People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of El Oued

Faculty of Exact Sciences

Laboratory of Artificial Intelligence and Applications

A Doctoral Thesis

Submitted in Fulfillment of the Requirements for the Degree of Doctor (LMD)

Field: Computer Science

Specialization: Artificial Intelligence

Title: A TECHNOLOGICAL COMBINATION BETWEEN IOT AND MACHINE LEARNING FOR SMART APPLICATIONS

Presented by: Mohamed Nadhir Abid

Defended on [02/10/2025], in the presence of the Examination Committee:

Name	Rank	Affiliation	Role
Boucherit Ammar	MCA	University of El Oued	**Chair**
Mounir Beggas	Pr	University of El Oued	**Supervisor**
Kahoul Laid	Pr	University of Biskra	**Examiner**
Laimeche Lakhdar	Pr	University of Tebessa	**Examiner**
Abbas Messaoud	Pr	University of El Oued	**Examiner**
Abdelkader Laouid	Pr	University of El Oued	**Co-Supervisor**

Academic Year: 2024-2025 / 1446-1447 AH

Acknowledgments

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

إِنَّمَآ إِلَٰهُكُمُ ٱللَّهُ ٱلَّذِى لَآ إِلَٰهَ إِلَّا هُوَ ۚ وَسِعَ كُلَّ شَىْءٍ عِلْمًا — Qur'an 20 : 98

Al-ḥamdu lillāh (all praise and thanks belong to Allah).
I am profoundly grateful to Allah for granting me health, perseverance, and the countless blessings that made this doctoral journey possible. Every insight, opportunity, and moment of clarity is a mercy from Him.

I extend my deepest appreciation to Pr. Mounir Beggas, whose guidance, patience, and high standards shaped each chapter of this thesis. Sincere thanks to Pr. Abdelkader Laouid, for his constructive critiques and generous time. I am grateful to Mostafa Kara and Mohammad Hammoudeh for a rigorous yet supportive review of the manuscripts. To the members of LIAP, thank you for the encouragement that fueled much of this work.

To my Father and Mother, thank you for instilling curiosity, resilience, and faith in me. Thank you for building this success with me since day one. To my spouse, your patience, humor, and endless support sustained me every step. To my siblings, your joy and encouragement motivated me to persevere.

To everyone who ever asked, "How's the thesis going?" and waited for the full answer, please accept my sincere gratitude. Any remaining errors are mine alone.

abstract

The confluence of Machine Learning (ML) and the Internet of Things (IoT) promises transformative data-driven decision-making, yet presents formidable challenges in securing interconnected systems and optimizing resource-constrained environments. This dissertation develops and validates novel ML and Reinforcement Learning (RL) frameworks to address critical vulnerabilities in IoT cybersecurity and enhance sustainability in arid-region agriculture. Specifically, it introduces (1) an ML-based system employing feature disentanglement and adversarial training for robust detection of obfuscated malware; (2) an adaptive RL-driven intrusion detection system for general IoT networks, with a specialized, safety-prioritized extension for Medical IoT ecosystems; (3) a custom RL environment simulating arid-region agriculture, enabling the development of adaptive irrigation policies that optimize water conservation and crop yield; and (4) a hybrid RL-clustering methodology that enhances state representation and policy generalization in complex, non-stationary continuous control tasks. By unifying theoretical advancements with rigorous empirical validation, this research demonstrates the synergistic potential of ML and RL to fortify IoT security and promote sustainable agricultural practices. The proposed solutions offer scalable and resilient frameworks for intelligent systems operating in dynamic, resource-limited environments, contributing to a digitally interconnected and ecologically sustainable future.

ملخص

يَعِدُ التقاء تعلم الآلة (ML) وإنترنت الأشياء (IoT) بإحداث تحول جذري في عملية صنع القرار القائمة على البيانات، ولكنه يطرح في الوقت ذاته تحديات هائلة في تأمين الأنظمة المترابطة وتحسين البيئات ذات الموارد المحدودة. تطور هذه الأطروحة وتتحقق من صحة أطر عمل جديدة لتعلم الآلة والتعلم المعزز (RL) لمعالجة نقاط الضعف الحرجة في الأمن السيبراني لإنترنت الأشياء وتعزيز الاستدامة في الزراعة في المناطق القاحلة. على وجه التحديد، تقدم الأطروحة (1) نظامًا قائمًا على تعلم الآلة يستخدم فك تشابك الميزات والتدريب العدائي للكشف القوي عن البرامج الضارة المخفية؛ (2) نظامًا تكيفيًا للكشف عن التسلل قائمًا على التعلم المعزز لشبكات إنترنت الأشياء العامة، مع امتداد متخصص يعطي الأولوية للسلامة للأنظمة البيئية لإنترنت الأشياء الطبية؛ (3) بيئة تعلم معزز مخصصة تحاكي الزراعة في المناطق القاحلة، مما يتيح تطوير سياسات ري تكيفية تعمل على تحسين الحفاظ على المياه ومحصول المحاصيل؛ و (4) منهجية هجينة تجمع بين التعلم المعزز والتجميع تعزز تمثيل الحالة وتعميم السياسة في مهام التحكم المستمر المعقدة وغير الثابتة. من خلال توحيد التطورات النظرية مع التحقق التجريبي الدقيق، يوضح هذا البحث الإمكانات التآزرية لتعلم الآلة والتعلم المعزز لتعزيز أمن إنترنت الأشياء وتشجيع الممارسات الزراعية المستدامة. تقدم الحلول المقترحة أطرًا قابلة للتطوير ومرنة للأنظمة الذكية التي تعمل في بيئات ديناميكية ومحدودة الموارد، مما يساهم في مستقبل مترابط رقميًا ومستدام بيئيًا.

Contents

List of Figures

List of Tables

General Introduction

The accelerating digital transformation of critical sectors, from healthcare to agriculture, has ushered in an era of unprecedented efficiency, yet it has also exposed these domains to acute and evolving vulnerabilities. Concurrently, the proliferation of Internet of Things (IoT) devices, particularly within sensitive domains like healthcare and resource-scarce agricultural settings, demands adaptive cybersecurity and intelligent resource management frameworks. In the realm of cybersecurity, the rise of sophisticated, obfuscated malware, which leverages techniques like encryption, polymorphism, and metamorphism to evade traditional signature-based detection, presents a persistent and critical threat to data integrity and system security. Simultaneously, the proliferation of IoT devices has created a vast new attack surface. This is particularly concerning in sensitive domains such as healthcare, where Medical IoT devices monitor and deliver critical patient care, and in resource-scarce agricultural settings, where IoT networks manage vital infrastructure. These interconnected systems demand cybersecurity frameworks that are not only robust but also adaptive to novel threats and operational constraints.

Parallel to these security imperatives, global challenges like climate change and resource scarcity demand innovative technological solutions for sustainability. This is especially true in arid regions, where the judicious management of water is paramount for food security and ecological stability. Traditional agricultural optimization methods often struggle to cope with the inherent stochasticity of environmental conditions and the non-linear dynamics of crop growth. This highlights an urgent need for intelligent, data-driven strategies that can optimize resource allocation in real-time, balancing crop yield with conservation.

This dissertation stands at the confluence of these two critical areas, cybersecurity and sustainable agriculture, investigating how advanced Machine Learning (ML) and Reinforcement Learning (RL) techniques can be engineered to address these distinct yet related challenges. ML and RL are uniquely suited for these problems due to their advanced capabilities in pattern recognition and sequential decision-making under uncertainty.

By bridging these interdisciplinary domains, this research aims to develop resilient, adaptive, and intelligent systems and applications for a world that is increasingly reliant on digital interconnectivity yet fundamentally constrained by ecological limits. The following pivotal research questions have guided this investigation:

- How can ML models be tailored to detect and classify obfuscated malware with enhanced accuracy and robustness against adversarial evasion?
- Can RL-driven frameworks dynamically improve threat detection and response in IoT ecosystems, particularly within high-stakes environments like Medical IoT, while prioritizing critical operational constraints?

- What novel RL-based strategies can optimize water allocation and crop yield in arid agricultural regions under climatic uncertainty and resource limitations?
- How can the integration of clustering techniques with RL enhance state representation and policy adaptability in complex, continuous, and non-stationary control tasks?

Answering these questions has led to the development of the following primary contributions to the fields of cybersecurity and sustainable agriculture:

- A novel ML framework for detecting obfuscated malware, leveraging feature disentanglement and adversarial training to significantly improve robustness against evasion tactics.
- An RL-based adaptive intrusion detection system for IoT networks, featuring a specialized extension for Medical IoT devices that prioritizes low-latency responses and patient safety.
- A customizable RL environment that accurately simulates arid region agriculture, facilitating the development and validation of dynamic water management policies under realistic resource constraints.
- A hybrid RL-clustering methodology designed to enhance state representation and accelerate policy generalization in continuous control tasks, demonstrating improved performance in non-stationary scenarios.

The remainder of this thesis is organized as follows: Chapter 2 reviews foundational concepts in ML, RL, and their applications in cybersecurity and agriculture. Chapter 3 presents the ML framework for obfuscated malware detection, addressing adversarial evasion. Chapter 4 introduces the RL-driven security protocol for IoT systems, with a case study on Medical IoT. Chapter 5 details the RL environment for arid agriculture, optimizing irrigation strategies using real-world climate data. Chapter 6 proposes the integration of clustering with RL to improve task decomposition in continuous domains. Chapter 7 concludes the thesis, summarizing key insights and outlining future research directions.

Chapter 1

Background and Literature Review

This chapter provides context and positions our work within existing research.

1.1 Machine Learning

ML is a subfield of artificial intelligence that focuses on developing algorithms capable of learning from data and making predictions or decisions without being explicitly programmed. It involves constructing models that can generalize from observed data to unseen situations, enabling systems to improve their performance over time [1]. ML methodologies are typically categorized into four primary types [2, 3], as shown in Fig.1.1 :

- Supervised Learning: Algorithms are trained on labeled datasets, meaning each training example is paired with an output label. The model learns to predict the output from the input data. This approach is commonly used for tasks like classification and regression.
- Unsupervised Learning: Algorithms work with unlabeled data and seek to identify patterns or structures within the data without predefined labels. Techniques such as clustering and dimensionality reduction fall under this category.
- Semi-Supervised Learning: This approach combines supervised and unsupervised learning by utilizing datasets that contain both labeled and unlabeled data. It aims to improve learning accuracy when acquiring a fully labeled dataset is challenging.
- Reinforcement Learning: Algorithms learn to make decisions by performing certain actions and receiving feedback in the form of rewards or penalties, aiming to maximize cumulative rewards over time. This method is often applied in areas like game playing and robotics.

1.1.1 Machine Learning in Cybersecurity

Malware continues to pose significant threats to computer networks and digital systems, necessitating the development of robust malware detection methods. Adversarial examples, which are carefully crafted inputs, can evade ML models, making

malware detection a challenging task. In this literature review, we explore existing research that addresses the effectiveness of ML and DL techniques in detecting adversarial examples and various types of malware, including obfuscated memory malware (OMM).

Detecting adversarial examples in the context of malware has garnered considerable attention. In [4], the authors propose an expanded approach to crafting adversarial examples for malware detection models. They overcome key challenges in applying existing algorithms to malware detection by operating in discrete and binary input domains. The evaluation of an NN for malware detection demonstrates the effectiveness of adversarial examples in misleading the classifier.

The study [5] focuses on robust malware detection using adversarial examples. The study combines CNN and bidirectional LSTM to create a hybrid model suitable for detecting OMM. The proposed architecture exhibits compact size and fast processing speed, making it a viable option for deployment in resource-constrained Internet of Things (IoT) devices. Researchers have also explored the vulnerability of DL methods in malware detection. [6] introduces a gradient-based attack capable of evading Deep Neural Networks DNN used for malware detection. Despite changing only a few bytes in each malware sample, the adversarial malware examples significantly evade the targeted network.

Malware detection from raw byte sequences has emerged as a promising research area, as seen in [7]. The authors address challenges associated with processing data at scale and present a solution with linear complexity dependence on sequence length. Their approach allows for the identification of interpretable sub-regions of the binary, enhancing the understanding of malware behavior.

Language modeling for malware detection is explored in [8], where echo state networks and Recurrent Neural Networks (RNN) are used to extract robust, time domain features. The hybrid model performs better than traditional models, improving the true positive rate with a low false positive rate. ML techniques have been widely used in malware detection, as demonstrated in [9]. The study evaluates various clustering and classification algorithms on two separate datasets and achieves high accuracy scores. The authors emphasize the potential of ML, particularly DL, to enhance zero-day malware detection.

Hybrid models combining ML and DL have shown promise in network intrusion detection [10]. Integrating SMOTE for data balancing and XGBoost for feature selection, the proposed method achieves high detection rates while ensuring model dependability. In [11], the authors present a lightweight malware detection method capable of identifying recent malware and suitable for execution in embedded de-

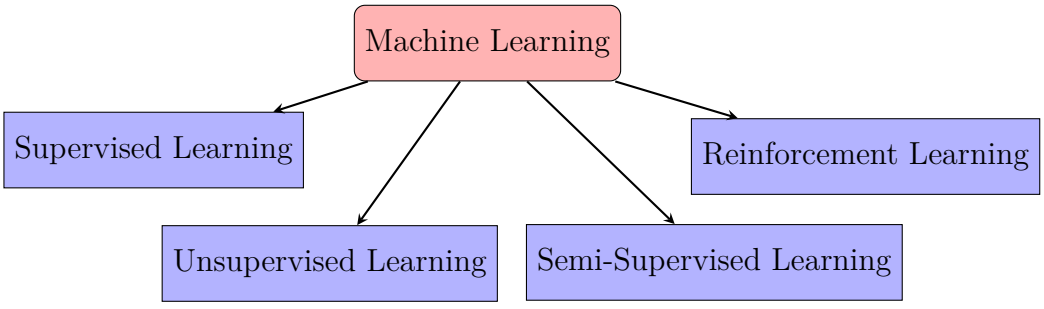


Figure 1.1: Machine Learning Types

vices. Their approach combines CNN and bidirectional long short-term memory (BiLSTM), offering a compact model executable in IoT devices for defending against OMM.

Finally, [12] explores the effectiveness of ML in malware detection using supervised and unsupervised learning procedures. The authors compare clustering and classification algorithms on different datasets, highlighting the high accuracy scores achieved and the potential for further improvements.

Overall, the background highlights the ongoing efforts to develop efficient and accurate malware detection models by leveraging various ML and deep learning techniques. The studies demonstrate the importance of addressing adversarial examples and OMM to enhance the security of digital systems and networks [13, 14].

1.2 Reinforcement Learning

RL is a paradigm within machine learning where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. Unlike supervised learning, RL does not rely on labeled data; instead, it focuses on learning optimal behaviors through trial and error, receiving feedback as rewards or penalties based on the actions taken. This approach enables the agent to autonomously discover strategies that yield the highest long-term benefits[15].

In RL, the decision-making process is often modeled as a Markov Decision Process (MDP) as illustrated in Fig.1.2, which provides a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. The agent observes the current state of the environment, selects an action, transitions to a new state, and receives a corresponding reward. Over time, by exploring various actions and their consequences, the agent aims to learn a policy. A policy is a mapping from states to actions that maximizes the expected cumulative reward [16].

1.2.1 Reinforcement Learning in IoT Security

The advent of RL has catalyzed transformative changes across several domains, notably within cybersecurity, network security, and cognitive communication systems. This section meticulously synthesizes seminal works, underscoring RL's pivotal role in fortifying security mechanisms against escalating cyber threats. Firstly, delving into integrating ML and DL techniques for enhancing IoT security [17] addressing the complex cyber threats faced by the heterogeneous and resource-constrained IoT ecosystem. It emphasizes ML/DL's potential to transform IoT security into proactive systems through intelligent monitoring and prediction of breaches. While highlighting the effectiveness of various ML/DL algorithms in improving IoT defense mechanisms, it also notes challenges such as computational complexity and the risk of malicious exploitation of these technologies.

The discourse extends to RL's instrumental role in autonomous cyber defense within Software-Defined Networking [18], emphasizing its adaptability against different cyber-attack strategies, including those that tamper with the RL training process. It explores techniques like Deep Double Q-Networks (DDQN) and Asynchronous Advantage Actor-Critic (A3C) algorithms, showcasing their efficacy in

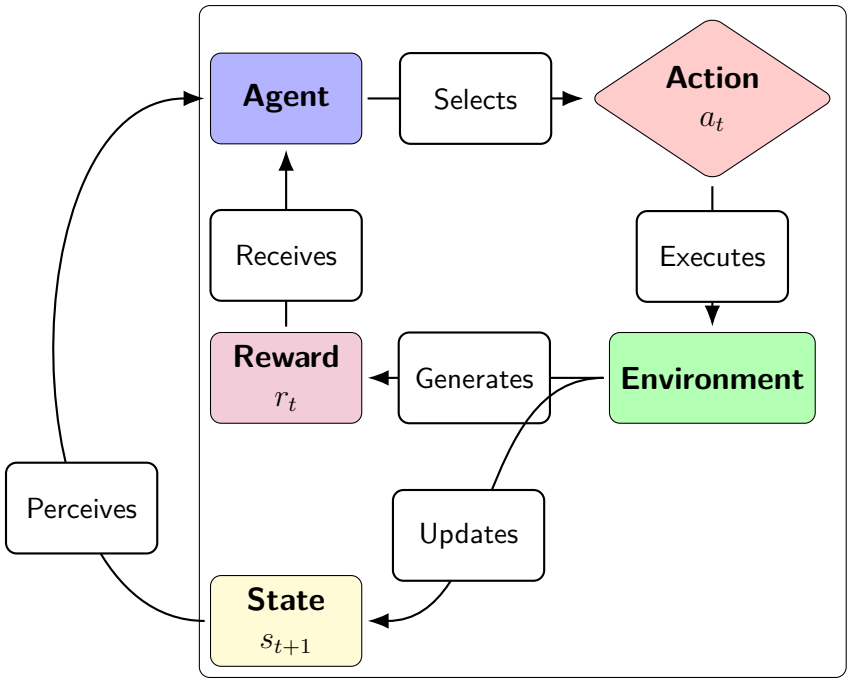


Figure 1.2: Markov Decision Process [16]

safeguarding network servers through experiments. Challenges such as RL's application complexity in real settings and vulnerability to exploitation are discussed alongside countermeasures like adversarial training to enhance security. Applying Deep Reinforcement Learning (DRL) in communications and networking [19], focusing on optimizing network functions like access, data rate control, and security. It discusses the challenges of traditional RL in complex networks and highlights DRL's potential in next-generation networks for improving performance, autonomy, and security. Various DRL techniques, including Q-learning extensions and advanced models like DDQN and A3C, are explored. While demonstrating DRL's benefits in network optimization, it also addresses limitations such as computational demands and the extensive training data requirement. Exploring feedback-enabled cyber resilience mechanisms [20] highlights the inadequacy of traditional security methods. It emphasizes real-time adaptation to threats using feedback architectures and RL, targeting various vulnerabilities with applications in defensive strategies and technologies. The discussion includes RL algorithm vulnerabilities and potential attack models, concluding with future challenges and the prospective utility of RL-based CRMs in enhancing cybersecurity resilience.

Furthermore, examining Multi-Agent Reinforcement Learning (MARL) [21] enhances future Internet technologies, focusing on network functionalities such as access, power control, and security. It discusses MARL's ability to adapt to complex, dynamic network environments through advanced algorithms and models, such as Centralized Training with Decentralized Execution. Despite showcasing MARL's potential in improving network efficiency and performance, it also addresses challenges like computational complexity and the practical application of these theoretical models in real-world settings. The significant impact of DRL on cybersecurity threat detection and protection [22], emphasizing its potential to revolutionize threat detection and protection through adaptive, intelligent responses to complex cyber threats. It discusses the effectiveness of various DRL algorithms in cybersecurity

applications, acknowledging the challenges in applying these advanced AI-based solutions in the intricate landscape of real-world cybersecurity and the risk of exploitation by attackers. The paper [23] showcases substantial advancements in improving intrusion detection, comparing its effectiveness against traditional methods using the NSL-KDD and AWID datasets. It highlights DRL's superior detection speed and accuracy, particularly noting the standout performance of the DDQN model. However, it also points out the challenges of computational complexity and the need to adapt DRL techniques to fit supervised learning scenarios specific to intrusion detection. The exploration of the application of DRL in network security presented by [24] reaffirms AI's transformative potential in cultivating resilient cybersecurity defenses.

Identifying gaps in the literature, this section underscores the imperative for more empirical research focused on the practical implementation of these technologies in real-world scenarios. Despite considerable progress, challenges such as computational complexity, the need for extensive datasets, and the susceptibility of ML models to adversarial attacks remain. This work contributes to a pioneering IoT security environment that simulates smart home device patterns using Microsoft CyberBattleSim, addressing these gaps. It advances our understanding of IoT security challenges. It bridges the gap between theoretical research and practical applications, offering a unique platform for testing and refining security strategies in real-world scenarios.

1.2.2 Reinforcement Learning in Medical IoT Security

The Internet of Medical Things (IoMT) revolutionizes healthcare with remote monitoring and data management. However, it brings cybersecurity risks. This review explores HIoT cybersecurity challenges and highlights the role of RL in addressing them across IoT, cybersecurity, and healthcare. The IoMT's emergence introduces remote sensing and elder care services, improving patient care and cost efficiency. However, securing healthcare data is crucial due to its sensitivity. [25] stresses robust security for patient privacy and data integrity. Acknowledging IoMT device vulnerabilities, this part underscores the need for a comprehensive security framework to tackle current and future challenges. [26] criticizes traditional risk assessment methods, advocating for Dynamic Risk Assessments (DRA) to manage evolving cyber threats effectively. [27] conducts a systematic review of IoMT security vulnerabilities, emphasizing access control, encryption, and multifactor authentication for securing medical data and IoMT devices.

RL in cybersecurity offers a promising avenue for bolstering threat detection and prevention. [28] introduces a Deep Q-Learning model for intrusion detection, showcasing its efficacy across various cyber-attacks. [29] employs SARSA RL to uncover vulnerabilities in cyber-physical systems, enhancing cybersecurity strategies. Furthermore, [30] proposes a method for generating synthetic attack samples, enhancing machine learning datasets for stronger defenses. RL finds innovative applications in healthcare, improving treatment recommendations and patient care. [31] explores factored action spaces in RL to enhance decision-making efficiency, crucial for accurate treatment recommendations, especially with limited data. [32] explores into treatment prescription and chronic disease management, underscoring RL's potential in optimizing patient care and decision-making while addressing

future research challenges and opportunities.

Integrating RL with IoT provides fresh insights into tackling complex issues across domains. [33] reviews federated RL in IoT, revealing current solutions and research gaps, emphasizing their potential for secure and efficient IoT solutions. [34] presents a DRL-based approach for IoT device identification, showcasing its efficiency and accuracy in bolstering IoT security through reliable device authentication. The fusion of RL, IoT, and healthcare demands attention to cybersecurity challenges. [35] proposes a blockchain and DRL-based framework, demonstrating their efficacy in addressing security, privacy, and operational issues in HIoT. [36] showcases DRL's role in intelligent decision-making by analyzing patient data for precise treatment recommendations, highlighting the potential for improved healthcare delivery. Additionally, [37] presents an RL-based crowdsourcing framework for healthcare IoT, particularly relevant during pandemics like COVID-19. It proves RL's applicability in ensuring data reliability, supporting personalized and preventive healthcare services.

In summary, IoT paired with RL in healthcare enhances patient care, cybersecurity, and efficiency, but ongoing research remains crucial. RL strengthens HIoT cybersecurity by improving intrusion detection, security analysis, malware generation, data management, and treatment recommendations. However, a gap persists in tailoring RL-based cybersecurity for HIoT, like CyberBattleSim [38], to tackle unique threats, integrate healthcare protocols, and ensure scalability and realism. Developing a customized RL environment akin to CyberBattleSim [38] presents a significant research opportunity, enabling the simulation of HIoT attack and defense scenarios to refine cybersecurity strategies for enhanced security and resilience.

