



## ADAPTIVE SUN TRACKING ALGORITHM USING WEIGHTLESS NEURAL NETWORKS FOR PHOTOVOLTAIC SYSTEMS

Guilheme Santos Souza

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Priscila M. V. Lima  
Felipe M. G. França

Rio de Janeiro  
Fevereiro de 2025

ADAPTIVE SUN TRACKING ALGORITHM USING WEIGHTLESS NEURAL  
NETWORKS FOR PHOTOVOLTAIC SYSTEMS

Guilheme Santos Souza

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO  
ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE  
ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO  
COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO  
GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E  
COMPUTAÇÃO.

Orientadores: Priscila M. V. Lima  
Felipe M. G. França

Aprovada por: Prof. Priscila M. V. Lima  
Prof. Diego Leonel Cadette Dutra  
Prof. Leandro Santiago de Araújo

RIO DE JANEIRO, RJ – BRASIL  
FEVEREIRO DE 2025

Santos Souza, Guilherme

Adaptive sun tracking algorithm using weightless neural networks for photovoltaic systems/Guilherme Santos Souza.

– Rio de Janeiro: UFRJ/COPPE, 2025.

XIII, 65 p.: il.; 29, 7cm.

Orientadores: Priscila M. V. Lima

Felipe M. G. França

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 61 – 65.

1. Renewable Energy.
2. Reinforcement Learning.
3. Weightless Neural Networks. I. M. V. Lima, Priscila *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.



# Agradecimentos

Meu agradecimento à Professora Priscila Lima, pela orientação cuidadosa e incentivo constante, e ao Professor Felipe França e ao Rafael Katopodis, pelas contribuições fundamentais e pelo suporte técnico durante este trabalho.

Aos meus pais Gustavo e Gilmara, meu mais profundo agradecimento por todo o apoio e amor incondicional. À minha noiva Francielle Sales, minha gratidão pelo companheirismo e paciência, sempre ao meu lado nos momentos mais difíceis.

Por fim, agradeço aos amigos, colegas e professores que contribuíram, de diversas formas, para a realização deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMO DE RASTREAMENTO DE LUZ SOLAR ADAPTATIVO  
USANDO REDES NEURAI SEM PESOS PARA SISTEMAS  
FOTOVOLTAICOS

Guilherme Santos Souza

Fevereiro/2025

Orientadores: Priscila M. V. Lima  
Felipe M. G. França

Programa: Engenharia de Sistemas e Computação

Combater as mudanças climáticas é um dos desafios mais importantes para a humanidade, segundo as Nações Unidas [1]. A transição energética desempenha um papel fundamental nesse esforço, pois mais de 80% da energia global ainda é baseada em combustíveis fósseis. No entanto, a geração fotovoltaica tem crescido exponencialmente e agora possui um Custo Nivelado de Energia (LCOE) competitivo.

Estudos recentes buscam maneiras de melhorar a eficiência dos campos fotovoltaicos, apesar dos limites de eficiência dos módulos [2]. Uma estratégia é o rastreamento solar, em que os painéis solares são conectados a eixos motorizados para manter a superfície no ângulo ótimo para a máxima irradiação solar. Uma grande quantidade do recurso disponível é perdida quando não se acompanha o movimento solar. Atualmente métodos estáticos são amplamente adotados, mas enfrentam dificuldades em ambientes com cobertura de nuvens e terreno irregular.

Este trabalho avalia o uso de aprendizado por reforço para enfrentar esses desafios. Através de um sistema de *loop* fechado, é possível melhorar a estratégia de posicionamento dos painéis solares, aprendendo com o *feedback* do próprio sistema. As principais estratégias de aprendizado por reforço foram aplicadas, com implementações baseadas em redes neurais sem pesos. O impacto de diferentes hiperparâmetros foi avaliado para obter uma compreensão mais profunda das forças e limitações do método proposto. Os resultados foram positivos, alcançando 99.11% da *performance* do método estático, mas obtendo melhor resultado em dias nublados e sem necessidade de alimentá-lo com detalhes sobre a topologia do ambiente. Isso indica

um potencial para novos estudos explorando outras estratégias de aprendizado por reforço e estressando mais o algoritmo em contextos mais desafiadores.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

## ADAPTIVE SUN TRACKING ALGORITHM USING WEIGHTLESS NEURAL NETWORKS FOR PHOTOVOLTAIC SYSTEMS

Guilheme Santos Souza

February/2025

Advisors: Priscila M. V. Lima

Felipe M. G. França

Department: Systems Engineering and Computer Science

Combating climate change is one of the most important challenges for humanity, according to the United Nations [1]. The energy transition plays a fundamental role in this effort, as more than 80% of global energy is still based on fossil fuels. However, photovoltaic generation has been growing exponentially and now has a competitive Levelized Cost of Energy (LCOE).

Recent studies are seeking ways to improve the efficiency of photovoltaic fields despite the efficiency limits of the modules [2]. One strategy is solar tracking, in which solar panels are connected to motorized axes to maintain the surface at the optimal angle for maximum solar irradiance. Currently, the challenge with solar tracking lies in environments with cloud cover and irregular terrain.

This work evaluates the use of reinforcement learning to address these challenges. Through a closed-loop system, it is possible to improve the positioning strategy of the solar panels, learning from the system's own feedback. The main reinforcement learning strategies were applied, with implementations based on weightless neural networks. The impact of different hyperparameters was evaluated to gain a deeper understanding of the strengths and limitations of the proposed method. The results were positive, achieving 99.11% of the performance of the static method while demonstrating superior outcomes on cloudy days and requiring no prior information about the environment's topology. This suggests potential for further studies exploring alternative reinforcement learning strategies and subjecting the algorithm to more challenging contexts.

# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Related works . . . . .	2
1.2 Goals and contributions . . . . .	3
1.3 Text Structure . . . . .	4
<b>2 Application domain knowledge</b>	<b>6</b>
2.1 Irradiance to electricity conversion . . . . .	6
2.2 Irradiance and solar positioning . . . . .	7
2.3 Sun tracking algorithm . . . . .	8
2.4 Shading and Backtracking . . . . .	10
2.5 Limitation of current tracking solutions . . . . .	11
<b>3 Background knowledge</b>	<b>13</b>
3.1 Reinforcement learning . . . . .	13
3.1.1 Learning the state-value function . . . . .	15
3.1.2 Q-Learning . . . . .	16
3.1.3 Learning the decision policy . . . . .	17
3.2 n-Tuple based models . . . . .	19
3.2.1 WiSARD . . . . .	20
3.2.2 Regression WiSARD . . . . .	21
3.2.3 Binary Encoding Strategies for n-Tuple Models . . . . .	21
3.3 Reinforcement learning with n-tuple models . . . . .	24
3.3.1 SARSA Weightless Neural Network . . . . .	25
3.3.2 n-Tuple REINFORCE . . . . .	26
3.3.3 n-Tuple Actor-Critic . . . . .	29
<b>4 Methodology</b>	<b>32</b>
4.1 Dataset Preparation for Solar Tracking Simulation . . . . .	33

4.2	Environment Simulator . . . . .	34
4.2.1	Decision agent representation . . . . .	34
4.2.2	Environment representation . . . . .	35
4.2.3	Reward . . . . .	36
4.2.4	Learning Loop and evaluation . . . . .	37
4.2.5	Inputs encoding . . . . .	38
4.3	Evaluation Metrics . . . . .	39
4.4	Weightless Q-Learning Neural Network . . . . .	39
4.5	Chapter Conclusion . . . . .	40
<b>5</b>	<b>Experimental Evaluation</b>	<b>42</b>
5.1	Dataset and Preprocessing . . . . .	42
5.1.1	Thermometer . . . . .	46
5.2	Experimental execution . . . . .	46
5.3	Results analysis . . . . .	47
5.3.1	State-value models comparison . . . . .	47
5.3.2	Policy models comparison . . . . .	53
5.4	Chapter Conclusion . . . . .	56
<b>6</b>	<b>Conclusion</b>	<b>59</b>
6.1	Future Work . . . . .	60
	<b>References</b>	<b>61</b>

# List of Figures

2.1	PV Module [3]. . . . .	6
2.2	Illustration of Earth’s translation and rotation [4]. . . . .	8
2.3	Solar zenith and azimuth angles [5]. . . . .	8
2.4	Direct component of the plane of array irradiance. . . . .	9
2.5	Partial shading examples with and without backtracking strategies. . . . .	11
3.1	Markov decision process representation [6]. . . . .	13
3.2	Actor-critic diagram [7] . . . . .	18
3.3	WiSARD schema [7] . . . . .	20
3.4	Linear thermometer representation, receiving three different inputs (1, 5, and 10), assuming 0 as minimum and 12 as maximum, the space is spitted in 4 clusters of size 3. . . . .	22
3.5	Circular thermometer representation, receiving three different inputs (1, 5, and 10), assuming 0 as minimum and 12 as maximum, the space is spitted in 4 clusters of size 3. . . . .	23
3.6	Comparison of thermometer encoding thresholds [8] . . . . .	23
4.1	Methodology fluxogram. . . . .	32
4.2	Experiment representation. . . . .	34
4.3	Agent representation. . . . .	35
4.4	Learning loop and evaluation representation. . . . .	37
4.5	WQNN model architecture [9]. . . . .	40
5.1	Weather station A606 location. . . . .	43
5.2	Adjustments on global horizontal irradiance measurements. . . . .	44
5.3	Inputs samples distribution. . . . .	45
5.4	Best state-value models timeseries. . . . .	49
5.5	Average horizontal ratio given the learning ratio on evaluation step. . . . .	49
5.6	Average horizontal ratio given the learning ratio on training iterations. . . . .	50
5.7	Average horizontal ratio given the decay ratio on evaluation step. . . . .	51
5.8	Average horizontal ratio given the decay ratio on training iterations. . . . .	51
5.9	Average horizontal ratio given the greed rate on evaluation step. . . . .	52

5.10	Average horizontal ratio given the greedy rate on training iterations. . .	53
5.11	Best policy search models timeseries. . . . .	54
5.12	Average horizontal ratio given the decay rate on evaluation step. . . .	55
5.13	Average horizontal ratio given the decay rate on training iterations. . .	55
5.14	Average horizontal ratio given the learning rate on evaluation step. . .	56
5.15	Average horizontal ratio given the learning rate on training iterations.	57

# List of Tables

5.1	Features of the public weather station A608 dataset, as provided by INMET. . . . .	43
5.2	Applicable parameters per model . . . . .	47
5.3	Top 5 n-Tuple SARSA . . . . .	48
5.4	Top 5 WQNN . . . . .	48
5.5	REINFORCE . . . . .	53
5.6	ACTOR CRITIC . . . . .	53
5.7	Top 10 Configurations Based on Performance Metrics . . . . .	57

# Chapter 1

## Introduction

The energy transition is a key element in addressing climate change, as fossil fuel-based electricity generation remains one of the largest contributors to carbon emissions. To achieve the goal of creating a low-carbon society, alternative energy sources must become increasingly efficient. This improvement not only ensures the feasibility of the transition but also positions it as a potentially profitable endeavor.

Addressing climate change is recognized by the United Nations as one of humanity's most pressing challenges [1]. Renewable energy sources play a crucial role in meeting this challenge by providing a sustainable solution to the growing global energy demand. According to projections from the International Energy Agency (IEA), renewable energy will become the largest source of electricity worldwide by 2025. Solar photovoltaics (PV), in particular, is the fastest-growing renewable energy technology, and by 2027, its installed capacity is expected to surpass all other sources of electricity generation globally [10].

Solar PV has emerged as one of the most promising alternatives to fossil fuels, offering one of the lowest Levelized Costs of Energy (LCOE) in the industry [11]. Despite its progress, there remains significant room for improvement. A primary challenge lies in maximizing the energy output given the available solar irradiance and the installed energy capacity. Enhancing the efficiency of solar PV systems is essential to making them a more competitive and sustainable energy source.

Solar tracking algorithms have been developed as a strategy to optimize energy capture by dynamically adjusting the orientation of PV panels to ensure that their surface remains perpendicular to the Sun's rays [12]. However, practical challenges arise in implementing these systems, particularly in scenarios with irregular terrain or obstacles, which can cause shading and significantly reduce module efficiency.

Research in PV solar systems focuses on improving overall production within the inherent efficiency limits of solar cells [2]. Sun tracking systems have been identified as a key solution to this challenge. By continuously adjusting the orientation of PV panels to follow the Sun's trajectory, these systems increase the irradiance on the

plane of the array, thereby generating more electricity. However, optimizing tracker positions becomes particularly challenging in cloudy environments. Studies have shown that while direct sun tracking can lead to higher energy yields over extended periods, it may result in losses during overcast days [13, 14].

Weightless Neural Networks (WNNs) are a class of machine learning models based on kernel methods. These models offer significant advantages over traditional weighted Artificial Neural Networks (ANNs), including greater flexibility and lower computational costs [6, 15]. WNNs are particularly well-suited for applications in solar tracking optimization, where the need for efficient, and real-time decision-making aligns with their lightweight computational framework.

By leveraging WNNs, solar tracking systems have the potential to adapt more effectively to dynamic environmental conditions, such as cloud cover or variable irradiance. This adaptability allows systems to operate closer to their optimal efficiency, even under less-than-ideal weather conditions. The integration of WNNs into solar tracking strategies represents a promising direction for future research, addressing both the practical challenges of shading and the broader goal of enhancing solar PV efficiency.

## 1.1 Related works

The work reported by AL-ROUSAN *et al.* proposed a detailed framework for classifying solar tracking technologies. Among these, the majority of current systems are categorized as active trackers, which utilize actuators controlled by algorithms to dynamically adjust the orientation of photovoltaic modules. Within active tracking, MUSTAFA *et al.* identified three primary strategies for control. Astronomical tracking relies on pre-calculated solar position data to determine the optimal alignment of modules based on the Sun's trajectory. Image processing tracking uses visual input from cameras to detect the Sun's position and adjust module orientation accordingly. In contrast, LDR-based tracking employs light-dependent resistors placed along the edges of photovoltaic modules, which provide real-time feedback to optimize alignment.

Recent advancements in machine learning (ML) have introduced new possibilities for enhancing solar tracking systems. According to PHIRI *et al.*, deep learning techniques have been increasingly explored in solar tracking, with applications often centered on simulated scenarios. Despite the promise shown by these methods, significant challenges remain, such as the lack of publicly available datasets and standardized evaluation metrics. Among the most common ML-based approaches are computer vision techniques for Sun detection and hybrid control systems that combine multiple tracking strategies, including LDR sensors, astronomical calculations,

and energy output monitoring, to achieve improved performance.

Reinforcement learning (RL), as defined by SUTTON e BARTO, is a promising methodology for optimizing control in systems modeled as Markov Decision Processes. However, its application to solar tracking remains under-explored. In related domains, PATARO *et al.* applied Q-Learning to optimize a Concentrated Solar Power (CSP) system, achieving superior performance by relying solely on plant measurements and avoiding reliance on complex system models. Similarly, ALAMRO *et al.* introduced a framework using Deep Q-Learning to optimize workload distribution in photovoltaic systems, demonstrating the flexibility of RL techniques in energy management.

The use of n-tuple models in reinforcement learning has also been investigated by KATOPODIS, who demonstrated their efficacy in achieving benchmark results with reduced computational costs and faster inference times. These qualities make n-tuple models particularly attractive for implementation on devices with limited computational resources, emphasizing their potential for lightweight and efficient solar tracking solutions. This body of research highlights the growing role of ML and RL in advancing solar tracking technologies while addressing the practical challenges of efficiency, adaptability, and computational constraints.

## 1.2 Goals and contributions

The research question of this work is to investigate the feasibility of the application of Weightless Neural Networks (WNNs) and reinforcement learning in controlling solar tracking systems to optimize the power output of photovoltaic (PV) panels. By leveraging WNNs, the aim is to design efficient and adaptable control algorithms that enhance energy capture without requiring additional hardware sensors. Supervised models are not suitable for the current application, since in a real scenario the optimal position is not known.

This research is structured around two specific goals and two key contributions:

### Specific Objectives

#### 1. Application of Reinforcement Learning for Solar Trackers Without Extra Sensors

Explore the use of reinforcement learning (RL) techniques to optimize the orientation of solar trackers. The goal is to eliminate the dependency on additional hardware sensors by using environment-derived data and RL algorithms to maximize energy capture.

#### 2. Integration of Weightless Neural Networks in Industrial Contexts

Investigate and validate the potential of WNNs in industrial-scale applications. This includes demonstrating their feasibility, efficiency, and adaptability in controlling solar tracking systems.

## Key Contributions

1. **Development and Open Publication of the Simulator Code** Design and implement a comprehensive simulation environment for evaluating solar tracker performance. The simulator integrates realistic weather models and PV system behavior to test and benchmark various control algorithms. The code is made publicly available to support further research and innovation in this domain.
2. **Development of a Q-Learning Algorithm Using N-Tuple Models (WQNN)** Create and implement a novel Q-Learning algorithm tailored for n-tuple models, referred to as the Weightless Q-Learning Neural Network (WQNN). This algorithm combines the advantages of reinforcement learning with the lightweight computational nature of WNNs, offering an efficient solution for real-time solar tracker optimization.

This work establishes a foundation for integrating advanced machine learning techniques in PV systems, contributing to the broader goal of enhancing renewable energy technologies.

## 1.3 Text Structure

Chapter 2 introduces the industrial application context, emphasizing the critical challenge of maximizing the irradiance received by photovoltaic (PV) modules to enhance their energy output. This sets the stage for the subsequent exploration of potential solutions.

