation heatmaps for Terminal 1, Terminal 2, and Terminal 3 (Figure 4.1) illustrate the relationships between active power demand, reactive power, solar power generation, and voltage fluctuations. These plots were generated using historical grid data, where voltage variations were analyzed alongside corresponding power values at each terminal. The heatmaps reveal key dependencies among these variables:

- **Terminal 1** shows a strong positive correlation between active power demand (P_L) and voltage fluctuations, indicating that load variations significantly influence voltage stability. A **moderate correlation** exists between solar power generation (P_S) and voltage, suggesting that while PV output affects voltage levels, other grid factors also play a role. Reactive power (Q) exhibits a weaker correlation, implying that it may not be the dominant factor for voltage regulation at this terminal.
- **Terminal 2** follows a similar trend, with active power demand being the primary factor affecting voltage stability. However, some reactive power components display negative correlations, indicating an inverse relationship between reactive power injections and voltage variations.
- **Terminal 3** presents a strong negative correlation (-0.94) between certain power components and voltage, highlighting a potential instability issue at this terminal. Unlike the other terminals, reactive power values show higher correlation coefficients, suggesting that voltage control strategies at this location might need to account for reactive power adjustments more prominently.

In addition to correlation analysis, feature dependency plots were generated to assess the non-linear impact of individual power components on voltage trends (Figure 4.2). These plots were derived from voltage and power measurements under various operating conditions and illustrate how voltage deviations depend on both active power demand and solar power generation.

The x-axis in these plots represents the power variables, either Load Power (PL) [W] or Solar Power (PS) [W], while the y-axis denotes Voltage (VT) [V] at different phases and terminals. This analysis helps in understanding how variations in power components influence voltage behavior.

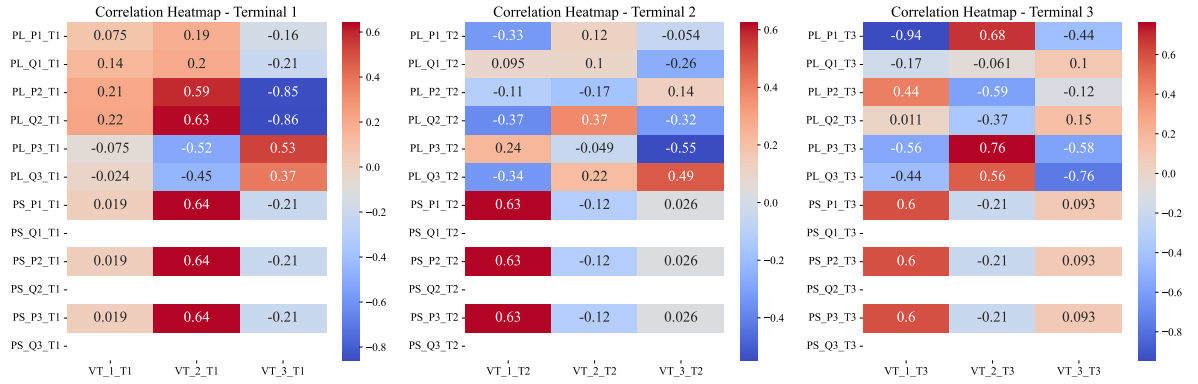

Fig. 4.1: Correlation heatmap

- **Load power vs. voltage dependency:** The plots indicate a non-linear relationship where voltage decreases as active power demand increases. This trend is consistent across different phases (V_{T1}, V_{T2}, V_{T3}), demonstrating that load fluctuations have a significant phase-dependent impact on voltage stability. For example, in Phase 3, when load power (PL_{P3}) varies between 0 W and 1200 W, the corresponding voltage fluctuates within 280 V to 340 V. Similarly, in Phase 1, load power (PL_{P1}) spans the same range, but the voltage response differs slightly, indicating that phase-wise power variations uniquely influence voltage levels.

- **Solar power vs. voltage dependency:** The effect of PV generation on voltage is more pronounced at higher power injection levels. For instance, when solar power (PS_{P3}) increases from 0 W to 2000 W, voltage fluctuations become more significant. This suggests that while solar power helps stabilize voltage by injecting active power into the grid, excessive PV penetration without adequate voltage regulation can cause undesirable variations.

These findings reinforce the importance of proactive voltage regulation strategies that account for load variations and high PV penetration levels. Additionally, the differences observed in correlation patterns across the three terminals indicate that voltage stability control measures need to be location-specific, particularly for Terminal 3, where reactive power plays a more critical role in voltage regulation. This analysis also supports the inclusion of temporal features in the predictive model, as past voltage values were found to significantly contribute to forecasting accuracy.

PV generation primarily impacts voltage stability under high penetration conditions, while load demand has a more consistent influence. Additionally, time-series analysis indicates that past voltage values contribute significantly to forecasting accuracy, reinforcing the importance of incorporating temporal features in the predictive model.

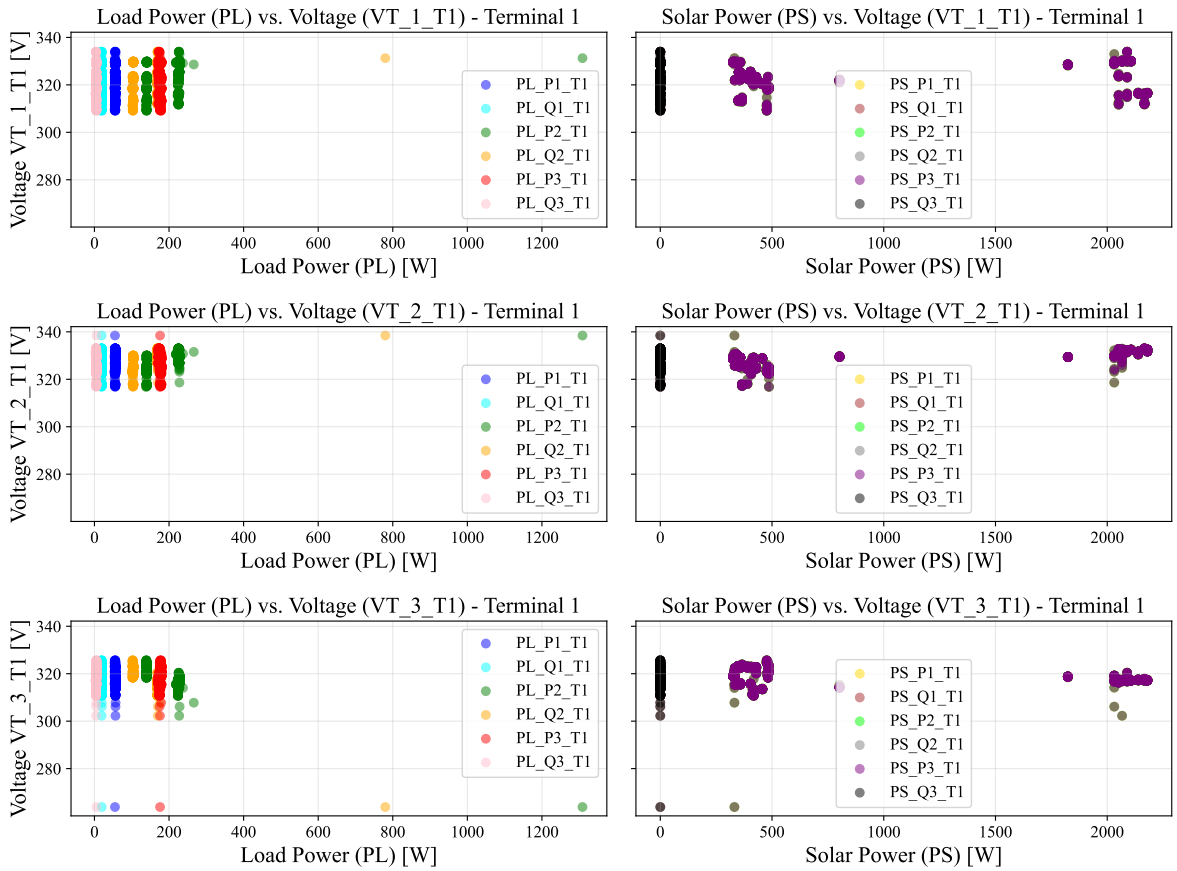

Fig. 4.2: Feature dependency

4.1.2 Forecasting accuracy metrics

To evaluate the effectiveness of the voltage forecasting model, the following performance metrics are used: MAE quantifies the average deviation between predicted and actual voltage values, providing an overall measure of forecasting accuracy. Root Mean Square Error (RMSE) penalizes larger deviations more heavily, making it a useful metric for identifying instances where the model struggles with extreme fluctuations. The R-squared (R^2) score determines how well the model explains variations in voltage behavior, offering insights into its predictive capability. Finally, computational efficiency is assessed to ensure that the model can generate forecasts in real-time scenarios without excessive delays.

The evaluation results are summarized in the following Table 4.1.

Interpretation of Results:

Tab. 4.1: Evaluation Metrics for Each Phase.

Metric	Phase 1	Phase 2	Phase 3
MAE (V)	1.7593	1.8972	4.5990
RMSE (V)	1.9852	2.1416	5.4900
R^2 Score	-5.3585	-7.1089	-52.2868
Prediction Time (ms)	2.500	2.800	3.200

- The MAE values, ranging from 1.7593 V to 4.5990 V, indicate the average prediction error across phases. Phase 3 exhibits the highest MAE, suggesting greater variability in its voltage fluctuations.
- The RMSE values (1.9852 V to 5.4900 V) suggest that the standard deviation of prediction errors is minimal for Phase 1 but significantly higher for Phase 3.
- The R^2 Scores, which are negative across all Phases, indicate that the models do not accurately capture the variance in voltage fluctuations. The large negative values, particularly for Phase 3 (-52.2868), suggest poor model performance in predicting voltage stability in that region.