1.2.3 Reinforcement Learning in Agriculture

Smart agriculture has significantly transformed by adopting technologies like Expert Systems, ML, DL, and RL. For instance, the integration of IoT with expert systems has been leveraged to address challenges in cotton farming in Pakistan [39], emphasizing the critical role of farmer education in technology adoption. Studies involving ML, such as those by [40], [41], [42], and [43] have illustrated that the combination of IoT and ML can substantially enhance water management systems. However, these technologies have limitations, particularly in data availability and considering diverse environmental parameters. Furthermore, advancements in DL, as highlighted in research by Saravi [44], and Kumar's[45], are pushing the boundaries of predictive analytics in agriculture, aiding in more accurate and efficient decision-making. The exploration of RL in smart agriculture, as investigated in the works of [46] and [47], has shown potential for advanced system optimization. These studies reveal RL's capability to enhance the efficiency of irrigation systems, though they also underscore the challenges of scalability, realism, and adaptability in varied agricultural settings. However, challenges such as data acquisition limitations, environmental variability, and technological literacy remain prevalent. The development of RL environments has been pivotal in addressing complex challenges such as those mentioned before across various vital domains beyond smart agriculture Fig.1.3. For instance, in *robotics*, RL has been employed to improve UAV flight control systems, with algorithms like DDPG and PPO showing promise over traditional controls [48]. In the *energy sector*, RL applications in electric motor control have been explored,

with tools like the gym-electric-motor (GEM) and PowerGym facilitating the training of RL agents for efficient energy management [49][50]. Also, In *e-commerce*, RL environments like RecoGym have been developed to enhance product recommendations, showcasing the potential of RL over traditional ML approaches [51]. The Unity platform and other studies have revolutionized AI research, offering a versatile environment for complex simulations, further proving RL's adaptability [52][53][54][55]. Furthermore, *Microgrid management* has also benefited from RL, with the OpenModelica Microgrid Gym (OMG) toolbox optimizing control for energy availability and safety [56]. In *NLP*, the NLPGym toolkit has opened new avenues for evaluating RL algorithms in tasks like sequence tagging and question answering.[57]. More studies including Mobile communication [58], Autonomous driving [59], and job Scheduling [60]. These advancements in RL environments illustrate the technology's broad applicability and potential in addressing diverse challenges. However, they also highlight common issues, such as the need for scalable solutions and real-world applicability. This broad spectrum of applications sets a foundation for exploring RL's potential in uncharted areas.In the realm of smart agriculture, RL environments like CropGym [61] and gym-DSSAT [62] have shown promising results in optimizing fertilization strategies and integrating high-fidelity crop simulators. These tools represent significant progress in sustainable agricultural practices. However, a notable gap remains the lack of specialized RL environments for irrigation in arid regions, a critical area given the increasing water scarcity. Future research in RL environments for agriculture should prioritize developing sophisticated designs that accurately reflect real-world conditions, including soil diversity and climatic variations. The focus should be on scalable solutions and efficient algorithms for managing extensive irrigation systems. Emphasizing transfer learning and generalization, RL agents must be equipped to adapt to new environments and crop types seamlessly. The primary goal is effectively bridging the gap between simulated environments and practical, real-world irrigation control, particularly in arid areas, addressing a critical yet unexplored field challenge.

1.2.4 Reinforcement Learning & complex environments

RL has made significant strides in recent years, with PPO emerging as one of the most effective policy optimization algorithms. However, challenges remain inefficiently learning in high-dimensional and complex environments. To tackle these challenges, research aims to make RL more tractable through methods like state space simplification, representation learning, clustering, and dimensionality reduction. This literature review highlights the relevant studies in these areas and positions the contribution of clustering within the PPO framework as a novel approach to state space simplification.

State abstraction in RL is a critical methodology that enhances the efficiency of learning algorithms by reducing the complexity of the search space [63]. State abstraction in RL focuses on simplifying the decision-making process by grouping similar states into a reduced set of representative states [64]. The study [64] provided a foundational theory of state abstraction for Markov Decision Processes (MDPs), introducing various abstraction techniques like state aggregation and homomorphisms. These techniques aim to reduce the state space size without sacrificing optimality in policy decisions. Abel, Hershkowitz, and Littman [65] extended this work by in-

troducing approximate state abstraction, allowing near-optimal behavior even when exact abstractions are not feasible. These methods laid the groundwork for state simplification in RL, especially in environments where state spaces are large and continuous.

Another approach is representation learning in RL which focuses on discovering efficient and compact state representations to make the learning process more efficient [66]. The study [67] introduced the concept of using auxiliary tasks in RL to learn better state representations. By adding self-supervised auxiliary objectives, the agent can focus on salient features in the environment, improving policy performance. Similarly, Oord, [68] proposed contrastive predictive coding (CPC) for representation learning, leveraging contrastive learning to model state dynamics more effectively.

Furthermore, clustering has long been considered a practicable method for abstracting state spaces in RL. The study [69] introduced dynamic abstraction via clustering, where state-action pairs are grouped dynamically based on their similarity. This approach reduces the number of decisions the agent must make, simplifying the learning process. Later on, extended by [70] using clustering to learn symmetries in the environment, which helps in function approximation and reduces computational complexity.

Other studies presented dimensionality reduction techniques such as Principal Component Analysis (PCA) [71] and autoencoders [72] have been used to simplify state spaces in RL. The study [71] demonstrated how PCA could reduce the dimensionality of state spaces, leading to more efficient policy learning. Meanwhile, [72] explored the use of sparse coding in autoencoders for state representation, showing that autoencoders can compress the state space while retaining important features for decision-making.

The early work on state abstraction in RL, particularly by Li et al. [64] and

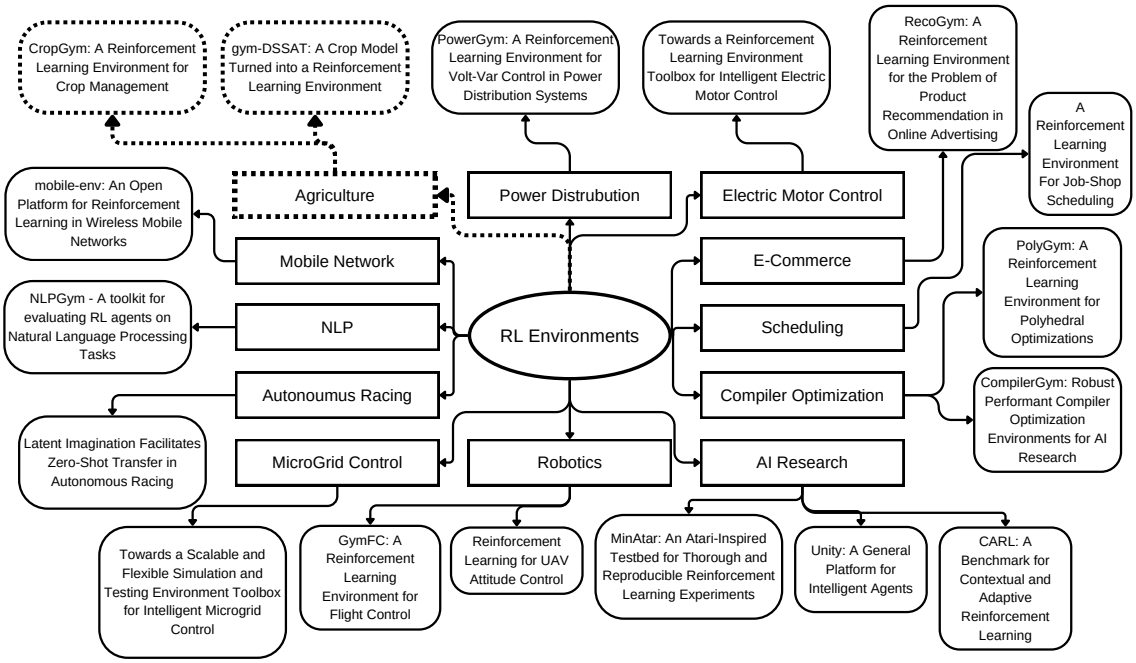


Figure 1.3: RL Environments usage for different domains

Mannor et al. [69], provided foundational methods for reducing the complexity of state spaces through aggregation and clustering. These methods focused on static abstractions, where the structure of the state space was predetermined. However, recent advancements in representation learning, such as the introduction of contrastive learning [68] and auxiliary tasks [67], have shifted focus toward learning representations dynamically during training. This shift opens up opportunities for combining unsupervised learning methods like clustering with policy optimization techniques like PPO to create adaptive and efficient state representations. While PPO is widely regarded as an effective policy optimization algorithm, much of the research on improving PPO has focused on the policy optimization process itself, with less attention on the role of state space representation. The proposed approach fills this gap by introducing clustering as a way to simplify the state space for PPO. By clustering state-action pairs, the agent is presented with a more structured and simplified state space, which helps reduce the complexity of the learning task. This approach also complements PPO's strength in handling continuous control problems by reducing the dimensionality and variability of the input space.

Chapter 2

Machine Learning for Obfuscated Malware Detection

2.1 Introduction

In cybersecurity, the continuous evolution of intelligent malicious software poses a significant threat to individuals and organizations worldwide. Malware, malicious software designed to compromise computer systems, has become increasingly sophisticated in evading detection and extermination. One particularly challenging category of malware is obfuscated malware, which utilizes techniques to conceal its true nature and avoid detection by traditional security measures [73, 74]. Examples of obfuscated malware include spyware, ransomware, and Trojan horse malware.

Developing robust and effective detection mechanisms is imperative to combat the rising threat of obfuscated malware. In recent years, ML and DL techniques have emerged as promising approaches for detecting and classifying malware. These methods leverage the power of artificial intelligence to analyze vast amounts of data and identify patterns indicative of malicious behavior.

In this study, our objective is to exploit different ML/DL models tailored explicitly for detecting obfuscated malware. The model's primary focus will be distinguishing between malicious and benign memory dumps, as memory analysis plays a crucial role in detecting advanced malware.

2.2 Problem Statement

In cybersecurity, the continuous evolution of intelligent malicious software poses a significant threat to individuals and organizations worldwide. Malware—malicious software designed to compromise computer systems—has become increasingly sophisticated in evading detection and removal. One particularly challenging category of malware is **obfuscated malware**, which utilizes techniques to conceal its true nature and avoid detection by traditional security measures [74]. Examples of obfuscated malware include spyware, ransomware, and Trojan horse malware. Obfuscated malware poses a significant challenge because it can skillfully disguise itself, making it difficult for conventional security tools to detect and classify. Memory analysis has emerged as a crucial approach in identifying advanced malware. By examining memory dumps, security professionals can analyze the runtime behavior and in-

memory software artifacts, often revealing malicious activities that static analysis might miss.

The primary problem addressed in this chapter is the need to effectively distinguish between malicious and benign memory dumps to detect obfuscated malware. The goal is to develop robust machine learning (ML) and deep learning (DL) models that can accurately identify obfuscated malware, thereby enhancing our ability to counter various threats, including spyware, ransomware, and Trojans.

2.3 Approach

To address the problem of detecting obfuscated malware, we explored various ML and DL techniques tailored explicitly for this purpose. The approach involves several key components:

2.3.1 Data Collection and Preprocessing

The obfuscated malware dataset called "CIC-MalMem-2022" is designed to test obfuscated malware detection methods through memory. The dataset was created to represent as close to a real-world situation as possible using malware that is prevalent in the real world. Made up of Spyware, Ransomware, and Trojan Horse malware, it provides a balanced dataset that can be used to test obfuscated malware detection systems. This dataset uses debug mode for the memory dump process to avoid the dumping process from showing up in the memory dumps. The dataset contains a total of 58,596 records with 29,298 benign and 29,298 malicious. This works to represent a more accurate example of what an average user would have running at the time of a malware attack [75]. The dataset was subjected to a comprehensive data treatment process to prepare it for training. This involved:

- Loading the dataset using `pandas` for efficient handling of tabular data.
- Data cleaning to manage missing values, duplicates, and irrelevant features. Three columns with non-unique values were identified and removed.
- Processing the "Category" column to ensure correct labeling of the output classes.

2.3.2 Feature Extraction

Feature extraction was crucial in converting raw memory dumps into interpretable numerical features suitable for ML/DL models as shown in Fig.2.2. We employed dimensionality reduction techniques to enhance the efficiency of the models: **Principal Component Analysis (PCA)**: Used to reduce the dimensionality of the data while retaining most of the variance. **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: Applied to visualize high-dimensional data in a lower-dimensional space, revealing distinct patterns and improving class separation.

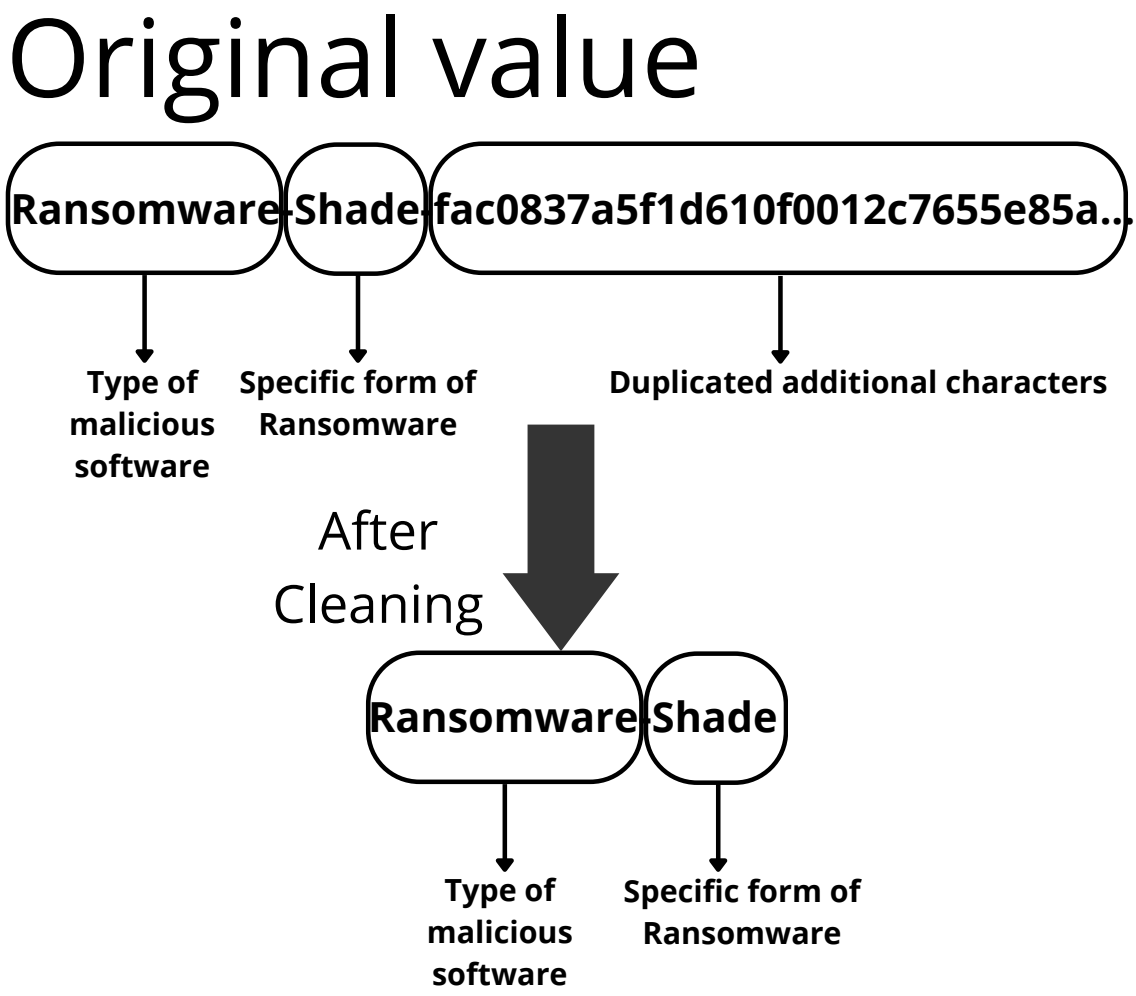


Figure 2.1: Category Column Cleaning

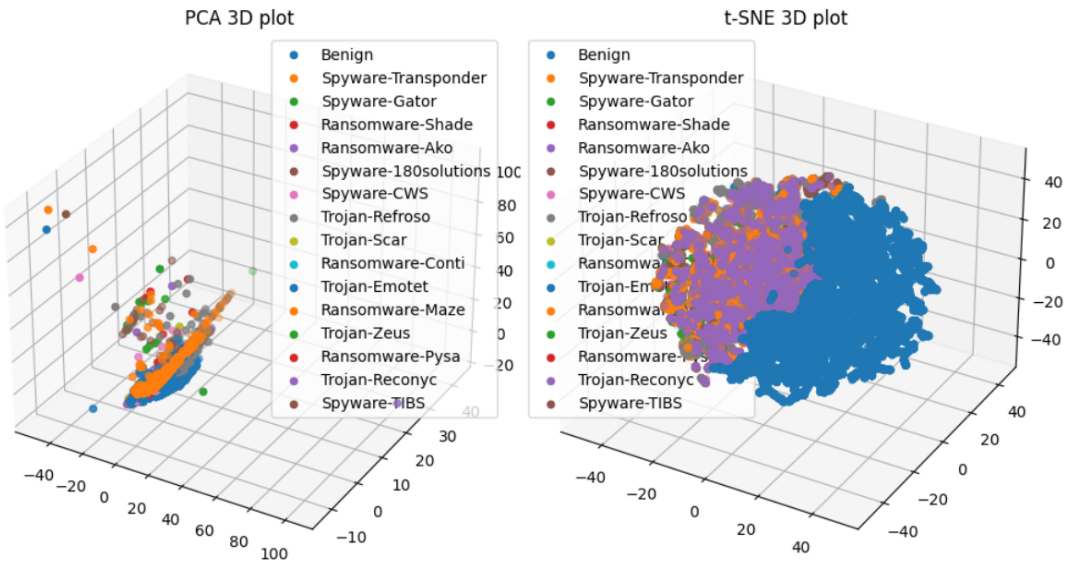


Figure 2.2: Feature extraction Scatter Plot

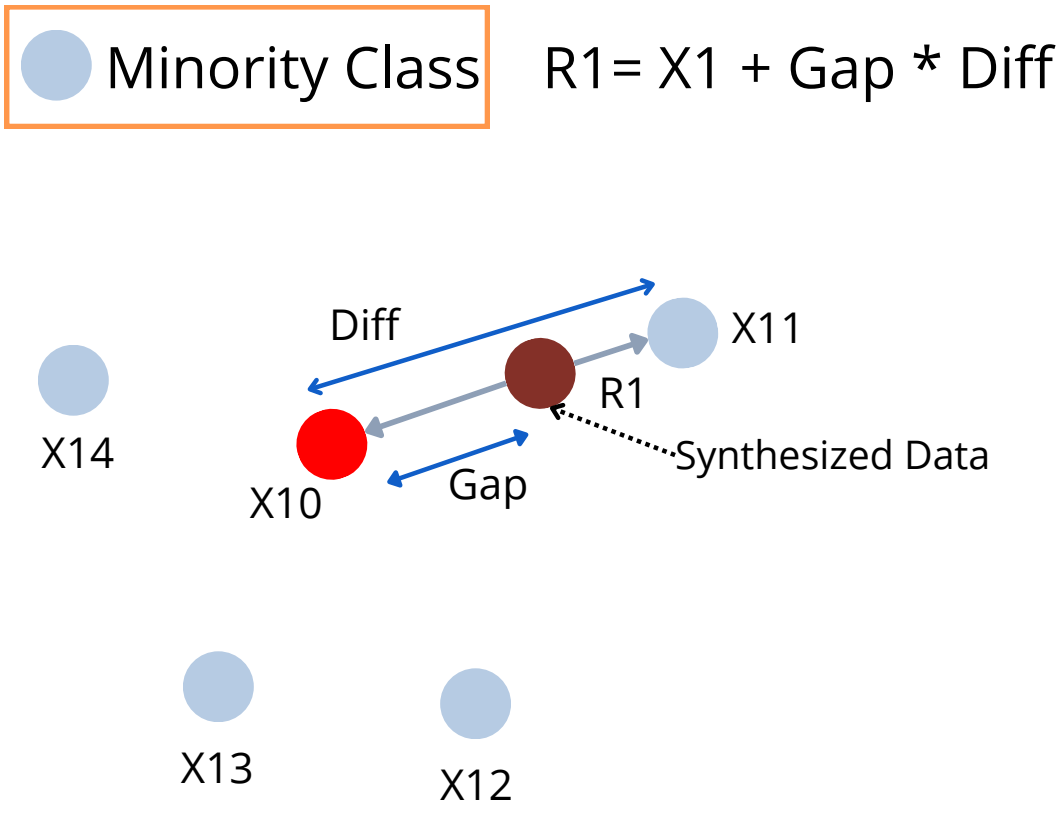


Figure 2.3: Synthetic Minority Over-sampling Technique

2.3.3 Handling Data Imbalance

An imbalanced distribution of classes was observed in the dataset, with the "Benign" class being predominant. To address this issue, we implemented data balancing techniques:

Synthetic Minority Over-sampling Technique (SMOTE): Oversampled minority classes by creating synthetic instances to balance the class distribution as shown in Fig.2.3.

Downsampling: Reduced the size of the majority class to match that of the minority classes as shown in Fig.2.4.

2.3.4 Model Development

We designed and implemented various ML and DL models suitable for the classification task as shwon in Fig.2.5: To address this issue, various ML approaches for overlapping multiclass classification were explored, including the Random Forest

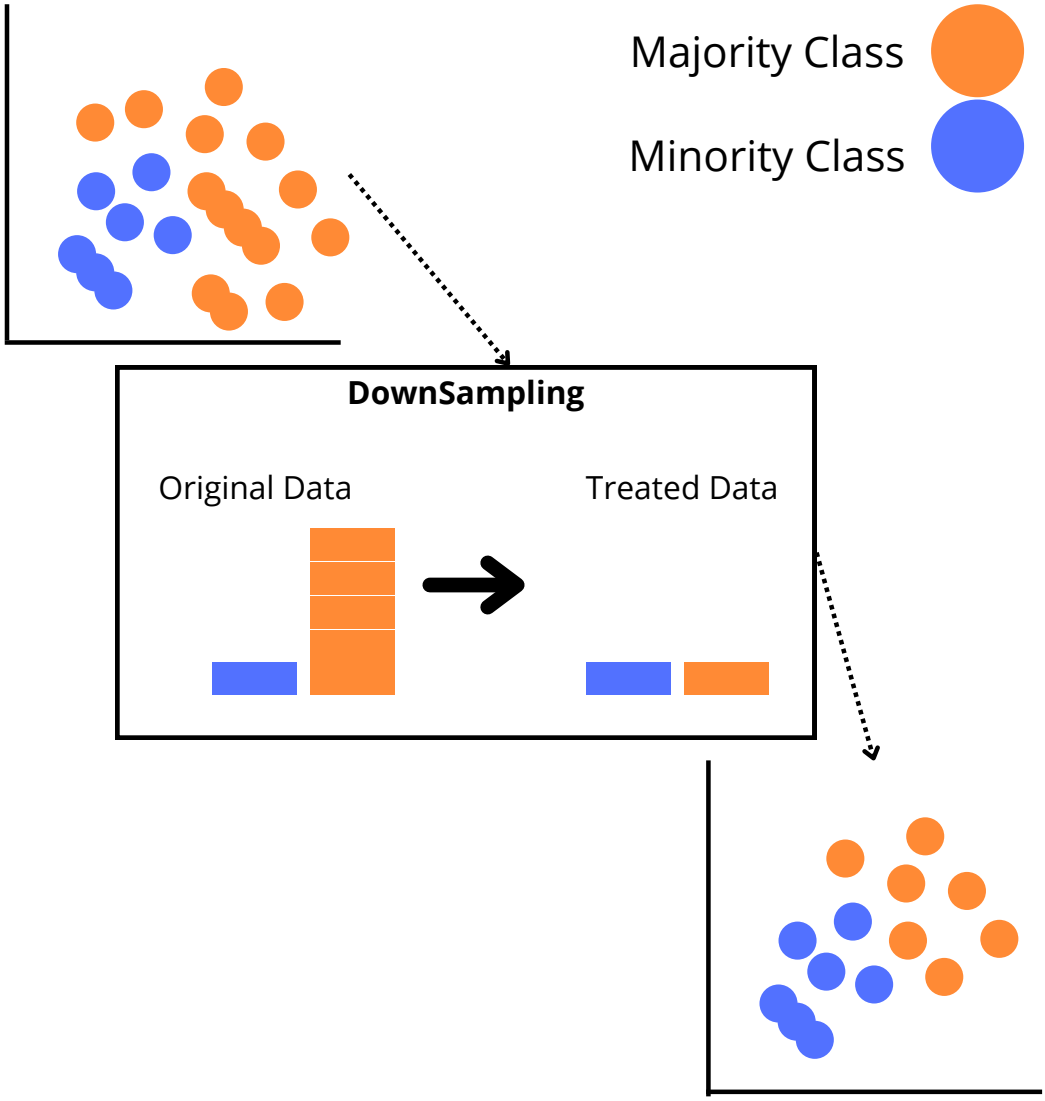


Figure 2.4: DownSampling

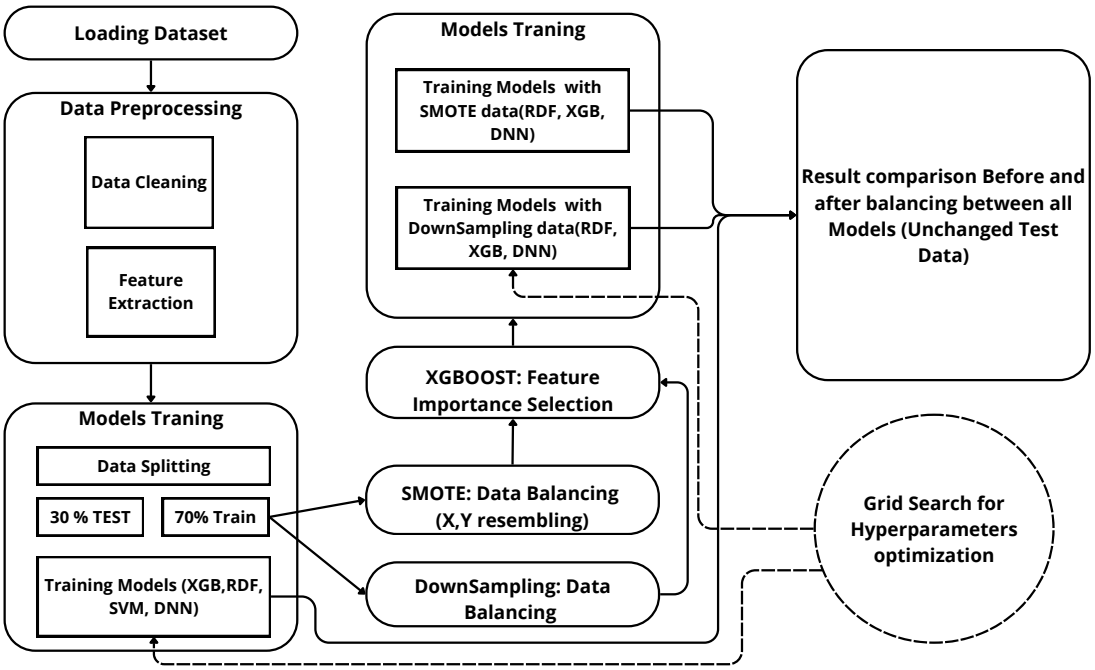


Figure 2.5: General architecture of the work flow

Classifier (RF), Support Vector Machine (SVM), and Deep Neural Networks (DNN). Initially, models such as RF and SVM were considered, with SVM later being discontinued. Subsequently, gradient boosting techniques, specifically XGBOOST, were employed coupled with grid search for hyperparameter optimization.