In Chapter 3, the theoretical foundation underpinning the proposed solution is presented. This chapter is organized into three key sections. The first section focuses on reinforcement learning, explaining its principles and relevance to optimization problems. The second section delves into n-tuple models, a computational framework that supports the development of efficient algorithms. The final section integrates these paradigms, demonstrating how reinforcement learning and n-tuple models can work together to address the challenges of solar tracking.

Chapter 4 outlines the methodology employed in this work, providing detailed descriptions of the systems developed to evaluate the algorithms. This includes the construction of a simulation environment tailored to the unique requirements

of the study. A dedicated section elaborates on the Weightless Q-Learning Neural Network, a novel approach designed and implemented during this research.

The results of the experimental evaluations are presented in Chapter 5, offering insights into the performance of the proposed algorithms. This chapter analyzes the effectiveness of the solutions in addressing the outlined challenges and compares their performance against baseline methods.

Finally, Chapter 6 synthesizes the findings of the study, highlighting the main conclusions and their implications. This chapter also identifies avenues for future research, proposing ways to extend and refine the current work to further advance the field.

# Chapter 2

## Application domain knowledge

Solar photovoltaic (PV) technology has emerged as a cornerstone for promoting sustainability and generating clean electricity. While advancements in PV materials have been made, the primary challenge to broader adoption lies in improving efficiency. A solar cell operates by converting light energy directly into electricity through the photovoltaic effect. The electrical characteristics of a solar cell - such as current, voltage, and resistance - vary based on the intensity and nature of light energy received, whether from natural sunlight or artificial sources. These cells are the foundational building blocks of photovoltaic modules.

### 2.1 Irradiance to electricity conversion



Figure 2.1: PV Module [3].

Solar cells (illustrated on Figure 2.1) are categorized based on their structural

composition. Single-junction solar cells consist of a single layer of light-absorbing material, whereas multi-junction solar cells employ multiple physical layers to optimize absorption and charge separation mechanisms. Monocrystalline cells currently achieve efficiencies of approximately 16%. Advances in material engineering and fabrication techniques have pushed multicrystalline silicon solar cell efficiency to 19.8%, while enhanced monocrystalline cells now reach efficiencies as high as 24.4% [20, 21].

The global irradiance on the tilted plane of a PV module, as determined by an irradiance processor model chain, is a critical input for the subsequent electrical simulation of PV systems. This parameter directly influences the output characteristics of PV cells. Specifically, the output current exhibits an approximately linear dependence on the incident irradiance, while the output voltage follows a logarithmic-like dependency. Consequently, the power output of PV modules is almost linearly dependent on irradiance levels under moderate to high illumination conditions [22]. This nearly linear relationship implies that the annual energy yield of PV modules is strongly correlated with the integral of irradiance over time on the module surface.

The conversion efficiency of solar modules is inherently limited by both electrical and optical losses. Optical losses are particularly influenced by the angle of light incidence relative to the module plane. The angle of incidence is governed by factors such as the orientation of the PV module, the time of year, and the geographical location. A suboptimal incidence angle can result in significant reflection losses, thereby reducing the amount of light absorbed by the module and ultimately affecting energy conversion efficiency. Understanding and mitigating these losses is critical for maximizing the performance of PV systems.

## 2.2 Irradiance and solar positioning

To evaluate the energy available for a solar system, it is crucial to understand the variation in the Sun's position relative to a specific location on Earth's surface. The amount of energy absorbed by a photovoltaic system directly depends on the angle of incidence between the Sun's rays and the module surface.

The Earth follows an elliptical orbit around the Sun, a movement known as translation, while simultaneously rotating around its axis. Due to the irregularities in Earth's shape, neither rotation nor revolution forms a perfect circle, introducing complexities in accurately predicting the Sun's trajectory.

Modeling the Sun's position relative to a specific point on Earth's surface is essential for predicting the incident irradiance on photovoltaic systems. The energy absorbed is proportional to the inner product of the solar beam vector and the surface's normal vector.

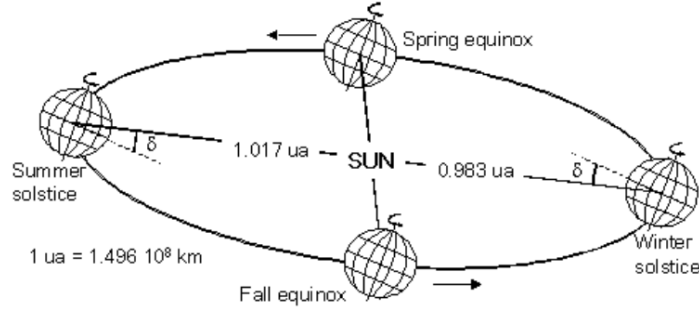


Figure 2.2: Illustration of Earth's translation and rotation [4].

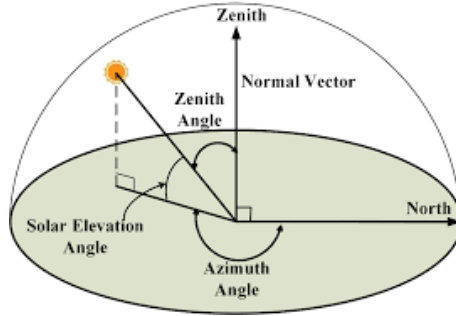


Figure 2.3: Solar zenith and azimuth angles [5].

To determine the Sun's position in the celestial sphere, two angles are typically used: zenith and azimuth. The zenith angle describes the Sun's elevation relative to the vertical, while the azimuth angle specifies its direction relative to true north. These angles, illustrated in Figure 2.3, are calculated using astronomical algorithms that account for Earth's rotation, revolution, and atmospheric refraction.

There are a few methods to model and estimate the solar angle, but we opted to focus on the astronomical algorithm published by the National Renewable Energy Laboratory U.S. (NREL) [23]. The advantage of this strategy is possibility to estimate the solar zenith and azimuth with uncertainties of  $\leq 0.01$ .

## 2.3 Sun tracking algorithm

To enhance solar energy collection, photovoltaic (PV) systems often incorporate sun tracking technologies. These systems use motors controlled by open-loop algorithms to adjust the module orientation based on solar position estimations. The goal is to optimize the Plane of Array Irradiance (POA), which is the sum of the direct and diffuse irradiance incident on the surface of the solar module, as described by Equation 2.1 [24].

$$POA = Irr_{direct} + Irr_{diffuse} \quad (2.1)$$

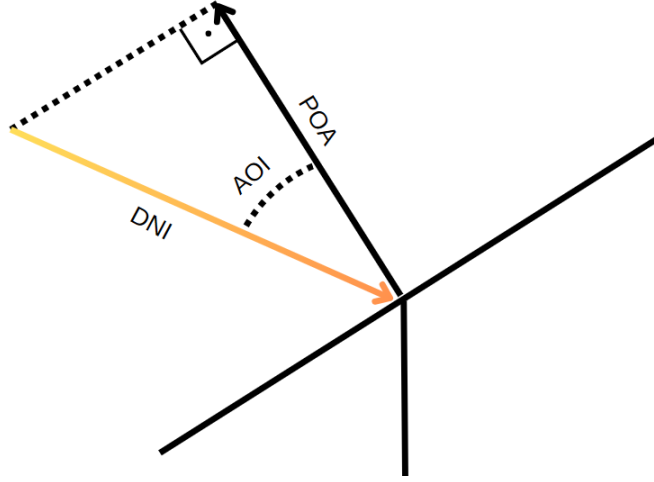


Figure 2.4: Direct component of the plane of array irradiance.

Here  $Irr_{direct}$  and  $Irr_{diffuse}$  are the direct and diffuse irradiance, respectively.

$$Irr_{direct} = dni * \cos(aoi) \quad (2.2)$$

Direct irradiance corresponds to sunlight arriving directly from the Sun's beam and is influenced by the angle of incidence ( $aoi$ ) between the Sun's rays and the PV module. The magnitude of direct normal irradiance ( $dni$ ) determines its intensity. Direct irradiance on a tilted surface is calculated using Equation 2.2, as illustrated in Figure 2.4.

$$aoi = \arccos(\cos(a) \cdot \cos(z_{e_{solar}}) + \sin(a) \cdot \sin(z_{e_{solar}}) \cdot \cos(az_{tracker} - az_{solar})) \quad (2.3)$$

The  $z_{e_{solar}}$  and  $az_{solar}$  on Equation 2.3, are the solar zenith and azimuth estimated by the solar position algorithm (SPA) [23]. The tilt angle  $a$  is the angle of the PV module relative to the horizontal plane, and  $az_{trackers}$  is azimuth of the PV array, typically 0 meaning that the PV array is pointed to the north.

Diffuse irradiance arises from the scattering and reflection of sunlight by atmospheric particles, the ground, and nearby objects. Unlike direct irradiance, it does not have a preferential direction, making its absorption less sensitive to the angle of incidence. Diffuse irradiance is typically modeled as the sum of two components: atmospheric scattering  $Irr_{sky}$  and the ground reflection  $Irr_{ground}$ , resulting on Equation 2.4.

$$Irr_{diffuse} = Irr_{ground} + Irr_{sky} \quad (2.4)$$

Both components are challenging to model analytically due to variability in environmental factors. For instance, atmospheric scattering depends on the concentration of aerosols, while ground reflection varies with the color and type of nearby vegetation.

One of the simplest approaches for estimating  $Irr_{sky}$  is the isotropic model proposed by Hottel and Woertz, which treats the sky as a uniform source of diffuse irradiance. This method calculates  $Irr_{sky}$  using the diffuse horizontal irradiance  $dhi$  and the module tilt angle  $a$ , as described in Equation 2.5 [25].

$$Irr_{sky} = dhi \cdot (1 + \cos(a)) \cdot 0.5 \quad (2.5)$$

Similarly, the ground diffuse irradiance is usually modeled in using the Equation 2.6. It assumes the ground reflection is uniform and proportional global horizontal irradiance  $ghi$ , [26]. The proportionality factor is called albedo  $alb$ , usually assumed to be empiric constant.

$$Irr_{ground} = ghi \cdot alb \cdot (1 - \cos(a)) * 0.5 \quad (2.6)$$

In summary, modeling  $POA$  irradiance involves a combination of direct measurements and theoretical estimates. Accurate predictions rely on understanding the interactions between direct sunlight, atmospheric scattering, and ground reflections.

## 2.4 Shading and Backtracking

One significant challenge in photovoltaic (PV) systems is the premature failure of modules caused by hot spot phenomena under partial shading conditions. Research indicates that under partial shading, PV cells may experience reverse breakdown, leading to localized temperatures as high as 400°C. These extreme temperatures not only degrade PV performance but also cause irreversible damage to the modules. In severe cases, this can result in catastrophic module failure or even fires. The extent of power loss due to shading is highly dependent on factors such as the shading pattern, irradiation levels, geographical location, and time of day. For instance, a single shaded cell in a module can reduce power output by up to 50%, while multiple shaded cells may lead to losses as high as 90% [27].

The primary goal of a tracker control algorithm is to maximize exposure to direct beam irradiance by minimizing the angle of incidence between the beam and the collector surface. However, this approach can lead to inter-row shading when the Sun is low on the horizon, as illustrated in Figure 2.5. When adjacent rows shade each other, significant power losses occur due to the compounded effects of partial

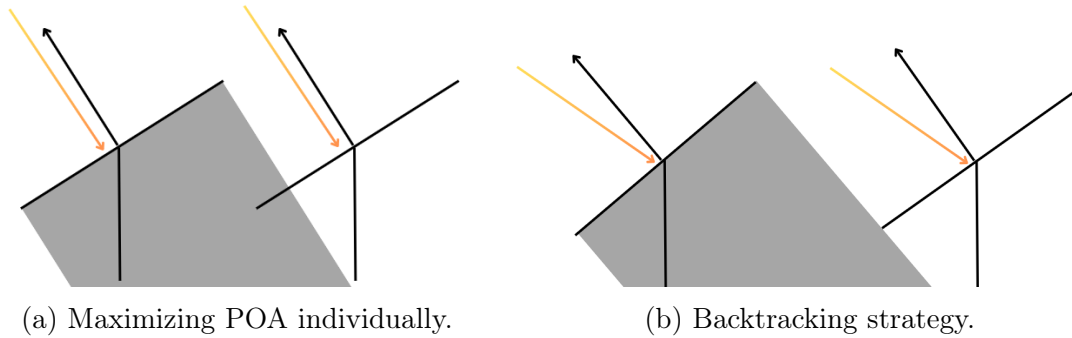


Figure 2.5: Partial shading examples with and without backtracking strategies.

shading, which exacerbate the hot spot phenomenon.

To mitigate this issue, an optimal backtracking rotation strategy is employed. The backtracking geometry is calculated based on the spacing between rows and the instantaneous solar position. Many commercial single-axis tracker systems implement a backtracking geometry similar to that described by Lorenzo, Narvarte, and Muñoz (2011), which assumes the tracker axes are contained within a horizontal plane, meaning no vertical offset is accounted for between rows [24].

While slope-aware backtracking algorithms have been developed to address varying terrain, such algorithms require precise knowledge of the system’s geometry, including the topography. This information is typically obtained through drone imaging in large-scale power plants, where accurate geometric data is challenging to acquire due to the complexity of the terrain. The reliance on precise geometric inputs highlights a limitation in deploying backtracking algorithms across large and uneven installations [24].

## 2.5 Limitation of current tracking solutions

Solar tracking systems (STS) have been extensively studied for their role in enhancing the efficiency of PV panels. These systems dynamically adjust the orientation of PV modules to maximize exposure to direct sunlight, thereby increasing energy output. Despite the significant progress in this area, there remain considerable gaps in understanding how emerging technologies can further optimize power capture. Most research to date has focused on traditional tracking systems, often neglecting the integration of advanced predictive algorithms and Internet of Things (IoT) technologies [28].

Existing studies have demonstrated that solar tracking systems can enhance energy output by improving the alignment of PV modules with the Sun’s position. However, these systems frequently lack the real-time adaptability required to respond effectively to varying environmental conditions, such as cloud cover, at-

atmospheric scattering, and sudden weather changes. This limitation restricts their efficiency, particularly in regions with dynamic and unpredictable climatic patterns [28].

Recent advancements in machine learning (ML) and artificial intelligence (AI) present a promising avenue for overcoming these limitations. These technologies can be leveraged to develop sophisticated control algorithms for solar tracking systems, enabling real-time adjustments based on predictive analytics and environmental data. By incorporating weather forecasts, irradiance predictions, and atmospheric models, AI-driven solar trackers can anticipate environmental changes and optimize their orientation proactively. Research in this domain remains nascent but holds transformative potential.

By addressing the limitations of traditional solar tracking systems through AI and ML integration, future developments can pave the way for more efficient, adaptive, and resilient solar energy solutions [28]. This approach not only maximizes energy yield but also enhances the sustainability and reliability of solar power generation in diverse environmental settings.

The next chapter describes the Reinforcement Learning, a machine learning paradigm focus on optimizing systems using real-time feedback.

# Chapter 3

## Background knowledge

This chapter provides the theoretical foundation that supports the proposed solution. In the context of optimizing photovoltaic performance through enhanced solar tracking, a clear understanding of the underlying principles is essential. Accordingly, the chapter is divided into three sections.

The first section discusses reinforcement learning by outlining its core principles and explaining its relevance to optimization problems. It examines the mechanisms of adaptive decision-making in dynamic and uncertain environments.

The second section presents n-tuple models, a computational framework known for its efficiency and scalability in algorithm development. This section reviews the structural and operational characteristics of n-tuple models and their role in forming effective solution strategies for solar tracking.

The final section integrates reinforcement learning with n-tuple models to address the challenges of real-world solar tracking systems. This synthesis provides the necessary background for the methodological developments and experimental evaluations detailed in subsequent chapters.

### 3.1 Reinforcement learning

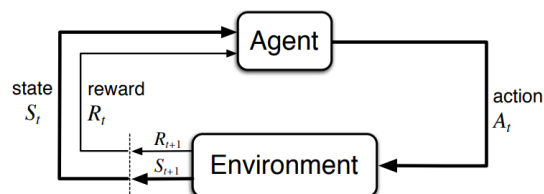


Figure 3.1: Markov decision process representation [6].

Reinforcement learning (RL) is a branch of machine learning focused on optimizing decision-making processes based on feedback from the environment. Its

mathematical background is based on the Markov Decision Process (MDP), illustrated in Figure 3.1. In an MDP, the environment represents the context in which an agent operates. The agent selects actions that affect the environment, leading to state transitions and generating feedback in the form of rewards  $R$ . The goal of the agent is to learn a strategy that maximizes the cumulative reward, denominated return  $J$ . The agent perceives the environment through a state representation, which serves as the basis for decision-making.

"Reinforcement learning problems involve learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. In an essential way they are closed-loop problems because the learning system’s actions influence its later inputs." [6]

A learning agent is an autonomous system designed to perform sequential actions within a given environment. Its primary objective is to learn a policy  $\pi$ , which defines a probability on taking action  $a$  giving an state  $s$ , in order to maximize the return.