These metrics provide an objective assessment of the model's ability to forecast voltage fluctuations across different grid conditions.

4.1.3 Graphical representation and analysis of forecasting performance

The accuracy of the forecasting model was evaluated through graphical visualizations comparing actual and predicted voltage values across three phases. These time-series plots illustrate the model's ability to capture trends and fluctuations in voltage profiles. Error distribution analysis and trend evaluations under varying grid conditions provide further insights into the model's performance. The testing was conducted under three key scenarios: high solar power generation with low load demand, high load demand with minimal solar power availability, and grid disturbances causing sudden voltage fluctuations.

To ensure realistic testing conditions, the forecasting models were evaluated using datasets containing 30 consecutive data points per evaluation cycle. This dataset structure aligns with the real-time data collection process of the Eye2Sky system, which provides a series of 31 data points every 30 seconds. This approach ensures that the model's performance is assessed in conditions similar to real-world applications.

The results, presented in Figures 4.3, 4.4, and 4.5, show that the model effectively captures overall voltage trends but struggles with small fluctuations, particularly during gradual voltage variations. Instead of closely tracking minor changes, the model maintains a steady prediction for extended periods and updates in discrete steps, likely due to the smoothing effect of the learning process. This behavior suggests that the model prioritizes

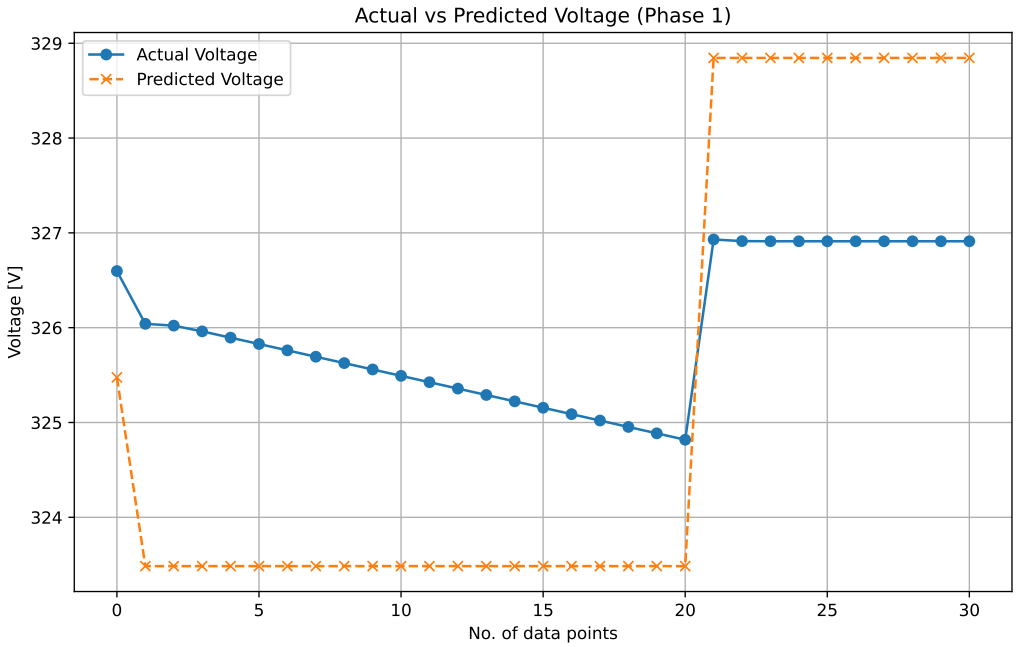

Fig. 4.3: Actual vs. Predicted Voltage (Phase 1)

larger voltage shifts over finer details, which can be beneficial in reinforcement learning but limits its precision in forecasting small variations. When voltage fluctuations are minimal, the predicted values remain relatively stable and sometimes fail to adapt to minor changes, leading to noticeable deviations. However, during significant voltage transitions, the model successfully detects and adjusts to the shifts, albeit in a stepwise manner rather than a smooth progression. Additionally, higher prediction errors were observed in scenarios with significant load variations, indicating that incorporating additional features such as real-time weather conditions or grid disturbances could improve accuracy.

Figure 4.3 illustrates the actual and predicted voltage for phase 1 across 30 data points. The model successfully follows the general voltage trend but exhibits a stepwise response instead of tracking minor variations. Similarly, Figure 4.4 presents the corresponding data for a different phase, revealing comparable behavior—stable predictions with abrupt adjustments rather than a continuous following of actual voltage. Finally, Figure 4.5 reinforces these observations, displaying a similar discrepancy where the model captures major transitions but struggles with finer fluctuations. These figures collectively highlight the model's strengths in detecting large-scale voltage changes and its limitations in responding to minor variations.

Overall, the forecasting model provides useful insights into voltage behavior but requires further refinements to enhance short-term prediction accuracy. Future improvements should focus on better capturing small voltage fluctuations and improving real-time adapt-

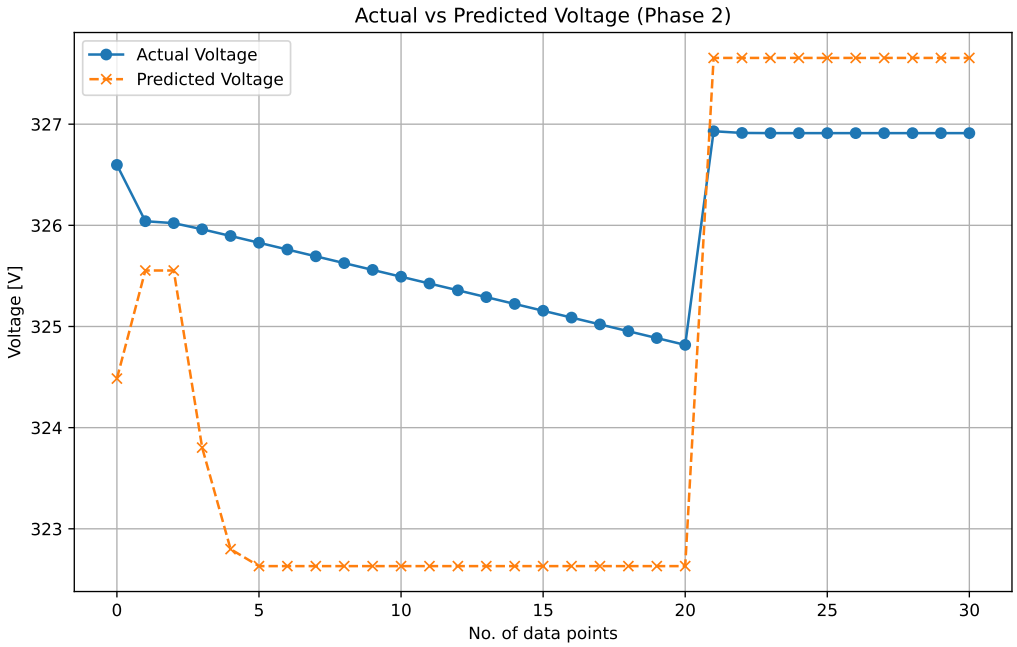

Fig. 4.4: Actual vs. Predicted Voltage (Phase 2)

ability to ensure a more precise response to gradual variations. Enhancing the model's sensitivity to minor changes while maintaining robustness in major transitions could lead to more effective voltage control strategies.

4.1.4 Discussion on forecasting model limitations

While the forecasting model demonstrates strong predictive capabilities, certain limitations must be considered. Forecasting errors can arise due to rapid fluctuations in solar power and load demand, making it challenging to capture sudden grid dynamics. Additionally, data inconsistencies, such as missing values and noisy observations, may introduce biases in predictions. Another challenge is the model's ability to generalize to unforeseen grid conditions, where external influences may disrupt expected trends. Furthermore, the assumption of a perfect load forecast is a limitation, as real-world implementations will inevitably encounter load prediction inaccuracies, affecting overall forecasting reliability. Additionally, obtaining real-time data from all terminals remains a challenge, as delays or missing observations could impact prediction accuracy.