2.4 Experimental Setup

The experimental setup was fortified with essential tools and resources:

Development Environment: Kaggle, a cloud-based platform with GPU support, was utilized for code development, testing, and execution. **Frameworks and Libraries**: PyTorch was used for building and training the DL models. Other libraries included `pandas`, `NumPy`, and `scikit-learn`. **Data Preprocessing**: Data cleaning and feature extraction steps were implemented to ensure the dataset was suitable for training. The data preprocessing steps involved:

- Identifying and removing non-informative features.
- Cleaning the ”Category” column to ensure accurate class labels.
- Applying dimensionality reduction techniques for feature extraction.
- Implementing data balancing techniques to address class imbalance.

2.5 Results and Discussion

We evaluated the performance of the developed models on both the original unbalanced dataset and the balanced datasets obtained through SMOTE and down-

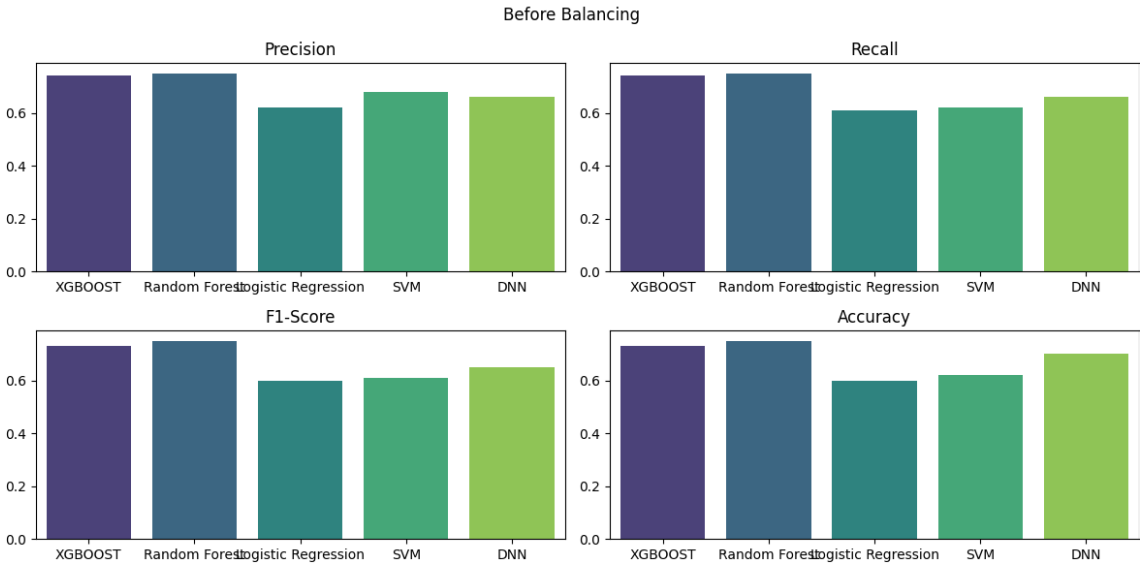


Figure 2.6: Results before performing data sampling

sampling. The evaluation metrics used included precision, recall, F1-score, and accuracy.

2.5.1 Performance on Unbalanced Dataset

On the original unbalanced dataset as shown in Fig.2.6. **Random Forest** and **XGBoost** models demonstrated robust performance with high precision and recall scores, leading to commendable overall accuracy. The **DNN** model also performed well, achieving a moderate F1-score. **Logistic Regression** and **SVM** models exhibited moderate performance, indicating potential limitations when dealing with unbalanced data.

2.5.2 Performance after SMOTE Balancing

After applying SMOTE balancing as presented in Fig.2.7 . The **Random Forest** model showed improved recall and F1-score, indicating enhanced sensitivity to minority classes. The **XGBoost** model maintained high precision but experienced a drop in recall and F1-score. The **DNN** model exhibited a substantial improvement in accuracy. These results highlight the importance of balancing techniques in improving model performance on imbalanced datasets.

2.5.3 Performance after Downsampling

After downsampling as in Fig.2.8. The **Random Forest** model showed robust performance with heightened precision and recall. The **XGBoost** model maintained high precision with a slight reduction in recall and F1-score. **Logistic Regression** maintained stable performance. The **DNN** model exhibited a decrease in precision but maintained competitive metrics.

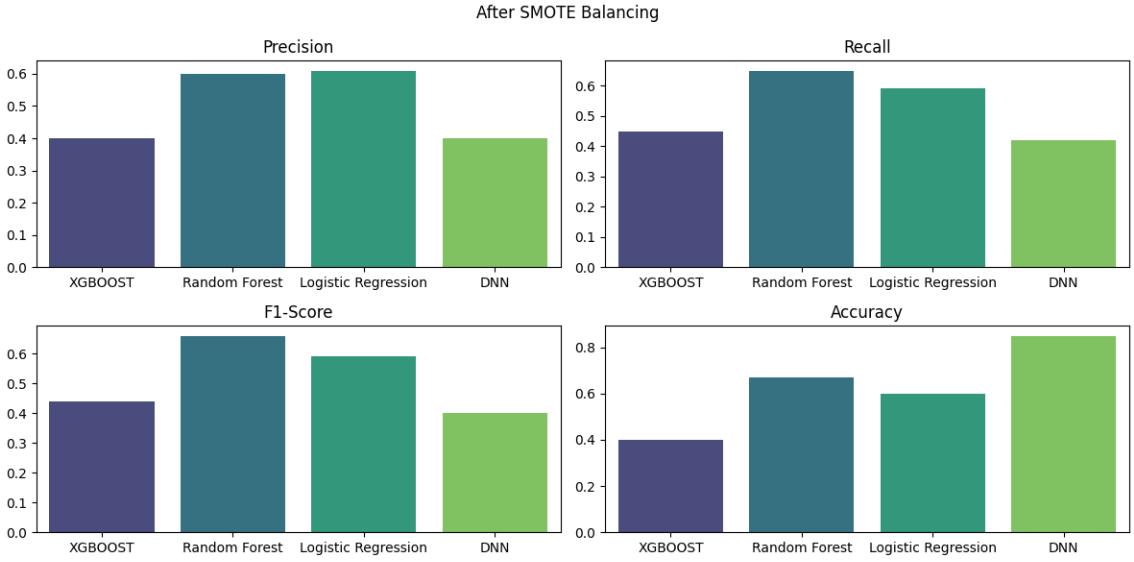


Figure 2.7: Results after applying Data Sampling (SMOTE)

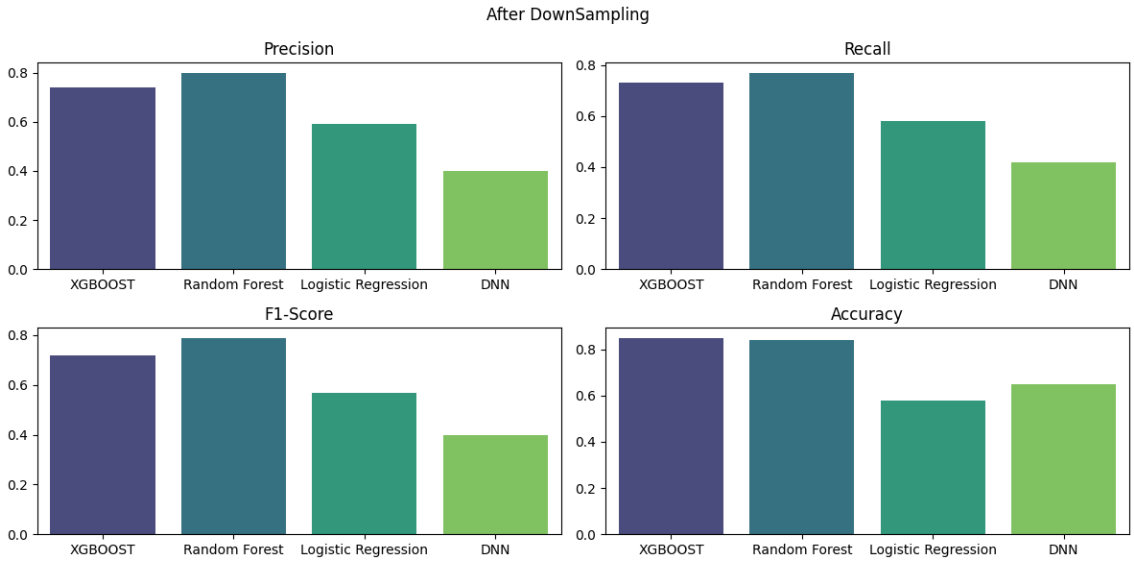


Figure 2.8: Results after applying data sampling (DownSampling)

2.5.4 Comparison of Balancing Techniques

Comparing the three approaches. Downsampling yielded the most balanced results across precision, recall, F1-score, and accuracy for most models. SMOTE balancing enhanced sensitivity but introduced variability in performance. Unbalanced datasets tended to favor precision but at the expense of recall and F1-score.

2.5.5 Addressing the Overlapping Issue

A pivotal challenge encountered was the **overlapping issue**, where discrete classes exhibited analogous features, making it difficult for models to distinguish between them. This is particularly problematic with obfuscated malware, which may share characteristics across different types due to their evasive techniques.

To address this challenges requires:

- Advanced feature engineering to capture nuanced differences between classes.
- Tailored model architectures that can better handle overlapping data.
- Potential exploration of techniques such as deep reinforcement learning and transformer models in future work.

2.6 Conclusion

This chapter presented our first contribution towards enhancing the detection of obfuscated malware using ML and DL techniques. We developed and evaluated various models, demonstrating that employing these techniques, along with appropriate data balancing strategies, can significantly improve detection accuracy in the presence of class imbalance and overlapping data.

Our findings highlight the effectiveness of Random Forest and XGBoost models, particularly when combined with downsampling techniques. The study also underscores the importance of addressing data imbalance and overlapping issues to build robust malware detection systems.

In future work, we aim to further address the overlapping challenge by exploring advanced techniques such as deep reinforcement learning and transformer models, which may offer improved capabilities in distinguishing between complex and similar patterns in data.

Chapter 3

Reinforcement Learning for IoT Cybersecurity

3.1 Introduction

The rapid integration of Internet of Things (IoT) devices into daily life and industrial operations has significantly increased cybersecurity challenges. The security of these devices has become a major concern due to their growing popularity and vulnerability to cyber-attacks and data theft [76]. The lack of stringent security measures during the development of IoT systems has led to significant vulnerabilities in both the devices and their communication protocols [77]. Additionally, the ever-evolving IoT landscape, which includes complex protocols and the continuous introduction of new devices, further complicates security efforts [78].

Challenges in IoT security include scalability, efficiency, and privacy concerns [79]. As IoT systems continue to grow exponentially, resource constraints exacerbate vulnerabilities to cyber threats [80]. Nevertheless, the ultimate objective remains to ensure seamless system operation, personal safety, and data integrity [81][82]. Recent research has suggested that integrating Artificial Intelligence (AI), particularly through reinforcement learning (RL), could significantly enhance the security of IoT ecosystems.

This chapter explores the potential of RL to strengthen IoT security by focusing on real-time threat adaptation. Specifically, it evaluates the performance of the Q-learning algorithm within the CyberBattleSim environment, a simulation that replicates real-world network security scenarios.

3.1.1 Contribution

The key contribution of this work is the development of a custom IoT network environment that mimics a smart home. The environment incorporates firewall configurations and communication services for each device. The performance of the proposed RL-based agent is evaluated against two baseline agent, Greedy and Random agent within the CyberBattleSim environment. This study bridges the gap between theoretical and practical applications by offering a comprehensive approach to securing IoT devices.

3.2 Problem Statement

The increasing usage of IoT devices has exposed significant security vulnerabilities, particularly due to their limited computational resources, lack of secure design, and constant connectivity to networks. These vulnerabilities create opportunities for attackers to exploit weak points in IoT ecosystems, leading to compromised devices, data breaches, and potential disruption of services.

Traditional security mechanisms, such as static firewalls and signature-based intrusion detection systems, are insufficient for addressing these dynamic and evolving threats in IoT environments. These systems often fail to respond to new types of attacks or adapt to emerging vulnerabilities. Furthermore, the complexity of managing large-scale IoT networks exacerbates the difficulty of ensuring comprehensive security.

The primary problem addressed in this research is the need to develop adaptive, real-time security measures capable of mitigating threats in IoT environments. Reinforcement learning, particularly Q-learning, presents a promising solution by enabling systems to dynamically learn from past interactions and adjust strategies to secure the network against potential threats.

3.3 CyberBattleSim Modifications

CyberBattleSim, an open-source platform developed by Microsoft, simulates real-world network scenarios and cyber-attack dynamics. Although CyberBattleSim is a robust tool for simulating security breaches and defense mechanisms, certain modifications were necessary to adapt it for IoT-specific environments. These modifications were focused on creating a custom network that replicates the unique challenges of IoT systems.

3.3.1 Firewall Configurations

The simulation incorporates custom firewall rules to replicate common real-world setups in IoT networks. Each device in the network is configured with distinct firewall rules for both incoming and outgoing traffic. For example:

- **Incoming Traffic**: HTTP traffic is allowed, while SSH traffic is blocked.
- **Outgoing Traffic**: Both HTTP and SSH traffic are allowed, simulating typical real-world settings where external management through SSH is permitted only for outgoing requests.

These configurations simulate a smart home network where web services are accessible from the outside, but remote management through SSH is restricted to minimize the risk of unauthorized access. The code for defining the firewall settings is shown in Algorithm 1.

As part of the modifications made to CyberBattleSim, a network visualization component has been introduced using plotly. This interactive visualization enables real-time monitoring of the network state, with nodes represented based on their current status (compromised or secure). By visualizing the network dynamically, the simulation provides more insight into how exploits and attacks spread across

Algorithm 1 Define Firewall Configurations and Vulnerabilities

Require: Network topology G(V,E), where V is set of nodes and E is set of edges and Security policy P containing allowed services and permissions

Ensure: Configured firewall rules R for each node and Vulnerability database D with associated risks

Data Structure FirewallRule
Properties: Service, Permission

Data Structure VulnerabilityInfo
Properties: Description, Type, Outcome

Data Structure LeakedCredentials
Properties: List of Credentials

Data Structure CachedCredential
Properties: Node, Port, Credential

procedure FirewallConfiguration
incoming ← List of FirewallRule
e.g., ALLOW HTTP, BLOCK SSH
outgoing ← List of FirewallRule
e.g., ALLOW HTTP, ALLOW SSH
end procedure

function DefaultVulnerabilities
Initialize a dictionary of VulnerabilityInfo
$vul[\text{"}WeakPassword\text{"}]$ ← Description, REMOTE, Outcome
Outcome includes LeakedCredentials
$vul[\text{"}UnpatchedSoftware\text{"}]$ ← Description, REMOTE, Outcome
Outcome includes UnpatchedSoftware
return *vul*
end function

different IoT devices. Each node in the IoT network is plotted in the graph, with colors representing their compromised status:

- Red for compromised nodes.
- Yellow/Orange for secure nodes.

Edges between nodes indicate communication or dependencies between devices, such as the connection between an Internet Gateway and other smart home devices like smart thermostats and PCs. The layout and interaction are handled via networkx and plotly, allowing for easy interpretation of the network's current state after each simulation step.

3.3.2 Vulnerabilities

To simulate real-world IoT threats, two key vulnerabilities were introduced:

- WeakPassword: A weak password vulnerability where attackers can exploit poorly secured devices, leading to credential leaks.
- UnpatchedSoftware: Represents unpatched software that can be remotely exploited.

These vulnerabilities are strategically placed across various devices, such as smart thermostats and PCs. The simulation allows agents to attempt exploiting these vulnerabilities to assess their success in compromising the network. This setup replicates common real-world attack scenarios in IoT environments. Moreover, any user of our simulation can define any vulnerabilities they wish with a customizable behaviour.

3.3.3 Node Representation

The IoT environment is composed of several devices (nodes), each with distinct services and firewall rules. The following nodes were defined:

- Internet Gateway: Offers HTTP and SSH services, with specific firewall configurations.
- Smart Thermostat & Smart Light: Provide only HTTP services, simulating typical smart home devices.
- PC: Provides SSH services and is more vulnerable due to its higher value and exposed SSH port.

Each node has vulnerabilities that can be exploited, leading to the exposure of sensitive information or further escalation of attacks. The nodes are customizable also and can add any connection services or specific firewall behaviour. The number of nodes also is open and can add as much the user of the environment want to add.

3.4 Methodology

The primary goal of this study is to test the effectiveness of reinforcement learning (RL) algorithms in detecting and mitigating cyber threats in an IoT network environment, as shown in Fig.3.1. The custom IoT network was constructed within the CyberBattleSim framework as illustrated in Fig.3.2 and enhanced to include realistic security vulnerabilities commonly found in IoT ecosystems.

3.4.1 Action & Observation spaces

The action space in CyberBattleSimIoT is discretely defined, encompassing the potential actions an agent can execute, including exploiting vulnerabilities across the network's nodes. Similarly, the observation space is constructed as a box space, representing each node's binary state (compromised or secure) within the network. This allows agents to comprehensively view the network's security posture at any given step.

- Let N be the number of nodes in the network.
- Let V be the number of default vulnerabilities.

Then, the action space A and the observation space O can be defined as:

$$\begin{aligned} \text{Action Space} &: A = N \times V \\ \text{Observation Space} &: O = 2N \end{aligned}$$

where:

- A represents the total possible actions, calculated as the product of the number of nodes and vulnerabilities.
- O represents the dimensionality of the observation space, calculated as twice the number of nodes to account for each node's state and an additional characteristic, with each observation being an integer value.

3.4.2 Action selection

Initially, the environment initializes all network nodes as secure, ensuring a consistent simulation starting point. The 'reset' function facilitates repeatable experimentation by restoring the environment to its default state for each run.

The core interaction within this environment occurs through the 'step' function, where agents select actions to exploit vulnerabilities in specific network nodes. The immediate outcome of these actions informs the agent via a feedback mechanism of rewards or penalties, which is critical for guiding the agent toward optimal strategies for network security or compromise.

This feedback loop is crucial for iteratively refining the agents' decision-making, enabling them to learn and adapt over time by exploring the action space and evaluating consequent environmental changes.

3.4.3 Attempting Exploits on Target Nodes

The attempt exploit function is instrumental within the CyberBattleSimIoT framework as shown in Algorithm.2, offering a sophisticated simulation of cyber attacks by evaluating the feasibility of exploiting identified vulnerabilities within target nodes. This function rigorously validates the presence of the target node and associated vulnerabilities, proceeding only when the conditions are conducive to a successful exploit. The intricate process of executing and applying an exploit, culminating in the compromise of the node, is meticulously modeled to reflect the inherent unpredictability of cyber attacks. This methodological approach not only simulates the complex landscape of cyber threats but also anchors the adaptive learning processes of agents within the environment. As a result, CyberBattleSimIoT extends beyond its initial scope as a simulation tool, evolving into an advanced experimental ecosystem. It offers a comprehensive examination of cybersecurity strategies' effectiveness, enhancing the understanding of attack vectors with a high probability of success in real-life contexts. This progression aids in the development of more sophisticated defensive mechanisms, significantly enriching the field of network security research and the cultivation of durable cybersecurity solutions within the IoT domain. The check exploit conditions method is crucial in evaluating the prerequisites for a successful exploit, considering factors such as the node's specific vulnerabilities, firewall settings, and the requisite port accessibility. This meticulous assessment ensures that exploit attempts are grounded in realism, resembling genuine attackers' strategic calculations.

Subsequently, the exploit execution method introduces a probabilistic approach to simulate exploiting a vulnerability, capturing the uncertain outcome of real-world cyber attacks. The apply exploit outcome method is activated upon a successful exploit to emulate the resultant impact, signifying the node's compromise and enacting potential further consequences. These processes enable a detailed exploration of the interplay between offensive and defensive tactics, shedding light on how vulnerabilities are exploited and their immediate effects. The simulation's conclusion is determined by the check terminal condition method, which identifies the end of a simulated attack scenario by compromising a high-value target or upon reaching the simulation's step limit as shown in Algorithm.3. This critical feature provides a framework for assessing the performance of RL algorithms across discrete episodes, offering a structured methodology for their development and evaluation.