The agent evaluates its decisions using two key functions, the state-value and action-state-value function, Equations 3.1 and 3.2. Both represent the expected return, giving the state  $s$ , and of taking the action  $a$  giving the state  $s$  and the policy  $\pi$ .

$$V_{\pi}(s) = E_{\pi} \left[ \sum_{i=0}^T R_{t+i+1} \mid S_t = s \right] \quad (3.1)$$

$$Q_{\pi}(a, s) = E_{\pi} \left[ \sum_{i=0}^T \gamma^i R_{t+i+1} \mid S_t = s, A_t = a \right], \quad (3.2)$$

Where  $T$  is the length of the sequence decisions, known as episode. In the scenarios where has not a determinate end, it is introduced decay rate  $\gamma$  that reduces the importance of future rewards, and the previous equations assume the forms of Equations 3.3 and 3.4

$$V_{\pi}(s) = E_{\pi} \left[ \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \mid S_t = s \right] \quad (3.3)$$

$$Q_{\pi}(a, s) = E_{\pi} \left[ \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \mid S_t = s, A_t = a \right], \quad (3.4)$$

The state-value function can be expressed recursively using the Bellman Equation 3.5, where  $s'$  is the next state. It provides a basis for iterative methods to compute  $V_{\pi}$ .

$$\begin{aligned}
V_\pi(s) &= E_\pi \left[ \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \middle| S_t = s \right] \\
&= E_\pi \left[ R_{t+1} + \gamma \sum_{i=0}^{\infty} \gamma^i R_{t+i+2} \middle| S_t = s \right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[ r + \gamma E_\pi \left[ \sum_{i=0}^{\infty} \gamma^i R_{t+1+i+1} \middle| S_t = s' \right] \right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma V_\pi(s')]
\end{aligned} \tag{3.5}$$

For reaching the best return, it is required to find the optimal policy  $\pi^*$ , that has the requirement on Equation 3.6.

$$V_{\pi^*}(s) \geq V_\pi(s) \quad \forall s \in S \tag{3.6}$$

Even though there may exist more than 1 optimal policy, all of them share the same optimal state-value and action-state function [6], referred to in Equations 3.7 and 3.8.

$$V_*(s) = \max_{\pi} V_\pi(s) \tag{3.7}$$

$$Q_*(s, a) = \max_{\pi} Q_\pi(s, a) \tag{3.8}$$

### 3.1.1 Learning the state-value function

Temporal difference (TD) control algorithm [6] learn directly from raw experience without a model of the environments dynamics. Given some experience following a policy, they update their estimate  $V$  of  $v$  for the non terminal states  $S_t$  occurring in that experience.

In the simplest form, it can be sintetased on Equations 3.9 and 3.10

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{3.9}$$

$$V(S_t) \leftarrow V(S_t) + \alpha \cdot \delta_t = V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{3.10}$$

---

**Algorithm 1** Q-learning

---

**Inputs:** Policy given a  $Q$  function and a state  $S$  -  $Policy(Q, S)$   
Initialization method to  $Q$  function -  $Initialize()$   
Environment update method -  $Environment(S, A)$   
Decay rate -  $\gamma$   
Learning rate -  $\alpha$

$Q(S, A) \leftarrow Initialize()$   
**for** episode from 1 to N **do**  
   $S_t \leftarrow S_0$   
  **for** t from 1 to T **do**  
     $A_t \leftarrow Policy(Q, S_t)$   
     $R_{t+1}, S_{t+1} \leftarrow Environment(S_t, A_t)$   
     $Q(S_t, A_t) \leftarrow (1 - \alpha) \cdot Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a)]$   
     $S_t \leftarrow S_{t+1}$   
  **end for**  
**end for**

---

### 3.1.2 Q-Learning

Q-learning is an off-policy temporal difference (TD) control algorithm [6]. Which means that it does not consider the effect of current policy on the learning process. Each state-action pair is stored in a Q-table and updated incrementally using a learning rate  $\alpha \in (0, 1]$  based on feedback from Temporal Difference learning.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \quad (3.11)$$

Here,  $Q(S_t, A_t)$  is updated based on the reward  $R_{t+1}$  = received after taking action  $A_t$  in state  $S_t$ , along with the maximum Q-value of the next state  $S_{t+1}$ . The learned Q-value approximates  $Q^*$ , the optimal action-value function, independent of the current policy. This independence simplifies analysis and supports convergence proofs.

While the policy influences which state-action pairs are visited and updated, correct convergence only requires that all pairs are continuously updated.

### SARSA

Algorithm 2 presents the SARSA (State-Action-Reward-State-Action) algorithm, a temporal-difference control method that operates in an on-policy manner. Unlike Q-learning, which is off-policy, SARSA continuously updates its estimate of the action-value function  $Q(s, a)$  based on the actions actually taken by the current behavior policy. This process enables the algorithm to adapt the policy toward

actions with higher Q-values as it interacts with the environment, thus gradually refining its decision-making capability [7].

---

**Algorithm 2** SARSA

---

**Inputs:** Policy given a  $Q$  function and a state  $S$  -  $Policy(Q, S)$   
Initialization method to  $Q$  function -  $Initialize()$   
Environment update method -  $Environment(S, A)$   
Update of model  $x$ , given output  $k$ , and inputs  $S$  and  $A$  -  $Update(x, S, A, k)$   
Decay rate -  $\gamma$   
Learning rate -  $\alpha$

$Q(S, A) \leftarrow Initialize()$   
**for** episode from 1 to  $N$  **do**  
     $S_t \leftarrow S_0$   
    **for**  $t$  from 1 to  $T$  **do**  
         $A_t \leftarrow Policy(Q, S_t)$   
         $R_t, S_{t+1} \leftarrow Environment(S_t, A_t)$   
         $A_{t+1} \leftarrow Policy(Q, S_{t+1})$   
         $Q(S_t, A_t) \leftarrow (1 - \alpha) \cdot Q(S_t, A_t) + \alpha [R_t + \gamma Q(S_{t+1}, A_{t+1})]$   
         $S_t \leftarrow S_{t+1}$   
    **end for**  
**end for**

---

The pseudo-code in Algorithm 2 begins with the initialization of the  $Q(S, A)$  matrix, typically set to zero. For each episode, the algorithm starts from an initial state  $S_0$ , and for each time step, an action  $A_t$  is selected based on the current policy, often implemented as an  $\epsilon$ -greedy strategy. The environment then returns a reward  $R_t$  and the subsequent state  $S_{t+1}$ , and the policy is used to determine the next action  $A_{t+1}$ . This transition allows the algorithm to update the Q-value associated with the state-action pair  $(S_t, A_t)$ , thereby incorporating the immediate reward and the estimated future rewards. The state is then updated, and the process repeats until the end of the episode.

The SARSA algorithm guarantees convergence to an optimal policy with probability 1, provided that all state-action pairs are visited infinitely often and the policy converges to a greedy policy over time. In practice, this can be achieved by gradually decreasing the exploration parameter  $\epsilon$ , for instance by setting  $\epsilon = \frac{1}{t}$ , although this theoretical result has not been formally published [6]. This approach ensures that the learning process remains focused on exploiting the best-known actions while still allowing for sufficient exploration during the early stages of training.

### 3.1.3 Learning the decision policy

Directly optimizing the policy provides a more intuitive and flexible framework for addressing reinforcement learning (RL) problems [6]. Unlike value-based methods,

which indirectly influence action selection through value functions, direct policy optimization leverages parameterized probability distributions to model the agent’s policy. The selection of an appropriate distribution family is influenced by the characteristics of the task’s action space, ensuring that the distribution’s support closely aligns with the environment’s action space [7].

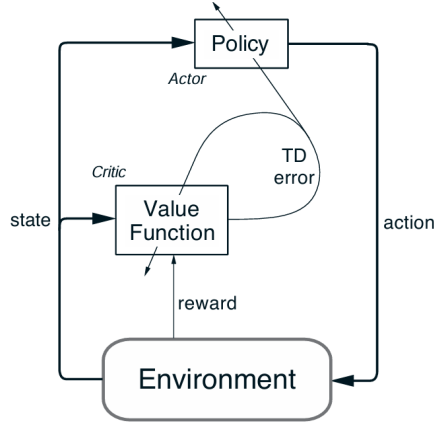


Figure 3.2: Actor-critic diagram [7]

Actor-Critic methods utilize two distinct structures: the actor, which explicitly represents the policy and selects actions, and the critic, which estimates the value function and provides feedback to the actor. The critic evaluates the quality of the actions taken by computing the Temporal Difference (TD) error, a crucial signal that guides the learning process. The TD error is defined as:

$$\delta_t = R_{t+1} + \gamma V_t(S_{t+1}) - V(S_t) \quad (3.12)$$

This TD error serves as a critique of the action  $A_t$  chosen in state  $S_t$ , helping the actor adjust its policy accordingly. Actions in the actor-critic framework are often generated using the Gibbs softmax method, which parametrizes the policy as follows:

$$\pi_t(a|s) = Pr\{A_t = a|S_t = s\} = \frac{e^{H_t(s,a)}}{\sum_b e^{H_t(s,b)}} \quad (3.13)$$

The  $H_t(s, a)$  represents the modifiable policy parameters at time  $t$ , indicating the preference for selecting action  $a$  in state  $s$ . These preferences are adjusted iteratively using the TD error. The simplest update rule is given by

$$H_{t+1}(S_t, A_t) = H_t(S_t, A_t) + \beta \delta_t \quad (3.14)$$

Alternatively, the update can include a correction term to account for the probability of selecting the current action under the existing policy:

$$H_t(S_t, A_t) = H_t(S_t, A_t) + \beta \delta_t [1 - \pi_t(A_t|S_t)] \quad (3.15)$$

In this formulation, the parameter  $\beta$  controls the step size of the updates. The update rule ensures that actions associated with positive TD errors—indicating better-than-expected outcomes—are reinforced, while the probabilities of suboptimal actions are gradually reduced. This dynamic adjustment strikes a balance between exploration and exploitation, allowing the agent to improve its policy over time and converge toward optimal behavior.

## 3.2 n-Tuple based models

The origins of the N-tuple Neural Network (NTNN) can be traced back to 1959, with the pioneering work of Bledsoe and Browning. NTNN-based models are categorized as Non-Parametric General Regression Models. These models aim to approximate any arbitrary function based solely on observed samples, making them versatile tools in machine learning [29]

NTNNs have been effectively applied in a variety of domains, including pattern recognition and function approximation. Their distinguishing features include:

1. Single-Layer Architecture: Simplifies network design while maintaining high functionality.
2. Capability for Non-Linear Mappings: Can model highly non-linear relationships between inputs and outputs.
3. Operational Simplicity: Easy to implement and train compared to deeper, more complex architectures.

According to Kolcz and Allinson, a modified NTNN architecture can function as a non-parametric kernel regression estimator. This enhancement allows NTNNs to approximate complex probability density functions (pdfs) or, in the limiting case, deterministic arbitrary function mappings [30].

Further advancements, originally proposed by Bledsoe and Bisson, involve recording the relative frequencies at which nodal memories are accessed during training. This frequency-based approach enhances the network’s ability to generalize by leveraging observed patterns more effectively [29].

The evolution of NTNNs showcases their potential as flexible, non-parametric tools for regression and classification. Their ability to approximate complex functions and probability distributions makes them valuable in scenarios requiring high adaptability and simplicity. These characteristics continue to drive their application in modern machine learning research and practice.

### 3.2.1 WiSARD

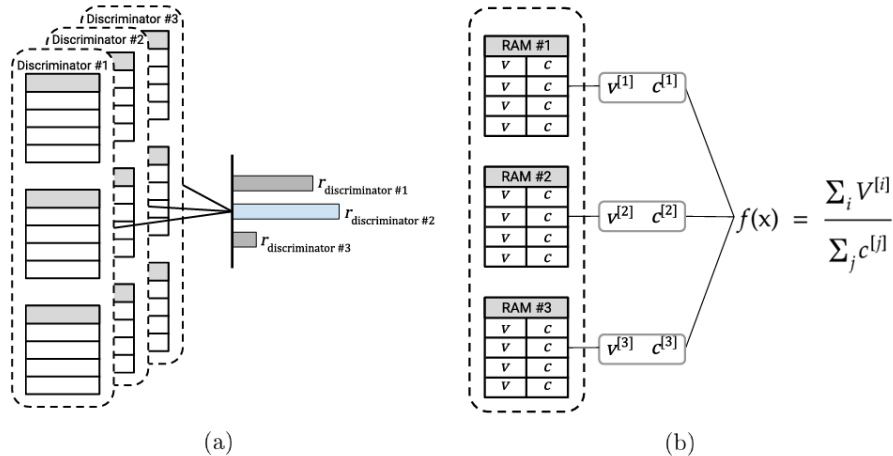


Figure 3.3: WiSARD schema [7]

The WiSARD (Wilkie, Stonham, and Aleksander Recognition Device) is a lightweight, supervised learning model derived from the N-tuple classifier [31]. It enables a straightforward implementation and a fast learning process, facilitated by a simple RAM access mechanism [32].

WiSARD’s architecture is tailored for multiclass classification, where each target class is represented by a structure called a discriminator. A discriminator consists of  $N$  RAM nodes, each addressed by a random n-tuple derived from the input. Unlike weighted neural networks, where knowledge is stored in the synaptic connections, WiSARD stores learned patterns directly in its memory nodes, giving rise to its classification as a RAM-based neural network [33].

#### Training Process

Training begins with the initialization of all memory positions to zero. For each training observation and its associated class.

The network selects the corresponding discriminator for the target class, updating every memory position addressed by the input random tuple by setting the value to 1. At the end of training, each memory position contains a 1 if it was accessed at least once during training and 0 otherwise [33].

#### Classification Process

When a new input pattern is presented:

1. Each discriminator computes a similarity score, which quantifies how closely the new pattern matches the patterns it has previously learned;

2. The similarity score is defined by the number of memory positions addressed by the input pattern;
3. The network selects the discriminator with the highest similarity score and assigns its class to the input pattern[33];

The generalization capability of WiSARD is influenced by the number of nodes  $N$  and the tuple size  $n$ . Larger values of  $N$  or smaller values of  $n$  improve generalization by reducing the likelihood of low similarity scores for slightly altered input patterns. This ensures that the classifier remains robust even when new patterns deviate slightly from the ones seen during training [33].

WiSARD’s simple, fast, and memory-efficient design makes it well-suited for tasks requiring rapid learning and decision-making. Its reliance on weightless architecture and RAM-based storage differentiates it from traditional neural networks, enabling unique advantages in pattern recognition and classification tasks.

### 3.2.2 Regression WiSARD

There is an alternative version adapted to perform regression task, called Regression WiSARD [34]. It is composed of a set of  $N$  RAM nodes. It receives discrete inputs and learns by counting the occurrence of  $n$ -tuple patterns and summing the respective output.

The regression network features a powerful one-pass training procedure and its learning is statistically consistent. The major advantage of utilizing the  $N$ -tuple architecture as a regression estimator is the fact that in this realization the training set points are stored by the network implicitly, rather than explicitly, and thus the operation speed remains constant and independent of the training set size. Therefore, the network performance can be guaranteed in practical implementations [30].

Its architecture is the same as the NTNN, consisting of a RAM discriminator where memory positions store a value and a counter, adopting the Equations 3.16 and 3.17 as respective update rules.

$$S \leftarrow y + S \tag{3.16}$$

$$C \leftarrow 1 + C \tag{3.17}$$

### 3.2.3 Binary Encoding Strategies for $n$ -Tuple Models

$N$ -tuple models rely on binary tuple inputs, making them inherently suited for similarity-based learning [7]. However, in many practical applications, the input

variables belong to the domain of real numbers rather than binary values. To bridge this gap, various binarization techniques have been developed. Each encoding method offers distinct characteristics, making them more or less suitable depending on the specific problem context.

One of the most widely used encoding strategies in Weightless Neural Networks (WiSARDs) is the thermometer encoding method. This approach converts scalar numerical values into categorical representations using binary tuples. The core principle behind thermometer encoding is that the Hamming distance between encoded values reflects their relative similarity. Among the most commonly applied thermometer encoding methods are linear, circular, and distributive thermometers, each designed to optimize learning performance in different scenarios.

### Linear Thermometer Encoding

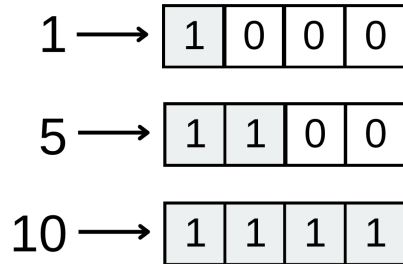


Figure 3.4: Linear thermometer representation, receiving three different inputs (1, 5, and 10), assuming 0 as minimum and 12 as maximum, the space is spitted in 4 clusters of size 3.

The linear thermometer encoding method is conceptually similar to a traditional mercury thermometer. A continuous numerical range is divided into uniformly spaced discrete intervals. The number of active bits in the binary representation increases as the input value progresses through these predefined intervals. This encoding scheme ensures that values belonging to the same interval share an identical binary representation, while values in adjacent intervals exhibit a gradual transition.

Figure 3.4 illustrates a linear thermometer encoding scheme with a thermometer size of 4 for a numerical range spanning from 0 to 12. In this configuration, the first interval covers values from 0 to 3, the second interval spans 3 to 6, the third covers 6 to 9, and values above 9 fall into the final interval. The primary advantage of this approach is its simplicity and ability to preserve the ordinal relationship between input values. However, it assumes that all input values are uniformly distributed, which may not always be the case in real-world datasets.

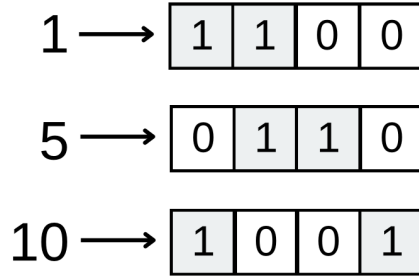


Figure 3.5: Circular thermometer representation, receiving three different inputs (1, 5, and 10), assuming 0 as minimum and 12 as maximum, the space is spitted in 4 clusters of size 3.

### Circular Thermometer Encoding

Circular thermometer encoding follows the same fundamental principle as the linear thermometer method by partitioning the numerical range into discrete intervals. However, instead of simply activating an increasing number of bits, the circular thermometer maintains a fixed number of active bits within a binary vector. These active bits shift within the vector as the input value moves across intervals, creating a continuous wrap-around effect.