Potential refinements include incorporating additional predictive features such as real-time weather conditions and grid frequency variations. Moreover, implementing ensemble learning methods could enhance model robustness by combining multiple prediction strategies to reduce error margins and improve generalization. Addressing load forecast-

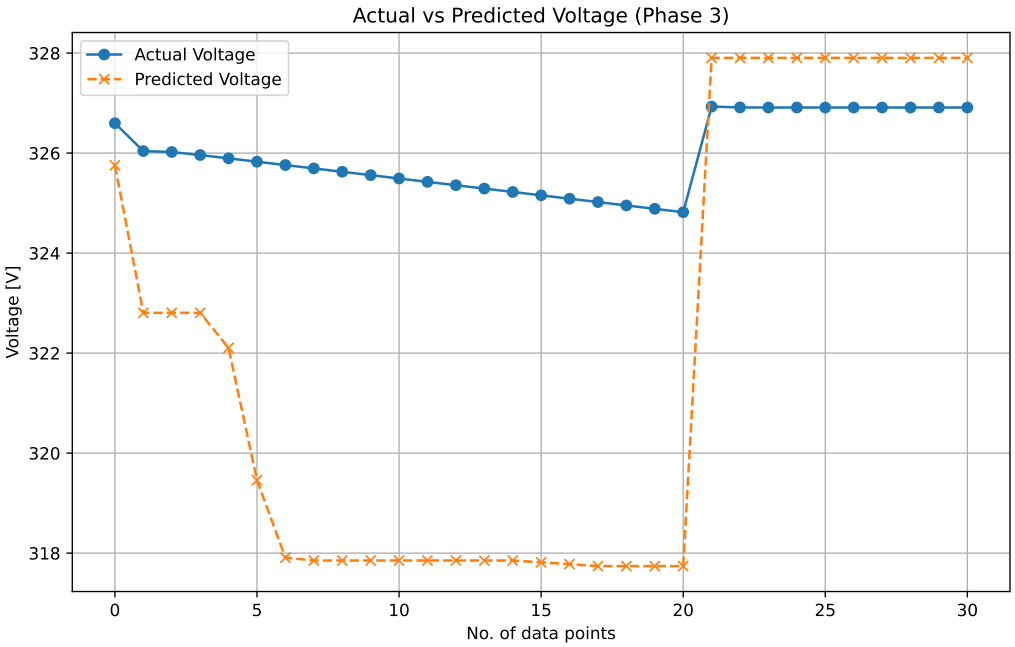

Fig. 4.5: Actual vs. Predicted Voltage (Phase 3)

ing uncertainties through probabilistic forecasting techniques may further improve model adaptability to real-world conditions.

In the context of RL-based voltage control, the accuracy of forecasting plays a crucial role. Reliable predictions enable the RL controller to take preemptive actions, enhancing grid stability. However, the RL control logic is primarily influenced by large or sudden voltage changes, as these require timely adjustments based on the model's forecasts. Minor or gradual variations can be managed directly by the RL controller itself, reducing dependency on fine-grained forecasting precision. This distinction helps balance computational complexity and real-time feasibility, ensuring that forecasting is performed efficiently without excessive delays. Future improvements should focus on optimizing the integration of forecasting into decision-making processes, ensuring that the model provides timely updates on significant voltage changes while allowing the RL controller to autonomously handle small fluctuations, ultimately leading to more effective voltage regulation strategies.

4.2 Performance evaluation of co-simulation framework

The co-simulation framework plays a crucial role in integrating the MATLAB-based grid model with the Python-based RL controller, enabling real-time voltage control decisions.

This section evaluates the efficiency of data exchange, synchronization challenges, and computational performance of the system. Despite the intended functionality, several limitations have emerged that impact the overall performance, particularly in achieving real-time coordination between the two simulators.

4.2.1 Data exchange and synchronization issues

The co-simulation setup relies on a queue-based communication mechanism to facilitate data exchange between MATLAB and Python. Initially, the objective was to retrieve 12 voltage values per time step, including observations from the supply terminal and all three load terminals. However, due to synchronization issues, this was not feasible. Instead, the system was restricted to receiving only three voltage observations at a time, corresponding to all three phases of a single load terminal.

One of the primary challenges encountered was the loss of data packets, leading to incomplete observations that affect the reliability of the RL controller's decision-making process. This issue is compounded by the mismatch in execution steps between the MATLAB simulation and the RL controller. Since both simulators operate independently within the Mosaik framework, this issue arises when the RL training and simulation execution operate in different threads. When both processes run sequentially within Mosaik's event-driven architecture, precise synchronization can be achieved. However, in the implemented setup, threading introduces timing inconsistencies, which lead to the loss of voltage state observations and result in the RL controller making control decisions based on outdated information. Additionally, delays in queue processing introduce further inconsistencies in the data received by the RL controller, resulting in an incomplete or outdated state representation.

4.2.2 Computational performance and multi-threading constraints

To handle the RL algorithm's training while simultaneously processing data exchange, a multi-threading approach was implemented. The RL training runs in a separate thread, allowing the main process to continue executing the simulation and exchanging data. While this setup prevents blocking operations, it introduces new challenges, particularly regarding synchronization and computational overhead.

A critical issue observed is that multi-threading interferes with the queue-based data exchange, occasionally leading to data loss or delays in processing observations. Since the RL training process operates faster than MATLAB's main execution loop, the training thread often processes new observations before MATLAB has completed its next simulation step. This results in state discrepancies where the RL agent updates its policy more frequently than MATLAB provides new grid observations, causing repeated control actions on the same state.

To quantify and maintain a consistent time difference, execution times were measured for both MATLAB's simulation step (T_{MATLAB}) and Python's RL training step (T_{RL}). The observed time difference (ΔT_{sync}) is defined as:

$$\Delta T_{\text{sync}} = T_{\text{MATLAB}} - T_{\text{RL}} \tag{4.4}$$

Experimental results showed that:

- The RL training thread is consistently faster than the main execution thread.
- The direct measured time difference ΔT_{sync} between the two threads is approximately 71.41 ms.
- This difference causes the RL agent to execute multiple training updates before MATLAB completes a single simulation step.

Since $T_{\text{RL}} < T_{\text{MATLAB}}$, the RL agent executes multiple training updates within one MATLAB execution cycle, which leads to inconsistencies where control actions are computed on outdated states.

The queue-based data exchange between MATLAB and Python further complicates synchronization. Since each *get()* operation on the queue removes the data point, the faster RL training thread sometimes retrieves an observation before MATLAB has generated the next step, resulting in:

- Repeated state evaluations, where the RL agent processes the same voltage readings multiple times.
- Control action redundancy, leading to unnecessary reactive power adjustments.

To address this, a time-lock mechanism is proposed, where the RL agent is restricted from executing multiple updates on the same state. Instead, it will be forced to wait until new data from MATLAB is available. This ensures that each control action corresponds to a distinct simulation step, reducing redundant decision-making.

The computational performance was analyzed based on execution time per simulation step and its impact on real-time decision-making. It was observed that maintaining a stable time difference between training and execution improves control stability. Future refinements should explore dynamic synchronization strategies, parallelized data pipelines, and efficient resource allocation to ensure that RL-based voltage control remains both adaptive and computationally efficient.

4.2.3 Discussion on co-simulation limitations and future optimizations

The observed synchronization challenges and computational constraints have significant implications for the effectiveness of the RL-based voltage control system. The loss of voltage observations limits the RL controller's ability to learn accurate state representations,

which in turn impacts its decision-making accuracy. Furthermore, delays in processing time-sensitive voltage data reduce the system's responsiveness, making it difficult to implement real-time corrective actions.

Several potential improvements can be explored to enhance the co-simulation framework. One approach is to optimize queue management techniques to improve data transfer efficiency and reduce the likelihood of data loss. Additionally, buffering mechanisms could be implemented to store incoming voltage observations temporarily, ensuring that no critical data is lost due to processing delays. Another possible solution involves refining the time-step alignment strategy to reduce discrepancies between MATLAB and Python, potentially through adaptive synchronization mechanisms that dynamically adjust the communication intervals between simulators.

Future modifications will aim to improve the reliability and efficiency of the co-simulation setup, enabling more robust and accurate real-time voltage control decisions. Addressing these challenges is crucial for ensuring that the RL-based controller functions as intended within the Mosaik-based framework, ultimately contributing to enhanced grid stability and performance.

4.3 Challenges in RL-based voltage control

The RL-based voltage control framework could not be successfully trained or executed due to critical issues in data synchronization, state observation, and computational constraints. These challenges prevented the RL controller from receiving consistent and complete training data, thereby making it impossible to initiate or evaluate the learning process. This section outlines the main challenges that prevented RL execution and discusses potential solutions that will be explored in future work.

4.3.1 Primary challenges preventing RL execution

The RL training process was not successfully executed due to inconsistencies in data exchange and synchronization between the MATLAB-based grid simulation and the Python-based RL controller. A primary issue was the incomplete state observations received by the RL agent. Originally, the controller was designed to process twelve voltage values per time step, covering all three phases at the supply and load terminals. However, communication constraints restricted data transmission to only three voltage values per time step, significantly reducing the available state information and limiting the RL agent's decision-making capability. Additionally, a measured time difference of 74.41 ms was observed between the training thread and the main execution thread, indicating that the training process was progressing faster than the main simulation. This desynchronization contributed to outdated or incomplete state representations, further hindering the RL controller's ability to learn and apply effective voltage control strategies.

Another significant challenge was the synchronization mismatch between MATLAB and Python. The MATLAB-based simulation executes independently in real time, while the

RL controller relies on queued observations to process actions. Due to the multi-threading approach and queue-based data exchange, MATLAB progressed through multiple time steps while the RL controller was still processing the previous step, leading to outdated or lost observations. This misalignment made it impossible for the RL controller to establish accurate relationships between its actions and the resulting voltage responses.

Additionally, delayed or missing control actions further exacerbated the problem. Since MATLAB did not pause to wait for RL-generated control actions, the computed actions were often outdated by the time they were applied. This rendered the reinforcement learning process ineffective, as the controller could not establish a stable feedback loop for learning. Computational constraints from multi-threading also contributed to the issue. The RL training process was executed in a separate thread to prevent blocking the data exchange, but this unintentionally interfered with the queue system, leading to further data loss and delayed updates.

4.3.2 Future solutions for improving RL implementation

Since the training process was never executed, the primary focus moving forward is to resolve synchronization issues and establish a stable data flow between MATLAB and Python before retraining the RL model. Enhancing synchronization between MATLAB and Python is one of the key steps in addressing this issue. A hybrid synchronization approach can be implemented, where MATLAB simulation execution is partially synchronized with RL-based decision-making, allowing for better alignment of control actions. Buffering techniques can also be introduced to store voltage observations in a structured manner before feeding them to the RL controller, ensuring that no critical data is lost due to step mismatches.