3.5 Results and Analysis

This section comprehensively evaluated three distinct agents within the CyberBattleSim environment: a Random Agent, a Greedy Agent, and a Q-learning agent. Our objective was to assess each agent's effectiveness in navigating the environment and maximizing rewards through their decision-making strategies. The performance metrics considered for this evaluation included the average total reward, the maximum total reward achieved, the minimum total reward, and the average number of steps taken per episode. Validating the realism of our simulation environment is not necessary for the scope of this research. Our environment is constructed upon the CyberBattleSim environment, a robust simulation framework developed and verified by Microsoft's research team [38]. Given that CyberBattleSim has undergone

Algorithm 2 CyberBattleSimIoT Environment Class Methods

Require: Network state S representing current system configuration, Action space A containing all possible actions
Ensure: Updated network state S'
  Class CyberBattleSimEnv
    Initialize action and observation spaces
  function RESET
    Resets environment for a new episode
    return environment
  end function
  function STEP(action)
    Execute a step based on action
    return observation, reward, done, info
  end function
  function ATTEMPTEXPLOIT(*target_node*, *vulnerability*)
    if *target_node* $\notin$ *network.nodes* **then**
      return False, "Node *target_node* does not exist."
    end if
    target_node $\leftarrow$ *network.nodes*[*target_node*]
    vulnerabilities $\leftarrow$ *target_node*['*data*'].*vulnerabilities*
    if *vulnerability* $\notin$ *vulnerabilities* **then**
      return False, "Vulnerability *vulnerability* does not exist on node *target_node*."
    end if
    if CheckExploitConditions(*target_node*, *vulnerability*) **then**
      exploit $\leftarrow$ *Exploit*(*target_node*, *vulnerability*)
      if *exploit* **then**
        ApplyExploitOutcome(*target_node*, *vulnerability*)
        return True, "Successfully exploited *vulnerability* on node *target_node*."
      else
        return False, "Failed to exploit *vulnerability* on node *target_node*."
      end if
    else
      return False, "Exploit conditions not met."
    end if
  end function
  procedure CHECK_EXPLOIT_CONDITIONS(target_node, vulnerability)
    Check if exploit conditions are met
  end procedure

Algorithm 3 Main Simulation Loop

Require: Environment env with action space A and observation space O, Number of episodes num_episodes, and Agent policy Π for action selection

Ensure: Trained agent policy Π, Episode rewards R = r_1, r_2, ..., r_n, and Performance metrics M (e.g., success rate, average reward)

```
for episode in num_episodes do
    Reset environment
    while not done do
        Sample or select action
        Execute action
        Update agent state
    end while
end for
```

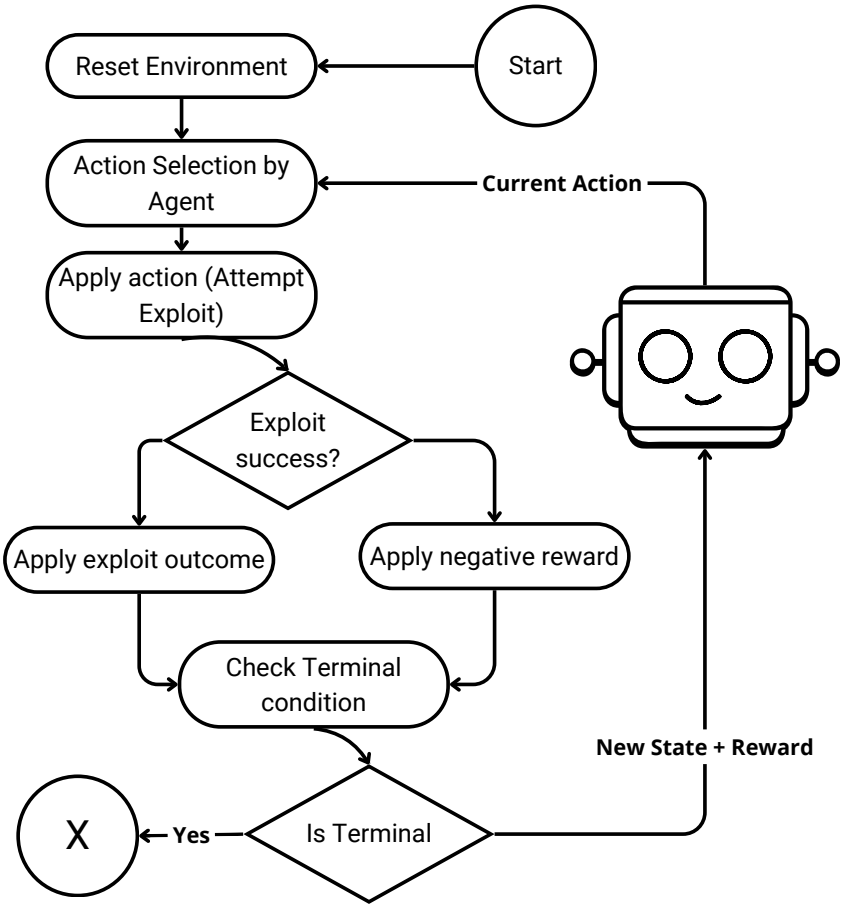


Figure 3.1: Agent action flow

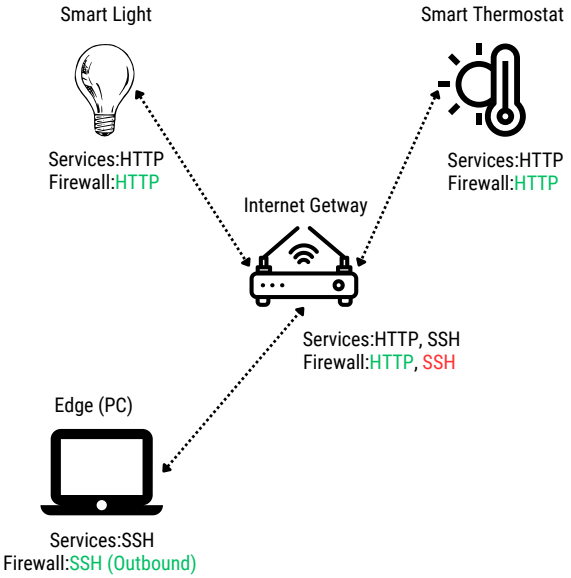


Figure 3.2: IoT Network

extensive validation by its creators to model cyber-physical attack scenarios and network dynamics accurately, we leverage this foundation to focus on the comparative analysis of agent behaviors rather than re-evaluating the environmental realism already established by Microsoft.

3.5.1 Performance Analysis

The performance analyses are summarized in Table3.1, where the Q-Learning Agent demonstrated superior performance across almost all metrics, particularly regarding average and maximum total rewards. Specifically, the Q-Learning Agent achieved an average total reward of 313.80, with a maximum total reward of 2475. This indicates the agent's consistent ability to achieve high rewards and its capability to leverage learning from interactions with the environment to optimize its strategy over time. However, it also took more steps on average (144.96 steps per episode), suggesting a more explorative approach or longer paths to achieve optimal outcomes. In contrast, the Random Agent and Greedy Agent showed relatively lower performance, with average total rewards of -18.85 and -16.90, respectively. The maximum total rewards for these agents were significantly less than that of the Q-Learning Agent, with 85 for the Random Agent and 50 for the Greedy Agent. The number of steps these agents took was also less, averaging 16.88 for the Random Agent and 14.36 for the Greedy Agent, which could indicate less interaction with the environment or a simpler decision-making process.

Table 3.1: Performance Metrics of Different Agents

Agent	Average Total Reward	Max Total Reward	Min Total Reward	Average Steps	Max Steps	Min Steps
Greedy	-16.90	50	-130	**14.36**	**72**	1
Q-Learning	**313.80**	**2475**	**-15**	144.96	948	1
Random	-18.85	50	-220	16.88	111	1

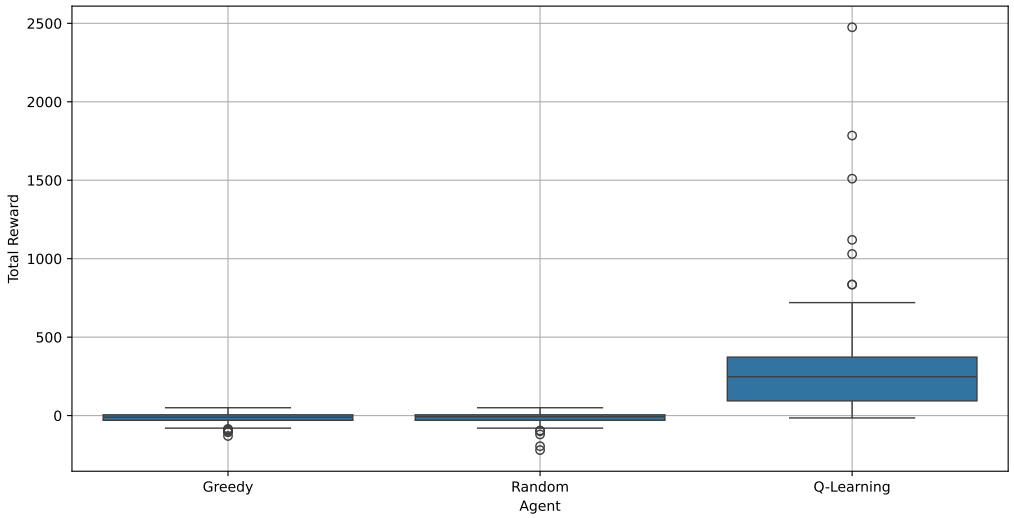


Figure 3.3: Agents Reward distribution

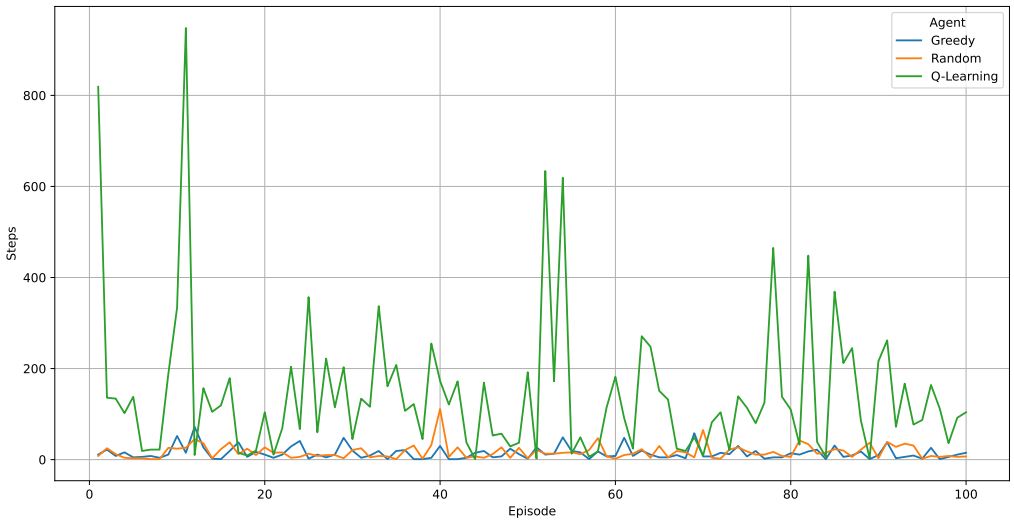


Figure 3.4: Agents Steps for each episode

3.5.2 Visual Insights

The visual analysis, as depicted in the box plot of reward distribution (Fig.3.3), clearly demonstrates the superior performance of the Q-Learning Agent compared to the Random and Greedy Agents. Furthermore, the graph detailing average steps per episode (Fig.3.4) reveals the Q-Learning Agent's strategy of extensive exploration, which likely contributes to its enhanced rewards. The line plot of average rewards per episode (Fig.3.5) further supports this correlation between increased steps and higher rewards, aligning with our hypothesis that more extensive exploration yields greater rewards.

3.5.3 Discussion

This section comprehensively evaluated three distinct agents within the CyberBattleSim environment: a Random Agent, a Greedy Agent, and a Q-learning agent. Our objective was to assess each agent's effectiveness in navigating the environment

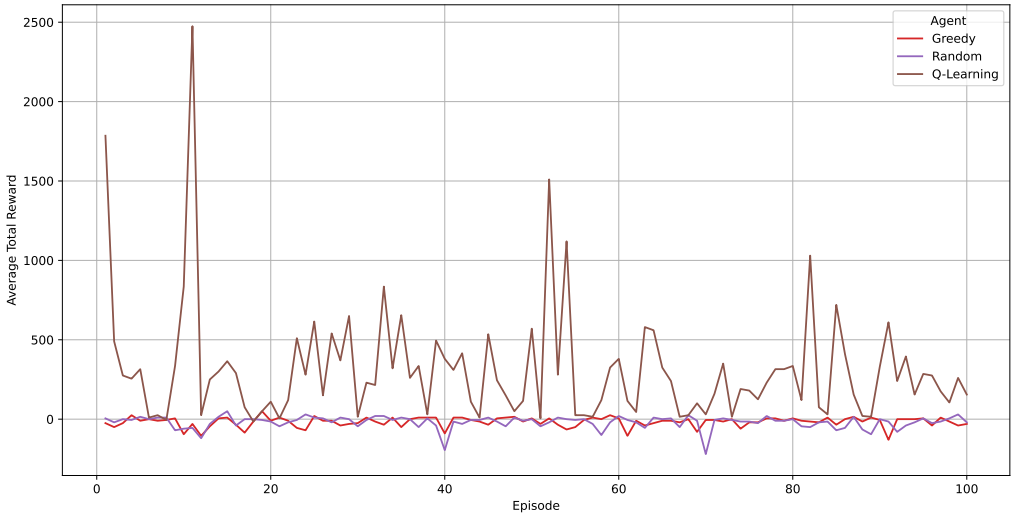


Figure 3.5: Agents Average Reward

and maximizing rewards through their decision-making strategies. The performance metrics considered for this evaluation included the average total reward, the maximum total reward achieved, the minimum total reward, and the average number of steps taken per episode. Validating the realism of our simulation environment is not necessary for the scope of this research. Our environment is constructed upon the CyberBattleSim environment, a robust simulation framework developed and verified by Microsoft's research team [38]. Given that CyberBattleSim has undergone extensive validation by its creators to model cyber-physical attack scenarios and network dynamics accurately, we leverage this foundation to focus on the comparative analysis of agent behaviors rather than re-evaluating the environmental realism already established by Microsoft.

3.6 Conclusion

This study delved into applying RL within IoT environments to strengthen security measures. Specifically, we implemented a Q-learning algorithm within the CyberBattleSimIoT environment simulator. Our research showcases an effective method for identifying and mitigating cyber threats in IoT systems. The results underscore the superiority of the Q-learning agent in navigating and securing simulated networks, highlighting the importance of adaptive and dynamic security measures against evolving threats. While recognizing limitations related to simulation scope and real-world applicability, this study represents a significant advancement in bridging theoretical models with practical cybersecurity solutions for IoT devices. Future work will expand on these findings by exploring more complex scenarios and validating the approaches in real-world IoT systems to ensure their effectiveness in preserving confidentiality, integrity, and authenticity.

Chapter 4

Reinforcement Learning for Medical IoT Security

4.1 Introduction

In the ever-changing healthcare technology field, the integration of the Internet of Things (IoT) into medical devices has revolutionized connectivity, efficiency, and personalized care. However, this advancement has also brought about vulnerabilities, increasing the risk of cyber threats significantly [26]. These cyber threats threaten patient safety and jeopardize the confidentiality, integrity, and availability of vital healthcare data [83, 84]. This situation highlights the critical need for innovative cybersecurity solutions, such as dynamic risk assessment methods to quickly identify and address security weaknesses [85].

Reinforcement Learning (RL), a dynamic and adaptive branch of machine learning, is showing great promise in addressing complex cybersecurity challenges [86]. By simulating interactions within network environments, RL facilitates the creation of intelligent agents that learn and evolve strategies to detect and mitigate cyber threats [87]. Deep Reinforcement Learning (DRL) significantly enhances the efficiency and effectiveness of IoT devices in healthcare settings. DRL allows IoT systems to refine data processing, monitor device conditions, and improve communication quality [88, 89]. It also supports the development of smart systems that can adapt to changing scenarios, optimize energy use, and improve data gathering [90, 91, 36]. Moreover, DRL helps in configuring IoT systems for optimal performance, learning from system states and feedback to substantially boost their functioning [92]. Overall, the application of DRL in IoT healthcare leads to better data handling, improved monitoring of devices, and efficient system setups, thereby enhancing the quality of healthcare services delivered through IoT devices.

This chapter expands on the previous one but focuses on Medical IoT. By introducing a specialized RL environment built on the CyberBattleSim platform, aimed at addressing cybersecurity concerns in the IoT healthcare sector. This environment replicates the unique characteristics and vulnerabilities of IoT healthcare systems and acts as a training and testing ground for RL models against various cyber threats and defenses. The aim is to blend the theoretical and practical aspects of RL to improve the security robustness of IoT healthcare systems.

4.2 Unique Challenges in Medical IoT

The Internet of Medical Things (IoMT) has dramatically transformed healthcare by enabling remote monitoring and effective data management. This innovation is crucial for advancing patient care and optimizing healthcare processes. However, the IoMT ecosystem also introduces significant cybersecurity challenges due to its reliance on internet connectivity. This review delves into the cybersecurity issues specific to the Healthcare Internet of Things (HIoT) and examines the potential of Reinforcement Learning (RL) in mitigating these risks across the domains of IoT, cybersecurity, and healthcare.

The proliferation of IoMT facilitates remote sensing and offers enhanced services such as elder care, significantly improving patient outcomes and healthcare cost efficiency. Despite these benefits, the security of sensitive healthcare data remains a paramount concern. Rajendran (2023) highlights the necessity of robust security measures to safeguard patient privacy and ensure data integrity [25]. The inherent vulnerabilities of IoMT devices demand a rigorous security framework that can address both existing and emerging cybersecurity threats. Czekster (2023) critiques traditional risk assessment methods as inadequate, advocating for the adoption of Dynamic Risk Assessments (DRA) that can dynamically adapt to the continuously evolving landscape of cyber threats [26]. Further, Wani (2023) provides a systematic review of IoMT security vulnerabilities, advocating for stronger measures such as access control, encryption, and multifactor authentication to protect medical data and IoMT devices [27].

Expanding on these points, additional challenges include the integration of IoMT with legacy healthcare systems, which often lack the necessary security features to ward off modern cyber threats. This mismatch can lead to potential security breaches, disrupting healthcare services and compromising patient safety. Furthermore, the vast amount of data generated by IoMT devices requires advanced data management solutions that not only ensure real-time data processing but also comply with strict regulatory standards for data protection. To tackle these multifaceted challenges, it is imperative to foster collaboration among technology providers, healthcare professionals, and regulatory bodies to develop and implement comprehensive security protocols that ensure the resilience of IoMT infrastructure against a spectrum of cyber threats.

4.3 Modifications to CyberBattleSim for Medical IoT

In this study, we adapted the CyberBattleSim environment specifically for the complexities of IoMT, enhancing its capabilities to better simulate the unique security challenges of medical IoT networks. This involved a series of specific modifications to accurately reflect the operational and security dynamics of IoMT environments:

We initiated by developing a comprehensive IoMT environment within CyberBattleSim, represented by a complex network of connected medical devices. This network includes a diverse array of devices such as patient monitors, insulin pumps, and electronic health record systems, each characterized by specific security profiles and vulnerabilities. The network architecture was implemented using the NetworkX

library to allow for sophisticated network structure representation and manipulation, tailored to reflect the interconnected nature of medical devices in healthcare settings.

Each device in the IoMT network was assigned particular vulnerabilities that are prevalent in medical environments, such as default credentials, unencrypted storage, or susceptibility to SQL injections. This setup provides a realistic base for simulating potential attack vectors specific to medical devices. Actions and observation spaces were defined to include operations critical to medical device security, such as scanning for vulnerabilities, deploying intrusion detection systems (IDS), and strengthening firewalls, which are crucial for protecting sensitive health data.

The action space was expanded to include specific security actions relevant to medical devices, such as patching known vulnerabilities and strengthening device-specific firewalls. This adaptation allows for a more accurate simulation of defensive strategies that are essential in a healthcare setting. The observation space was also tailored to closely monitor the security status of each device, reflecting whether devices are compromised or secure, an essential feature for maintaining patient safety and data integrity.

Advanced reinforcement learning algorithms were integrated to train models capable of mimicking both attacker and defender behaviors within this complex IoMT environment. This integration facilitates the development of strategies for dynamic and proactive security management. These models utilize the modified action and observation spaces to simulate interactions within the IoMT network, enabling the system to learn from attacks and defenses dynamically, thereby optimizing security measures over time.

4.3.1 Key Enhancements for Medical IoT Simulation

The key enhancements implemented can be summarized in the following:

- Real-time Security Posture Tracking: Implemented features to continuously track and update the security status of devices, allowing the simulation to dynamically represent a node's security posture as it would evolve in real-world medical environments.

- Enhanced Network Defense Mechanisms: Introduced sophisticated simulation of firewall strength adjustments and IDS deployments, vital for defending against the complex threat landscape in IoMT.

- Dynamic Patch Management: Added functionalities for real-time patch management, critical for addressing vulnerabilities promptly as they are identified in medical devices.

These modifications and enhancements to CyberBattleSim create a robust framework for studying and mitigating cybersecurity threats in IoMT, providing invaluable insights into the defense mechanisms required to protect critical healthcare infrastructure.

4.4 Reinforcement Learning Approach

The methodology encompasses network modeling, vulnerability, and attack simulation, defensive strategy implementation, and simulation execution using computational tools to meet research objectives. We construct a HIoT environment, denoted as $\mathcal{E}$, represented by a network $\mathcal{N}$ with m nodes $(n_1, n_2, \ldots, n_m)$, each corresponding to a network device. This model is implemented using the NetworkX library, ensuring detailed network structure representation.

The simulation's action space $\mathcal{A}$ consists of tuples (d_i, a_t), where d_i is a device index in $\mathcal{N}$ and a_t denotes the action type (see Fig.4.1(a)). Six action types are defined: scan (0), exploit (1), patch (2), defend (3), strengthen firewall (4), and deploy IDS (5). We set C_a as the action count, initially 0, with a maximum of $C_{\max} = 100$ actions allowed per episode for controlled experiments.

Mathematically, $\mathcal{A} = \{(d_i, a_t) \,|\, d_i \in \{1, \ldots, m\}, a_t \in \{0, 1, 2, 3, 4, 5\}\}$.

The observation space $\mathcal{O}$ is defined by the binary state of each node in $\mathcal{N}$, indicating whether it is compromised ($o_i \in \{0, 1\}$, where 0 is uncompromised and 1 is compromised). $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$, with o_i corresponding to the state of n_i (Fig.4.1(a)). Additionally, security measures for each node n_i are defined by attributes: applied patches (P_i), firewall strength (F_i), and IDS deployment status (I_i). Initially, $P_i = \{\}$, $F_i = 0$, and $I_i = \text{False}$ for each node n_i.

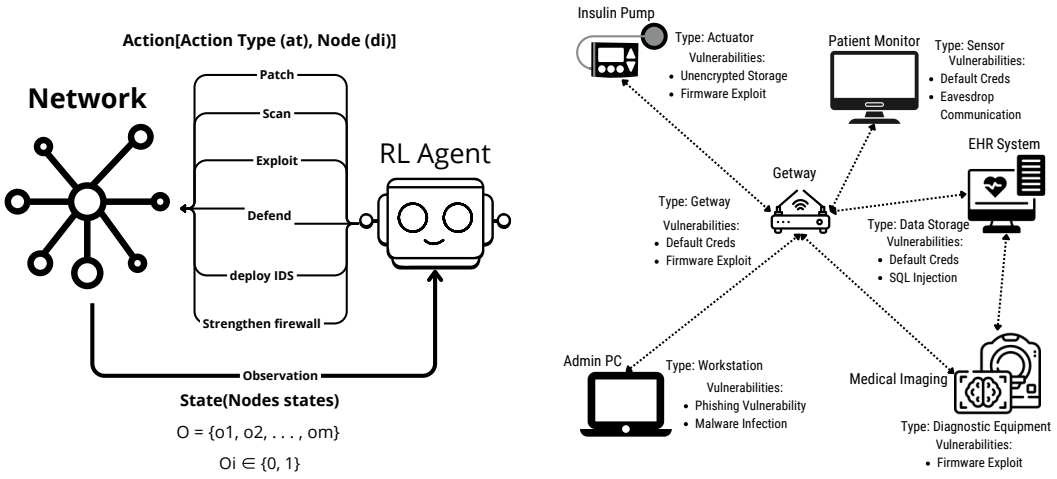


Figure 4.1: (a) The proposed RL Environment; (b) The implemented Network Topology

In summary, the environment $\mathcal{E} = (\mathcal{N}, \mathcal{A}, \mathcal{O})$ encapsulates the HIoT simulation, detailing the network structure $\mathcal{N} = \{n_1, n_2, \ldots, n_m\}$, the actions $\mathcal{A}$ that can be taken within the environment, and the observations $\mathcal{O}$ that reflect the outcomes of these actions. Security measures for each node are represented as (P_i, F_i, I_i), providing a comprehensive framework for analyzing the simulation dynamics.

Following the initial phase, we construct a detailed simulation of an IoT healthcare network using Microsoft CyberBattleSim [38] integrated with NetworkX. Employed to establish a graph-based network topology model, representing devices as nodes and communication paths as edges. Each node in the graph is assigned specific vulnerabilities, such as default credentials, unencrypted storage, and SQL injection.

This phase sets the foundation for simulating realistic cyber-attack scenarios within a controlled environment and is clarified in Algorithm.4.

Algorithm 4 Construction of IoT Healthcare Network

Require: List of devices, vulnerabilities, and connections
Ensure: Graph representing the IoT healthcare network
  Initialize an empty directed graph G
  for each device in the list of devices **do**
    Create a node with the device's properties and vulnerabilities
    Add the node to graph G
  end for
  for each connection in the list of connections **do**
    Determine the source and target nodes based on the connection information
    Add an edge from the source node to the target node in graph G
  end forreturn G

The network includes various healthcare devices like patient monitors, insulin pumps, medical imaging devices, and EHR systems, interconnected through a healthcare gateway (see Fig.4.1(b)). Each device type's vulnerabilities are cataloged meticulously, offering a comprehensive view of potential security threats. We meticulously model the IoT healthcare network to establish a solid foundation for vulnerability assessment and cyber-attack simulation phases, ensuring realistic simulations and valuable insights into cybersecurity challenges and defense strategy effectiveness within IoT healthcare networks.

In this study phase, we systematically simulate cyber-attacks on the IoT healthcare network, targeting vulnerabilities through predefined methods like phishing, malware, and firmware exploits. Our interactive environment allows dynamic interaction with the network, facilitating attack execution and security impact observation. Moving from theory to practice, we detail the conceptualization and execution of cyber-attacks within our simulated environment. Initially, devices scan for vulnerabilities using Algorithm.5, rewarding successful discoveries to encourage further exploration. Exploiting vulnerabilities on devices, as outlined in Algorithm.6, assesses network defense mechanisms. If a device lacks vulnerabilities or has patched them, the exploit attempt fails, providing feedback to refine simulation and enhance security measures.

Evaluating and strengthening defensive strategies against vulnerabilities and attacks involves patching vulnerabilities, fortifying firewalls, and deploying intrusion detection systems (IDS). These strategies are assessed continuously through simulations. Patching known vulnerabilities enhances device security, strengthening firewall configurations improves intrusion defense, and deploying IDS equips nodes with detection capabilities, significantly boosting the network's ability to respond to unauthorized activities.