In Figure 3.5, the same numerical range as in Figure 3.4 is depicted but encoded using a circular thermometer with a fixed block size of  $y = 2$ . As the input value progresses, the active bits shift within the array until they reach the rightmost boundary. Once this boundary is reached, the active bits wrap around and reposition themselves at the leftmost edge of the vector. This property is particularly beneficial for encoding angular values, where numerical values near  $360^\circ$  should be considered closer to  $0^\circ$  rather than  $180^\circ$ . By preserving circular continuity, this encoding method ensures that proximity relationships in cyclic data are correctly represented.

### Distributive Thermometer Encoding

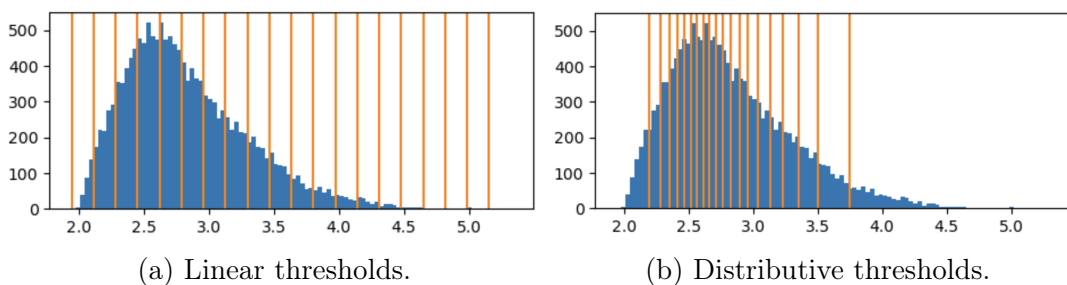


Figure 3.6: Comparison of thermometer encoding thresholds [8]

The distributive thermometer encoding method, first proposed by BACELLAR *et al.*, was specifically designed to enhance the performance of WiSARD-based mod-

els. Unlike linear thermometers, which assume a uniform distribution of data across intervals, distributive thermometers dynamically adjust their threshold values based on the statistical distribution of the dataset. This approach ensures that each encoding interval contains approximately the same number of samples.

The key difference between linear and distributive thermometer encoding is highlighted in Figure 3.6. While the linear method divides the numerical range into equal-width intervals, it can lead to some intervals containing significantly more data points than others. In contrast, the distributive thermometer method concentrates threshold values in regions with higher data density, thereby refining the model’s ability to capture subtle variations in frequently occurring values.

By allocating more encoding granularity to densely populated regions of the input space, distributive thermometer encoding enhances model precision where it matters most. This results in improved learning efficiency and greater differentiation between critical input values, making it particularly advantageous in scenarios where data is skewed or non-uniformly distributed.

### 3.3 Reinforcement learning with n-tuple models

The N-tuple regression network (NTRN), originally designed for supervised learning tasks, operates under assumptions that align well with this context. In supervised learning, it is often assumed that the entire set of samples is available beforehand, and the objective is to uncover a deterministic mapping from these samples. This framework typically presupposes a fixed joint distribution of data. However, these assumptions are poorly suited to reinforcement learning (RL), where the agent learns through interaction with an environment, and the data distribution evolves dynamically as the agent explores and improves its policy.

In RL, the agent begins with no optimal policy and often selects suboptimal actions, resulting in visits to poor states. Over time, as it interacts with the environment, the agent gathers evidence and refines its behavior, gradually improving its policy to maximize cumulative rewards. This characteristic of dynamic, interaction-driven learning fundamentally differs from the static nature of supervised learning, where the training data is fixed and predetermined.

Another limitation of the NTRN arises from its reliance on minimizing the mean squared error (MSE) as its learning objective. The update rule of the model is intrinsically tied to this metric, which may not align well with the goals of RL. Reinforcement learning often requires a more dynamic approach to updates, where recent experiences may need to carry greater weight to adapt quickly to changing policies and environments.

To address these limitations, Katopodis proposed a modified learning model [7],

maintaining the architecture of the Regression WiSARD but introducing a novel update rule and output function. A new hyperparameter, the forgetting factor  $\phi$ , is introduced, defined within the interval  $[0, 1]$ . This factor plays a crucial role in enabling the model to adapt dynamically to recent experiences.

$$C \leftarrow 1 + C \quad (3.18)$$

$$S \leftarrow y + \phi \cdot S \quad (3.19)$$

The proposed update rule modifies the memory content by first multiplying the existing value by the forgetting factor  $\phi$  and then incrementing it with the value of the new observation. Mathematically, this process ensures that the contribution of past observations decays exponentially, while recent observations have a more significant influence on the approximation. This approach not only enforces forgetting but also enhances the model’s ability to adapt to the evolving data distribution in RL scenarios, aligning the learning process with the dynamic nature of reinforcement learning environments.

$$f_t(x) = \begin{cases} \frac{\sum_{i=1}^N v_t^{[i]}(x)}{\sum_{i=1}^N s_t^{[i]}(x)} & \sum_{i=1}^N s_t^{[i]}(x) \neq 0 \\ 0 & \text{Otherwise} \end{cases} \quad (3.20)$$

$$s_t^{[i]}(x) = \begin{cases} \frac{1 - \phi c_t^{[i]}(x)}{c_t^{[i]}(x)} & c_t^{[i]}(x) > 0 \text{ and } \phi \neq 1 \\ c_t^{[i]}(x) & \text{Otherwise} \end{cases} \quad (3.21)$$

Other issues is that Action-value functions take two inputs, while n-tuple regression architectures expect a single binary pattern. One possible solution that merits exploration is to jointly encode state and action into a pattern. However, a much simpler approach is handling state and action separately. The separation could be understood as modeling a state-value function for each possible action. With n-tuple systems, this means having multiple RAM discriminators. Alternatively, a single discriminator could be used, but its memory positions would store vectors of lengths corresponding to the number of actions.

### 3.3.1 SARSA Weightless Neural Network

The SARSA Weightless Neural Network extends the core principles of the tabular SARSA algorithm into a neural network framework, incorporating the advantages of weightless neural architectures. This approach utilizes  $m$ -step bootstrapped returns as update targets, allowing it to balance learning stability and responsiveness. By embedding SARSA within a neural framework, the model achieves greater scalability

and flexibility while maintaining the essential on-policy learning characteristics of its tabular counterpart.

In this formulation, policies are implicitly defined through the action-value function or its approximation. The action selection process adheres to the  $\epsilon$ -greedy exploration strategy, where the agent predominantly chooses the action with the highest estimated value but occasionally explores alternative actions to ensure adequate exploration of the state-action space.

The  $m$ -step variation of the algorithm introduces a delay in updating the model until  $n$  steps have elapsed. This bootstrapped update mechanism enhances the stability of the learning process, especially in environments where rewards are delayed or sparse. Such a design ensures that the agent has sufficient time to observe the consequences of its actions before making adjustments to its internal models.

Algorithm 3 details the implementation of the n-Tuple  $m$ -step SARSA algorithm. Initially, the function  $f(S, A)$ , which represents the output of the Regression WiSARD model adapted for reinforcement learning, is initialized. For each episode, the agent begins at an initial state  $S_0$  and selects an initial action  $A_0$  according to the current policy derived from  $f$ . As the episode unfolds, the agent interacts with the environment: at each time step, it receives a reward and transitions to a new state  $S_{t+1}$ , and then selects a subsequent action  $A_{t+1}$  using the same policy. After accumulating rewards over  $m$  steps, the algorithm computes a return  $G$  that integrates these rewards, possibly bootstrapped with the estimated Q-value at a future state. This return is then used to update the function  $f$  at the corresponding state-action pair. The iterative updating continues until the episode terminates, progressively refining the agent’s approximation of the optimal action-value function.

### 3.3.2 n-Tuple REINFORCE

The n-Tuple REINFORCE algorithm is a policy gradient method that integrates the principles of the REINFORCE algorithm into the n-Tuple framework. In this approach, the policy parameters are optimized through gradient-based methods to maximize the expected return starting from the initial state. Given that the n-Tuple network is designed to minimize its objective function, the algorithm applies a negative gradient to align the minimization process with the goal of maximizing cumulative rewards. This inversion ensures that updates to the model parameters drive the policy towards selecting actions that yield higher returns.

Mathematically, the objective function for the n-Tuple REINFORCE algorithm is expressed in Equation 3.22.

$$R[f_t, S_t, A_t, G_t] = -G_t \ln \pi_{f_t}(A_t|S_t) \quad (3.22)$$

---

**Algorithm 3** n - Tuple m-step SARSA

---

**Inputs:** Policy given a  $Q$  function and a state  $S$  -  $Policy(Q, S)$   
Initialization method to  $f$  function -  $Initialize(\phi)$   
Environment update method -  $Environment(S, A)$   
Update of model  $x$ , given output  $k$ , and inputs  $S$  and  $A$  -  $Update(x, S, A, k)$   
Forgetting factor -  $\phi$   
Decay rate -  $\gamma$   
Learning rate -  $\alpha$

```
 $f \leftarrow Initialize(\phi)$ 
for episode from 1 to N do
   $S_t \leftarrow S_0$ 
   $A_0 \leftarrow Policy(f, S_t)$ 
   $t \leftarrow 0$ 
   $\tau \leftarrow -\infty$ 
   $T \leftarrow \infty$ 
  while  $\tau < T - 1$  do
    if  $t < T$  then
       $R_t, S_{t+1} \leftarrow Environment(S_t, A_t)$ 
      if  $S_{t+1}$  is terminal then
         $T \leftarrow t + 1$ 
      else
         $A_{t+1} \leftarrow Policy(f, S_{t+1})$ 
      end if
    end if
     $\tau \leftarrow t - m + 1$ 
    if  $\tau \geq 0$  then
       $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+m, T)} \gamma^{i-\tau-1} R_i$ 
      if  $\tau + m < T$  then
         $G \leftarrow G - \gamma^m f(S_{\tau+m}, A_{\tau+m})$ 
      end if
       $f = Update(f, S_\tau, A_\tau, G)$ 
    end if
     $t \leftarrow t + 1$ 
  end while
end for
```

---

Where  $G_t$  is the return from time  $t$ , and  $\pi_{f_t}(A_t|S_t)$  represents the probability of taking action  $A_t$  in state  $S_t$  given the current policy parameters  $f_t$ . The gradient of the objective with respect to  $f_t$  is computed in Equation 3.23.

$$\Delta_{f_t} R[f_t, S_t, A_t, G_t] = -G_t \Delta_{f_t} \ln \pi_{f_t}(A_t|S_t) \quad (3.23)$$

This formulation ensures that when the return  $G_t$  is high, the probability of the corresponding action is increased, while a low return will decrease that probability. Such a mechanism drives the policy to converge towards more rewarding actions over time, effectively improving performance through gradient descent in the weightless neural network framework.

---

**Algorithm 4** n - Tuple REINFORCE

---

**Inputs:** Policy given a  $Q$  function and a state  $S$  -  $Policy(Q, S)$   
Initialization method to  $g$  function -  $Initialize(\phi)$   
Environment update method -  $Environment(S, A)$   
Update of model  $x$ , given output  $k$ , and inputs  $S$  and  $A$  -  $Update(x, S, A, k)$   
Forgetting factor -  $\phi$   
Decay rate -  $\gamma$   
Learning rate -  $\alpha$

$g \leftarrow Initialize(\phi)$   
**for** episode from 1 to N **do**  
  **while**  $t < T$  **do**  
     $R_{t+1}, S_{t+1} \leftarrow Environment(S_t, A_t)$   
    **if**  $S_{t+1}$  is terminal **then**  
       $T \leftarrow t + 1$   
    **else**  
       $A_{t+1} \leftarrow Policy(g, S_{t+1})$   
    **end if**  
     $t \leftarrow t + 1$   
  **end while**  
  **for**  $h = 0, 1, \dots, T-1$  **do**  
     $G \leftarrow \sum_{i=h+1}^T \gamma^{i-h-1} R_i$   
     $g = Update(g, S_h, A_h, \omega G \Delta_g \ln \pi_g(A_h|S_h))$   
  **end for**  
**end for**

---

The pseudocode in Algorithm 4 outlines the implementation of the n-Tuple REINFORCE algorithm. Initially, the parameterized function  $g$  is initialized, and for each episode, the agent interacts with the environment by selecting actions and receiving rewards until the episode terminates. Following the interaction phase, the algorithm computes the return  $G$  for each time step and subsequently updates the policy parameters using the gradient of the log-probability of the taken actions, scaled by the return. This update rule ensures that the adjustments made to the

function  $g$  are directed towards maximizing expected returns. The integration of policy gradient techniques within the n-Tuple framework not only leverages the efficiency of weightless neural networks but also provides a robust method for optimizing decision-making in complex environments.

### Gradient Descent for Kernel Methods

The parameterization of policies in reinforcement learning allows for the use of various probability distribution families. This flexibility enables the model to adapt to the specific requirements of different action spaces. For effective sampling, the distribution must closely align with the task’s action space, ensuring compatibility and efficiency.

In this work, discrete and finite action spaces are the primary focus. The policy is parameterized using a softmax-based distribution, which transforms raw preferences into probabilities, as shown in Equation 3.24. The corresponding functional gradient is described in Equation 3.25.

$$\pi_f(A|S) = \frac{\exp(f(s)[a])}{\sum_{i=1}^l \exp(f(s)[i])} = \frac{\exp(f(s)^T \hat{e}_a)}{\sum_{i=1}^l \exp(f(s)^T \hat{e}_i)} \quad (3.24)$$

$$\ln \pi_f(A|S) = f(s)^T \hat{e}_a - \ln \sum_{i=1}^l \exp(f(s)^T \hat{e}_i) \quad (3.25)$$

$$\begin{aligned} \Delta_f \ln \pi_f(A|S) &= \Delta_f f(s)^T \hat{e}_a - \Delta_f \ln \sum_{i=1}^l \exp(f(s)^T \hat{e}_i) \\ &= \Delta_f f(s)^T \hat{e}_a - \Delta_f \ln \sum_{i=1}^l \exp(f(s)^T \hat{e}_i) \\ &= \left[ \hat{e}_a - \sum_{i=1}^l \pi_f(i|S) \hat{e}_i \right] k(S, \cdot) \end{aligned}$$

The integration of softmax-based policies with functional gradients facilitates efficient optimization of the policy parameters, aligning the learning process with the overarching goal of maximizing expected returns. This framework ensures that the n-Tuple architecture can effectively model complex decision-making processes in reinforcement learning.

### 3.3.3 n-Tuple Actor-Critic

The n-Tuple Actor-Critic method combines the strengths of the SARSA and REINFORCE algorithms by integrating value-based and policy-based learning strategies.

The actor component updates the policy strategy using principles derived from REINFORCE, while the critic component updates the state-value function in a manner analogous to  $n$ -Step SARSA. This dual structure ensures that the method benefits from the stability of value-based updates and the flexibility of policy-based optimization.

---

**Algorithm 5** n - Tuple Actor Critic

---

**Inputs:** Policy given a  $Q$  function and a state  $S$  -  $Policy(Q, S)$   
Initialization method to  $f$  function -  $InitializeValueNet(\phi)$   
Initialization method to  $g$  function -  $InitializePolicyNet(\phi)$   
Environment update method -  $Environment(S, A)$   
Update of model  $x$ , given output  $k$ , and inputs  $S$  and  $A$  -  $Update(x, S, A, k)$   
Forgetting factor -  $\phi$   
Decay rate -  $\gamma$   
Learning rate -  $\alpha$

```

 $f \leftarrow InitializeValueNet(\phi)$ 
 $g \leftarrow InitializePolicyNet(\phi)$ 
for episode from 1 to N do
   $S_t \leftarrow S_0$ 
   $A_0 \leftarrow Policy(f, S_t)$ 
   $t \leftarrow 0$ 
   $\tau \leftarrow -\infty$ 
   $T \leftarrow \infty$ 
  while  $\tau < T - 1$  do
    if  $t < T$  then
       $R_{t+1}, S_{t+1} \leftarrow Environment(S_t, A_t)$ 
      if  $S_{t+1}$  is terminal then
         $T \leftarrow t + 1$ 
      else
         $A_{t+1} \leftarrow Policy(g, S_{t+1})$ 
      end if
    end if
     $\tau \leftarrow t - m + 1$ 
    if  $\tau \geq 0$  then
       $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+m, T)} \gamma^{i-\tau-1} R_i$ 
      if  $\tau + m < T$  then
         $G \leftarrow G - \gamma^m f(S_{\tau+m}, A_{\tau+m})$ 
      end if
       $\sigma \leftarrow G - f(S_\tau, A_\tau)$ 
       $f = Update(f, S_\tau, A_\tau, G)$ 
       $g = Update(g, S_h, A_h, \omega \sigma \Delta_g \ln \pi_g(A_h | S_h))$ 
    end if
     $t \leftarrow t + 1$ 
  end while
end for

```

---

As shown in Algorithm 5, during each episode, the environment is sampled in a sequential manner. The agent starts from an initial state  $S_0$  and selects an action  $A_0$  according to the current policy, which is computed by the policy network  $g$ . As the episode progresses, the environment returns subsequent rewards and states, and the critic network  $f$  is updated using bootstrapped returns over  $m$ -step intervals. The temporal-difference error, computed as the difference between the accumulated return and the current value estimate, is then used to update both the critic and the actor. The critic’s update refines the estimation of the action-value function, while the actor’s update adjusts the policy parameters in the direction that maximizes expected returns. This iterative process continues until the termination of the episode, ensuring that both networks converge towards their respective optimal representations.

The combined approach is expected to result in more stable learning compared to standalone REINFORCE methods, as the critic provides an additional layer of guidance to the actor. By leveraging the unique properties of the n-Tuple framework, the Actor-Critic method effectively balances exploration and exploitation while ensuring efficient convergence to optimal policies.