Another crucial improvement involves refining the observation and action space. Instead of requiring all twelve voltage observations, the model can focus on the most critical state variables that have the highest impact on grid stability. Additionally, using synthetic datasets to pre-train the RL model before integrating it into real-time simulation can help improve learning efficiency. Queue management and multi-threading strategies also need to be adjusted. Modifying queue structures to prevent message loss and ensuring that all received observations are processed before taking the next RL action can improve consistency. Furthermore, adjusting the execution order so that RL actions are applied immediately after voltage observations are received will help maintain real-time decision-making capabilities.

Beyond technical modifications, alternative approaches for RL training should also be explored. One potential solution is offline RL training, where a dataset of past simulation results is used to pre-train the model before deploying it in real-time. This would reduce the dependence on perfect synchronization during the initial training phases. Additionally, hybrid AI-based control strategies that combine RL with rule-based controllers can be investigated to mitigate synchronization challenges while maintaining efficient voltage control.

4.3.3 Summary of RL challenges and next steps

The inability to execute the RL training process was primarily due to synchronization mismatches, incomplete state observations, and multi-threading constraints. These issues prevented the RL controller from receiving valid training data, rendering the learning process ineffective. Moving forward, resolving data exchange issues and stabilizing the co-simulation framework will be prioritized before reattempting RL training. Once these challenges are addressed, the RL algorithm can be optimized for voltage stability control, leveraging predictive insights from the voltage forecasting model.

The findings from this chapter highlight the strengths and limitations of the proposed voltage forecasting and RL-based voltage control framework. While the forecasting model demonstrated structured input analysis and promising predictive capabilities, the RL-based controller faced major challenges in execution due to synchronization issues and data inconsistencies. The co-simulation framework successfully enabled MATLAB-Python integration; however, in the specific setup used in this research, real-time data exchange was impacted by asynchronous execution and multi-threading complexities. It is important to note that these synchronization challenges arise due to the chosen implementation of separate simulators running in parallel. When simulators are executed within a single-threaded environment without RL integration, real-time execution is not inherently an issue for the Mosaik platform. The next chapter will provide a comprehensive conclusion of the research and outline future directions for refining the RL-based control approach and addressing the identified challenges.

5 Conclusion and future work

The integration of renewable energy sources, such as PV systems, into power grids introduces challenges in maintaining voltage stability due to the variability in generation. This research aimed to address this issue by developing an RL-based voltage control framework (Section 3.7) that leverages short-term PV forecasts. Additionally, a machine learning-based voltage forecasting model (Section 3.5) was implemented to predict voltage profiles based on anticipated PV power generation and load demand. A co-simulation framework (Section 3.6) was established using MATLAB and Python, facilitated by Mosaik, to train a voltage control strategy in a simulated environment before potential deployment in a hardware setup. This setup allows for future replacement of the protocol converter and the grid model with real hardware, ensuring a seamless transition from simulation to practical implementation. However, due to significant synchronization issues and limitations in data exchange, the RL model could not be successfully trained and executed in this study. Despite these challenges, important insights were gained regarding the application of the co-simulation platform, particularly in its integration with reinforcement learning for voltage control. These findings provide valuable input for future research at the DLR Institute, highlighting areas for improvement in co-simulation stability, data handling, and real-time decision-making. This chapter summarizes the key contributions of the study, discusses encountered challenges and limitations, and outlines potential future research directions to enhance the proposed approach.

5.1 Summary of key contributions

This research successfully developed a machine learning-based voltage forecasting model capable of predicting voltage fluctuations based on short-term PV power and load demand forecasts. The model demonstrated its potential for predictive voltage control, offering a data-driven approach to managing grid stability. As detailed in Section 3.5, the forecasting model was designed to improve the RL-based controller's decision-making process by providing timely predictions of voltage variations.

A MATLAB-Python co-simulation framework was implemented to facilitate real-time voltage control experiments using RL. This setup enabled real-time data exchange between the MATLAB-based grid simulator and the Python-based control logic, providing

a foundation for testing AI-based voltage regulation strategies (Section 3.6). The modular structure of the co-simulation allows for future enhancements, including real-time synchronization improvements and integration with additional control methodologies.

While the RL-based voltage control strategy could not be fully executed due to synchronization and data exchange limitations, this study provided key insights into the complexities of integrating RL with real-time power grid simulations (Section 4.3). The investigation into multi-threading, queue-based data handling, and time-step mismatches highlighted significant bottlenecks affecting RL performance. These findings contribute to improving co-simulation stability, optimizing data handling mechanisms, and refining AI-driven voltage control strategies.

5.2 Research limitations

Despite its contributions, this study encountered several challenges that impacted the successful implementation of the RL-based voltage control strategy. One of the primary limitations was the synchronization mismatch between MATLAB and Python, which led to inconsistencies in data exchange and step execution. As discussed in Section 3.6.3, these mismatches resulted in delayed or missing observations, preventing the RL controller from receiving a stable and continuous stream of grid state data. Additionally, constraints in queue-based data handling restricted the system to retrieving only three voltage observations per step instead of the intended twelve, significantly reducing the amount of available state information. This issue prevented the RL controller from establishing a reliable learning process and making effective control decisions, limiting its ability to regulate voltage effectively.

Another significant limitation was the computational complexity of real-time RL training. The need for high-frequency control decisions introduced substantial computational overhead, requiring a balance between accuracy and real-time feasibility. While multi-threading was implemented to allow parallel execution of RL training and simulation, it inadvertently interfered with the queue system, causing data loss and outdated state updates. The RL training process was consistently faster than MATLAB's execution, leading to mismatched control actions, which further reduced the overall effectiveness of the learning framework. These computational bottlenecks posed challenges in maintaining a stable and responsive RL-based voltage control system, underscoring the need for improved scheduling mechanisms and asynchronous training strategies.

Furthermore, while the voltage forecasting model successfully predicted general voltage trends, it struggled with handling rapid fluctuations and unexpected grid disturbances. Since the model relied on predicted PV power and load demand, any inaccuracies in these forecasts directly impacted voltage prediction accuracy. This introduced uncertainties that could affect the RL controller's decision-making process. Future improvements should explore probabilistic forecasting techniques or ensemble learning methods to enhance the model's robustness under dynamic grid conditions.

These limitations highlight key areas where further refinements are necessary to achieve reliable, real-time RL-based voltage control. Future research should focus on resolving synchronization mismatches, improving data handling mechanisms, and optimizing forecasting accuracy to enhance the effectiveness of AI-driven voltage regulation strategies.

5.3 Future research directions

To overcome the challenges identified in this study, future research should focus on three key areas: improving the co-simulation framework, refining the RL-based voltage control strategy, and enhancing the accuracy of voltage forecasting models. One of the most critical improvements involves resolving synchronization issues between MATLAB and Python to enable reliable real-time data exchange. Implementing event-driven communication mechanisms, such as ZeroMQ or shared memory approaches, could reduce timing mismatches and data loss. Additionally, introducing a structured data buffering system would allow temporary storage of voltage observations before feeding them to the RL controller, ensuring a more stable and continuous learning process. These refinements will enhance the robustness of co-simulation frameworks for AI-driven grid stability applications.

Further advancements in reinforcement learning-based voltage control are also necessary. Since real-time RL training was hindered by computational constraints and synchronization issues, offline pre-training using historical simulation data could provide the RL agent with a strong initial policy before transitioning to real-time learning. This would reduce the model's reliance on continuous real-time updates, improving stability and convergence. Exploring alternative RL architectures, such as model-based reinforcement learning or hybrid rule-based RL approaches, may also enhance training efficiency and decision accuracy. Refining the observation and action space by selecting only the most relevant voltage parameters could further optimize real-time voltage control while reducing computational complexity.

Additionally, improvements in voltage forecasting will be essential to support proactive voltage regulation strategies. While the forecasting model successfully captured general voltage trends, its accuracy can be improved by incorporating real-time weather conditions, grid frequency variations, and historical voltage fluctuations as additional predictive features. Moreover, exploring probabilistic forecasting techniques or ensemble learning methods could help quantify uncertainties in PV and load forecasts, allowing the RL controller to make more reliable voltage control decisions even in uncertain conditions.

By addressing these challenges, future research can bridge the gap between simulation-based RL training and real-world grid control applications. The insights from this study provide a strong foundation for further development, ensuring that AI-driven voltage stability strategies can be effectively integrated into modern power grids. Advancing co-simulation techniques, refining RL methodologies, and improving forecasting accuracy

will be key to achieving scalable, real-time, and intelligent voltage regulation for future energy systems.

5.4 Final conclusion

This research explored reinforcement learning-based voltage control for smart grids, integrating a machine learning-driven voltage forecasting model with a MATLAB-Python co-simulation framework. The forecasting model effectively predicted voltage fluctuations based on short-term PV and load forecasts, providing a foundation for proactive grid management. The model was tested using 30 data points per iteration, reflecting the Eye2Sky system's data update cycle. While evaluation metrics, as discussed in Section 4.1.2, demonstrated strong predictive accuracy, the model struggled to capture small voltage fluctuations, often smoothing out minor variations instead of closely following them. This behavior, likely due to the model's reliance on past values and the smoothing effect of the learning process, highlights the need for further refinement, particularly in handling gradual changes and fine-scale voltage variations.