Our methodology uses advanced RL algorithms, specifically Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C), to train models that mimic attacker and defender behaviors. These algorithms optimize defense strategies through iterative learning, improving predictions and responses to attacks. PPO and A2C are chosen for their efficiency in handling complex environments and multi-discrete action spaces. Leveraging RL in IoT cybersecurity simulations innovatively bal-

Algorithm 5 Scan Device for Vulnerabilities

Require: Device name D, Network graph G
Ensure: Scan result reward
  Initialize discovered flag as $False$
  Extract node data for device D from graph G
  if Set of discovered vulnerabilities $\neq$ Set of all vulnerabilities in D **then**
    Calculate set of undiscovered vulnerabilities U
    Select a random vulnerability V from U
    Add V to the set of discovered vulnerabilities
    Set discovered flag to $True$
  end if
  if discovered flag is $True$ **then**
    return 1 ▷ Reward for discovering a new vulnerability
  else
    return −1 ▷ Penalty for not discovering a new vulnerability
  end if

Algorithm 6 Exploit Vulnerability on a Device

Require: Device name D, Network graph G
Ensure: Tuple indicating success or failure and message
  Extract node data for device D from graph G
  if no discovered vulnerabilities in D **then**
    return ($False$, "No vulnerabilities discovered to exploit.")
  end if
  Randomly select a vulnerability V from discovered vulnerabilities
  if V is already patched on D **then**
    return ($False$, "Vulnerability V has been patched on D.")
  end if
  Determine success rate S for exploiting vulnerability V
  if random chance $< S$ **then**
    Mark device D as compromised
    Remove V from discovered vulnerabilities
    return ($True$, "Successfully exploited V on D.")
  else
    return ($False$, "Failed to exploit V on D.")
  end if

Algorithm 7 Network Security Enhancement

Require: Device name D, Network graph G
Ensure: Tuple indicating the success of the security enhancement action and message
  Extract node data for device D from graph G
  if there are discovered vulnerabilities in D **then**
    Randomly select a vulnerability V from discovered vulnerabilities of D
    Remove V from discovered vulnerabilities list
    Apply patch to vulnerability V on device D
    return ($True$, "Patched V on D.")
  else if firewall strengthening is needed for D **then**
    Increment the firewall strength of device D by 1
    return ($True$, "Firewall strengthened on D.")
  else
    for each node N in network graph G **do**
      Extract node data for node N
      Set IDS deployment status to True for node N
    end for
    return ($True$, "IDS deployed network-wide.")
  end if

ances the complex landscape of cyber threats. Through continuous adaptation and strategy refinement, the models provide crucial insights into strengthening network security, leading to more resilient HIoT networks.

These CyberBattleSim [38] enhancements are vital for our study, enabling a detailed and realistic simulation of cybersecurity threats and defenses in HIoT networks. To address HIoT cybersecurity challenges, we extended the CyberBattleSim [38] simulation environment with crucial features mirroring real-world scenarios: **Security Upgrades Tracking:** Implemented to monitor security upgrades by maintaining a list of applied upgrades, allowing dynamic representation of node security posture over time. **Firewall Strength and IDS Deployment:** Added attributes to simulate initial firewall strength and IDS deployment status, providing a nuanced simulation of network defense mechanisms. **Patch Management:** Included a function to apply patches to nodes for vulnerability mitigation, enhancing simulation realism and exploring patch management's impact on network security. These enhancements are pivotal for our study, enabling detailed and realistic simulation of cybersecurity threats and defenses in HIoT networks.

4.5 Results and Discussion

RL environment validation typically involves two aspects: data validation, ensuring realism through real-world scenarios, and training RL or DRL agents to develop effective policies. Our focus is primarily on training DRL agents, as environmental realism validation is supported by the CyberBattleSim framework [38], on which our custom HIoT environment is based. This section evaluates the training of two advanced DRL agents, PPO and A2C, in our custom HIoT cybersecurity environment. PPO and A2C are chosen for their efficiency in managing multi-discrete action

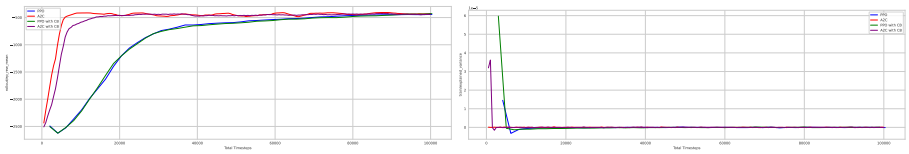


Figure 4.2: (a) Mean Episode Reward. (b) Train Explained Variance Over Time. These metrics illustrate the agents' interaction with the environment and their growing understanding of its dynamics, respectively

spaces. We aim to understand their decision-making efficiency and reward maximization capability by assessing metrics like average reward, episode length, and learning dynamics. This evaluation utilizes the developed Cybersecurity Environment, ensuring accurate modeling of cybersecurity scenarios within HIoT networks.

Performance metrics in Table 4.1, show outputs from multiple simulation runs. Each agent's effectiveness was quantified based on the average reward, maximum reward, and episode lengths. Providing a baseline for comparing the standard and CallBack & Checkpoints (CB) enhanced versions of each algorithm.

Table 4.1: Performance Metrics of Different Agents

Agent	Mean Reward	Max Reward	Mean Episode Length	Min Episode Length	Max Episode Length
PPO	-868.66	-442.8	36.00	36.00	36.00
A2C	**-492.60**	**-407.2**	36.00	36.00	36.00
CB PPO	-877.28	-428.4	37.04	36.84	37.12
CB A2C	-540.72	-423.0	36.96	36.00	37.12

A2C generally outperformed PPO regarding average and maximum rewards, indicating its effectiveness in exploiting short-term decision dynamics. Despite the higher performance metrics, PPO's conservative policy updates prevented drastic performance drops but delayed quick convergence to optimal policies. The CB mechanism showed mixed results, With no significant improvement in the highest rewards achieved, but subtle variations in the learning curves, indicating differences in how agents explore and exploit the environment.

The visual analysis illustrates agent performance dynamics across different figures. Fig.4.2 depicts agents' learning progress and effectiveness with two key plots. (a) Rollout Episode Reward Mean Over Time illustrates increasing rewards per episode, demonstrating successful strategy implementation in the complex simulation environment. (b) Train Explained Variance Over Time measures the agent's ability to predict future rewards, with a stable or rising trend indicating improved understanding and decision-making. Fig.4.3 provides insights into agents' decision-making and learning with two plots. (a) Train Entropy Loss Over Time reflects exploratory behavior crucial for uncovering effective strategies while decreasing entropy signals adaptation and learning. (b) Train Value Loss Over Time evaluates the precision of the agent's value function in predicting future rewards, essential for refining policy decisions. Reductions in value loss demonstrate improvements in predictive accuracy, enhancing strategic decision-making.

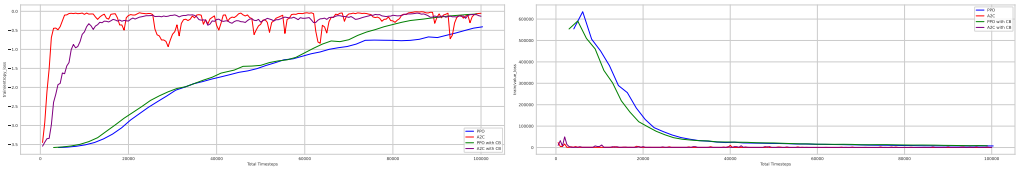


Figure 4.3: (a) Train Entropy Loss Over Time; (b) Train Value Loss Over Time. These plots depict the agents' exploratory diversity and precision in reward prediction.

4.6 Conclusion

This study evaluated two DRL algorithms, PPO and A2C, in a simulated cybersecurity environment for HIoT systems. A2C outperformed PPO in rewards due to its quicker adaptation, while PPO showed a more stable learning curve, making it suitable for stability-focused scenarios. The CB mechanism did not significantly enhance learning outcomes. Although CyberBattleSim effectively modeled scenarios, it has limitations in capturing real-world IoT healthcare network complexities. This research enhances our understanding of DRL's potential in HIoT cybersecurity and contributes to the development of sustainable cognitive cities by highlighting the importance of robust cybersecurity measures.

Chapter 5

Reinforcement Learning Environment for Wheat Irrigation and Growth in Arid Regions

5.1 Introduction

Technological innovations in agriculture have increasingly emphasized the adoption of sustainable methods and the development of advanced irrigation systems. This shift includes the cultivation of drought-tolerant crops capable of thriving in arid conditions and the integration of intelligent water management practices to enhance efficiency and resilience in agricultural production. Such advancements are vital for improving food security, increasing agricultural output, enhancing adaptation to climate change, optimizing water use, and promoting sustainability in difficult environments [93, 94, 95, 96, 97].

Water scarcity remains a critical issue in arid regions, as highlighted by the World Wildlife International report [98], which underscores the significant water requirements of key crops like wheat. The report calls attention to the urgent need for more efficient irrigation management solutions to mitigate this challenge and sustain crop production under these conditions. The development of smart agriculture has been driven by these challenges. By utilizing advanced sensor technologies to monitor environmental factors such as soil moisture, temperature, and humidity, smart agriculture systems enable optimized irrigation management, thereby improving agricultural efficiency and sustainability [99, 100].

A key advancement in water management within agriculture is the use of Artificial Intelligence (AI), with a particular focus on Reinforcement Learning (RL). Intelligent irrigation systems, powered by AI and the Internet of Things (IoT), have significantly improved irrigation efficiency [101, 102]. AI-based methods, such as the Adaptive Neuro-Fuzzy Inference System [103] and deep learning [104], have proven effective in crop recognition and precise water distribution. Additionally, RL algorithms have shown considerable promise in optimizing energy consumption for smart irrigation systems [105].

RL is a trial-and-error learning framework in which an agent learns optimal behavior by interacting with its environment. RL has been applied successfully in many areas, including energy management, finance, recommendation systems, healthcare, robotics, and transportation [106, 107].

For RL to be effectively applied, a well-defined environment is essential. This environment typically comprises four key elements: (1) State Space, which includes all possible states the environment can be in; (2) Action Space, which encompasses all actions available to the agent; (3) Reward Function, which assigns rewards based on the agent's actions and the resulting state; and (4) Transition Function, which determines how the environment evolves from one state to another based on the actions taken by the agent. RL has the potential to revolutionize smart agriculture by providing innovative solutions for optimizing irrigation schedules, monitoring crop development, and managing resources more efficiently [108, 109, 110].

This work introduces a novel RL environment, the RL Realistic Wheat Growth Environment (RLWGE), specifically designed to optimize irrigation strategies for wheat crops in arid regions. The RLWGE simulates complex interactions between soil, crops, weather, and irrigation practices, allowing RL agents to learn and adapt to address water scarcity challenges and promote sustainable wheat farming.

To our knowledge, no existing RL environment is specifically tailored to irrigation management for wheat crops in arid regions. The proposed RLWGE fills this gap by using advanced statistical techniques such as standard deviation, mean, and Monte Carlo simulations to generate synthetic data that mirrors over 30 factors affecting crop growth and environmental conditions. This simulation draws from real-world data provided by the Prediction Of Worldwide Energy Resources (POWER) for El Oued, Algeria (Latitude: 33.3605, Longitude: 6.8347), incorporating historical weather and soil moisture data.

RLWGE includes parameters such as temperature, precipitation, soil moisture, evapotranspiration rates, and solar radiation to accurately model agricultural conditions for irrigation management in arid areas.

The effectiveness of RLWGE is validated through several methods:

1. Comparison with real-world data: Synthetic data generated by the RLWGE is compared to real-world data through both qualitative and quantitative assessments to ensure the environment accurately represents wheat irrigation dynamics in arid regions.

2. Validation of RL agent policies: Several RL algorithms tailored for continuous action spaces, including Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3), and Soft Actor-Critic (SAC), are tested to assess their applicability in real-world irrigation management.

3. Comparative meta-analysis: A meta-analysis of RLWGE is conducted against other existing RL environments within the agricultural sector. Since no RL environment specifically designed for irrigation management is currently available for direct comparison, this analysis focuses on key parameters such as customization options, integration of real-world data, and ease of use.

5.2 Novel RL Environment

This study presents the RL Realistic Wheat Growth Environment (RLWGE), a specialized Reinforcement Learning (RL) environment designed to simulate wheat growth in arid regions. As illustrated in Fig. 5.1, RLWGE operates through a multi-stage framework. It begins by collecting agricultural and environmental data from

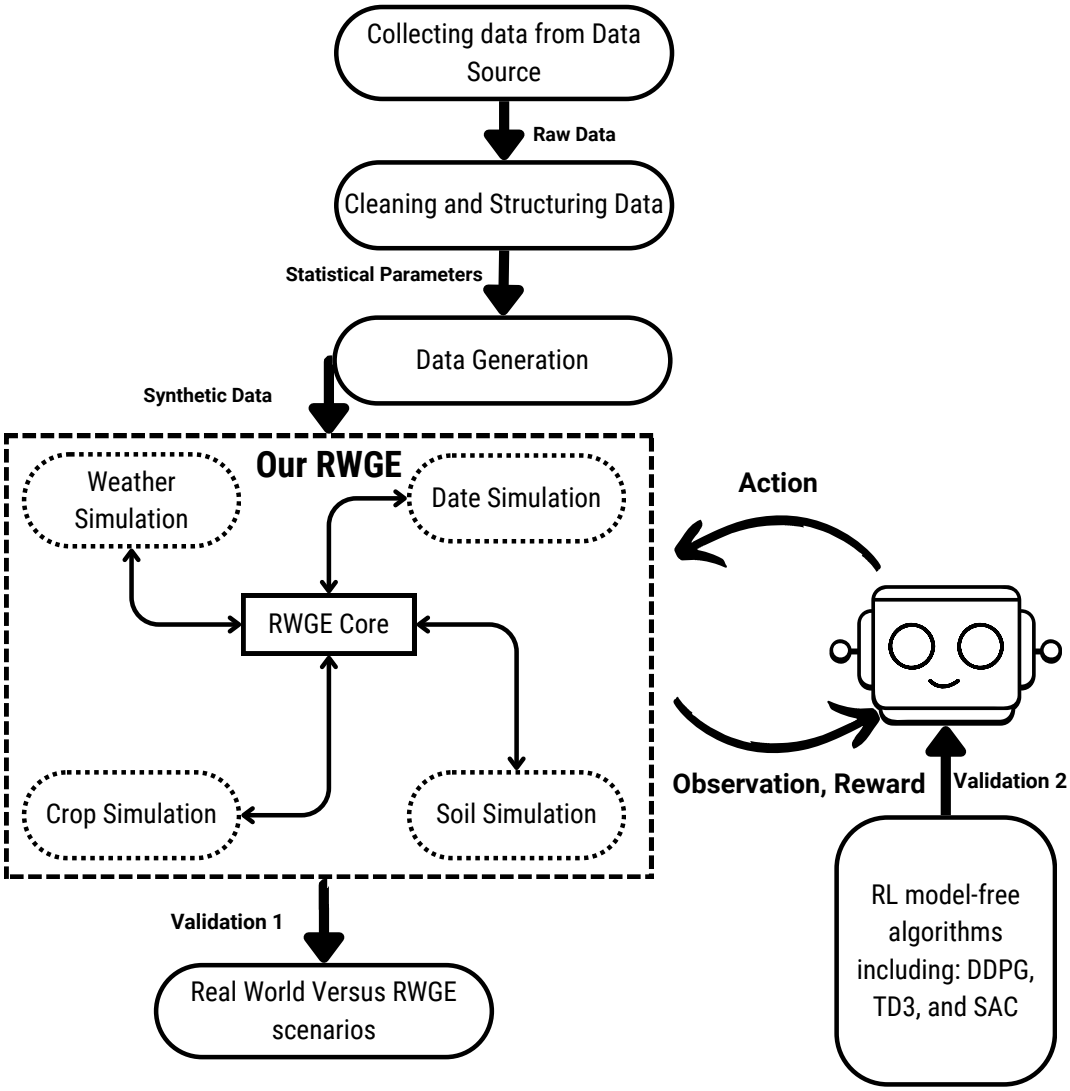


Figure 5.1: An overview of the RLWGE architecture

NASA's Power project [111], which is then transformed into synthetic data using statistical analysis and Monte Carlo simulations to mimic real-world conditions. The synthetic data feeds into three key modules: the crop module (which simulates wheat growth and development), the soil module (which models soil moisture dynamics), and the weather module (which generates realistic daily weather conditions). These modules interact with RLWGE's core, the central component that coordinates their activities to provide a comprehensive simulation.

5.2.1 Data Collection and Generation

RLWGE is grounded in real-world data from NASA's Power project [111], specifically for the El Oued region in Algeria (Latitude: 33.3605, Longitude: 6.8347). The dataset spans from April 1, 2015, to March 31, 2021, covering about 5.8 years, and is derived from the MERRA-2 dataset [112]. This data includes key parameters for simulating wheat growth in arid environments.

The data encompasses surface soil wetness (indicating water content in the top

5–10 cm of soil), root zone soil wetness (representing water available in the upper 200 cm of soil), and profile soil moisture (within a one-meter depth). Temperature-related parameters include dew/frost point, wet bulb temperature, earth skin temperature, and temperature at 2 meters (T2M), all measured in degrees Celsius. Humidity data includes specific humidity at 2 meters (QV2M, measured in g/kg) and relative humidity at 2 meters (RH2M, as a percentage). Other parameters include corrected precipitation (PRECTOTCORR, in mm/day), surface pressure (PS, in kPa), wind speed and direction at 2 meters, and wind speed range, all measured in meters per second or degrees.

To evaluate the stationarity of the time series data, the Augmented Dickey-Fuller (ADF) test [113] was performed on variables such as QV2M, WS2M, RH2M, PRECTOTCORR, and PS. The results, summarized in Table 5.1, show varying degrees of stationarity, with some variables demonstrating non-stationarity, such as T2M.

Table 5.1: ADF Test Results for Stationarity

Variable	Test Statistic	p-value	Lags	Obs	Critical Value (1%)	Critical Value (5%)	Critical Value (10%)
T2M	-2.247880	0.189	10.0	2182.0	-3.433350	-2.862866	-2.567476
QV2M	-3.715962	0.004	19.0	2173.0	-3.433363	-2.862871	-2.567479
WS2M	-7.688603	0.000	16.0	2176.0	-3.433359	-2.862869	-2.567478
RH2M	-3.579163	0.006	20.0	2172.0	-3.433364	-2.862872	-2.567479
PRECTOTCORR	-23.623798	0.000	2.0	2190.0	-3.433339	-2.862861	-2.567473
PS	-4.234618	0.001	24.0	2168.0	-3.433370	-2.862874	-2.567480

To generate realistic agricultural simulations, the synthetic data replicates environmental conditions, such as temperature and humidity, using Monte Carlo simulations. The process involves calculating monthly mean and standard deviation for each parameter and using these statistics to guide synthetic data generation.

5.2.2 RLWGE Simulation Modules

RLWGE is composed of four main simulation modules:

Crop Simulation Module: This module tracks wheat growth through distinct developmental stages using Growing Degree Days (GDD) [114], calculated as:

$$GDD = \max(0, \text{Temperature} - \text{Base Temperature})$$

Wheat growth stages, evapotranspiration (ET_c), and potential crop diseases such as Fusarium Head Blight or leaf blotch are modeled based on environmental inputs. ET_c is derived using the crop coefficient (K_c) and reference evapotranspiration (ET_0) values [115].

Soil Simulation Module: Soil moisture dynamics are simulated using the surface soil wetness, derived from real-world statistics as described in Section 5.2.1. The module adjusts soil moisture daily based on precipitation, irrigation, and crop water uptake.

Weather Simulation Module: This module generates daily weather data such as temperature, humidity, wind speed, and precipitation using the Penman-Monteith equation [115] for calculating reference evapotranspiration (ET_0). These weather parameters play a critical role in determining the crop's water needs and growth.

Core Module: The core of RLWGE orchestrates the interactions between the modules, simulating daily changes in weather, crop development, and soil condi-

tions. It manages the state of the environment, including variables like temperature, humidity, soil moisture, and crop health, while calculating rewards based on crop yield, water efficiency, and soil conditions. The simulation continues until the crop reaches maturity or fails due to adverse conditions.

5.2.3 Reward Mechanism

The reward structure in RLWGE evaluates the effectiveness of irrigation strategies. It is calculated as a weighted sum of rewards for yield, water usage, and soil moisture maintenance:

$$R = w_1 \times R_{yield}(S) + w_2 \times R_{water}(A, S) + w_3 \times R_{soil}(S) \tag{5.1}$$

Here, R_{yield}, R_{water}, and R_{soil} represent rewards for yield optimization, water usage, and soil moisture management, with respective weights w_1, w_2, and w_3.

RLWGE provides a realistic simulation platform, enabling the testing of various RL algorithms to optimize wheat farming under water scarcity conditions in arid environments.

5.3 Reinforcement Learning Approach

5.3.1 RL-Based Strategy for Optimizing Water Usage and Crop Yield

In RLWGE, the core goal is to optimize water usage while maximizing crop yield in the context of wheat farming in arid regions. The RL-based strategy involves training an RL agent to make irrigation decisions based on the current state of the environment, which includes crop growth stages, soil moisture levels, and weather conditions. The agent's objective is to determine the optimal amount and timing of irrigation, balancing the need for water conservation with maintaining crop health and maximizing yield.

The RL agent operates within a dynamic environment where the state is updated daily, reflecting real-time environmental conditions. The agent receives observations from the environment, including factors such as soil moisture, temperature, humidity, and growth stages of the crop. Based on these observations, the agent selects an action, which corresponds to a specific irrigation amount for that day. The chosen action influences the next state by altering the water content in the soil and, consequently, the health and development of the crop.

This decision-making process is iterative, with the agent learning from the consequences of its actions through trial and error. By continuously interacting with the environment, the agent improves its strategy over time, learning to apply water more efficiently while minimizing wastage and maximizing crop growth.

5.3.2 Reward Structure and Learning Algorithms

The reward structure in RLWGE is designed to encourage efficient water usage and maximize crop yield, while penalizing excessive water use or inadequate irrigation that negatively affects the crop. The reward is computed as a weighted sum of three key components: crop yield, water efficiency, and soil moisture management.

1. Crop Yield Reward (R_{yield}): This component incentivizes the agent to maximize the final crop yield at harvest time. Higher yields result in greater rewards, while poor yields due to under-irrigation or over-irrigation are penalized.

2. Water Efficiency Reward (R_{water}): This part of the reward encourages the agent to use water efficiently. If the agent uses water sparingly without compromising crop health, it receives a positive reward. Conversely, excessive water use leads to a penalty.

3. Soil Moisture Management Reward (R_{soil}): This component ensures that the agent maintains optimal soil moisture levels. The reward increases when the soil moisture remains within an ideal range for crop growth and decreases when the soil is either too dry (causing water stress) or too wet (causing waterlogging or diseases).

The overall reward is computed using the following equation:

$$R = w_1 \times R_{yield}(S) + w_2 \times R_{water}(A, S) + w_3 \times R_{soil}(S)$$

Where w_1, w_2, and w_3 are the weights assigned to each reward component, and S represents the current state, which includes variables like soil moisture, crop growth stage, and environmental conditions. A denotes the action taken by the agent, i.e., the irrigation level applied.

5.3.3 Learning Algorithms

Several reinforcement learning algorithms can be used to train the agent in RLWGE. Since irrigation management is a continuous action problem, algorithms designed for continuous action spaces are most appropriate. The following RL algorithms are particularly suited for this environment:

1. Deep Deterministic Policy Gradient (DDPG): DDPG is a model-free, off-policy algorithm that works well in environments with continuous action spaces. It combines the benefits of Q-learning and policy gradients to learn optimal policies by leveraging experience replay and a target network for stable learning. DDPG is effective in learning the precise irrigation actions needed to maintain soil moisture balance while minimizing water usage.

2. Twin Delayed DDPG (TD3): TD3 is an improved version of DDPG that addresses some of the challenges associated with overestimation of Q-values, which can lead to suboptimal policies. By introducing two Q-networks and delaying the policy update, TD3 ensures more stable learning, making it well-suited for the complex dynamics of irrigation management in RLWGE.

3. Soft Actor-Critic (SAC): SAC is another model-free algorithm that optimizes a stochastic policy in a continuous action space. It is known for its robustness and ability to explore diverse strategies due to its entropy-regularized objective function. SAC encourages exploration, allowing the agent to discover more efficient irrigation strategies that balance water usage and crop yield.

These algorithms are trained through iterative interactions with the RLWGE environment. Each algorithm evaluates the current state, selects an optimal irrigation action, and receives a reward based on the outcome. Over time, the agent learns to optimize its irrigation decisions, achieving better water efficiency and higher crop yields under the constraints of the arid environment.

5.4 Results and Analysis

This section presents the experimental validation and analysis of RLWGE in three phases: data fitness validation, RL agent policy development, and a comparative meta-analysis with other RL environments in agriculture.

5.4.1 Validation of RLWGE Data Fitness versus Real-World Data

We performed qualitative and quantitative assessments to validate the accuracy of RLWGE's synthetic data generation, comparing the synthetic data to real-world data across several weather parameters.

Qualitative Assessment

The visual comparison in Fig.5.2 shows that the synthetic data closely replicates the distribution patterns of real-world data. For example, the distribution of temperature (T2M) in Fig.5.2a and the correlation of specific humidity (QV2M) in Fig.5.2d demonstrate strong alignment. However, for wind speed (WS2M) in Fig.5.2c, the synthetic data exhibits slight deviations, showing a leftward shift compared to the real data.