# Chapter 4

## Methodology

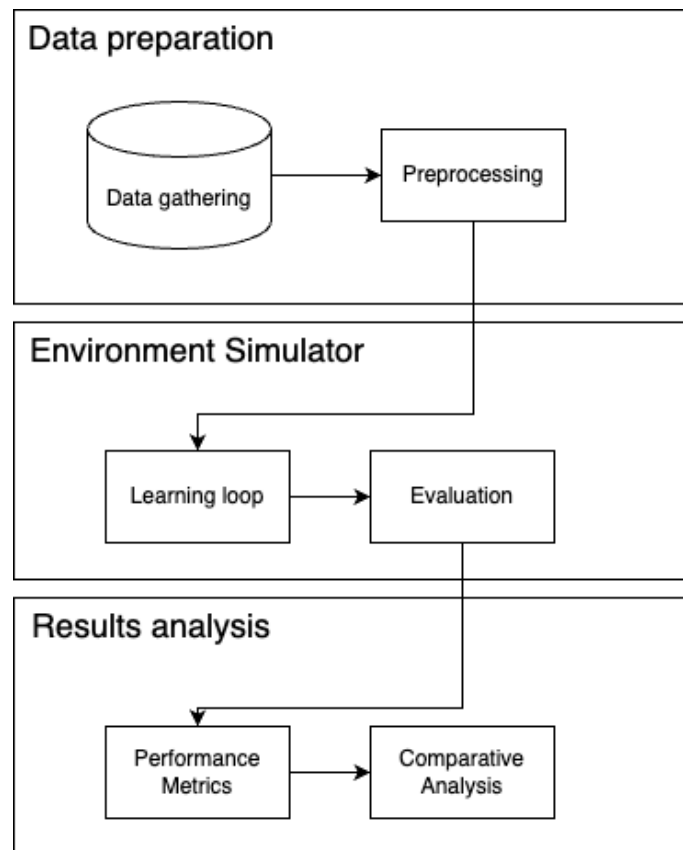


Figure 4.1: Methodology fluxogram.

This chapter provides a description of the methodology adopted in this research. It details the steps and components involved in the development, implementation, and evaluation of reinforcement learning algorithms for solar tracking systems. The overall process was divided into three primary phases, as illustrated in Figure 4.1: data preparation, environment simulator, and result analysis.

The first step, data preparation, focused on acquiring relevant datasets containing global horizontal irradiance (GHI) measurements from geographically suitable

locations. These datasets formed the foundation for modeling the environmental conditions required for the study. The preprocessing step followed ensures the data's quality and reliability through anomaly detection and correction techniques. This step was essential for minimizing biases and ensuring the subsequent phases were built on robust data.

The experimental execution phase involved developing a custom simulation environment to replicate real-world conditions. Within this controlled setting, four reinforcement learning algorithms were applied to optimize the performance of solar tracking systems. Each method was exposed to a learning loop and was tested in an evaluation set.

The final phase, result analysis, systematically compared the performance metrics to identify strengths, limitations, and opportunities for improvement across the evaluated algorithms.

Additionally, this chapter provides a detailed examination of the Weightless Q-Learning Neural Network (WQNN), a novel algorithm introduced in this work. The WQNN integrates the principles of Weightless Neural Networks with the Q-Learning reinforcement learning method to optimize decision-making processes. This new algorithm completes the implementation by introducing an off-policy method to the comparison.

## 4.1 Dataset Preparation for Solar Tracking Simulation

Simulating the behavior of solar tracking systems requires datasets with high temporal granularity to accurately capture the rapid changes in solar positions and irradiance levels. This precision is essential to prevent shading effects and to maximize the energy captured by photovoltaic (PV) modules. For this study, a maximum granularity of one hour was established as the baseline for the dataset, reflecting a balance between data availability and computational efficiency.

To achieve the level of detail required, the dataset was further refined by reducing its granularity to five-minute intervals, which is the industry standard for solar energy systems. This refinement was performed through interpolation of the original hourly data points. Interpolation certifies that the temporal resolution of the dataset is sufficient to simulate the rapid adjustment, enabling the evaluation of tracking algorithms under realistic operational conditions.

To ensure the integrity and reliability of the dataset, the measured global horizontal irradiance (GHI) values were cross-verified against theoretical clear-sky GHI values. The clear-sky model provides an upper limit for realistic irradiance mea-

measurements, serving as a reference to identify and correct anomalies in the data. The Python library *PVLib* was employed for this purpose, as it offers robust tools to compute both theoretical meteorological irradiance under clear-sky conditions and astronomical solar positions. By specifying the geographic location and time period, the library accurately calculates solar zenith and azimuth angles, as well as theoretical irradiance values.

The resulting dataset combines proper temporal granularity with corrected and validated irradiance measurements, making it suitable for simulating the performance of various solar tracking algorithms. The enhanced dataset reflects the dynamic behavior of solar irradiance and the corresponding environmental conditions, providing a reliable foundation for evaluating the effectiveness of different tracking strategies under diverse scenarios.

## 4.2 Environment Simulator

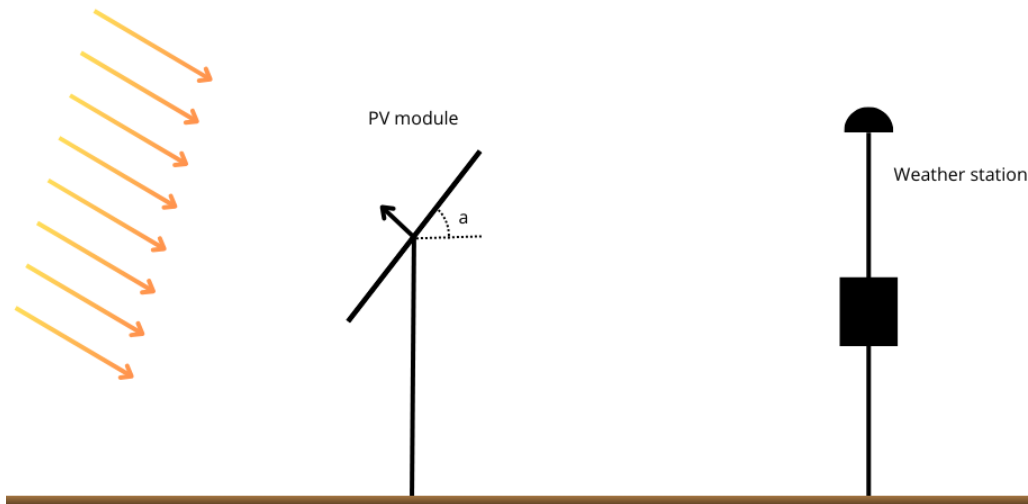


Figure 4.2: Experiment representation.

The environment simulator was designed to evaluate the agent’s decision-making capabilities under realistic conditions. Figure 4.2 illustrates the experimental setup, which incorporates environmental and physical parameters for accurate simulations. It modeled the interaction between a weather station measuring GHI and a photovoltaic (PV) module with a tilt angle controlled by a solar tracker

### 4.2.1 Decision agent representation

As discussed in Chapter 3, the representation of the agent is the cornerstone of reinforcement learning systems. The agent receives the current state representation as

input and determines its actions based on a decision algorithm. Figure 4.3 provides a visual summary of the agent representation adopted in this work.

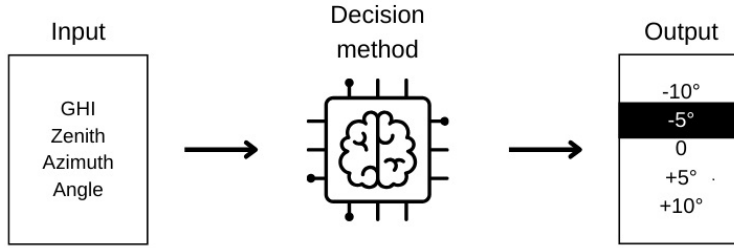


Figure 4.3: Agent representation.

The state representation is composed of four variables: global horizontal irradiance  $ghi$ , zentih  $ze$ , azimuth  $az$  and current tracking angle  $a$ . The first three variables ( $ghi$ ,  $ze$  and  $az$ ) are independent of the agent’s behavior. Specifically,  $ghi$  is measured by the weather station,  $ze$  and  $az$  are estimated using the Solar Position Algorithm (SPA). In contrast, the tracking angle ( $a$ ) is a control variable directly influenced by the agent’s decisions, as it determines the orientation of the PV module.

To simplify the decision-making process, the agent decides from a discrete set of five possible actions. Introducing more options or adopting a continuous representation would significantly increase the exploration space, requiring more iterations for effective learning. In this work, the agent adjusts the tracking angle ( $a$ ) by selecting a value ( $\delta_i$ ), as described by the following equation:

$$a_{next} = a_{current} + \delta_i \quad (4.1)$$

The five discrete actions adopted in this work are:  $\delta_1 = -10$ ,  $\delta_2 = -1$ ,  $\delta_3 = 0$ ,  $\delta_4 = 5$ , and  $\delta_5 = 10$ . These increments represent adjustments to the tracking angle in degrees.

## 4.2.2 Environment representation

The environment representation for this study is designed to simulate the behavior of a photovoltaic (PV) system by calculating the plane-of-array (POA) irradiance as the system’s response while providing inputs—global horizontal irradiance ( $ghi$ ), solar zenith angle ( $ze$ ), and solar azimuth angle ( $az$ )—to the decision-making agent.

POA irradiance is calculated as the sum of its two main components: direct irradiance ( $Irr_{direct}$ ) and diffuse irradiance ( $Irr_{diffuse}$ ), as expressed in Equation 4.2 - replicating definition on section 2.3.

$$POA = Irr_{direct}(dni, a, ze, az) + Irr_{diffuse}(dhi, a) \quad (4.2)$$

Direct irradiance represents sunlight arrived directly from the Sun’s beam, while diffuse irradiance accounts for sunlight scattered by atmospheric particles and reflected by the ground. The accurate estimation of these components requires the computation of direct normal irradiance ( $dni$ ) and diffuse horizontal irradiance ( $dhi$ ), which is derived from available data.

To estimate  $dni$  and  $dhi$ , this study employs the *PVLib* implementation of Erbs’ method [35], a well-established empirical model. The Erbs method estimates the diffuse fraction ( $df$ ), which represents the portion of  $ghi$  attributed to diffuse irradiance. This estimation is based on the ratio of  $ghi$  to a theoretical extraterrestrial irradiance, taking into account the position of the Sun relative to the Earth’s surface. Once the diffuse fraction is determined,  $dhi$  can be calculated using Equation 4.3:

$$dhi = df \cdot ghi \quad (4.3)$$

Subsequently,  $dni$  is derived by subtracting  $dhi$  from  $ghi$  and dividing the result by the cosine of the solar zenith angle ( $ze$ ), as shown in Equation 4.4:

$$dni = \frac{(ghi - dhi)}{\cos(ze)} \quad (4.4)$$

This approach ensures that  $dni$  and  $dhi$  are accurately computed, enabling the reliable estimation of POA irradiance.

### 4.2.3 Reward

The environment’s response is another important element in the Markov Decision Process (MDP), as it guides the agent toward the desired behavior. In a real-world scenario, the ultimate objective would be to optimize the power output of the PV system. However, this approach introduces additional complexities associated with estimating the electromechanical behavior of the system. To address this challenge, this work focuses on optimizing the plane-of-array irradiance (POA), as the power output is roughly proportional to this parameter.

To enhance the agent’s learning process, the irradiance values were scaled by a factor of three. This adjustment increases the distance between suboptimal solutions, making it easier for the agent to discern and learn the optimal behavior. The final reward function, incorporating this scaling, is described by Equation 4.5.

$$r = \left(\frac{POA}{1200}\right)^3 \quad (4.5)$$

This approach simplifies the learning process while maintaining alignment with the physical realities of PV system optimization. It ensures that the agent’s decisions are guided toward maximizing energy capture, ultimately improving the efficiency and effectiveness of solar tracking systems.

#### 4.2.4 Learning Loop and evaluation

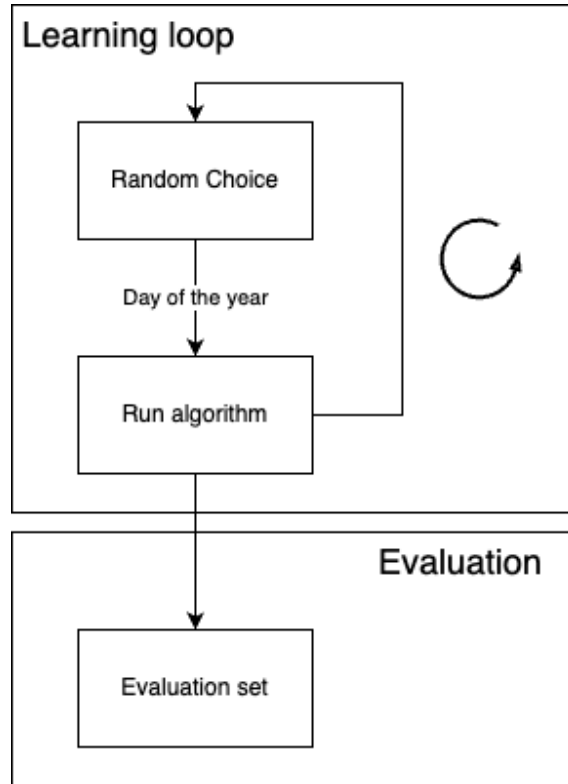


Figure 4.4: Learning loop and evaluation representation.

Figure 4.4 provides a detailed depiction of the model evaluation framework within the simulation environment. The learning loop is structured around episodic sampling, where each episode corresponds to a complete day from the dataset. This approach was selected because, in the context of solar tracking applications, each day’s data is entirely independent of the subsequent day. Consequently, achieving global optimality necessitates the aggregation of optimal performance across individual days. This episodic structure simplifies the reinforcement learning process, allowing the models to focus on optimizing daily performance without being influenced by temporal dependencies across multiple days.

A specific operational constraint was introduced to reflect real-world conditions: during non-irradiance hours, specifically between 8:00 p.m. and 4:00 a.m., the system is required to maintain a "safe position," where the tracking angle is fixed at  $a = 0$ . This constraint ensures that the tracking mechanism does not engage unnecessarily during periods without solar irradiance, thereby preserving energy and

reducing mechanical wear. Incorporating this rule into the simulation enhances the realism of the training environment and aligns the model’s behavior with practical operational standards in photovoltaic (PV) systems.

For the evaluation phase, a dedicated subset of the dataset was selected to assess model performance. This evaluation set was uniformly sampled across the entire time series to ensure that it accurately represents the variability and patterns observed throughout the year. Given the pronounced seasonality in solar irradiance data, a purely random sampling approach could inadvertently overrepresent certain seasons while underrepresenting others. Uniform sampling mitigates this risk, providing a balanced dataset that captures the full spectrum of seasonal variations. Importantly, the same evaluation set was applied consistently to all models after their respective training phases. This standardized evaluation protocol ensures a fair comparison of model performance, allowing for objective assessments of each algorithm’s strengths and limitations under identical conditions.

#### 4.2.5 Inputs encoding

The encoding of samples into binary tuples followed the methodology established on SOUZA *et al.*. To ensure an efficient representation of different input variables, distinct binarization strategies were applied based on the characteristics of each variable. For environmental condition variables, the distributive thermometer encoding method was employed. This choice was motivated by its ability to dynamically allocate encoding thresholds based on the statistical distribution of the dataset, thereby enhancing precision in regions where variations are more relevant.

In contrast, the tracking angle was encoded using a linear thermometer. This decision was based on the fact that the tracking angle is a direct result of the learning process rather than an inherent environmental condition. The linear thermometer provides a straightforward, ordered representation that preserves the relative magnitude of different angle values while maintaining simplicity in the encoding process.

Circular thermometer encoding was not applied in this study due to the physical constraints of the system. Unlike conventional angular measurements, where values wrap around at  $360^\circ$ , the tracking angle and solar angles in this system do not complete a full cycle. This characteristic made circular encoding unnecessary. Further details on the encoding methodology and its implications for learning performance are provided in Section 5.1.1.

### 4.3 Evaluation Metrics

After the learning step, each model was applied in a sequence of 24 days, (every 15 days, starting on January 1st) for comparing the results of the multiple configurations. Three evaluation metrics were adopted:

The models' performance was evaluated using two correlated metrics:

#### 1. Horizontal Ratio (Hor.)

$$\text{Ratio Hor.} = \frac{\sum POA}{\sum GHI} \quad (4.6)$$

This metric quantifies the extra irradiance received by the modules compared to fixed-position modules. A higher value indicates better performance.

#### 2. Astronomical Ratio (Astro.)

$$\text{Ratio Astro.} = \frac{\sum POA}{\sum POA_{astro}} \quad (4.7)$$

This metric compares the irradiance received by the modules to the theoretical maximum predicted by the astronomical model. Similarly, a higher value stands for superior performance.

The transposition gain  $t$  measures the relative gain of solar tracking over a fixed panel. The cumulative reward  $R$  is the target of the optimization of reinforce learning so is a useful metric for understand the results.

### 4.4 Weightless Q-Learning Neural Network

The Weightless Q-Learning Neural Network (WQNN) is reinforcement learning algorithm integrating the Regression WiSARD within the Q-learning strategy developed in this study. Its results were presented on the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) [9].

This algorithm complements the work Reinforcement Learning with Weightless Neural Networks [7], by introducing an off-policy learning strategy based on n-tuple methods. It is also an alternative implementation, based on regular Regression WiSARD, this modular strategy facilitate the application improvements on the base model, such as bleach [34].