However, synchronization issues, delayed state observations, and queue-based data exchange inefficiencies prevented the successful execution of real-time RL training. These challenges underscored the complexities of integrating AI-based control strategies into power grid operations, particularly in multi-simulator environments. Despite these limitations, this study made significant contributions by implementing multi-threading and queue-based data handling within a co-simulation framework—an approach not previously explored in the proposed context. The insights gained on synchronization challenges, computational bottlenecks, and the impact of data exchange inefficiencies provide a valuable reference for future research aimed at refining MATLAB-Python co-simulation techniques for real-time voltage control applications.

Future advancements should focus on improving synchronization methods, implementing offline RL pre-training, and enhancing forecasting accuracy through probabilistic modeling. Additionally, refining voltage forecasting models to improve predictions under dynamic grid conditions will be essential for ensuring reliable RL-based voltage control. By addressing these challenges, AI-driven voltage stability strategies can become more practical for real-world deployment. This research serves as a foundation for further exploration into intelligent grid control, contributing to the development of more adaptive, stable, and scalable power systems in the era of renewable energy integration.

Appendix

A.1 Co-simulation pseudocode

This is the pseudocode for a co-simulation system between MATLAB/Simulink and a controller component using the Mosaik framework.

Algorithm 1 Co-Simulation Main Process

```
1: function RunCoSimulation
2:     config ← LoadConfiguration()
3:     world ← CreateMosaikWorld()
4:     matlab_sim ← world.StartSimulator("matlab", port=8011)
5:     controller_sim ← world.StartSimulator("controller", port=8013)
6:     ConnectSimulators(world, matlab_sim, controller_sim)
7:     world.Run(until=sim_end_time)
8: end function
```

Algorithm 2 MATLAB Simulator Flow

```
1: function MatlabSimulatorStep(time, inputs)
2:     // Get voltage measurements from Simulink model
3:     voltage_data ← GetVoltageMeasurements()
4:     // Send voltage data to controller
5:     QueueVoltageData(voltage_data)
6:     // Get power control inputs from controller
7:     power_data ← GetPowerControlData()
8:     // Apply power controls to Simulink model
9:     if power_data ≠ NULL then
10:        ApplyPowerControls(power_data)
11:    else
12:        ApplyDefaultControls()
13:    end if
14:    // Advance simulation time
15:    next_time ← time + time_step
16:    return next_time
17: end function
```

Algorithm 3 Controller Simulator Flow

```
function ControllerSimulatorStep(time, inputs)
    // Process voltage data from MATLAB
    voltage_data ← ProcessVoltageInputs(inputs)
    // Compute optimal power actions
    power_actions ← ComputeOptimalActions(voltage_data)
    // Send power actions to MATLAB
    QueuePowerActions(power_actions)
    // Advance simulation time
    next_time ← time + time_step
    return next_time
end function
```

Algorithm 4 Queue-Based Data Exchange

```
function QueueVoltageData(voltage_data)
    voltage_queue.put(voltage_data, timeout=0.1)
    return SUCCESS QueueFull
    ClearQueue(voltage_queue)
    voltage_queue.put(voltage_data)
    return SUCCESS
end function
function GetPowerActions
    power_data ← power_queue.get(timeout=0.1)
    return power_data QueueEmpty
    return NULL
end function
```

Algorithm 5 Flask Interface Routes

```
function InitializeFlaskServers
    // MATLAB server (port 8011)
    RegisterRoute("/", HandleInitRequest)
    RegisterRoute("/sendudp", HandleSendUDP)
    RegisterRoute("/recvudp", HandleReceiveUDP)
    RegisterRoute("/start", HandleStartSimulation)
    RegisterRoute("/stop", HandleStopSimulation)
    // Controller server (port 8013)
    RegisterRoute("/", HandleControllerInit)
    RegisterRoute("/sendudp", HandleControllerSendUDP)
    RegisterRoute("/recvudp", HandleControllerReceiveUDP)
end function
```

Algorithm 6 UDP Data Exchange

```
1: function SendPowerControl(terminal, phase, value)
2:     // Map terminal and phase to index
3:     index ← MapToIndex(terminal, phase)
4:     // Send via UDP
5:     data ← [index, value]
6:     SendUDPRequest("/sendudp", data)
7: end function
8: function *(GetVoltageMeasurement)terminal, phase
9:     // Map terminal and phase to index
10:    index ← MapToIndex(terminal, phase)
11:    // Request via UDP
12:    response ← SendUDPRequest("/recvudp", index)
13:    return ParseResponse(response)
14: end function
```

Queue management pseudocode

This is the pseudocode for the queue management system used in the co-simulation framework. The system provides thread-safe communication between the MATLAB simulator and the RL controller through a centralized queue architecture.

Algorithm 7 Queue Manager Core Functions

```
voltage_queue ← Queue(maxsize=2)
power_queue ← Queue(maxsize=2)
function PutVoltageData(data)
    success ← FALSE
    errors ← 0
    if voltage_queue.full() then
        voltage_queue.get_nowait()                    ▷ Remove oldest
    end if
    voltage_queue.put(data)
    success ← TRUE
    return success
end function
function GetVoltageData
    data ← NULL
    if NOT voltage_queue.empty() then
        data ← voltage_queue.get(timeout=0.5)
    end if
    return data
end function
function PutPowerData(data)
    success ← FALSE
    if power_queue.full() then
        power_queue.get_nowait()                      ▷ Remove oldest
    end if
    power_queue.put(data)
    success ← TRUE
    return success
end function
function GetPowerData
    data ← NULL
    if NOT power_queue.empty() then
        data ← power_queue.get(timeout=0.5)
    end if
    return data
end function
```

Algorithm 8 Queue Manager Maintenance Functions

```
function ClearQueues
    while NOT voltage_queue.empty() do
        voltage_queue.get_nowait()
    end while
    while NOT power_queue.empty() do
        power_queue.get_nowait()
    end while
    Log("All queues cleared")
end function
function ResetQueues
    ClearQueues()
    test_data ← {"test": "reset_check"}
    voltage_queue.put_nowait(test_data)
    result_v ← voltage_queue.get_nowait()
    power_queue.put_nowait(test_data)
    result_p ← power_queue.get_nowait()
    if result_v = test_data AND result_p = test_data then
        return TRUE
    else
        LogError("Queue reset verification failed")
        return FALSE
    end if
end function
```

Algorithm 9 Safe Queue Operations

```
function SafePutVoltage(data)
    success ← FALSE
    message ← ""
    success ← QueueManager.PutVoltageData(data)
    if success then
        message ← "Success"
    else
        message ← "Queue full"
    end if
    return success, message
end function
function SafeGetVoltage
    data ← QueueManager.GetVoltageData()
    if data ≠ NULL then
        return data, "Success"
    else
        return NULL, "Queue empty"
    end if
end function
function SafePutPower(data)
    success ← FALSE
    message ← ""
    success ← QueueManager.PutPowerData(data)
    if success then
        message ← "Success"
    else
        message ← "Queue full"
    end if
    return success, message
end function
function SafeGetPower
    data ← QueueManager.GetPowerData()
    if data ≠ NULL then
        return data, "Success"
    else
        return NULL, "Queue empty"
    end if
end function
```

Algorithm 10 Queue Verification

```
function VerifyQueueHealth
    test_data ← {"test": "health_check"}
    success_v, _ ← SafePutVoltage(test_data)
    result_v, _ ← SafeGetVoltage()
    voltage_healthy ← success_v AND result_v = test_data
    success_p, _ ← SafePutPower(test_data)
    result_p, _ ← SafeGetPower()
    power_healthy ← success_p AND result_p = test_data
    return voltage_healthy AND power_healthy
end function
function VerifyQueueState
    voltage_size ← QueueManager.voltage_queue.qsize()
    power_size ← QueueManager.power_queue.qsize()
    if voltage_size ≥ QueueManager.voltage_queue.maxsize then
        LogWarning("Voltage queue is full")
        return FALSE
    end if
    if power_size ≥ QueueManager.power_queue.maxsize then
        LogWarning("Power queue is full")
        return FALSE
    end if
    test_data ← {"test": "state_verify"}
    success_v1, _ ← SafePutVoltage(test_data)
    result_v, _ ← SafeGetVoltage()
    voltage_ok ← success_v1 AND result_v = test_data
    success_p1, _ ← SafePutPower(test_data)
    result_p, _ ← SafeGetPower()
    power_ok ← success_p1 AND result_p = test_data
    return voltage_ok AND power_ok
end function
```

Algorithm 11 Queue Monitor System

```
running ← FALSE
monitor_thread ← NULL
stats ← { 'voltage': {'puts': 0, 'gets': 0, 'errors': 0}, 'power': {'puts': 0, 'gets': 0, 'errors': 0} }
last_check ← CurrentDateTime()
function StartMonitoring
    if NOT running then
        running ← TRUE
        monitor_thread ← CreateThread(MonitorLoop)
        monitor_thread.daemon ← TRUE
        monitor_thread.start()
        Log("Queue monitoring started")
    end if
end function
function StopMonitoring
    running ← FALSE
    if monitor_thread ≠ NULL then
        monitor_thread.join(timeout=1.0)
        Log("Queue monitoring stopped")
    end if
end function
```