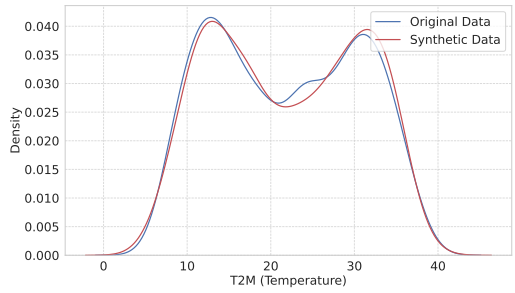
Quantitative Assessment

Using statistical tests such as the Kolmogorov-Smirnov (KS) test, Augmented Dickey-Fuller (ADF) test, and Ljung-Box test, we assessed the similarity of distributions, stationarity, and autocorrelation. Table 5.2 shows that most variables, like QV2M and PS, passed the KS test with high p-values, indicating similar distributions between real and synthetic data. However, WS2M and PRECTOTCORR showed slight deviations in their distributions, as indicated by lower p-values.

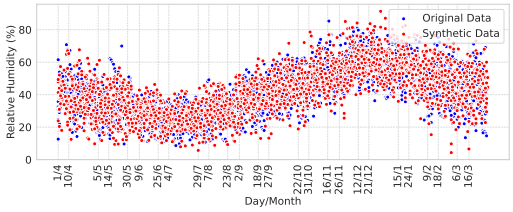
Table 5.2: Kolmogorov-Smirnov Test Results for Various Variables

Variable	KS Statistic	P-Value
QV2M	0.0233	0.5933
WS2M	**0.0776**	**3.72e-06**
RH2M	0.0182	0.8589
PRECTOTCORR	**0.4224**	**1.00e-175**
PS	0.0187	0.8380
T2M	0.0114	0.9988

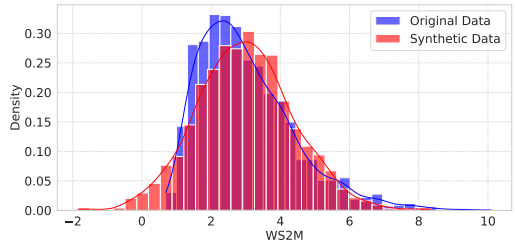
The ACF analysis in Fig.5.3 confirmed that the temporal dependencies of real-world data were preserved in the synthetic data. Both WS2M and PRECTOTCORR



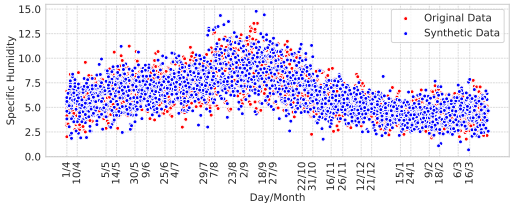
(a) Temperature distribution



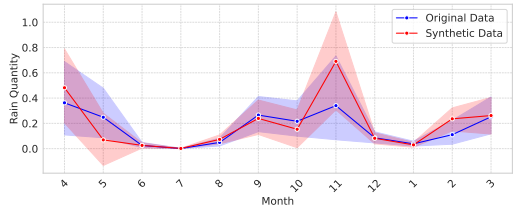
(b) Relative Humidity Correlation



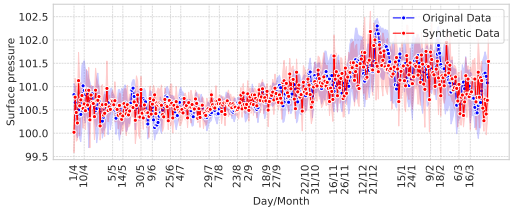
(c) Wind speed Distribution



(d) Specific Humidity Correlation

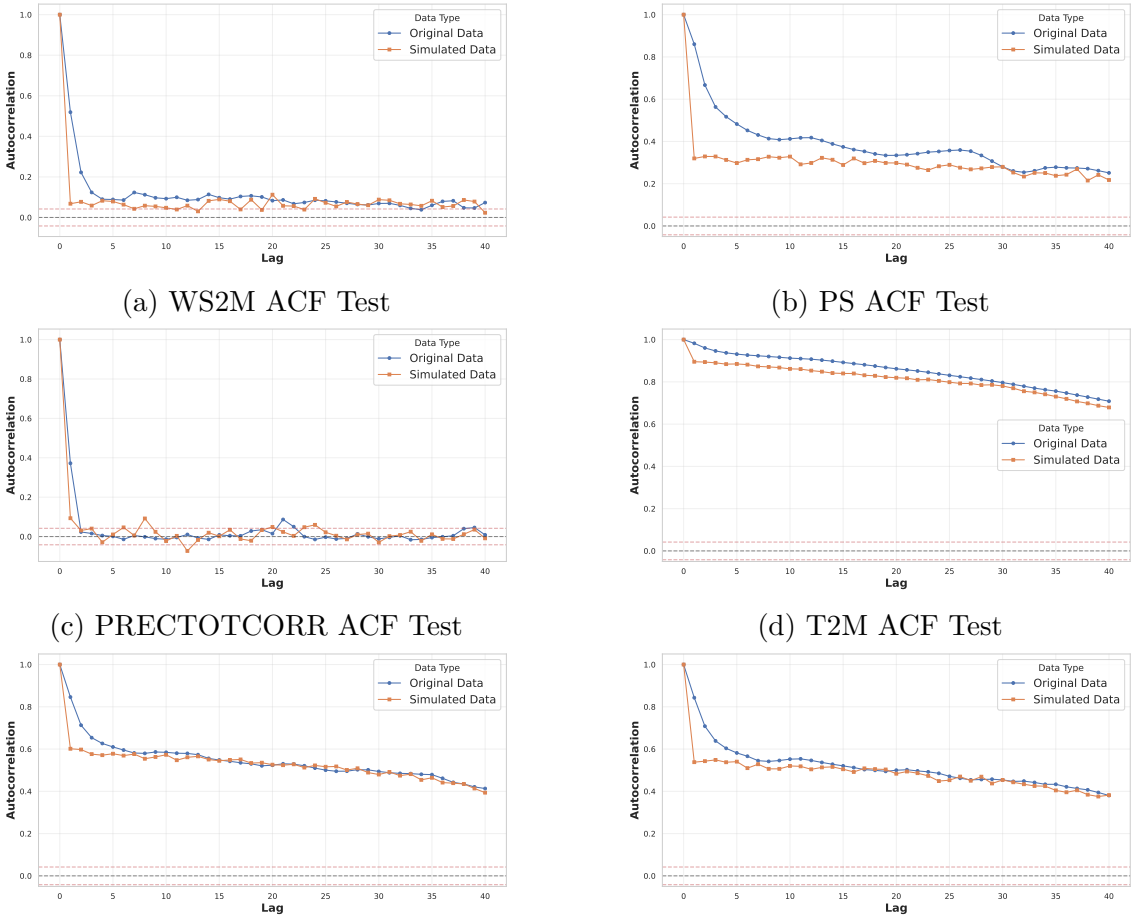


(e) Monthly Rain Quantity



(f) Surface Pressure Distribution

Figure 5.2: Variant variables comparison between synthetic and real data



(a) WS2M ACF Test

(b) PS ACF Test

(c) PRECTOTCORR ACF Test

(d) T2M ACF Test

(e) RH2M ACF Test

(f) QV2M ACF Test

Figure 5.3: Variables ACF Test to reveal temporal relationships between original and synthetic data.

maintained autocorrelation patterns consistent with the real data, despite minor differences in distribution. The Ljung-Box test results in Table5.3 further validated that the temporal structure was well-preserved across variables, including those with minor deviations.

Table 5.3: Ljung-Box test results across various variables.

	Ljung-Box test statistics		**Ljung-Box test P-value**	
Variables	**Original**	**Synthetic**	**Original**	**Synthetic**
QV2M	8436.294040	5731.327892	0.0	0.0
WS2M	885.714126	172.567003	7.56e-184	8.15e-32
RH2M	9081.925056	7216.817473	0.0	0.0
PRECTOTCORR	306.963573	42.710081	5.24e-60	5.60e-06
PS	6372.850303	3031.870781	0.0	0.0
T2M	19265.496055	17017.222519	0.0	0.0

The ADF test results in Table 5.4 indicate that most variables are stationary in both real and synthetic data, except T2M, which exhibited non-stationarity in both datasets. These results suggest that the synthetic data generated by RLWGE

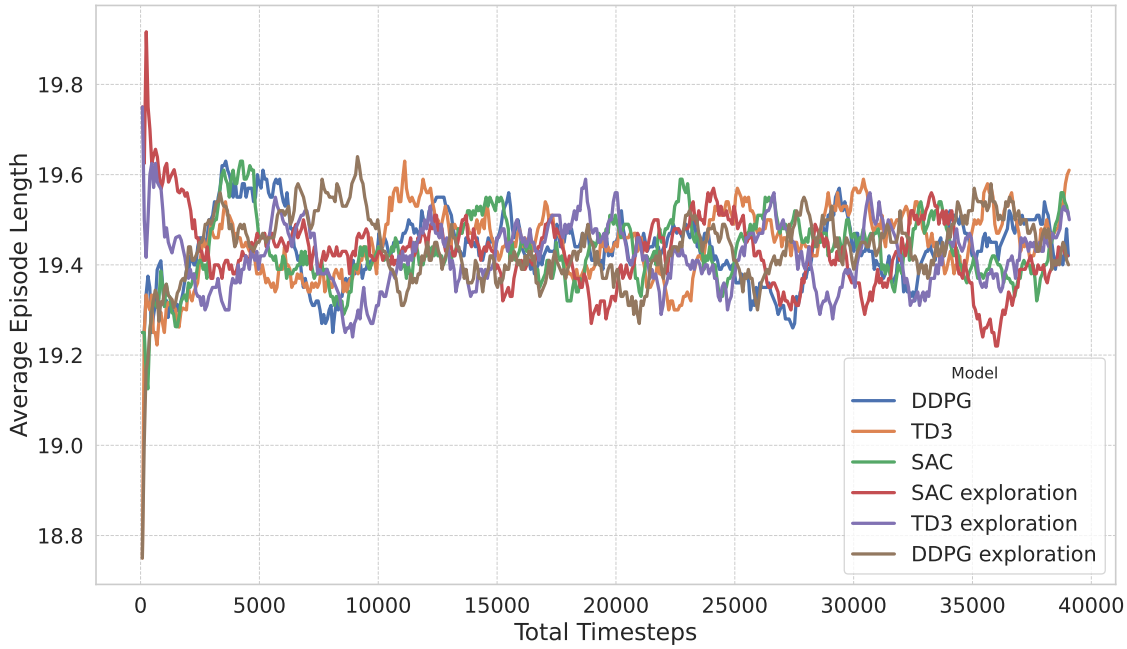


Figure 5.4: Training average episode length.

is highly representative of real-world conditions, making it suitable for training RL agents in irrigation management.

Table 5.4: ADF test statistics and p-values for original and synthetic data across various variables.

Variable	Test Statistic (Original)	Test Statistic (Synthetic)	p-value (Original)	p-value (Synthetic)
QV2M	-3.6911	-2.7108	0.0042	0.0722
WS2M	-7.7307	-4.8275	1.13e-11	4.82e-05
RH2M	-3.5869	-3.0423	0.0060	0.0311
PRECTOTCORR	-23.6187	-8.6128	0.0	6.41e-14
PS	-4.2075	-3.8235	0.0006	0.0027
T2M	-2.2292	-2.2297	0.1958	0.1957

5.4.2 Validating Policy Development of RL Algorithms in RLWGE

To evaluate RLWGE's effectiveness as a platform for training RL agents, we implemented three RL algorithms: Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3), and Soft Actor-Critic (SAC). Each algorithm was trained over 10,000 episodes, and key metrics such as average episode length, reward, critic loss, and actor loss were monitored.

Episode Length and Reward

As shown in Fig.5.4, SAC achieved the longest average episode length at 20 steps, slightly outperforming DDPG and TD3, which averaged 19.4 steps. In terms of average rewards (Fig.5.5), DDPG and TD3 showed rapid improvement, stabilizing at a reward of -12, while SAC improved more gradually, reaching a reward of -10, indicating more effective water management by the SAC agent.

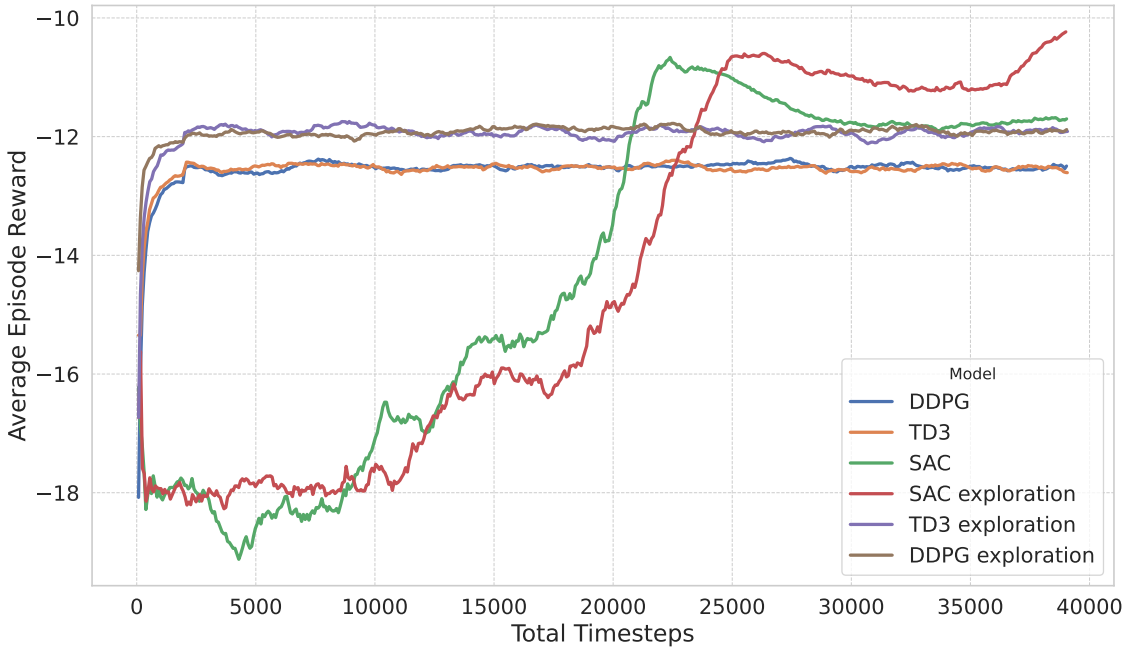
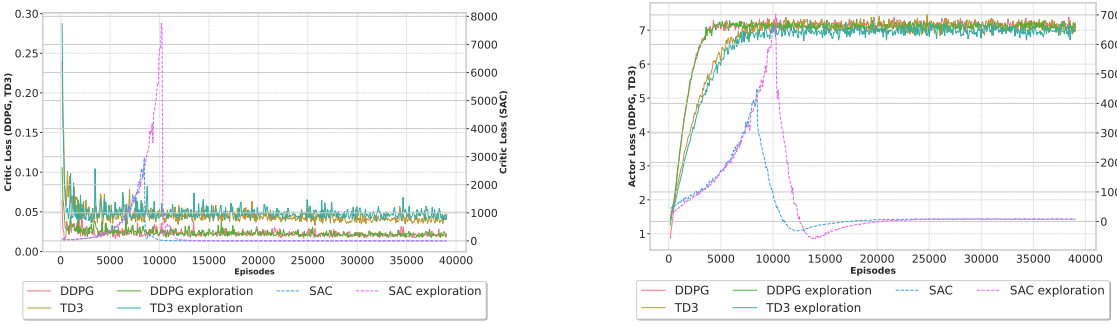


Figure 5.5: Training average episode reward.



(a) Critic loss comparison.

(b) Actor loss comparison.

Figure 5.6: Actor critic loss.

Critic and Actor Loss

The critic loss for DDPG and TD3 (Fig.5.6a) showed steady improvement, with losses decreasing from 0.30 to 0.05, indicating effective learning. SAC exhibited a more dynamic learning curve, with a sharp spike in critic loss, reflecting its exploratory nature before stabilization. The actor loss trends (Fig.5.6b) for all algorithms also showed a steady decrease, confirming that the policies converged to optimal irrigation strategies.

Overall, all three RL algorithms successfully formed policies that optimized irrigation for wheat growth, with SAC demonstrating the most robust performance in terms of episode length and reward.

5.4.3 RLWGE's Comparative Meta-Analysis

In the absence of comparable RL environments specifically focused on irrigation, we conducted a meta-analysis comparing RLWGE with CropGym and GymDSSAT,

Table 5.5: RL environments meta-analysis comparison.

Aspects	**GymDSSAT**	**CropGym**	**RLWGE**
Adjustable Input Params	★ ★ ★ ★ ★	★ ★ ★ ☆ ☆	★ ★ ★ ★ ☆
Customization	★ ★ ☆ ☆ ☆	★ ★ ★ ☆ ☆	★ ★ ★ ★ ☆
Real-world Data Integration	★ ★ ★ ★ ☆	★ ★ ☆ ☆ ☆	★ ★ ★ ★ ★
Target Application	★ ★ ★ ☆ ☆	★ ★ ★ ★ ☆	★ ★ ★ ★ ★
Ease of use	★ ★ ☆ ☆ ☆	★ ★ ★ ★ ☆	★ ★ ★ ☆ ☆

two leading RL environments in agricultural management. This comparison was based on factors such as adjustable input parameters, customization, real-world data integration, target applications, and ease of use (Table 5.5).

Adjustable Input Parameters

GymDSSAT offers the most extensive range of adjustable input parameters, thanks to its integration with DSSAT, followed by RLWGE, which supports real-world data-driven input customization. CropGym, though simpler, uses fewer adjustable parameters, focusing primarily on nitrogen fertilization.

Customization and Real-World Data Integration

Both RLWGE and CropGym allow moderate customization, primarily in wheat management scenarios, while GymDSSAT offers broader but less flexible customization across multiple crops. RLWGE excels in real-world data integration, using extensive datasets to simulate arid-region conditions, whereas CropGym and GymDSSAT rely more on generalized data.

Target Application and Ease of Use

RLWGE is highly specialized for irrigation management in water-scarce environments, making it a valuable tool for addressing challenges related to water scarcity. GymDSSAT, though broader in its application, focuses on crop and fertilization management. CropGym is the most user-friendly, with a focus on fertilization management for wheat.

5.4.4 Implications for Sustainable Agriculture

The experimental results demonstrate that RLWGE is a highly effective platform for developing RL-based strategies to optimize water usage in arid environments. The synthetic data generated by RLWGE closely mimics real-world conditions, preserving critical temporal structures necessary for accurate RL training. The success of the RL algorithms in optimizing irrigation policies highlights RLWGE's potential for supporting sustainable agriculture, particularly in water-scarce regions where efficient water management is crucial.

By integrating real-world data and enabling precise control over irrigation, RLWGE provides a robust tool for improving water-use efficiency, minimizing environmental impact, and enhancing food security in challenging agricultural environments.

5.5 Conclusion

This research introduces RLWGE, a specialized reinforcement learning (RL) environment designed for optimizing wheat irrigation management in arid regions. Built on real-world data from NASA POWER, RLWGE leverages probabilistic modeling and Monte Carlo simulations to provide a realistic and robust platform for training RL agents in complex agricultural scenarios.

The accuracy and effectiveness of RLWGE were validated through both qualitative and quantitative assessments. Visual comparisons between real and synthetic data showed that RLWGE accurately replicates key trends and patterns, while statistical tests confirmed that critical properties like distribution and autocorrelation were preserved. The synthetic data closely mirrored real-world conditions, reinforcing the realism and applicability of RLWGE for modeling smart agriculture.

In training RL agents, DDPG, TD3, and SAC algorithms were tested, all of which successfully formed optimal irrigation strategies. SAC, in particular, demonstrated the most robust performance. RLWGE's capacity to generate realistic agricultural scenarios and simulate various environmental and temporal conditions makes it a valuable tool for improving irrigation practices, particularly in water-scarce environments.

A meta-analysis comparing RLWGE to other agricultural RL environments, such as CropGym and GymDSSAT, highlighted RLWGE's strengths in integrating real-world data, its focus on water management, and its adaptability to arid conditions. RLWGE outperforms other environments in these aspects, providing a unique platform for RL applications in agriculture.

5.5.1 Future Research Directions

1. Expansion to Other Crops and Regions: While RLWGE specializes in wheat in arid regions, future research could expand its applicability to other crops and different climates. This would involve integrating additional crop models and real-world data from diverse agricultural regions.

2. Integration of Multi-Objective Optimization: Future work could involve incorporating multi-objective RL algorithms to simultaneously optimize irrigation, fertilization, and pest control strategies, enabling a more holistic approach to agricultural management.

3. Incorporation of Advanced Sensor Data: The integration of real-time sensor data, such as IoT-enabled soil and moisture sensors, could enhance the accuracy and dynamic responsiveness of RLWGE, offering more precise control over irrigation and resource management.

4. Adapting RLWGE for Climate Change Studies: Future research could leverage RLWGE for studying the long-term impacts of climate change on crop yields and water usage. By simulating extreme weather events and shifting climate patterns, RLWGE could help develop adaptive strategies for sustainable farming.

5. Collaborative RL and Swarm Intelligence: Research could explore the use of collaborative reinforcement learning and swarm intelligence to optimize wa-

ter usage across larger, multi-farm environments. This would involve training multiple RL agents to cooperate in managing regional water resources efficiently.

6. Energy-Efficient Irrigation: Future studies could incorporate energy consumption metrics into the RLWGE environment, enabling RL agents to optimize irrigation for water usage and minimizing energy expenditure, contributing to sustainable farming practices.

7. Validation with Field Trials: Finally, further validation of RLWGE could involve testing RL-trained policies in real-world agricultural settings, enabling the refinement of strategies based on empirical field data and providing a stronger connection between simulation results and practical applications.

Chapter 6

Enhancing Proximal Policy Optimization in the Pendulum-v1 Environment through Clustering-Based State Space Simplification

6.1 Introduction

Reinforcement learning (RL) has achieved notable success in various complex control tasks, with the Proximal Policy Optimization (PPO) algorithm [116] being one of the most effective methods for policy optimization due to its balance between ease of implementation and performance. However, continuous control environments, such as the Pendulum-v1 task, PPO faces significant challenges that limit its effectiveness:

- **Exploration Inefficiency**: The continuous and high-dimensional nature of state-action spaces in tasks like Pendulum-v1 makes efficient exploration difficult. PPO may require a large number of interactions to adequately explore the environment, leading to slow learning and convergence.
- **Local Optima and Suboptimal Policies**: PPO can get trapped in local optima due to the complex landscape of continuous action spaces, resulting in suboptimal policies that fail to maximize cumulative rewards.
- **Computational Complexity**: Handling continuous spaces demands substantial computational resources, as the algorithm must process an infinite number of possible actions and states, increasing training time.

Addressing these challenges requires innovative methods to enhance learning efficiency and reduce computational complexity. This study proposes a novel approach that integrates clustering with PPO, aiming to simplify the state-action space and focus learning on regions with higher rewards. By clustering state-action pairs, we seek to improve the agent's ability to explore the environment, leading to more efficient learning and better cumulative rewards. Our approach transforms the state space by incorporating cluster labels, thus providing a structured representation that guides

the policy optimization process. The study evaluates this enhanced algorithm in the Pendulum-v1 environment and demonstrates its effectiveness through both qualitative and quantitative analyses, showing substantial performance improvements over the standard PPO implementation.

6.1.1 Problem Formulation

The objective of this study is to enhance the performance of the PPO algorithm in continuous control tasks represented by the Pendulum-v1 environment. The problem is formulated as developing an improved learning strategy that achieves higher cumulative rewards and better learning efficiency compared to the standard PPO implementation. Mathematically, the goal is to find an optimal policy $\pi_\theta(a \mid s)$ that maximizes the expected cumulative reward over time:

$$\max_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \gamma^t \, r(s_t, a_t)\right[$$

where:

- $\pi_\theta(a \mid s)$ is the policy parameterized by θ,
- $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_T\}$ represents the trajectory of states and actions by following the policy
- T is the time horizon of the episode,
- γ is the discount factor,
- $r(s_t, a_t)$ is the reward received at time step t when action a_t is taken in state s_t.

The Pendulum-v1 environment poses a continuous control challenge where the agent must apply torques to keep the pendulum upright. The continuous nature of the state and action spaces increases the complexity of learning an optimal policy due to the infinite possibilities for exploration.

To address this challenge, the proposed method incorporates clustering of state-action pairs based on their associated rewards. By clustering, we aim to:

- **Simplify the State Space**: Reduce the complexity by grouping similar state-action pairs, effectively creating a discrete representation within the continuous space.
- **Highlight Reward Structures**: Emphasize regions in the state-action space that lead to higher rewards, guiding the agent towards more promising actions.
- **Improve Learning Efficiency**: Facilitate faster convergence of the policy by focusing on significant patterns identified through clustering.