In the architecture on Figure 4.5, the agent receives measurements from the environment as input (on the left of image), each are encoded in binary representations.

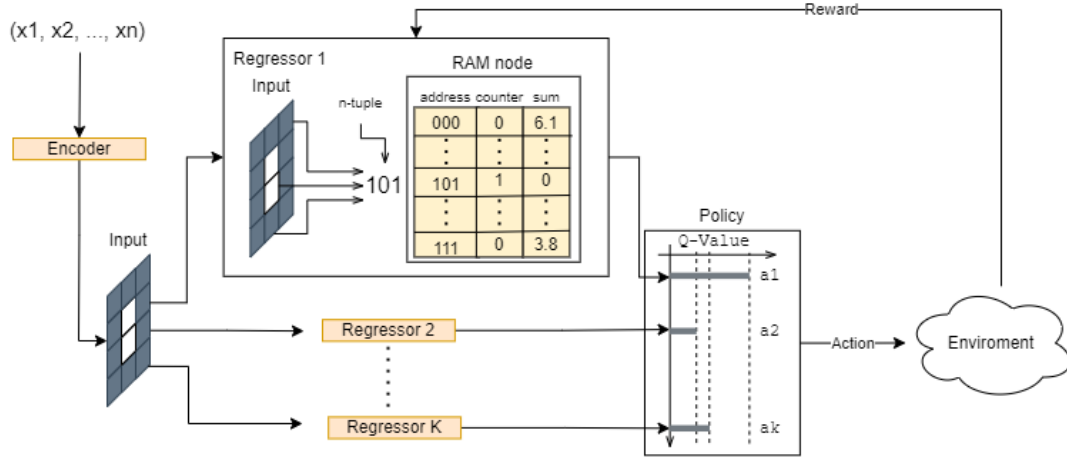


Figure 4.5: WQNN model architecture [9].

For each action available there is a Regression WiSARD model estimating the respective Q-value. Then, the agent chooses between the action with maximum Q or explore the other randomly, according to the exploration rate  $\epsilon$ .

The Algorithm 6 describes in detail the method. While the episode is not over, the agent evaluates the expected cumulative reward for each of the  $i^{th}$  action from the  $k$  possibilities,  $q_{ai}$ , chooses the action according to the greedy rate  $\epsilon$ , estimates the maximum return of next state, and updates the regressor relative to the selected action.

## 4.5 Chapter Conclusion

In this chapter, we explored the foundational aspects of the study, detailing the environment in which the agent operates and strives for maximum reward. We began by introducing the environment where the agent interacts, highlighting the challenges it faces and the strategies it adopts in pursuit of the highest possible reward. The core concepts behind the learning process were laid out, setting a clear picture of how the agent navigates its surroundings.

Towards the latter part of the chapter, we delved into the specifics of our custom implementation of Q-learning, which incorporates the Regression WiSARD model. This innovative approach allowed us to adapt Q-learning in a way that leverages the strengths of the WiSARD architecture, aiming for more effective decision-making and reward maximization.

---

**Algorithm 6** WQNN algorithm [9]

---

**Inputs:** Get initial state method  $S$  -  $GetInitialState()$   
Encoding method  $Q$  function -  $encode()$   
Prediction method given a regressor  $x$  and input  $y$  -  $Evaluate(x, y)$   
Update of model  $x$ , given output  $k$ , and input  $S$  -  $Update(x, S, k)$   
Decay rate -  $\gamma$   
Learning rate -  $\alpha$   
Greedy rate -  $\epsilon$

```
done  $\leftarrow$  false
state  $\leftarrow$   $GetInitialState()$ 
while done = false do
  for  $i$  from 1 to  $k$  do
     $q_{ai} = Evaluate(regressor_{ai}, encode(state))$ 
  end for
   $maxAction \leftarrow max_{ai}(q_{ai})$ 
  if  $Random() < \epsilon$  then
     $action \leftarrow maxAction$ 
  else
     $action \leftarrow Sample(a_1, a_2, \dots, a_k)$ 
  end if
   $done, r, nextState \leftarrow Environment(action)$ 
   $Q^* \leftarrow max_{ai}(Evaluate(regressor_{ai}, encode(nextState)))$ 
   $Update(regressor_{action}, encode(state), (1 - \alpha) \cdot Q + \alpha * (r + \gamma \cdot Q^*))$ 
   $state \leftarrow nextState$ 
end while
```

---

# Chapter 5

## Experimental Evaluation

In this chapter, we delve into the execution and results of the experiments outlined in the previous chapter. The experimental process begins with selecting a suitable dataset. The dataset must include features that accurately represent the agent’s environment and behavior in a real-world setting. Once the acquired the dataset, data cleaning and preprocessing are crucial steps. These tasks aim to mitigate the impact of noise and ensure the data quality is sufficient for reliable modeling and analysis.

The next phase involves setting up the experimental environment. This includes writing scripts to simulate the environment and implementing the algorithms under evaluation. Specifically, we implemented the following algorithms: Weightless Q-Learning Neural Network (WQNN), n-Tuple Sarsa, n-Tuple REINFORCE, and n-Tuple Actor-Critic.

Each algorithm was integrated into the simulator to enable consistent evaluation under identical experimental conditions. During this step, we also ensured that all relevant metrics and intermediate results were stored systematically for later analysis.

After running the simulations, the last step was to analyze the results. This involved examining the performance of each algorithm in terms of predefined metrics, comparing their effectiveness, and identifying any notable patterns or anomalies in the data. The insights gained from this analysis are crucial for understanding the behavior of the algorithms and their potential applicability in real-world scenarios.

### 5.1 Dataset and Preprocessing

The simulation of a realistic environment relied on data from the public weather station A608, provided by INMET, as shown in Figure 5.1. This station is operated by the Meteorology Laboratory (LAMET) at UENF, Macaé. The original dataset, which has an hourly granularity, includes the features listed in Table 5.1.

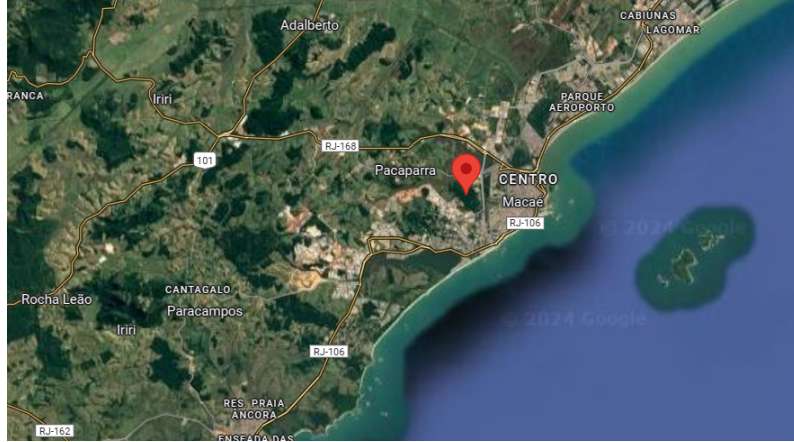


Figure 5.1: Weather station A606 location.

Table 5.1: Features of the public weather station A608 dataset, as provided by INMET.

Column	Description
Data	Date
Hora (UTC)	Hour (Coordinated Universal Time)
Radiacao (KJ/m <sup>2</sup> )	Global Horizontal Irradiance ( $KJ/m^2$ )
Temp. Ins. (°C)	Average Temperature (°C)
Temp. Max. (°C)	Maximum Temperature (°C)
Temp. Min. (°C)	Minimum Temperature (°C)
Umi. Ins. (%)	Average Relative Humidity (%)
Umi. Max. (%)	Maximum Relative Humidity (%)
Umi. Min. (%)	Minimum Relative Humidity (%)
Pto Orvalho Ins. (°C)	Average Dew Point Temperature (°C)
Pto Orvalho Max. (°C)	Maximum Dew Point Temperature (°C)
Pto Orvalho Min. (°C)	Minimum Dew Point Temperature (°C)

To enhance the dataset, additional variables were calculated using the Solar Positioning Algorithm (SPA) from the PVLlib library, developed by NREL. This extension added key solar characteristics such as zenith, azimuth, and astronomical tracker model data. These features were essential for accurately capturing the environment’s response to the agent’s state.

Anomalies in irradiance measurements were identified and corrected by comparing the recorded data with the clear-sky model provided by PVLlib. Since actual irradiance values should not exceed the clear-sky estimates, the corrected global horizontal irradiance (GHI) values were computed using Equation 5.1:

$$ghi = \min(ghi_{actuals}, ghi_{cls}) \quad (5.1)$$

In this equation,  $ghi$  is the corrected global horizontal irradiance,  $ghi_{actuals}$  denotes the measured values from the weather station, and  $ghi_{cls}$  is the clear-sky model estimate. This correction process is illustrated in Figure 5.2.

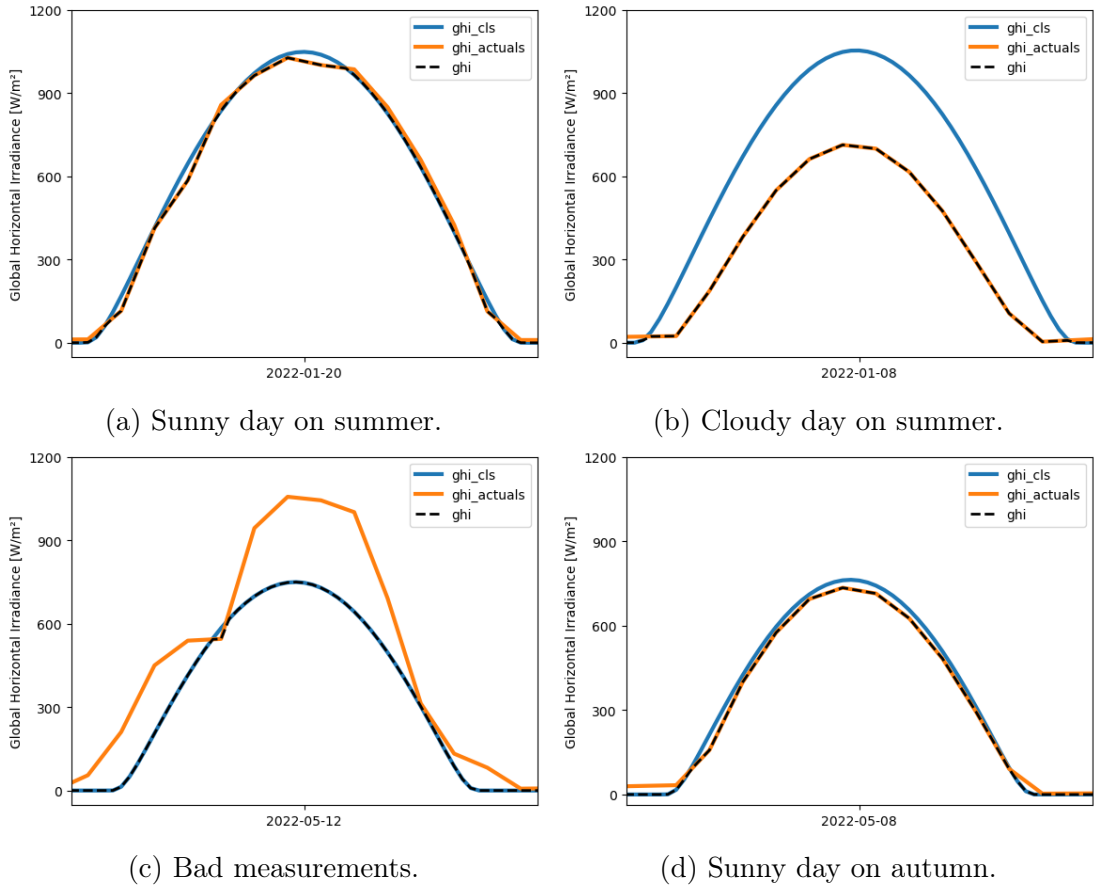
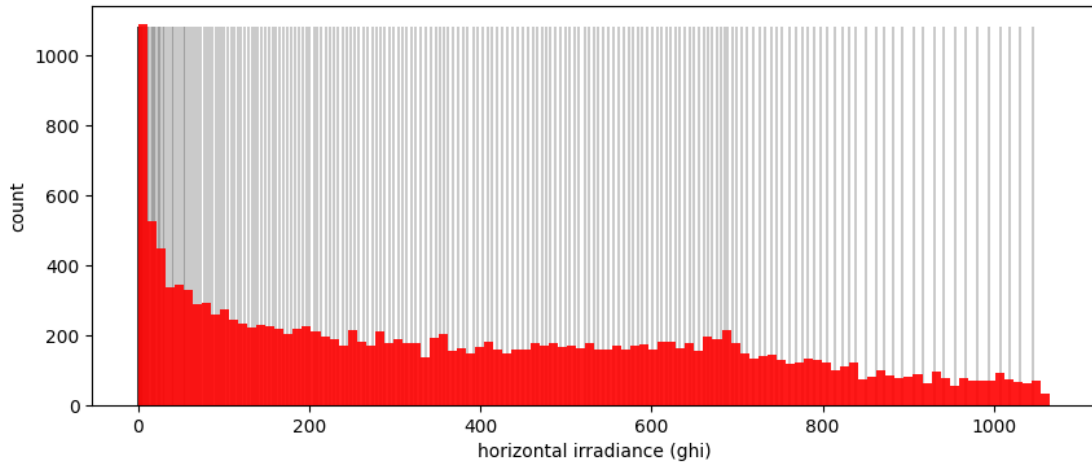
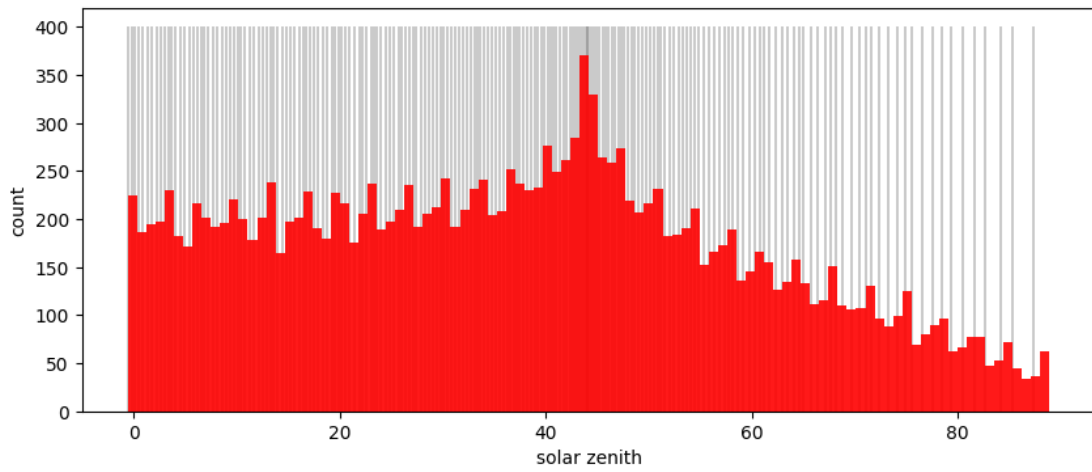


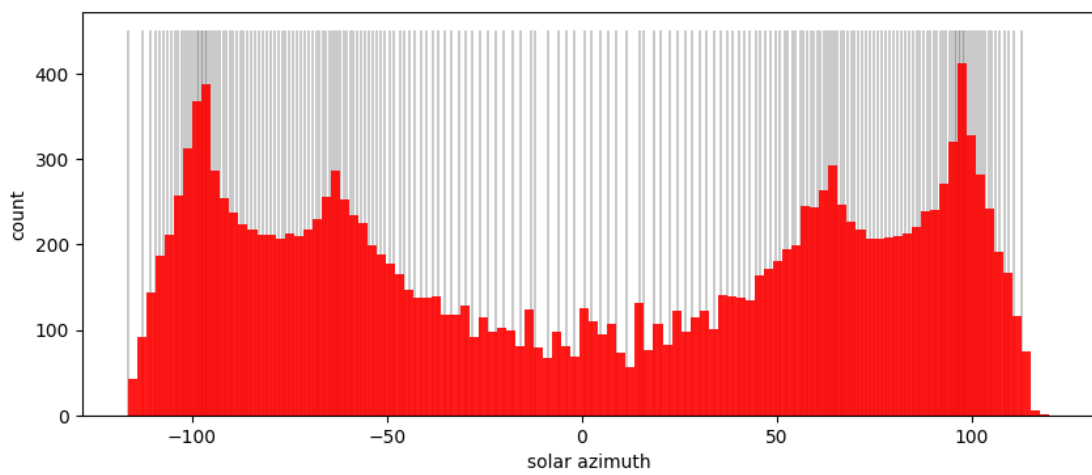
Figure 5.2: Adjustments on global horizontal irradiance measurements.



(a) *ghi* samples distribution.



(b) Solar zenith *ze* samples distribution.



(c) Solar azimuth *ze* samples distribution.

Figure 5.3: Inputs samples distribution.

### 5.1.1 Thermometer

The binarization of the input variables was carried out using distributive thermometer encodings with sizes of 600, 240, and 240 for the variables *ghi* (global horizontal irradiance), *ze* (solar zenith angle), and *az* (solar azimuth angle), respectively. The tracking angle was binarized using a non-linear thermometer of size 240. These encoding sizes were adopted based on the configurations presented in [9], following a simple manual optimization process. It was observed that increasing the representation size improved model performance, likely due to the enhanced resolution in capturing data nuances. As a result, the regression models receive an input vector consisting of 1560 binary bits.

Figure 5.3 illustrates the sample distribution of the input variables, derived from the adopted dataset. Each variable exhibits a distinct, non-uniform distribution, reflecting the diverse environmental and system conditions captured during data collection.