Algorithm 12 Queue Monitoring Loop

```
function MonitorLoop
    while running do
        voltage_size ← QueueManager.voltage_queue.qsize()
        power_size ← QueueManager.power_queue.qsize()
        Log("Queue Status - Voltage: " + voltage_size)
        Log("Queue Status - Power: " + power_size)
        Log("Queue Stats - Voltage: " + stats['voltage'])
        Log("Queue Stats - Power: " + stats['power'])
        if voltage_size ≥ QueueManager.voltage_queue.maxsize then
            LogWarning("Voltage queue is full")
        end if
        if power_size ≥ QueueManager.power_queue.maxsize then
            LogWarning("Power queue is full")
        end if
        Sleep(5)                                    ▷ Check every 5 seconds
    end while
end function
function GetStats
    return { 'stats': stats, 'last_check': last_check, 'voltage_size': QueueManager.voltage_queue.qsize(), 'power_size': QueueManager.power_queue.qsize() }
end function
function ResetStats
    stats ← { 'voltage': {'puts': 0, 'gets': 0, 'errors': 0}, 'power': {'puts': 0, 'gets': 0, 'errors': 0} }
    last_check ← CurrentDateTime()
    Log("Queue statistics reset")
end function
```

Data Flow

The queue management system facilitates bidirectional data flow between components:

1. **Voltage Data Flow:**
   - MATLAB collects voltage measurements
   - Data placed in voltage_queue via SafePutVoltage()
   - RL controller retrieves via SafeGetVoltage()
   - Controller processes data to determine actions
2. **Power Data Flow:**
   - RL controller determines optimal power controls
   - Actions placed in power_queue via SafePutPower()
   - MATLAB retrieves actions via SafeGetPower()
   - MATLAB applies power control to the simulation

Error Handling and Recovery

The queue management system implements robust error handling mechanisms:

1. **Queue Overflow Protection:**
   - When full, oldest data is automatically removed
   - Ensures new data is processed even under heavy load
   - Prevents deadlocks due to full queues
2. **Timeout Handling:**
   - Get operations use timeouts (0.5 seconds)
   - Returns NULL if queue is empty (no blocking)
   - Components can use default values if needed
3. **Error Recovery:**
   - VerifyQueueHealth() detects queue issues
   - ResetQueues() restores queues to clean state
   - ClearQueues() removes all data for fresh start
4. **Monitoring:**

- Continuous tracking of queue status
- Detects fill levels, throughput, and error rates
- Provides statistics for performance analysis

A.1.1 Thread safety

All queue operations are designed to be thread-safe:

- Queue implementations use atomic operations
- Error handling for concurrent environments
- Monitors run in separate non-blocking threads

A.1.2 Performance optimization

The queue system is optimized for real-time performance:

- Small queue sizes (maxsize=2) to minimize latency
- Automatic overflow handling to prevent blocking
- Timeouts on all operations to prevent deadlocks
- Minimal logging during normal operation

A.1.3 Robust error management

The system uses a layered approach to error management:

- Low-level Queue Manager with basic error reporting
- Mid-level Queue Handlers with detailed error tracking
- High-level verification functions for system-wide checks
- Continuous monitoring for proactive error detection

A.2 Reinforcement learning controller pseudocode

This is the pseudocode for a reinforcement learning controller designed for grid voltage management in a co-simulation system. The controller uses a hierarchical control strategy to maintain grid voltages within safe limits while optimizing power usage from various sources.

Algorithm 13 RL Controller training process

```
function TrainRLController
    config ← LoadSystemConfiguration()
    voltage_env ← CreateRLEnvironment(config)
    voltage_predictor ← InitializeMLPredictor(config.ml_model_mode)
    battery_system ← InitializeBatterySystem(config.battery)
    var_controller ← InitializeVARController(config.voltage)
    agent ← CreateRLAgent("PPO", voltage_env.observation_space, voltage_env.action_space)
    load_profiles ← LoadProfiles(config.paths.load_profiles_dir)
    solar_data ← SolarData(config.paths.solar_data_path)
    voltage_queue ← CreateQueue(maxsize=1)
    power_queue ← CreateQueue(maxsize=1)
    while not stop_event do
        voltage_data ← GetFromQueue(voltage_queue)
        if voltage_data = NULL then
            voltage_data ← GetDefaultVoltageData()
        end if
        voltage_env.update_voltage_data(voltage_data)
        sim_time ← voltage_data.sim_time
        current_voltages ← ExtractVoltagesFromData(voltage_data)
        future_loads ← load_profiles.GetFutureLoads(sim_time)
        solar_power ← solar_data.GetPowerAtTime(sim_time)
        predicted_voltages ← voltage_predictor.predict_future_voltage(future_loads, solar_power.future)
        observation ← CreateObservation(current_voltages, predicted_voltages, solar_power.current, future_loads)
        action ← agent.predict(observation)
        control_actions ← ProcessVoltageControl(current_voltages, predicted_voltages, solar_power, action)
        power_actions ← FormatPowerActions(control_actions)
        PutInQueue(power_queue, power_actions)
        reward ← CalculateReward(current_voltages, predicted_voltages, control_actions)
        agent.train_step(observation, action, reward)
        if IsEpisodeComplete(voltage_env) then
            ResetEnvironment(voltage_env)
            SaveModelCheckpoint(agent)
        end if
    end while
    SaveModel(agent, config.paths.output_dir)
end function
```

Algorithm 14 Process RL Actions into Control Signals

```
function ProcessVoltageControl(current_voltages, predicted_voltages, solar_power, action)
    control_actions ← {}
    for terminal_idx, terminal in {0:"T1", 1:"T2", 2:"T3"} do
        terminal_voltages ← current_voltages[terminal]
        idx_start ← terminal_idx × 6
        terminal_action ← action[idx_start:idx_start + 6]
        is_low ← AnyVoltageBelowLimit(terminal_voltages, config.voltage.lower_limit)
        is_high ← AnyVoltageAboveLimit(terminal_voltages, config.voltage.upper_limit)
        if is_low then
            control ← HandleLowVoltage(terminal, solar_power, terminal_action)
        else if is_high then
            control ← HandleHighVoltage(terminal, terminal_action)
        else
            control ← terminal_action
        end if
        future_voltages ← predicted_voltages[terminal]
        future_low ← AnyVoltageBelowLimit(future_voltages, config.voltage.lower_limit)
        future_high ← AnyVoltageAboveLimit(future_voltages, config.voltage.upper_limit)
        if future_low OR future_high then
            control ← AdjustForPredictions(control, future_low, future_high)
        end if
        control_actions[terminal] ← control
    end for
    return control_actions
end function
```

Algorithm 15 Handle Low Voltage Condition

```
function HandleLowVoltage(terminal, solar_power, action)
    solar_available ← CalculateTotalSolarPower(solar_power)
    power_needed ← CalculateRequiredPower(action)
    if solar_available ≥ power_needed then
        return CreateSolarOnlyControl(solar_available, power_needed)
    end if
    battery ← GetBatteryForTerminal(terminal)
    if battery.GetSOC() > battery.min_soc then
        remaining_power ← power_needed - solar_available
        battery_power ← MIN(remaining_power, battery.max_discharge_rate)
        return CreateSolarBatteryControl(solar_available, battery_power)
    end if
    return RequestPowerFromOtherTerminals(terminal, power_needed)
end function
```

Algorithm 16 Handle High Voltage Condition

```
function HandleHighVoltage(terminal, action)
    excess_power ← CalculateExcessPower(action)
    battery ← GetBatteryForTerminal(terminal)
    if battery.GetSOC() < battery.max_soc then
        charge_power ← MIN(excess_power, battery.max_charge_rate)
        return CreateBatteryChargeControl(charge_power)
    end if
    if OtherTerminalsNeedPower() then
        sharing_plan ← CreatePowerSharingPlan(terminal, excess_power)
        return CreatePowerSharingControl(sharing_plan)
    end if
    var_controller ← GetVARControllerForTerminal(terminal)
    q_required ← var_controller.calculate_reactive_power(GetAverageVoltage(terminal),
excess_power)
    return CreateVARCompensationControl(q_required)
end function
```

Algorithm 17 Calculate RL Reward

```
function CalculateReward(current_voltages, predicted_voltages, control_actions)
    reward ← 0.0
    for all terminal, voltages in current_voltages do
        for all voltage in voltages do
            if voltage WITHIN limits then
                reward ← reward + 10.0
            else
                violation ← MIN(ABS(voltage - lower_limit), ABS(voltage - upper_limit))
                reward ← reward - violation × 5.0
            end if
        end for
    end for
    for all terminal, action in control_actions do
        if action.solar_power > 0 then
            reward ← reward + 2.0
        end if
        if action.battery_power ≠ 0 then
            reward ← reward - 0.5 × ABS(action.battery_power / max_battery_power)
        end if
        if action.reactive_power ≠ 0 then
            reward ← reward - 0.3 × ABS(action.reactive_power / max_reactive_power)
        end if
    end for
    for all terminal, voltages in predicted_voltages do
        for all voltage in voltages do
            if voltage WITHIN limits then
                reward ← reward + 5.0
            else
                violation ← MIN(ABS(voltage - lower_limit), ABS(voltage - upper_limit))
                reward ← reward - violation × 2.0
            end if
        end for
    end for
    return reward
end function
```

Algorithm 18 Create Observation Vector

```
function CreateObservation(current_voltages, predicted_voltages, solar_power, future_loads)
    observation ← [ ]
    AddSupplyVoltages(observation, current_voltages)
    AddTerminalVoltages(observation, current_voltages)
    AddPredictedVoltages(observation, predicted_voltages)
    AddLoadValues(observation, future_loads)
    AddSolarValues(observation, solar_power)
    AddBatteryAndVARStates(observation)
    return observation
end function
```