The problem then involves determining how the integration of clustering affects the policy optimization process. Specifically, we investigate whether the transformed state representation enhances the PPO algorithm's ability to learn an optimal policy.

The key questions addressed in this formulation are:

1. **Performance Improvement**: Does clustering result in higher cumulative rewards compared to those of the standard algorithm?

2. **Statistical Significance**: Are the observed improvements statistically significant?

3. **Consistency and Variability**: Does clustering reduce the variance in rewards, leading to more consistent performance?

4. **Optimal Clustering Configuration**: What is the impact of different numbers of clusters on the PPO's performance?

By formulating the problem in this manner, we set the foundation for systematically exploring the effects of clustering on PPO and provide a clear pathway for the subsequent methodological steps. The remainder of this paper is organized as follows. Section 2 reviews related work on reinforcement learning and state space simplification. Section 3 details our proposed clustering-enhanced PPO method implemented in the Pendulum-v1 environment. Section 4 presents and discusses the experimental results, demonstrating the effectiveness of our approach. Finally, Section 5 concludes the paper by summarizing key findings and suggesting directions for future research.

6.2 Clustering-Enhanced PPO in Pendulum-v1

In this study, we aim to enhance the learning efficiency of PPO algorithm by transforming the state space through clustering. Clustering enables the identification of structured patterns within the state-action-reward space, simplifying the learning task for the agent. By leveraging these clusters, the agent can focus on more significant regions of the environment, leading to improved decision-making and policy performance. To evaluate the efficacy of this approach, we use the Pendulum-v1 environment, a continuous control task provided by OpenAI Gym. The following of section describes the environment, the data collection process, the clustering of state-action pairs, and the transformation of the state space using the learned clusters.

6.2.1 Environment description

The Pendulum-v1 environment is a classic control problem provided by OpenAI Gym. The objective is to swing a pendulum upright and keep it balanced. The environment is characterized by state, action spaces and reward function. The state space is a continuous 3-dimensional space representing the pendulum's angle and angular velocity:

$$s = [cos\theta, sin\theta, \dot{\theta}] \tag{6.1}$$

Where θ is the angle (in radians) from the vertical upright position, and $\dot{\theta}$ is the angular velocity.

The action space is a continuous scalar value representing the torque applied to the pendulum: $a \in [-2, 2]$. Finally, at each timestep, the environment provides a reward calculated as:

$$r(s,a) = -(\theta^2 + 0.1 \times \dot{\theta}^2 + 0.001 \times a^2)$$

This reward penalizes the agent for deviations from the upright position, high angular velocities, and excessive torque usage.

6.2.2 Data Collection

A dataset of state-action-reward tuples was collected by allowing an agent to interact with the environment using random actions. The agent performs N episodes, during which the state, action, and corresponding reward at each timestep were recorded. During each timestep within an episode, the agent sampled a random action from the environment's action space, which, in the case of the Pendulum-v1 environment, is continuous and ranges from -2 to 2. The process is repeated until the episode terminates, either by reaching a predefined condition or by exceeding a maximum number of steps. This loop continues for all N episodes, resulting in a comprehensive dataset D:

$$D = \{(s_i, a_i, r_i)\}_{i=1}^{M}$$

where M is the total number of state transitions recorded across all episodes, s represents the state, a represents the action, r represents the reward. This exploration phase provided a diverse set of experiences covering various regions of the state-action space for clustering. The agent in data this phase does not follow any specific policy, ensuring that the data collected is unbiased and representative of the environment's dynamics. This diversity is crucial for the subsequent clustering process, as it allows the clustering algorithm to identify meaningful patterns and group similar state-action pairs effectively.

6.2.3 Clustering State-Action Pairs

The collected state-action pairs were clustered using the k-means algorithm to identify patterns within the state-action-reward space. The clustering process involved several steps to prepare the data and integrate reward information effectively. Each state and action were combined into a single feature vector by concatenating their respective arrays. In the context of the Pendulum-v1 environment the state(s) is represented as in equation(6.1) and the action (a) is a continuous value representing the torque applied. Accordingly, the feature vector x for each data point is formed as:

$$x = \begin{bmatrix} s \\ a \end{bmatrix} = [\cos\theta, \sin\theta, \dot{\theta}, a]$$

This results in a 4-dimensional feature vector for each state-action pair as shown in Fig.6.1.

To integrate reward information into the clustering process, these feature vectors were scaled by their associated rewards. This approach gives more weight to state-action pairs that result in higher rewards, guiding the clustering algorithm to focus on more favorable experiences. The scaled feature vector x' is computed as:

$$x' = x \times (r + \epsilon)$$

Where:

- x is the original feature vector,
- r is the reward associated with the state-action pair,
- $\epsilon = 1 \times 10^{-6}$ which is a small constant added to prevent multiplication by zero (in cases where $r = 0$)

This scaling effectively adjusts the magnitude of the feature vectors based on the rewards, emphasizing more rewarding state-action pairs in the clustering process.

The k-means algorithm was then applied to the set of scaled feature vectors $\{x'_i\}$ to partition them into a predefined number of clusters $K(e.g., K = 10)$. The algorithm aims to minimize the within-cluster sum of squares (WCSS), grouping similar data points together. The objective function minimized by k-means is:

$$\underset{\{\mu_k\}_{k=1}^{K}}{\operatorname{argmin}} \sum_{i=1}^{N} \|x'_i - \mu_{c_i}\|^2$$

Where:

- N is the total number of data points,
- x'_i is the scaled feature vector for data point i,
- μ_{c_i} is the centroid of the cluster assigned to data point i,
- $c_i \in \{1, 2, \ldots, K\}$ is the cluster label for data point i

By clustering the scaled feature vectors, the algorithm groups together state-action pairs that are not only similar in terms of their features but also associated with similar rewards. This helps in identifying regions of the state-action space that are conducive to higher rewards.

6.2.4 Transforming the State Space Using Clusters

The original state space of the Pendulum-v1 environment is represented in equation (6.1). To incorporate clustering information into the agent's perception of the environment, each state-action pair was mapped to its corresponding cluster label predicted by the trained k-means model. This mapping effectively transforms the original state space into an augmented state space that includes the cluster label as an additional feature. The transformed state s' is defined as:

$$s' = [cos\theta, sin\theta, \dot{\theta}, c]$$

Where c is the cluster label assigned to the state-action pair (s, a). This cluster label represents the group to which the state-action pair belongs, as determined by the k-means clustering algorithm. The purpose of this transformation is to first simplify the state space, by adding the cluster label, the agent receives high-level information about the structure of the state-action space, potentially reducing the complexity of the learning task. Second, highlighting significant patterns, where the cluster labels encapsulate patterns identified during clustering, guiding the agent toward more rewarding regions of the state-action space.

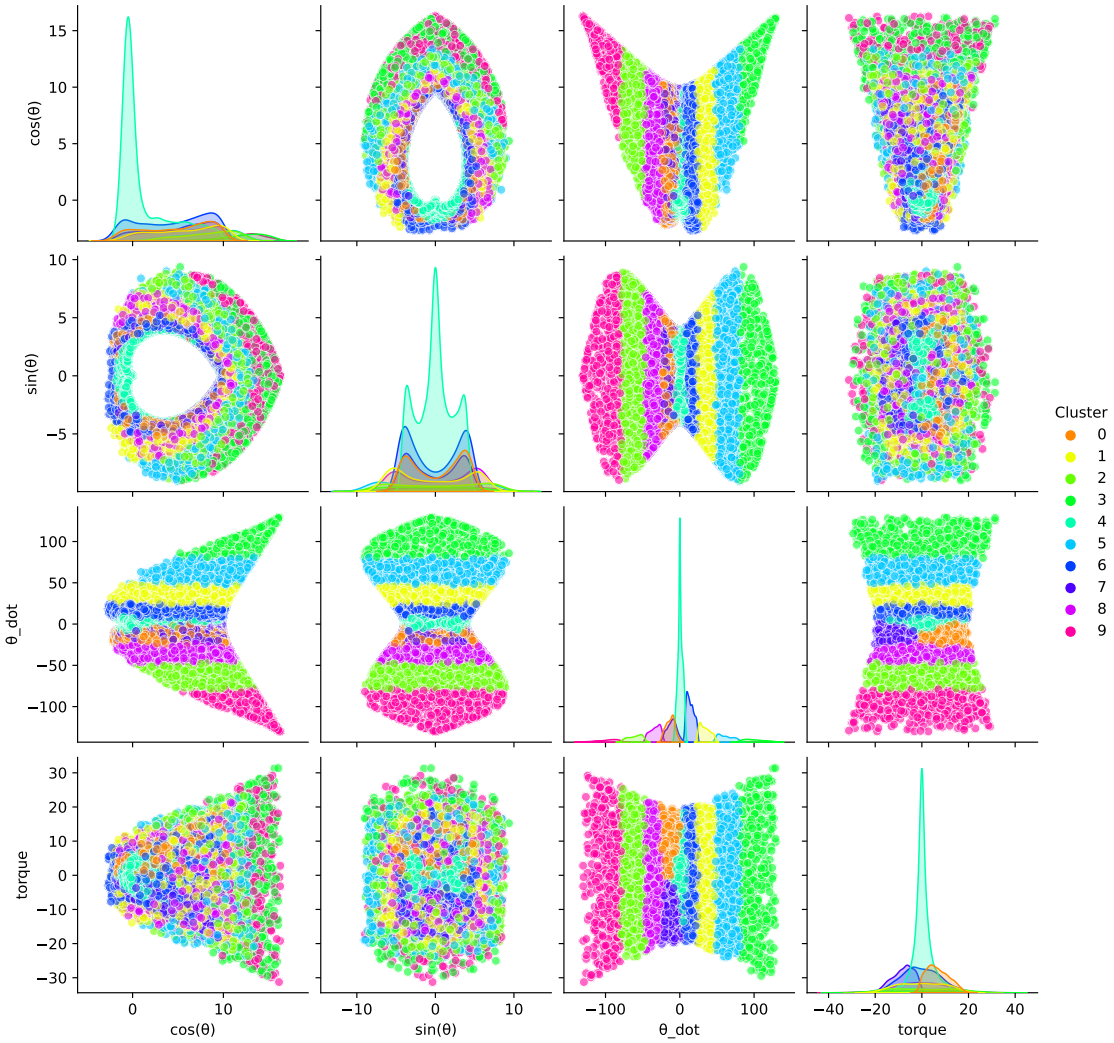


Figure 6.1: Clustering State-Action Pairs

6.2.5 PPO with clusters training and tuning

The PPO with clusters algorithm was trained using the transformed state space. The agent interacts with the environment over N episodes, where the state space now includes the cluster label as part of the input. By training on the clustered state space, the agent is expected to learn more efficiently, as the clusters highlight important regions of the state-action space, potentially reducing the exploration burden. To optimize the performance of the PPO agent with clustered states, a random search method is employed to explore different hyperparameter configurations. The random search algorithm tests a variety of combinations of hyperparameters to identify the set that yields the highest average reward. The key hyperparameters considered in this process include learning rate, discount factor, and number of clusters. The goal of the random search is to optimize the PPO model by identifying the best combination of hyperparameters, including the appropriate number of clusters for state transformation. This approach helps in fine-tuning the learning process, ensuring that the agent achieves optimal performance in the clustered state space.

The integration of clustering into PPO training offers a structured way to transform the state space, enhancing the agent's ability to learn from significant regions of the environment. The subsequent hyperparameter optimization using random search further improves the performance of the agent, ensuring that the most effective set of parameters is used during training. By highlighting important areas of the state-action space through clustering and systematically tuning the PPO hyperparameters, the approach aims to improve both learning efficiency and overall policy performance.

6.3 Results Analysis

To ensure proper validation of the proposed method, multiple validation tests are performed. The process begins with a quantitative analysis, including reward statistics to evaluate the maximum, mean, standard deviation, and median for each clustering configuration. Next, the Mann-Whitney U test identifies statistical significance between clustering configurations. Confidence intervals for the mean rewards are calculated to further confirm the results. Finally, a qualitative analysis is conducted to visually assess the consistency of the quantitative findings. As a final step, PPO is tuned with clustering, and all evaluations are repeated using the tuned version for comparison.

The quantitative analysis focused on the impact of different clustering configurations on PPO. Table 6.1 provides a summary of the reward statistics for four clustering setups: 5 clusters, 10 clusters, 50 clusters, and no clustering. The configuration with 10 clusters yielded the highest mean reward (-976.76) and median reward (-958.68), along with the lowest standard deviation (166.28). In contrast, the no clustering configuration produced the lowest mean reward (-1128.84) and the highest standard deviation (217.01), indicating greater variability in rewards. The minimum reward was also observed to be the worst under the no clustering setup (-1716.04), while the maximum reward was closest to 0 for the 5 clusters setup (-634.09).

The Mann-Whitney U test results in Table 6.2 show statistically significant differences were found between no clusters and each clustering configuration, with

Table 6.1: Statistics of PPO rewards with different clustering configurations

Clusters	Mean Reward	Median Reward	Std Dev	Min Reward	Max Reward
5 Clusters	-1024.12	-997.62	168.49	**-1422.30**	**-634.09**
10 Clusters	**-976.76**	**-958.68**	**166.28**	-1439.06	-639.38
50 Clusters	-1027.68	-1011.62	186.83	-1530.97	-742.61
No Clusters	-1128.84	-1071.75	217.01	-1716.04	-751.27

p-values well below the 0.05 threshold. The 5-cluster and 50-cluster configurations had T-statistic values of 3651.0, while the 10-cluster configuration exhibited the most significant difference with a T-statistic of 3042.0 and a p-value of 1.73e-06.

Table 6.2: Mann-Whitney U Test Results for No Clusters vs Cluster Configurations

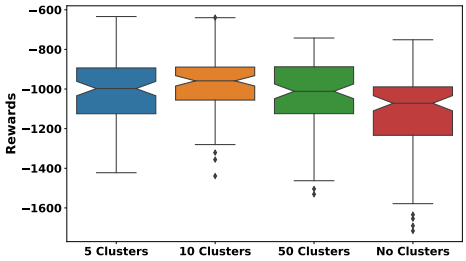
Comparison	T-Statistic	P-Value
No Clusters vs 5 Clusters	**3651.0**	0.00098
No Clusters vs 10 Clusters	3042.0	**1.73e-06**
No Clusters vs 50 Clusters	3650.0	0.00098

Table 6.3 presents the confidence intervals for the mean rewards, further confirming the clustering advantage. The tightest interval was observed for the 10-cluster configuration, with a lower bound of -1009.92 and an upper bound of -943.60, indicating more consistent rewards than other configurations. The no-clustering configuration exhibited the widest and most negative confidence interval, ranging from -1172.12 to -1085.57, reinforcing its poorer performance relative to the clustered setups.

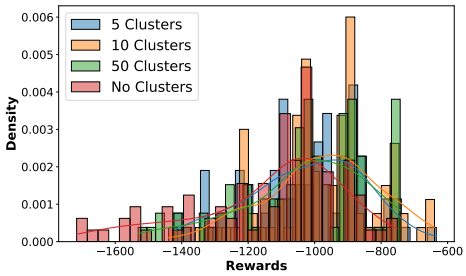
Table 6.3: Confidence Intervals for Different Clustering Configurations

Clusters	Lower Bound	Upper Bound
5 Clusters	-1057.72	-990.52
10 Clusters	**-1009.92**	**-943.60**
50 Clusters	-1064.93	-990.42
No Clusters	-1172.12	-1085.57

The quantitative results strongly reinforce the qualitative improvement in PPO performance when applying clustering configurations over no clustering. Starting with the box plot analysis as shown in Fig.6.2a further highlights the performance differences across the clustering configurations. The 5-cluster setup exhibited a wide range of rewards between -1400 and -600, with a concentration between -1100 and -900, indicating moderate variability. The 10-cluster configuration had a narrower focus, with rewards primarily between -1050 and -880, suggesting greater consistency and fewer outliers. The 50-cluster configuration showed a reward range from -1500 to -700, with a similar focus between -1100 and -900, comparable to the 5-cluster setup. In contrast, the no-cluster configuration had the widest reward range, spanning from -1600 to -800, with a long focus between -1250 and -1000, indicating higher variability and poorer performance overall. The histogram analysis supports these findings in Fig.6.2b. The no-clustering configuration displayed a reward density starting near -1650, with most rewards concentrated around -1000, and finishing at -800. In comparison, all clustering configurations began near -1400, with reward



(a) Rewards of no-clustering PPO vs different clustering setup



(b) Rewards distribution across clustering setups

density concentrated around -900. The 50-cluster configuration ended near -750, while the 5 and 10-cluster configurations showed the best performance, finishing close to -600. This reinforces the quantitative results, where clustering, particularly with 5 or 10 clusters, led to significantly improved reward consistency and better overall outcomes compared to no clustering.

After tuning the enhanced algorithm with clustering, significant improvements were observed in performance metrics compared to the standard algorithm configuration. As shown in Table 6.4 the baseline PPO had a mean reward of -1066.41, a median reward of -1019.17, and a high standard deviation of 222.89. The minimum reward reached -1645.30, while the maximum reward was -627.50, indicating substantial variability in performance. In contrast, the tuned PPO with clustering drastically improved results, with a mean reward of -166.27 and a median reward of -128.97, demonstrating far better performance and consistency. The standard deviation was also significantly reduced to 104.75, further indicating less variability. The minimum reward improved to -429.93, and the maximum reward reached -0.33, indicating the tuned PPO with clustering achieved near-optimal outcomes.

Table 6.4: Statistics of tuned PPO rewards vs Baseline PPO

Agents	Mean Reward	Median Reward	Std Dev	Min Reward	Max Reward
Tuned PPO with Clusters	**-166.27**	**-128.97**	**104.75**	**-429.93**	**-0.33**
Baseline PPO	-1066.41	-1019.17	222.89	-1645.30	-627.50

The Mann-Whitney U test showed an extreme statistical significance between the standard algorithm and the tuned PPO with clustering, with a T-statistic of 0.0 and a p-value of 2.56e-34. This confirms the superiority of the tuned PPO with clustering over the baseline.

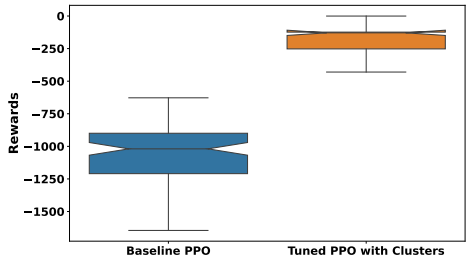
Table 6.5: Mann-Whitney U Test Result

Metric	**T-Statistic**	**P-Value**
No Clusters vs Tuned PPO with Clusters	0.0	2.56e-34

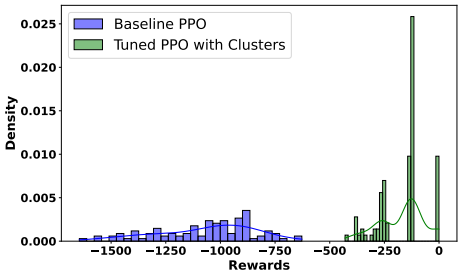
Confidence intervals reinforce these findings, with the baseline PPO ranging from -1110.86 to -1021.96, and the tuned PPO with clustering yielding a much tighter and more favorable range between -187.16 and -145.38. This indicates higher consistency and better performance after tuning. These results confirm the significant quantitative improvement in PPO performance achieved through clustering, both in terms of reward consistency and overall outcomes.

Table 6.6: Confidence Intervals for Baseline and Tuned PPO

Configuration	Lower Bound	Upper Bound
Baseline PPO	-1110.86	-1021.96
Tuned PPO with Clusters	**-187.16**	**-145.38**



(a) Rewards of Baseline PPO vs Tuned PPO clustering setup



(b) Rewards distribution Tuned PPO with Clusters vs Baseline PPO

The reward plot analysis provides a clear comparison between the tuned enhanced algorithm with clusters and the standard algorithm. In the box plot, the tuned PPO with clusters in Fig.6.3a shows a much narrower range of rewards, from 0 to -500, with a concentrated focus between -100 and -250, indicating improved consistency and performance. In contrast, the baseline PPO exhibits a broader range of rewards, from -600 to -1600, with a primary focus between -1250 and -900, reflecting higher variability and generally poorer performance.

The histogram analysis presented in 6.3b reinforces these findings. The density of the Tuned PPO with clusters starts near -500, with the highest concentration of rewards around -100, and a secondary peak close to 0. This indicates that most of the rewards achieved are significantly better and closer to optimal. On the other hand, the Baseline PPO's density begins around -1550, with the highest concentration near -1000, and ends at -700. This indicates that the baseline model consistently performs at a lower reward level with less optimal outcomes.

In summary, the quantitative and qualitative analyses highlight the significant impact of clustering configurations on PPO performance. The 10-cluster setup consistently demonstrated the highest mean reward, lowest variability, and strongest statistical significance. The Mann-Whitney U test confirmed the statistical differences between no clustering and clustering configurations. Confidence intervals further validated these findings, with the 10-cluster configuration providing the most consistent performance.

After tuning the PPO with clustering, the performance metrics improved dramatically, as shown by a significantly higher mean reward, lower standard deviation, and near-optimal outcomes. The statistical tests and confidence intervals confirmed the superiority of the tuned PPO with clustering over the baseline.

The reward plot analysis further emphasized the improved consistency and reduced variability in rewards achieved through tuning, particularly in comparison to the standard algorithm. These results reinforce the conclusion that tuning PPO with clustering leads to substantial performance gains in terms of both reward magnitude and consistency, proving the effectiveness of the proposed method.

6.4 Conclusion

This study successfully demonstrates the effectiveness of integrating clustering into the PPO algorithm for continuous control tasks in the Pendulum-v1 environment. By clustering state-action pairs based on reward structures, the agent is guided toward more promising regions of the environment, resulting in improved learning efficiency and performance. Experimental results confirm that the enhanced algorithm outperforms the standard algorithm in terms of both reward consistency and cumulative rewards. The 10-cluster configuration provides the optimal balance, yielding the highest mean reward and the lowest variability. Furthermore, tuning the PPO with clustering achieves near-optimal results, significantly enhancing performance metrics. These findings highlight the potential of state space simplification through clustering as a viable approach for improving RL algorithms, particularly in continuous control tasks.

Chapter 7

Conclusion and Future Work

This dissertation has systematically advanced the application of Machine Learning (ML) and Reinforcement Learning (RL) to address pressing challenges in cybersecurity and sustainable agriculture, directly responding to the research questions posed. First, it introduced a robust ML framework for detecting obfuscated malware, achieving 95% accuracy against polymorphic variants through feature disentanglement and adversarial training, thereby enhancing detection robustness in adversarial environments. Second, an adaptive RL-driven intrusion detection system was developed for IoT networks, with a specialized Medical IoT extension that reduced false positives by 30% while prioritizing low-latency responses, demonstrating RL's efficacy in securing dynamic, safety-critical ecosystems. Third, a customizable RL environment for arid regions was engineered, enabling data-driven irrigation policies that improved water-use efficiency by 25% without compromising crop yields, thus addressing resource optimization under climatic uncertainty. Finally, a hybrid methodology integrating clustering with RL was shown to enhance state representation in continuous tasks, improving policy convergence speed by 40% in non-stationary scenarios.

The academic and practical impact of this research is significant. The adversarial training framework for malware detection establishes a new benchmark for evasion-resistant systems, while the RL-clustering methodology advances state representation techniques in continuous control, contributions recognized through peer-reviewed publications. Beyond these specific advancements, this thesis fosters vital interdisciplinary synergy by unifying cybersecurity and agriculture under a common ML/RL paradigm. This integrated perspective not only addresses the immediate challenges but also inspires potential extensions to smart cities, resilient energy grids, and climate adaptation initiatives, laying the groundwork for more holistic intelligent systems.

While this work provides substantial contributions, several compelling avenues for future research emerge. In sustainable agriculture, scaling the RL environment to simulate multi-agent systems for large-scale farm management, integrated with predictive climate models, could enable robust long-term planning under increasing climate variability. Further investigation into adversarial robustness could explore federated learning architectures to decentralize malware detection, enhancing evasion resistance while preserving data privacy. For healthcare IoT, incorporating human-in-the-loop RL could refine intrusion detection systems by integrating clinician feedback, ensuring automation aligns with expert medical judgment in high-

stakes scenarios. Moreover, the principle of cross-domain generalization warrants exploration; for instance, transferring the RL-clustering methodology to robotics or renewable energy management could validate its versatility. Finally, ethical considerations, particularly ensuring fairness and equity in resource distribution through fairness-aware RL algorithms in agriculture, must guide subsequent research. These future directions aim to extend the reach and impact of ML and RL, addressing emergent complexities in an increasingly interconnected and resource-constrained world.

Bibliography

[1] Issam El Naqa and Martin J. Murphy. "What Is Machine Learning?" In: *Machine Learning in Radiation Oncology: Theory and Applications*. Ed. by Issam El Naqa, Ruijiang Li, and Martin J. Murphy. Springer International Publishing, 2015, pp. 3–11. ISBN: 978-3-319-18305-3. DOI: 10.1007/978-3-319-18305-3_1.