The distribution of *ghi*, depicted in Figure 5.3a, shows a higher concentration of values near zero. This pattern can be attributed to two main factors. First, seasonal variations play a significant role, particularly the position of the sun during different times of the year. Higher irradiance values are predominantly observed in the summer months when the solar zenith angle is greater, allowing more direct sunlight. Second, the presence of clouds significantly reduces irradiance levels, even on days when high solar exposure would be expected. This combination of seasonal and meteorological influences results in a skewed distribution with a concentration of low irradiance values.

The distribution of the solar zenith angle exhibits a pronounced peak around  $45^\circ$ , which is closely related to seasonal variations. Values closer to  $90^\circ$  occur predominantly during the summer when the sun reaches its highest point in the sky. In contrast, zenith angles up to  $45^\circ$  are observed throughout the year. During winter, the maximum daily zenith angle tends to hover around  $45^\circ$ . Due to the sinusoidal nature of the zenith angle's variation throughout the day, the rate of change in this region is relatively low, leading to a denser clustering of data points around this angle range.

## 5.2 Experimental execution

The simulator was implemented using the Julia Programming Language (v1.9.3). Its primary design allows for the execution of a fixed number of random episodes (5000) and 25 predetermined evaluation scenarios. The complete time series of each execution was stored in CSV format.

The simulations were repeated for each combination of applicable parameters. Table 5.2 outlines the parameters considered for each model.

Table 5.2: Applicable parameters per model

Model	$\gamma$	$\omega$	$\epsilon$	$\zeta$	$n$	$\phi$
N-tuple Actor-Critic	x	x		x		
N-tuple REINFORCE	x	x		x	x	
N-tuple SARSA	x	x	x	x	x	
<b>WQNN</b>	x	x	x	x		x

The following parameters and their respective values were evaluated:

- Learning rate  $\omega$  - {0.25, 0.5, 0.7, 0.9, 0.98}
- Decay rate  $\gamma$  - {0.25, 0.5, 0.7, 0.9, 0.98}
- Greedy rate  $\epsilon$  - {0.25, 0.5, 0.7, 0.9, 0.98}
- Tuple size  $\zeta$  - {20, 40, 80, 160, 320, 720}
- Steps number  $n$  - {1, 2, 4, 8, 16}
- Forgetting factor  $\phi$  - {0.25, 0.5, 0.7, 0.9, 0.98}

The experiments were conducted on a personal laptop running Windows 11 Pro. The system is equipped with an Intel Core i7-1255U processor and 16 GB of RAM. This configuration provided sufficient computational power for running the simulations efficiently.

## 5.3 Results analysis

For clarity and systematic evaluation, we divided the results into two groups: State value models (WQNN, and N-tuple SARSA), and Policy models (N-tuple Actor-Critic, and N-tuple REINFORCE)

This section concludes with a summary of overall observations, providing a comprehensive understanding of the models' behaviors and their dependency on parameter configurations. This structured approach ensures a clear comparison across both groups and helps identify the most effective models and parameter settings.

### 5.3.1 State-value models comparison

Tables 5.3 and 5.4 present the top five configurations of the n-Tuple SARSA and WQNN models in terms of performance.

Table 5.3: Top 5 n-Tuple SARSA

Position	$\omega$	$\gamma$	$n$	$\zeta$	$\epsilon$	Ratio Hor.	Ratio Astro.
1	0.25	0.25	4.00	160.00	0.90	1.09	0.99
2	0.70	0.50	4.00	40.00	0.90	1.09	0.98
3	0.25	0.25	4.00	80.00	0.98	1.09	0.98
4	0.70	0.90	4.00	80.00	0.98	1.08	0.98
5	0.25	0.50	2.00	80.00	0.90	1.08	0.98

Table 5.4: Top 5 WQNN

Position	$\omega$	$\gamma$	$\zeta$	$\epsilon$	$\phi$	Ratio Hor.	Ratio Astro.
1	0.90	0.25	160.00	0.98	0.25	1.10	0.99
2	0.70	0.25	40.00	0.98	0.98	1.09	0.98
3	0.25	0.25	40.00	0.70	0.25	1.08	0.98
4	0.90	0.70	160.00	0.98	0.25	1.08	0.98
5	0.98	0.25	160.00	0.90	0.98	1.08	0.97

Both models demonstrated the ability to achieve near-optimal performance, with an astronomical ratio ( $Astro. \geq 99\%$ ). While the WQNN model achieved the highest performance overall, the n-Tuple SARSA model exhibited more consistent results, as evidenced by lower variance across its top five configurations. This consistency suggests that n-Tuple SARSA is more robust to parameter changes, whereas WQNN may require more precise tuning to reach optimal performance.

Figure 5.4 illustrates the POA irradiance time series for the best-performing configuration of each model. During periods of high irradiance, both models closely approximated the astronomical model. In cloudy conditions, both models captured approximately up to 5% more irradiance compared to the baseline.

Certain parameter values consistently appeared in the top-performing configurations for both models, indicating their importance for optimal performance: tuple size  $\zeta = 160$ , decay rate  $\gamma = 0.25$ , learning rate  $\omega = 0.25$ , greedy rate  $\epsilon = 0.98$ .

### Effects of the learning rate on learning

The learning rate is a parameter that directly affects the stability and adaptability of the algorithm. A higher learning rate enables the agent to update its internal state-value function more rapidly, allowing it to react quickly to changes in the environment. However, this comes at the cost of increased noise, which can lead to instability and hinder convergence.

Figure 5.5 illustrates the effect of the learning rate on the average horizontal ratio during the evaluation phase. For the n-Tuple SARSA model, performance consistently degrades as the learning rate increases. This suggests that higher learning rates introduce excessive noise, preventing the model from effectively capturing

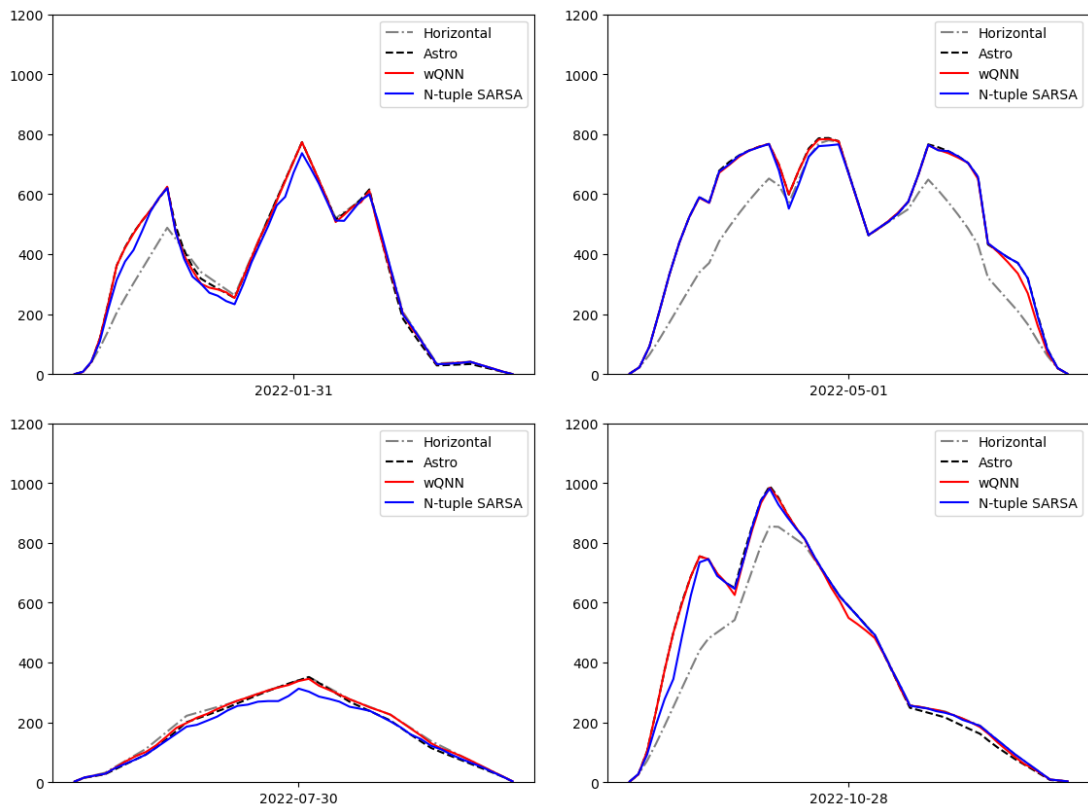


Figure 5.4: Best state-value models timeseries.

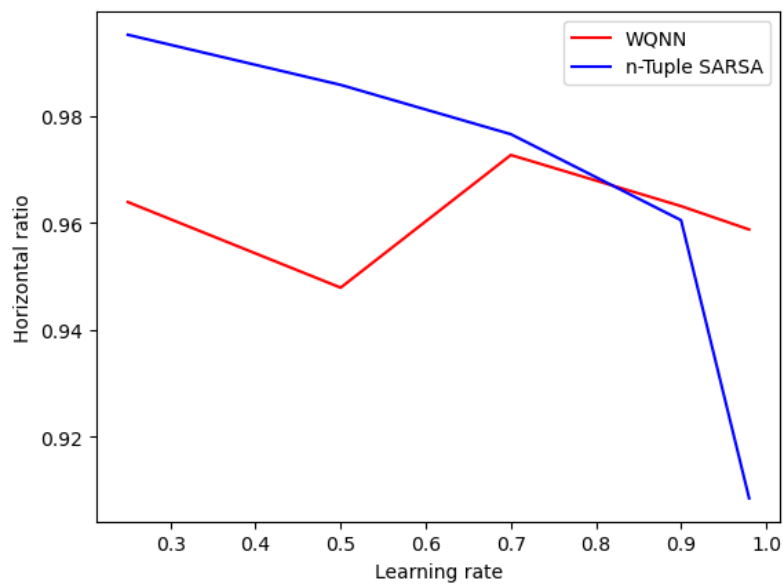


Figure 5.5: Average horizontal ratio given the learning ratio on evaluation step.

the underlying patterns. In contrast, the WQNN model shows minimal sensitivity to changes in the learning rate, indicating that this parameter is less significant for WQNN’s performance.

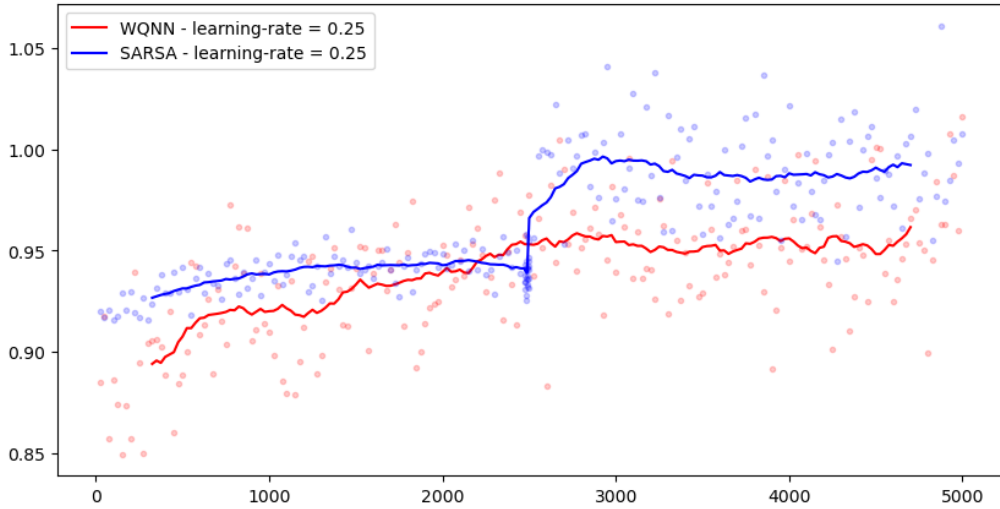


Figure 5.6: Average horizontal ratio given the learning ratio on training iterations.

The learning curve in Figure 5.6 further supports these observations. Throughout the training process, the n-Tuple SARSA model with the lowest learning rate consistently outperforms the WQNN model. This indicates that a lower learning rate not only enhances stability but also enables the n-Tuple SARSA to achieve superior long-term performance. Conversely, WQNN’s performance remains relatively unaffected by the learning rate, suggesting a different sensitivity to parameter tuning.

### Effects of the decay rate on learning

The Decay rate ( $\gamma$ ) is the parameter that controls the balance between immediate rewards and the expected maximum cumulative reward. A lower decay rate places more emphasis on immediate rewards, making the agent more short-sighted. Conversely, a higher decay rate encourages the agent to prioritize long-term gains.

highlights the impact of the decay rate on model performance. For the WQNN model, the highest efficiency is seen at  $\omega = 0.7$ , indicating an optimal balance between immediate and future rewards at this value. In contrast, the n-Tuple SARSA model shows a slight performance improvement at the highest decay rate,  $\omega = 0.98$ , favoring long-term reward accumulation. Interestingly, these results differ from Table 5.4, where most top-performing WQNN configurations used  $\omega = 0.25$ . This discrepancy suggests a higher variance in WQNN performance across different decay rate settings.

Figure 5.8 shows the learning progression of both models. While both WQNN

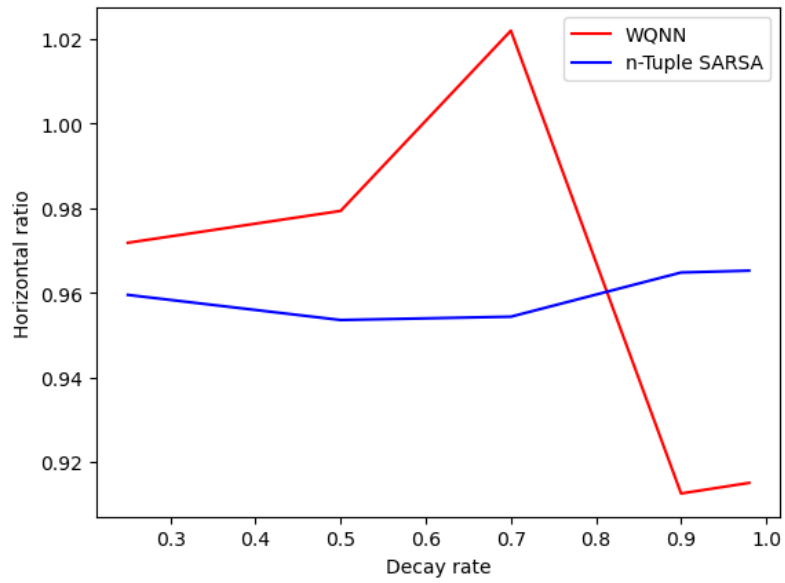


Figure 5.7: Average horizontal ratio given the decay ratio on evaluation step.

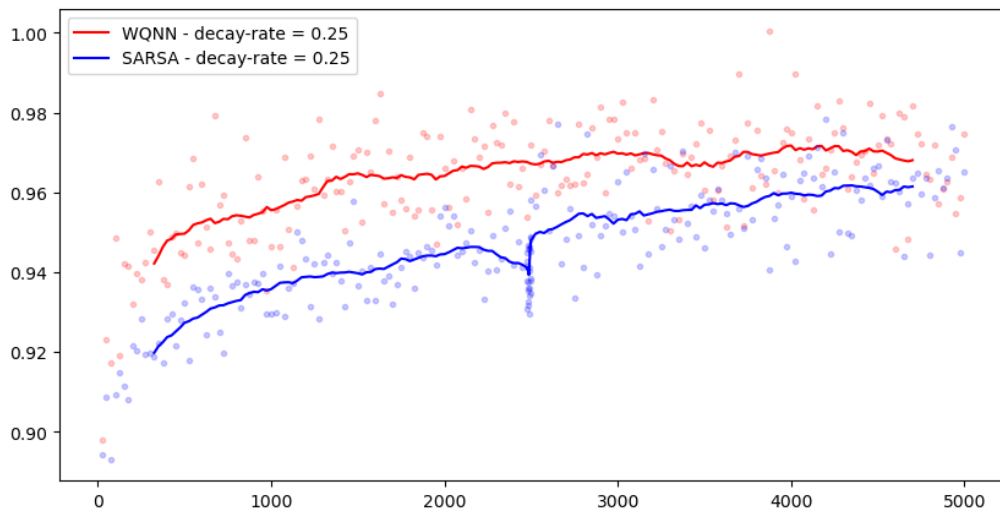


Figure 5.8: Average horizontal ratio given the decay ratio on training iterations.

and n-Tuple SARSA successfully learn over time, WQNN converges significantly faster. However, given more training iterations, it is likely that the n-Tuple SARSA could eventually surpass WQNN, particularly at higher decay rates where it shows gradual improvement.

### Effects of the greedy rate on learning

In state-value methods, the decision policy is predefined. In this context, the greedy rate ( $\epsilon$ ) determines the balance between exploitation (choosing actions based on the maximum expected return) and exploration (selecting random actions to discover potentially better options). A higher greedy rate implies more exploitation and less exploration.