Algorithm 19 Format Power Actions

```
function FormatPowerActions(control_actions)
    formatted_actions ← {}
    for all terminal, action in control_actions do
        p_values ← action.power_sources.p_values
        formatted_actions["P_3ph_" + terminal] ← {
            terminal + "_P1": p_values[0],
            terminal + "_P2": p_values[1],
            terminal + "_P3": p_values[2]
        }
        q_values ← action.power_sources.q_values
        formatted_actions["Q_3ph_" + terminal] ← {
            terminal + "_Q1": q_values[0],
            terminal + "_Q2": q_values[1],
            terminal + "_Q3": q_values[2]
        }
    end for
    return formatted_actions
end function
```

Data Flow

The RL controller manages bidirectional data flow between components:

1. **Input Data:**
   - Voltage measurements from MATLAB (9 values: 3 terminals × 3 phases)
   - Load forecasts from the load profile processor
   - Solar power forecasts from the Eye2Sky processor
2. **Internal Processing:**
   - Voltage prediction using ML models
   - RL agent for optimal action selection
   - Hierarchical control strategy implementation
3. **Output Data:**
   - Active power control signals (P) for each terminal and phase
   - Reactive power control signals (Q) for each terminal and phase
   - Battery charging/discharging commands

Implementation Details

Observation Space

The observation vector includes:

- Current voltages (12 values: 4 nodes × 3 phases)
- Predicted voltages (3 values for T1-only mode, 9 for all-terminals mode)
- Load values (18 values: 3 terminals × 6 values)
- Solar values (6 values: P1, Q1, P2, Q2, P3, Q3)
- Battery and VAR states (9 values: 3 terminals × 3 values)

Action Space

The action space consists of 18 dimensions:

- Active power control (P) for each terminal and phase (9 values)
- Reactive power control (Q) for each terminal and phase (9 values)

Reward Function

The reward function considers:

- Voltage stability within pre-defined limits
- Power source prioritization (solar > grid > battery)
- Minimal battery usage to extend lifetime
- Minimal reactive power usage for efficiency
- Predicted future voltage stability

A.3 Voltage Prediction Model - Pseudocode

Algorithm 20 Model Configuration

```
Set input_directory, output_directory, model_directory
Set train_test_split = 0.2
Set random_seed = 42
Set cross_validation_folds = 5
Set processor_cores = min(9, available_cores)
Set hyperparameters = {
    n_estimators: [50, 100, 200],
    max_depth: [3, 5, 7],
    min_samples_split: [2, 5, 10]
}
```

Algorithm 21 Data Processing

```
procedure ProcessData
    files_list ← list files in input_directory
    processed_data ← empty list
    for each file in files_list in parallel do
        df ← read_csv(file)
        df.columns ← GenerateColumnNames()
        processed_data.append(df)
    end for
    combined_data ← concatenate(processed_data)
    return combined_data
end procedure
```

Algorithm 22 Column Name Generation

```
procedure GenerateColumnNames
    // Load power features
    pl_cols ← ["PL_P{p}_T{t}" for t=1-3, p=1-3]
    pl_cols ← pl_cols + ["PL_Q{p}_T{t}" for t=1-3, p=1-3]
    // Solar power features
    ps_cols ← ["PS_P{p}_T{t}" for t=1-3, p=1-3]
    ps_cols ← ps_cols + ["PS_Q{p}_T{t}" for t=1-3, p=1-3]
    // Voltage targets
    vt_cols ← ["VT_{i}_T{t}" for t=1-3, i=1-3]
    return pl_cols + ps_cols + vt_cols
end procedure
```

Algorithm 23 Training Data Preparation

```
procedure PrepareTrainingData(data)
    // Extract features and targets
    feature_cols ← columns starting with 'PL_' or 'PS_'
    target_cols ← columns starting with 'VT_'
    X ← data[feature_cols]
    y ← data[target_cols]
    // Split into training and testing sets
    X_train, X_test, y_train, y_test ←
        train_test_split(X, y, test_size=0.2)
    return X_train, X_test, y_train, y_test, feature_cols, target_cols
end procedure
```

Algorithm 24 Single Target Model Training

```
procedure TrainSingleTarget(target, X_train, y_train, X_test, y_test)
    // Create grid search for hyperparameter tuning
    grid_search ← GridSearchCV with:
        estimator = RandomForestRegressor()
        param_grid = hyperparameters
        cv = cross_validation_folds
        scoring = 'r2'
    // Fit model to training data
    grid_search.fit(X_train, y_train[target])
    best_model ← grid_search.best_estimator_
    // Evaluate model
    y_pred ← best_model.predict(X_test)
    score ← r2_score(y_test[target], y_pred)
    return target, best_model, score
end procedure
```

Algorithm 25 Complete Model Training Process

```
procedure TrainModels(X_train, X_test, y_train, y_test, target_cols)
    // Train models for each target in parallel
    results ← ParallelExecute(
        TrainSingleTarget(target, X_train, y_train, X_test, y_test)
        for target in target_cols
    )
    // Collect results
    models ← {target: model for target, model, score in results}
    scores ← {target: score for target, model, score in results}
    // Save results
    SaveToFile(models, "models.pkl")
    SaveToFile(scores, "evaluation_scores.json")
    return models, scores
end procedure
```

Algorithm 26 Main Pipeline Execution

```
procedure Main
    SetupLogging()
    // Process data
    data ← ProcessData()
    // Prepare training data
    X_train, X_test, y_train, y_test, feature_cols, target_cols ←
        PrepareTrainingData(data)
    // Train and evaluate models
    models, scores ← TrainModels(X_train, X_test, y_train, y_test, target_cols)
    Log("Pipeline completed successfully")
end procedure
```

Lists

List of Tables

List of Figures

References

[1] S. Zhou, Y. Li, C. Jiang, Z. Xiong, J. Zhang, and L. Wang, "Enhancing the resilience of the power system to accommodate the construction of the new power system: Key technologies and challenges," *Frontiers in Energy Research*, vol. 11, p. 1 256 850, Sep. 5, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fenrg.2023.1256850/full (visited on 01/21/2025).

[2] J. Schaible, B. Nouri, L. Höpken, T. Kotzab, M. Loevenich, N. Blum, A. Hammer, J. Stührenberg, K. Jäger, C. Becker, and S. Wilbert, "Application of nowcasting to reduce the impact of irradiance ramps on PV power plants," *EPJ Photovoltaics*, vol. 15, p. 15, 2024. [Online]. Available: https://www.epj-pv.org/10.1051/epjpv/2024009 (visited on 01/21/2025).

[3] S. Rahman, S. Saha, M. Haque, S. Islam, M. Arif, M. Mosadeghy, and A. Oo, "A framework to assess voltage stability of power grids with high penetration of solar PV systems," *International Journal of Electrical Power & Energy Systems*, vol. 139, p. 107 815, Jul. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0142061521010310 (visited on 01/21/2025).

[4] "Reducing the impact of irradiance ramps on PV power production - a techno-economic analysis of nowcasting," ResearchGate. (), [Online]. Available: https://www.researchgate.net/publication/374087918_Reducing_the_Impact_of_Irradiance_Ramps_on_PV_Power_Production_-_A_Techno-Economic_Analysis_of_Nowcasting (visited on 03/03/2025).

[5] P. Kundur, *Power system stability and control*, Nachdr. New York: McGraw-Hill, 200, 1176 pp.

[6] J. Hossain and A. Mahmud, Eds., *Renewable Energy Integration: Challenges and Solutions*, Green Energy and Technology, Singapore: Springer Singapore, 2014. [Online]. Available: https://link.springer.com/10.1007/978-981-4585-27-9 (visited on 01/21/2025).

[7] "World energy outlook 2021 – analysis," IEA. (Oct. 13, 2021), [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2021 (visited on 02/26/2025).

[8] C. Steinbrink, M. Blank-Babazadeh, A. El-Ama, S. Holly, B. Lüers, M. Nebel-Wenner, R. P. Ramírez Acosta, T. Raub, J. S. Schwarz, S. Stark, A. Nieße, and S. Lehnhoff, "CPES testing with mosaik: Co-simulation planning, execution and analysis," *Applied Sciences*, vol. 9, no. 5, p. 923, Mar. 5, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/923 (visited on 07/29/2024).

[9] "Generation plants on the low-voltage grid (VDE-AR-N 4105)." (), [Online]. Available: https://www.vde.com/vde-ar-n-4105-2018 (visited on 03/04/2025).

[10] K. Kumar and B. Jaipal, "The role of energy storage with renewable electricity generation," in *Electric Grid Modernization*, M. Ghofrani, Ed., IntechOpen, Jul. 13, 2022. [Online]. Available: https://www.intechopen.com/chapters/75183 (visited on 01/21/2025).

[11] O. Alsayegh, S. Alhajraf, and H. Albusairi, "Grid-connected renewable energy source systems: Challenges and proposed management schemes," *Energy Conversion and*

Management, vol. 51, no. 8, pp. 1690–1693, Aug. 2010. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0196890409004920 (visited on 01/21/2025).

[12] L. Bird, M. Milligan, and D. Lew, "Integrating variable renewable energy: Challenges and solutions," NREL/TP-6A20-60451, 1097911, Sep. 1, 2013, NREL/TP–6A20–60 451, 1 097 911. [Online]. Available: http://www.osti.gov/servlets/purl/1097911/ (visited on 01/21/2025).

[13] F. Katiraei, R. Iravani, N. Hatziargyriou, and A. Dimeas, "Microgrids management," *IEEE Power and Energy Magazine*, vol. 6, no. 3, pp. 54–65, May 2008, Conference Name: IEEE Power and Energy Magazine. [Online]. Available: https://ieeexplore.ieee.org/document/4505827 (visited on 01/21/2025).