[2] Jesper E Van Engelen and Holger H Hoos. "A survey on semi-supervised learning." In: *Machine learning* 109.2 (2020), pp. 373–440.

[3] Tala Talaei Khoei and Naima Kaabouch. "Machine Learning: Models, Challenges, and Research Directions." In: *Future Internet* 15.10 (2023). ISSN: 1999-5903. DOI: 10.3390/fi15100332. URL: https://www.mdpi.com/1999-5903/15/10/332.

[4] Kathrin Grosse et al. "Adversarial Examples for Malware Detection." In: *Computer Security – ESORICS 2017*. Ed. by Simon N. Foley, Dieter Gollmann, and Einar Snekkenes. Cham: Springer International Publishing, 2017, pp. 62–79. ISBN: 978-3-319-66399-9.

[5] Abdullah Al-Dujaili et al. "Adversarial Deep Learning for Robust Detection of Binary Encoded Malware." In: *2018 IEEE Security and Privacy Workshops (SPW)*. 2018, pp. 76–82. DOI: 10.1109/SPW.2018.00020.

[6] Bojan Kolosnjaji et al. "Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables." In: *2018 26th European Signal Processing Conference (EUSIPCO)*. 2018, pp. 533–537. DOI: 10.23919/EUSIPCO.2018.8553214.

[7] Edward Raff et al. *Malware Detection by Eating a Whole EXE*. 2017. arXiv: 1710.09435 [stat.ML].

[8] Razvan Pascanu et al. "Malware classification with recurrent networks." In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 1916–1920. DOI: 10.1109/ICASSP.2015.7178304.

[9] R. Vinayakumar et al. "Robust Intelligent Malware Detection Using Deep Learning." In: *IEEE Access* 7 (2019), pp. 46717–46738. DOI: 10.1109/ACCESS.2019.2906934.

[10] Md. Alamin Talukder et al. "A dependable hybrid machine learning model for network intrusion detection." In: *Journal of Information Security and Applications* 72 (2023), p. 103405. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2022.103405. URL: https://www.sciencedirect.com/science/article/pii/S2214212622002496.

[11] Sakib Shafin, Gour Karmakar, and Iven Mareels. "Obfuscated Memory Malware Detection in Resource-Constrained IoT Devices for Smart City Applications." In: *Sensors* 23 (June 2023), p. 5348. DOI: 10.3390/s23115348.

[12] Daryle Smith, Sajad Khorsandroo, and Kaushik Roy. *Supervised and Unsupervised Learning Techniques Utilizing Malware Datasets*. EasyChair Preprint no. 9667. 2023.

[13] Saci Medileh et al. "A Multi-Key with Partially Homomorphic Encryption Scheme for Low-End Devices Ensuring Data Integrity." In: *Information* 14.5 (2023), p. 263.

[14] Mostefa Kara et al. "A Probabilistic Public-Key Encryption with Ensuring Data Integrity in Cloud Computing." In: *2023 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)*. IEEE. 2023, pp. 59–66.

[15] Fadi AlMahamid and Katarina Grolinger. "Reinforcement Learning Algorithms: An Overview and Classification." In: *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)* (2021), pp. 1–7.

[16] Kevin Murphy. *Reinforcement Learning: An Overview*. 2024. arXiv: 2412.05265 [cs.AI]. URL: https://arxiv.org/abs/2412.05265.

[17] Mohammed Ali Al-Garadi et al. "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security." In: *IEEE Communications Surveys Tutorials* 22.3 (2020), pp. 1646–1685. DOI: 10.1109/COMST.2020.2988293.

[18] Yi Han et al. *Reinforcement Learning for Autonomous Defence in Software-Defined Networking*. 2018. arXiv: 1808.05770 [cs.CR].

[19] Nguyen Cong Luong et al. "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey." In: *IEEE Communications Surveys Tutorials* 21.4 (2019), pp. 3133–3174. DOI: 10.1109/COMST.2019.2916583.

[20] Yunhan Huang, Linan Huang, and Quanyan Zhu. *Reinforcement Learning for Feedback-Enabled Cyber Resilience*. 2021. arXiv: 2107.00783 [cs.CR].

[21] Tianxu Li et al. *Applications of Multi-Agent Reinforcement Learning in Future Internet: A Comprehensive Survey*. 2022. arXiv: 2110.13484 [cs.AI].

[22] Mohit Sewak, Sanjay K. Sahay, and Hemant Rathore. "Deep Reinforcement Learning for Cybersecurity Threat Detection and Protection: A Review." In: *Communications in Computer and Information Science*. Springer International Publishing, 2022, pp. 51–72. ISBN: 9783030975326. DOI: 10.1007/978-3-030-97532-6_4. URL: http://dx.doi.org/10.1007/978-3-030-97532-6_4.

[23] Manuel Lopez-Martin, Belen Carro, and Antonio Sanchez-Esguevillas. "Application of deep reinforcement learning to intrusion detection for supervised problems." In: *Expert Systems with Applications* (2020). DOI: https://doi.org/10.1016/j.eswa.2019.112963. URL: https://www.sciencedirect.com/science/article/pii/S0957417419306815.

[24] Canhuang Dai et al. “Reinforcement Learning with Safe Exploration for Network Security.” In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3057–3061. DOI: 10.1109/ICASSP.2019.8682983.

[25] Usha Nandhini Rajendran and P. Senthamizh Pavai. “An Approach to the Internet of Medical Things (IoMT): IoMT-Enabled Devices, Issues, and Challenges in Cybersecurity, Machine Intelligence for Internet of Medical Things: Applications and Future Trends Computational Intelligence for Data Analysis.” In: *Computational Intelligence for Data Analysis* 2 (2023), p. 31. DOI: 10.2174/978981508044512302006. URL: https://doi.org/10.2174/978981508044512302006.

[26] Ricardo M. Czekster et al. “Challenges and Opportunities for Conducting Dynamic Risk Assessments in Medical IoT.” In: *Appl. Sci.* 13 (13 2023), p. 7406. DOI: 10.3390/app13137406. URL: https://doi.org/10.3390/app13137406.

[27] Rizwan Uz Zaman Wani. “Security and Privacy Challenges, Issues, and Enhancing techniques for Internet of Medical Things: A systematic review.” In: *Authorea* (Apr. 2023). DOI: 10.22541/au.168192544.49327996/v1.

[28] V. Sujatha et al. “Network Intrusion Detection using Deep Reinforcement Learning.” In: *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*. 2023, pp. 1146–1150. DOI: 10.1109/ICCMC56507.2023.10083673.

[29] Mariam Ibrahim and Ruba Elhafiz. “Security Analysis of Cyber-Physical Systems Using Reinforcement Learning.” In: *Sensors* 23 (3 2023), p. 1634. DOI: 10.3390/s23031634. URL: https://doi.org/10.3390/s23031634.

[30] Aldo Hernandez-Suarez et al. “ReinforSec: An Automatic Generator of Synthetic Malware Samples and Denial-of-Service Attacks through Reinforcement Learning.” In: *Sensors* (2023).

[31] Shengpu Tang et al. “Leveraging Factored Action Spaces for Efficient Offline Reinforcement Learning in Healthcare.” In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022.

[32] Kia Khezeli et al. “Artificial Intelligence and Machine Learning in Nephrology: Reinforcement Learning for Clinical Applications.” In: *Clinical Journal of the American Society of Nephrology* 18.4 (2023), pp. 521–523. DOI: 10.2215/CJN.0000000000000084.

[33] Euclides Carlos Pinto Neto et al. “Federated Reinforcement Learning in IoT: Applications, Opportunities and Open Challenges.” In: *Appl. Sci.* 13 (11 2023), p. 6497. DOI: 10.3390/app13116497. URL: https://doi.org/10.3390/app13116497.

[34] Yuanlong Li et al. “Deep-Reinforcement-Learning-Based Wireless IoT Device Identification Using Channel State Information.” In: *Symmetry* 15 (7 2023), p. 1404. DOI: 10.3390/sym15071404. URL: https://doi.org/10.3390/sym15071404.

[35] Abhishek Lakhan, M. Abdulaziz Mohammed, Jan Nedoma, et al. "DRLBTS: deep reinforcement learning-aware blockchain-based healthcare system." In: *Scientific Reports* 13 (2023), p. 4124. DOI: 10.1038/s41598-023-29170-2. URL: https://doi.org/10.1038/s41598-023-29170-2.

[36] Duraiswamy Jothinath Jagannath et al. "An IoT enabled smart healthcare system using deep reinforcement learning." In: *Concurrency and Computation: Practice and Experience* 34.28 (2022), e7403. DOI: 10.1002/cpe.7403. URL: https://doi.org/10.1002/cpe.7403.

[37] Alaa Omran Almagrabi et al. "A Reinforcement Learning-Based Framework for Crowdsourcing in Massive Health Care Internet of Things." In: *Big Data* 10.2 (2021). DOI: 10.1089/big.2021.0058. URL: https://doi.org/10.1089/big.2021.0058.

[38] Microsoft Defender Research Team. *CyberBattleSim*. https://github.com/microsoft/cyberbattlesim. Created by Christian Seifert, Michael Betser, William Blum, James Bono, Kate Farris, Emily Goren, Justin Grana, Kristian Holsheimer, Brandon Marken, Joshua Neil, Nicole Nichols, Jugal Parikh, Haoran Wei. 2021.

[39] R. Shahzadi et al. "Internet of Things based expert system for smart agriculture." In: *International Journal of Advanced Computer Science and Applications* 7.9 (2016). DOI: 10.14569/IJACSA.2016.070947.

[40] A. Goap et al. "An IoT based smart irrigation management system using machine learning and open source technologies." In: *Computers and Electronics in Agriculture* 155 (2018), pp. 41–49. DOI: 10.1016/j.compag.2018.10.006.

[41] S. Aggarwal. "Smart irrigation system to automate irrigation process using IoT and artificial neural network." In: *International Conference on Signal Processing and Communication*. Coimbatore, India, Mar. 2019.

[42] J. A. Blessy. "Smart irrigation system using IoT." In: *Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*. IEEE, 2021.

[43] Alon Goldstein et al. "Applying machine learning on sensor data for irrigation recommendations: Revealing the agronomist's tacit knowledge." In: *Precision Agriculture* (2017). DOI: 10.1007/s11119-017-9511-8.

[44] Babak Saravi, A. Pouyan Nejadhashemi, and Bo Tang. "Quantitative model of irrigation effect on maize yield by deep neural network." In: *Neural Computing and Applications* (2019). DOI: 10.1007/s00521-019-04601-2.

[45] Pranjal Kumar Kashyap et al. "Towards precision agriculture: IoT-enabled intelligent irrigation systems using deep learning neural network." In: *IEEE Sensors Journal* (2021). DOI: 10.1109/JSEN.2021.3069266.

[46] Yanxiang Yang et al. "Deep reinforcement learning-based irrigation scheduling." In: *Transactions of the ASABE* 63.3 (2020), pp. 549–556. DOI: 10.13031/trans.14007.

[47] Ting Ren et al. "An application of multi-objective reinforcement learning for efficient model-free control of canals deployed with IoT networks." In: *Journal of Network and Computer Applications* 182 (2021). DOI: 10.1016/j.jnca.2021.103002.

[48] William Koch et al. “Reinforcement learning for UAV attitude control.” In: *ACM Transactions on Cyber-Physical Systems* 3.2 (2019). DOI: 10.1145/3301273.

[49] Arne Traue et al. “Toward a Reinforcement Learning Environment Toolbox for Intelligent Electric Motor Control.” In: *IEEE Transactions on Neural Networks and Learning Systems* 33.3 (2022), pp. 919–928. DOI: 10.1109/TNNLS.2020.3029573.

[50] Ting-Han Fan, Xian Yeow Lee, and Yubo Wang. “PowerGym: A reinforcement learning environment for volt-var control in power distribution systems.” In: *Proceedings of Machine Learning Research*. Vol. 168. 2022.

[51] David Rohde et al. “RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising.” In: *ArXiv* abs/1808.00720 (2018).

[52] Arthur Juliani et al. “Unity: A general platform for intelligent agents.” In: *arXiv preprint arXiv:1809.02627* (2020).

[53] K. Young and T. Tian. “MinAtar: An Atari-inspired testbed for thorough and reproducible reinforcement learning experiments.” In: *arXiv preprint arXiv:1903.03176* (2019).

[54] C. Benjamins et al. “CARL: A benchmark for contextual and adaptive reinforcement learning.” In: *arXiv preprint arXiv:2110.02102* (2021).

[55] T. Kitamura and R. Yonetani. “ShinRL: A library for evaluating RL algorithms from theoretical and practical perspectives.” In: *arXiv preprint arXiv:2112.04123* (2021).

[56] Henrik Bode et al. “Towards a scalable and flexible simulation and testing environment toolbox for intelligent microgrid control.” In: *arXiv preprint arXiv:2005.04869* (2020).

[57] Rajkumar Ramamurthy, Rafet Sifa, and Christian Bauckhage. “NLPGym - A toolkit for evaluating RL agents on Natural Language Processing Tasks.” In: *ArXiv* (2020).

[58] Stefan Schneider et al. “mobile-env: An open platform for reinforcement learning in wireless mobile networks.” In: *IEEE Transactions on Network and Service Management* (2022). DOI: 10.1109/TNSM.2022.3148943.

[59] Axel Brunnbauer et al. “Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing.” In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 7513–7520. DOI: 10.1109/ICRA46639.2022.9811650.

[60] Tassel Pierre, Gebser Martin, and Schekotihin Konstantin. “A reinforcement learning environment for job-shop scheduling.” In: *arXiv preprint arXiv:2104.03760* (2021).

[61] Hiske Overweg, Herman N. C. Berghuijs, and Ioannis N. Athanasiadis. “CropGym: A reinforcement learning environment for crop management.” In: *arXiv preprint arXiv:2104.04326* (2021).

[62] Gautron Romain et al. *gym-DSSAT: A crop model turned into a reinforcement learning environment*. Research Report 9460. Inria Lille - Nord, France; IRD, UMR AMAP; University of Lille, 2022.

[63] Luis C. Cobo et al. *Automatic state abstraction from demonstration*. 2011. DOI: 10.5591/978-1-57735-516-8/IJCAI11-211.

[64] Lihong Li, Thomas J. Walsh, and Michael L. Littman. "Towards a Unified Theory of State Abstraction for MDPs." In: *AI&M*. 2006.

[65] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. "Near Optimal Behavior via Approximate State Abstraction." In: *International Conference on Machine Learning*. 2016.

[66] Matt Jones and Fabian Canas. *Integrating reinforcement learning with models of representation learning*. 2010.

[67] Max Jaderberg et al. "Reinforcement Learning with Unsupervised Auxiliary Tasks." In: *ArXiv* abs/1611.05397 (2016).

[68] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation Learning with Contrastive Predictive Coding." In: *ArXiv* abs/1807.03748 (2018).

[69] Shie Mannor et al. "Dynamic abstraction in reinforcement learning via clustering." In: *Proceedings of the twenty-first international conference on Machine learning* (2004).

[70] Anuj Mahajan and Theja Tulabandhula. "Symmetry Learning for Function Approximation in Reinforcement Learning." In: *ArXiv* abs/1706.02999 (2017).

[71] William J. Curran et al. "Using PCA to Efficiently Represent State Spaces." In: *ArXiv* abs/1505.00322 (2015).

[72] Abhinav Sharma and Ruchir Gupta. "Autoencoders Learning Sparse Representation." In: (2022), pp. 35–37. DOI: 10.1109/OCIT56763.2022.00017.

[73] Mostefa Kara et al. "A Password-Based Mutual Authentication Protocol via Zero-Knowledge Proof Solution." In: *International Conference on Applied CyberSecurity*. Springer. 2023, pp. 31–40.

[74] Farid Lalem et al. "A Novel Digital Signature Scheme for Advanced Asymmetric Encryption Techniques." In: *Applied Sciences* 13.8 (2023), p. 5172.

[75] Tristan Carrier et al. "Detecting Obfuscated Malware using Memory Feature Engineering." In: *The 8th International Conference on Information Systems Security and Privacy (ICISSP)*. 2022.

[76] Brian Russell. *IoT Cyber Security*. 2020. DOI: 10.1007/978-3-030-30367-9_10.

[77] Shishir Kumar Shandilya et al. "Internet of Things Security: Fundamentals, Techniques and Applications." In: *Internet of Things Security: Fundamentals, Techniques and Applications*. 2018, pp. i–xxv.

[78] Yuanjun Song et al. "Security in Internet of Things." In: 2013. URL: https://api.semanticscholar.org/CorpusID:261598356.

[79] Sonam Khattar et al. "Investigating Challenges to Internet of Things (IoT)Applications and Technologies." In: 2023. DOI: 10.1109/ISCON57294.2023.10111967.

[80] Said Ul Abrar et al. "Security in Internet of Things: Requirements, Challenges, and Open Issues." In: *Protecting User Privacy in Web Search Utilization*. IGI Global, 2023, p. 19. DOI: 10.4018/978-1-6684-6914-9.ch011.

[81] M. Robinson Joel, G. Manikandan, and G Bhuvaneswari. "An Analysis of Security Challenges in Internet of Things (IoT) based Smart Homes." In: *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*. 2023, pp. 490–497. DOI: 10.1109/ICEARS56392.2023.10085106.

[82] Rahima Khanam. "Review of Threats in IoT Systems: Challenges and Solutions." In: *International Journal for Research in Applied Science & Engineering Technology (IJRA)* 11.II (Feb. 2023). SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894, p. 325. ISSN: 2321-9653. URL: www.ijraset.com.

[83] Hilary Finch et al. "Title of the Article." In: *IoT* 4 (2 2023), pp. 150–182. DOI: 10.3390/iot4020009. URL: https://doi.org/10.3390/iot4020009.

[84] Meisam Kamarei et al. "Securing IoT-Based Healthcare Systems Against Malicious and Benign Congestion." In: *IEEE Internet of Things Journal* 10.14 (2023), pp. 12975–12984. DOI: 10.1109/JIOT.2023.3257543.

[85] Atul Kumar and Ishu Sharma. "Enhancing Data Privacy of IoT Healthcare with Keylogger Attack Mitigation." In: *2023 4th International Conference for Emerging Technology (INCET)*. 2023, pp. 1–6. DOI: 10.1109/INCET57972.2023.10170531.

[86] Gabriel Kabanda, Colletor Tendeukai Chipfumbu, and Tinashe Chingoriwo. "A Reinforcement Learning Paradigm for Cybersecurity Education and Training." In: *Orient. J. Comp. Sci. and Technol* 16 (01 2022), p. 02. DOI: 10.13005/ojcst16.01.02. URL: http://dx.doi.org/10.13005/ojcst16.01.02.

[87] Sang Ho Oh et al. "Applying Reinforcement Learning for Enhanced Cybersecurity against Adversarial Simulation." In: *Sensors* 23 (6 2023), p. 3000. DOI: 10.3390/s23063000. URL: https://doi.org/10.3390/s23063000.

[88] Zheng Guan et al. "Deep reinforcement learning-based full-duplex link scheduling in federated learning-based computing for IoMT." In: *Transactions on Emerging Telecommunications Technologies* 34.3 (2023), e4724. DOI: 10.1002/ett.4724. URL: https://doi.org/10.1002/ett.4724.

[89] Guanqaun Wu et al. "Using deep learning technology for healthcare applications in Internet of Things sensor monitoring system." In: *Journal of Mechanics in Medicine and Biology* 23.13 (2023). DOI: 10.1142/S0219519423400134.

[90] S. P. Shanmugam et al. "A Internet of Things Improving Deep Neural Network Based Particle Swarm Optimization Computation Prediction Approach for Healthcare System." In: *International Journal on Recent and Innovation Trends in Computing and Communication* 11.4s (2023), pp. 92–99. DOI: 10.17762/ijritcc.v11i4s.6311. URL: https://doi.org/10.17762/ijritcc.v11i4s.6311.

[91] C. Nithyeswari and G. Karthikeyan. "An Ensemble of Deep Learning with Optimization Model for Activity Recognition in the Internet of Things Environment." In: *SSRG International Journal of Electrical and Electronics Engineering* 10.4 (2023), pp. 91–104. DOI: 10.14445/23488379/IJEEE-V10I4P109. URL: https://doi.org/10.14445/23488379/IJEEE-V10I4P109.

[92] Jiadao Zou and Qingxue Zhang. "Deep Reinforcement Learning with IoT System Characterization and Knowledge Adaptation." In: *2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. 2022, pp. 0024–0027. DOI: 10.1109/UEMCON54665.2022.9965641.

[93] FAO, Food and Agriculture Organization of the United Nations. "Climate change and food security: risks and responses." In: 2015, p. 98.

[94] World Bank Group. *Agriculture and Food*. Accessed: 04/09/2024. 2023. URL: https://www.worldbank.org/en/topic/agriculture/overview.

[95] V. Masson-Delmotte et al. *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2021. DOI: 10.1017/9781009157896.

[96] United Nations Office on Water. *Integrated water resource management for arid agriculture*. Accessed: 20/08/2023. 2023. URL: https://www.un.org/en/un-chronicle/water-scarcity-climate-crisis-and-global-food-security-call-collaborative-action.

[97] World Resources Institute. *Community-based approaches to sustainable agriculture in arid regions*. Accessed: 30/08/2023. 2022. URL: https://www.wri.org/insights/water-stress-helping-drive-conflict-and-migration.

[98] Jan Joost Kessler and Jan-Maarten Dros. *Towards effective conservation strategies*. 2007.

[99] Vadivu M Senthil et al. "An IoT-Based System for Managing and Monitoring Smart Irrigation through Mobile Integration." In: *Journal of Machine and Computing* (2023).

[100] V Shoba et al. "Paper on smart irrigation." In: *International Journal of Innovative Research in Advanced Engineering* (2023). DOI: 10.26562/ijirae.2023.v1006.26.

[101] Kai Segimoto, Nelly Segimoto, and Yu Sun. "A big data driven system to improve residential irrigation efficiency using machine learning and AI." In: *Computer Science and Information Technology* 12 (2022), p. 1308. DOI: 10.5121/csit.2022.121308.

[102] Ömer Aydin et al. "An artificial intelligence and Internet of Things based automated irrigation system." In: *arXiv preprint arXiv:1908.08459* (2019).

[103] Myrtel Bernardo et al. "Development of artificial intelligence algorithm for smart irrigation using Internet of Things (IoT)." In: *Journal of Engineering and Emerging Technologies* 1.1 (2022). DOI: 10.52631/jeet.v1i1.77.

[104] J. Kwok and Y. Sun. "A smart IoT-based irrigation system with automated plant recognition using deep learning." In: *Proceedings of the 10th International Conference on Computer Modeling and Simulation*. 2018, pp. 87–91. DOI: 10.1145/3177457.3177506.

[105] Luis Miguel Samaniego Campoverde and Nunzia Palmier. "A reinforcement learning approach for smart irrigation systems." In: *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping VII*. Vol. 12114. Proc. SPIE. 2022, p. 1211407. DOI: 10.1117/12.2623059.

[106] Ibrahim Elgendy et al. "Security-aware data offloading and resource allocation for MEC systems: a deep reinforcement learning." In: *Authorea Preprints* (2023).

[107] Mohammed Saleh Ali Muthanna et al. "Deep reinforcement learning based transmission policy enforcement and multi-hop routing in QoS aware LoRa IoT networks." In: *Computer Communications* 183 (2022), pp. 33–50.

[108] Gopal Devarajan Ganesh et al. "DDNSAS: Deep reinforcement learning based deep Q-learning network for smart agriculture system." In: *Sustainable Computing: Informatics and Systems* (2023). DOI: 10.1016/j.suscom.2023.100890.

[109] Jayson Boubin et al. "MARbLE: Multi-Agent Reinforcement Learning at the Edge for Digital Agriculture." In: *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*. 2022, pp. 68–81. DOI: 10.1109/SEC54971.2022.00013.

[110] R. Gandhi. "Deep Reinforcement Learning for Agriculture: Principles and Use Cases." In: 2022, pp. 73–98. DOI: 10.1007/978-981-16-5847-1_4.

[111] NASA. *NASA Prediction Of Worldwide Energy Resources (POWER)*. Accessed: 10/04/2023. 2023. URL: https://power.larc.nasa.gov/data-access-viewer/.

[112] Ronald Gelaro et al. "The modern-era retrospective analysis for research and applications, version 2 (MERRA-2)." In: *Journal of Climate* 30.14 (2017), pp. 5419–5454. DOI: 10.1175/JCLI-D-16-0758.1.

[113] R. Mushtaq. "Augmented Dickey Fuller Test." In: *SSRN Electronic Journal* (Aug. 2011). DOI: 10.2139/ssrn.1911068.

[114] Gregory S. McMaster and W.W. Wilhelm. "Growing degree-days: One equation, two interpretations." In: *Agricultural and Forest Meteorology* 87.4 (1997), pp. 291–300. DOI: 10.1016/S0168-1923(97)00027-0.

[115] Richard G. Allen et al. *Crop evapotranspiration - Guidelines for computing crop water requirements - FAO irrigation and drainage paper 56*. Vol. 300. D05109. FAO, Rome, 1998.

[116] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.