[14] J. Machowski, J. Bialek, and J. Bumby, "Power system dynamics. stability and control," Jan. 1, 2012.

[15] N. G. Hingorani and L. Gyugyi, *Understanding FACTS: concepts and technology of flexible AC transmission systems*. New York: IEEE Press, 2000, 1 p.

[16] H. Farhangi, "The path of the smart grid," *IEEE Power and Energy Magazine*, vol. 8, no. 1, pp. 18–28, Jan. 2010, Conference Name: IEEE Power and Energy Magazine. [Online]. Available: https://ieeexplore.ieee.org/document/5357331 (visited on 01/21/2025).

[17] M. Liserre, T. Sauter, and J. Y. Hung, "Future energy systems: Integrating renewable energy sources into the smart power grid through industrial electronics," *IEEE Industrial Electronics Magazine*, vol. 4, no. 1, pp. 18–37, Mar. 2010, Conference Name: IEEE Industrial Electronics Magazine. [Online]. Available: https://ieeexplore.ieee.org/document/5439057 (visited on 01/21/2025).

[18] S. Massoud Amin and B. Wollenberg, "Toward a smart grid: Power delivery for the 21st century," *IEEE Power and Energy Magazine*, vol. 3, no. 5, pp. 34–41, Sep. 2005, Conference Name: IEEE Power and Energy Magazine. [Online]. Available: https://ieeexplore.ieee.org/document/1507024 (visited on 01/21/2025).

[19] K.-i. Kondo and J. Baba, "Reactive power control by use of boost critical conduction mode power factor correction converter for suppressing voltage rise in distribution network," in *2014 16th European Conference on Power Electronics and Applications*, Lappeenranta, Finland: IEEE, Aug. 2014, pp. 1–9. [Online]. Available: http://ieeexplore.ieee.org/document/6910758/ (visited on 03/02/2025).

[20] B. Kroposki, R. Lasseter, T. Ise, S. Morozumi, S. Papathanassiou, and N. Hatziargyriou, "Making microgrids work," *IEEE Power and Energy Magazine*, vol. 6, no. 3, pp. 40–53, May 2008, Conference Name: IEEE Power and Energy Magazine. [Online]. Available: https://ieeexplore.ieee.org/document/4505826 (visited on 01/21/2025).

[21] S. Li, J. Drgona, S. Abhyankar, and L. Pileggi, *Power grid behavioral patterns and risks of generalization in applied machine learning*, Jun. 1, 2023. arXiv: 2304.10702[eess]. [Online]. Available: http://arxiv.org/abs/2304.10702 (visited on 03/04/2025).

[22] A. Pinto, L.-C. Herrera, Y. Donoso, and J. A. Gutierrez, "Enhancing critical infrastructure security: Unsupervised learning approaches for anomaly detection," *International Journal of Computational Intelligence Systems*, vol. 17, no. 1, p. 236,

Sep. 10, 2024. [Online]. Available: https://doi.org/10.1007/s44196-024-00644-z (visited on 01/21/2025).

[23] J. R. Vázquez-Canteli and Z. Nagy, "Reinforcement learning for demand response: A review of algorithms and modeling techniques," *Applied Energy*, vol. 235, pp. 1072–1089, Feb. 1, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261918317082 (visited on 01/21/2025).

[24] M. Y. Arafat, M. J. Hossain, and M. M. Alam, "Machine learning scopes on microgrid predictive maintenance: Potential frameworks, challenges, and prospects," *Renewable and Sustainable Energy Reviews*, vol. 190, p. 114 088, Feb. 1, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032123009462 (visited on 01/21/2025).

[25] A. Talhar Belge, S. Gupta, S. Alegavi, V. Singh, and K. Shukla, "Advancements, challenges, and future prospects of smart grid technology in india," *Frontiers in Artificial Intelligence*, vol. 7, 2024. [Online]. Available: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1475604 (visited on 01/21/2025).

[26] "Safe AI: Technology between safety and efficiency," MHP Management- und IT-Beratung. (), [Online]. Available: https://www.mhp.com/en/insights/blog/post/safe-ai (visited on 03/05/2025).

[27] M. S. Bhutta, Y. Li, M. Abubakar, F. M. Almasoudi, K. S. S. Alatawi, M. R. Altimania, and M. Al-Barashi, "Optimizing solar power efficiency in smart grids using hybrid machine learning models for accurate energy generation prediction," *Scientific Reports*, vol. 14, no. 1, p. 17 101, Jul. 24, 2024, Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-024-68030-5 (visited on 01/21/2025).

[28] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction,"

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature14236 (visited on 01/21/2025).

[30] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3, pp. 219–354, 2018. arXiv: 1811.12560[cs]. [Online]. Available: http://arxiv.org/abs/1811.12560 (visited on 01/21/2025).

[31] L. Quakernack, M. Kelker, and J. Haubrock, "Deep reinforcement learning for autonomous control of low voltage grids with focus on grid stability in future power grids," in *2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Novi Sad, Serbia: IEEE, Oct. 10, 2022, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/9960416/ (visited on 06/26/2024).

[32] H. Momen and S. Jadid, "Resilience enhancement of power distribution system using fixed and mobile emergency generators based on deep reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 137, p. 109 118, Nov. 1,

2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197624012764 (visited on 01/21/2025).

[33] N. F. P. Dinata, M. A. M. Ramli, M. I. Jambak, M. A. B. Sidik, and M. M. Alqahtani, "Designing an optimal microgrid control system using deep reinforcement learning: A systematic review," *Engineering Science and Technology, an International Journal*, vol. 51, p. 101 651, Mar. 1, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098624000375 (visited on 07/15/2024).

[34] A. B. S. Rufino and F. Saraiva, "Use of Augmented Random Search Algorithm for Transmission Line Control in Smart Grids - A Comparative Study with RNA-based Algorithms," in *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, ISSN: 2763-9061, SBC, Sep. 25, 2023, pp. 924–938. [Online]. Available: https://sol.sbc.org.br/index.php/eniac/article/view/25754 (visited on 07/15/2024).

[35] K. von Maydell, J. Petznik, H. Behrends, T. Esch, M. Ahmed, A. Rubio, L. Uhse, R. Völker, S. Unglaube, S. Geißendörfer, F. Schuldt, and C. Agert, "The networked energy systems emulation center at the german aerospace center dlr – bridging the gap between digital simulation and real operation of energy grids," English, *At-Automatisierungstechnik*, vol. 70, no. 12, pp. 1072–1083, 2022.

[36] "Wolkenkamera-Netzwerk Eye2Sky." (), [Online]. Available: https://www.dlr.de/de/ve/forschung-und-transfer/infrastruktur/labore_infrastrukturen/eye2sky (visited on 01/21/2025).

[37] M. Schlemminger, T. Ohrdes, E. Schneider, and M. Knoop, *WPuQ*, version 2.0, Nov. 5, 2021. [Online]. Available: https://zenodo.org/records/5642902 (visited on 03/02/2025).

[38] S. Shabou, *Chapter 5 Outlier detection in Time series | Time Series with R*. [Online]. Available: https://s-ai-f.github.io/Time-Series/outlier-detection-in-time-series.html (visited on 10/02/2024).

[39] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, 2nd edition. Melbourne: OTexts, 2018.

[40] M. Sengupta, A. Habte, S. Wilbert, C. Gueymard, and J. Remund, "Best practices handbook for the collection and use of solar resource data for solar energy applications: Third edition," NREL/TP-5D00-77635, 1778700, MainId:29561, Apr. 1, 2021, NREL/TP–5D00–77 635, 1778700, MainId:29561. [Online]. Available: https://www.osti.gov/servlets/purl/1778700/ (visited on 02/25/2025).

[41] A. RAFFIN, Q. Gallouédec, N. Dormann, A. Gleave, Anssi, A. Pasquali, J. Rocamonde, M. Ernestus, P. Helm, Corentin, Q. Sinclair, T. Simonini, T. Rohrer, S. Tio, R. Tangri, T. Dörr, Wilson, S. H. Wang, S. Toyer, R. Gavrilescu, P. M. Scheikl, P. Kothari, O. Kachaiev, B. Raml, C. Schindlbeck, C. Huang, D. Kerr, G. Passault, J.-H. Ewers, and M. Duclusaud, *DLR-RM/stable-baselines3: V2.5.0: New algorithm (SimBa in SBX) and NumPy 2.0 support*, version v2.5.0, Jan. 27, 2025. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.8123988 (visited on 03/02/2025).

[42] N. Beg, M. Ahmed, K. Derendorf, F. Schuldt, and S. Geißendörfer, "Distributed co-simulation of networked hardware-in-the-loop power systems," in *2023 IEEE PES*

Innovative Smart Grid Technologies Europe (ISGT EUROPE), Grenoble, France: IEEE, Oct. 23, 2023, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/10408193/ (visited on 07/24/2024).

[43] J. Dong, Y. Xue, M. Olama, T. Kuruganti, J. Nutaro, and C. Winstead, *Distribution Voltage Control: Current Status and Future Trends*. Jun. 1, 2018, 1 p., Pages: 7.

[44] P. G. Balakrishnan, R. Ramesh, and T. Prem Kumar, "Safety mechanisms in lithium-ion batteries," *Journal of Power Sources*, vol. 155, no. 2, pp. 401–414, Apr. 21, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775305016629 (visited on 03/02/2025).