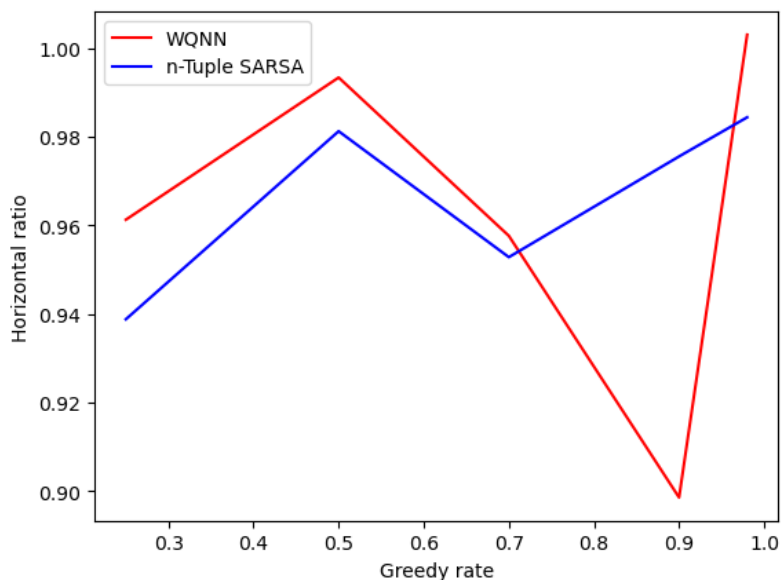


Figure 5.9: Average horizontal ratio given the greed rate on evaluation step.

Figure 5.9 shows the impact of the greedy rate on the learning performance of both WQNN and n-Tuple SARSA. While the general trend is similar for both methods, an outlier is observed at  $\epsilon = 0.9$  for WQNN. The best performance was achieved with the highest greedy rate ( $\epsilon = 0.98$ ), which aligns with expectations. In a stable environment, once the optimal policy is learned, exploration is less beneficial, as it may lead to unnecessary production losses without the prospect of discovering new optimal actions.

Figure 5.10 illustrates the learning curves for both models under same greedy rate. Both WQNN and n-Tuple SARSA exhibit similar responses to the greedy rate throughout the training process. While WQNN shows slightly higher dispersion and marginally better performance, the overall learning dynamics and curve shapes are comparable. This indicates that the greedy rate influences both models in a

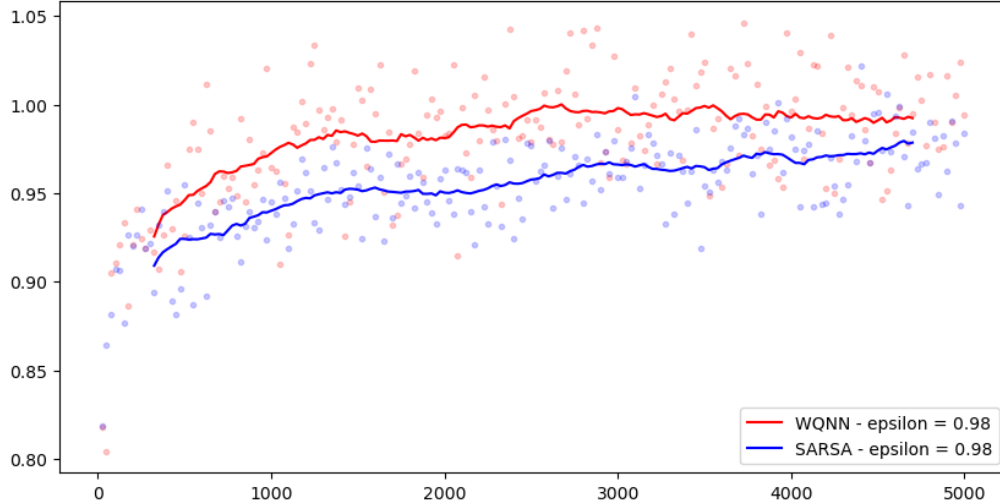


Figure 5.10: Average horizontal ratio given the greedy rate on training iterations.

consistent manner, with no significant performance advantage favoring one over the other.

### 5.3.2 Policy models comparison

In contrast to the state-value models, the policy-search methods struggled to achieve competitive results. As shown in Tables 5.5 and 5.6, their performance often fell below that of the static horizontal module, highlighting significant limitations in their learning process.

Table 5.5: REINFORCE

Position	$\omega$	$\gamma$	$\zeta$	Ratio Hor.	Ratio Astro.
1	0.98	0.70	160.00	1.00	0.90
2	0.98	0.50	160.00	0.99	0.89
3	0.70	0.50	320.00	0.99	0.89
4	0.70	0.70	160.00	0.98	0.89
5	0.50	0.70	160.00	0.98	0.88

Table 5.6: ACTOR CRITIC

Position	$\omega$	$\gamma$	$n$	$\zeta$	Ratio Hor.	Ratio Astro.
1	0.50	0.70	8.00	20.00	1.02	0.92
2	0.98	0.70	2.00	160.00	1.00	0.90
3	0.98	0.25	1.00	10.00	1.00	0.90
4	0.70	0.98	1.00	80.00	0.96	0.87
5	0.25	0.25	4.00	160.00	0.93	0.84

Figure 5.11 highlights the instability of these methods. While they occasionally matched optimal performance under clear-sky conditions, their performance was

inconsistent across different days. Similar variability was observed during cloudy periods, where the models performed well on some days but poorly on others.

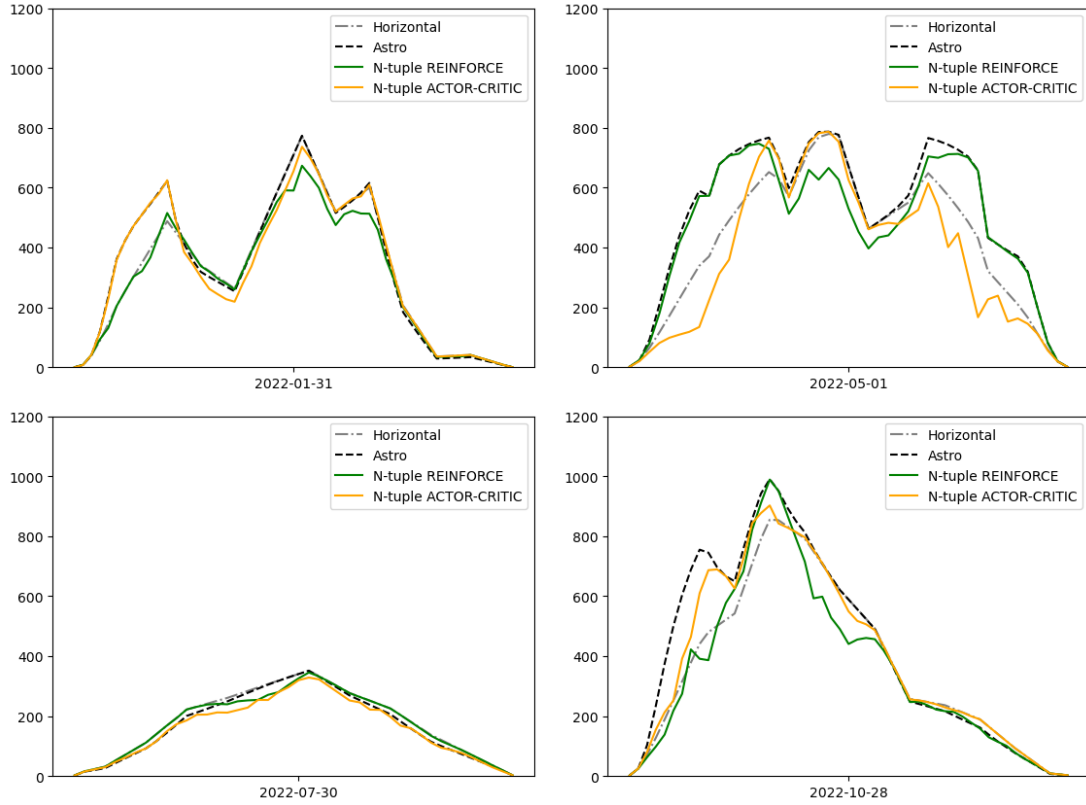


Figure 5.11: Best policy search models timeseries.

These patterns suggest that both REINFORCE and Actor-Critic models struggle to balance exploration and exploitation effectively. This imbalance may lead to suboptimal learning, where the models fail to generalize their policies across varying environmental conditions. Consequently, their performance remains unstable, limiting their applicability in scenarios with dynamic weather conditions.

Eventough the following parameter values appeared in the top-performing configurations for both models: tuple size  $\zeta = 160$ , decay rate  $\gamma = 0.70$ , learning rate  $\omega = 0.98$ .

### Effects of the decay rate

Despite the instability observed in policy-search models, the impact of individual parameters is evident. One notable example is the Decay rate ( $\gamma$ ), where optimal performance is achieved around  $\omega = 0.7$ , as shown in Figure 5.12.

The learning curve in Figure 5.13 shows an almost linear improvement as training progresses, indicating that the models continue to learn incrementally over time. However, the rate of improvement is exceptionally slow, making it impractical to wait for full convergence within a reasonable time-frame.

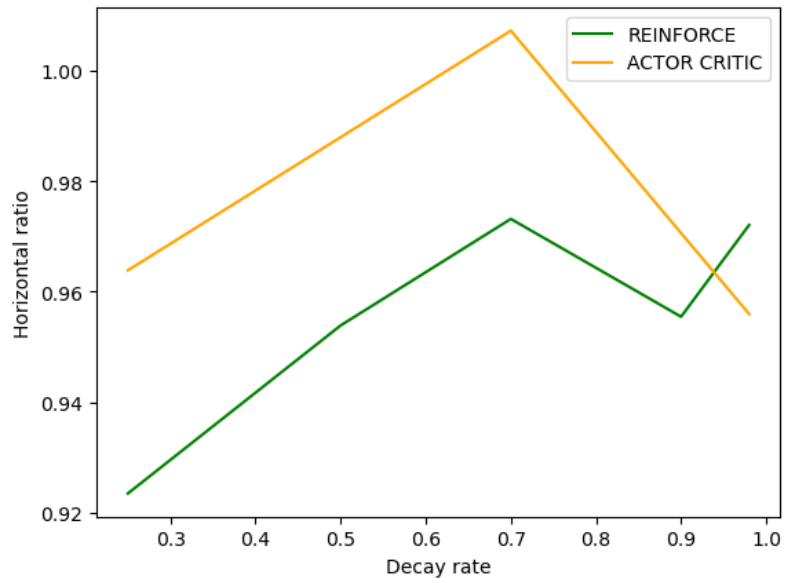


Figure 5.12: Average horizontal ratio given the decay rate on evaluation step.

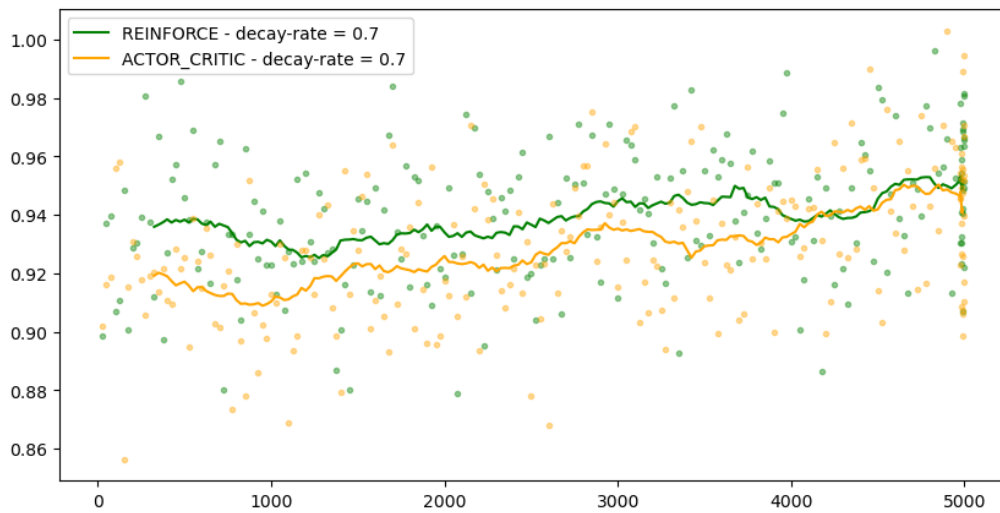


Figure 5.13: Average horizontal ratio given the decay rate on training iterations.

## Effects of the learning rate

Unlike state-value models, policy-search models benefit from higher learning rates. As shown in Figure 5.5, increasing the learning rate improves the average horizontal ratio during the evaluation phase. This suggests that these models require more rapid updates to their internal parameters to adapt effectively, which may be linked to their inherently slower improvement rate.

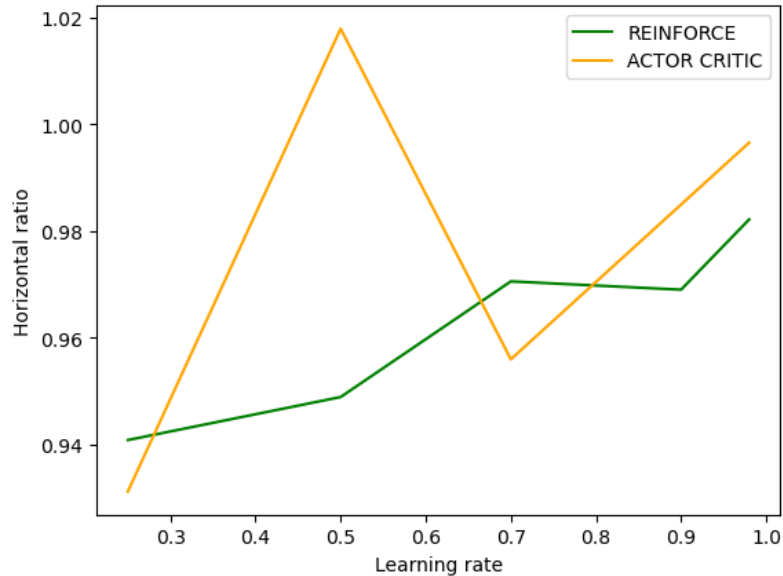


Figure 5.14: Average horizontal ratio given the learning rate on evaluation step.

Figure 5.15 illustrates the learning curve for the highest tested learning rate. Similar to the trend observed in Figure 5.13, the models exhibit a gradual improvement over time. However, the n-Tuple Actor-Critic model shows a noticeably faster convergence compared to other policy-search models. This indicates that while high learning rates accelerate learning, their impact varies across different model architectures.

## 5.4 Chapter Conclusion

Table 5.7 presents the top 10 configurations based on their performance, reinforcing the conclusions drawn throughout this chapter.

State-value models clearly outperformed other approaches. The Weightless Q-Learning Neural Network (WQNN) achieved the highest overall performance. However, n-Tuple SARSA was prominently featured, comprising 60% of the top 10 configurations. This highlights its reliability and consistency across different parameter settings.

Although neither model surpassed the total return of the astronomical model

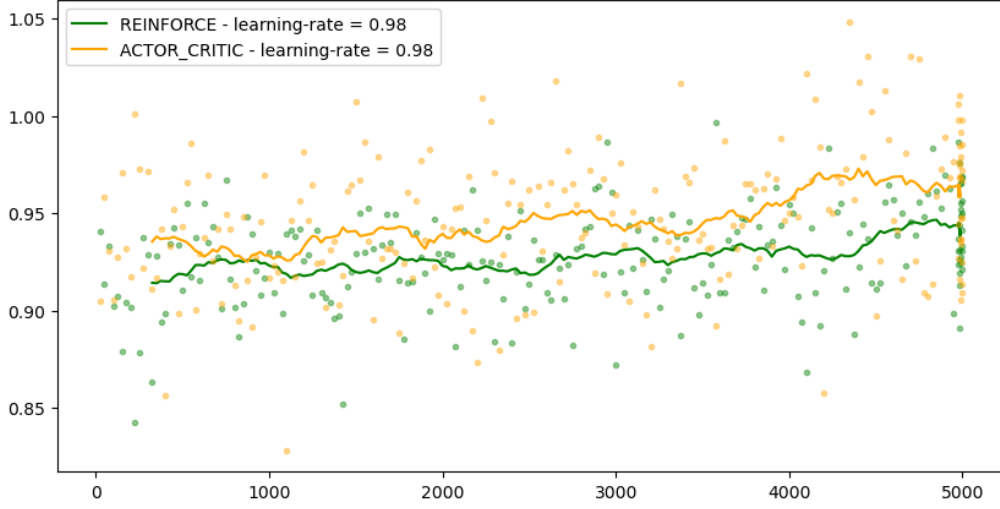


Figure 5.15: Average horizontal ratio given the learning rate on training iterations.

Table 5.7: Top 10 Configurations Based on Performance Metrics

Position	model	$\omega$	$\gamma$	$n$	$\zeta$	$\epsilon$	$\phi$	Ratio Hor.	Ratio Astro.
1	<b>WQNN</b>	0.90	0.25	-	160.00	0.98	0.25	1.10	0.99
2	SARSA	0.25	0.25	4.00	160.00	0.90	-	1.09	0.99
3	SARSA	0.70	0.50	4.00	40.00	0.90	-	1.09	0.98
4	<b>WQNN</b>	0.70	0.25	-	40.00	0.98	0.98	1.09	0.98
5	SARSA	0.25	0.25	4.00	80.00	0.98	-	1.09	0.98
6	<b>WQNN</b>	0.25	0.25	-	40.00	0.70	0.25	1.08	0.98
7	SARSA	0.70	0.90	4.00	80.00	0.98	-	1.08	0.98
8	SARSA	0.25	0.50	2.00	80.00	0.90	-	1.08	0.98
9	<b>WQNN</b>	0.90	0.70	-	160.00	0.98	0.25	1.08	0.98
10	SARSA	0.25	0.25	1.00	80.00	0.90	-	1.08	0.97x

in the evaluation set, they achieved comparable results under real-world conditions. Both state-value methods provided a some performance boost under cloudy conditions, which demonstrates their practical applicability without requiring strong environmental assumptions.

The top configurations confirm the important role of specific parameters such as low decay rates ( $\omega = 0.25$ ) and high greedy rates ( $\epsilon = 0.90$  or  $0.98$ ) in maximizing performance.

These findings reinforce the robustness of state-value models, especially WQNN and n-Tuple SARSA, in achieving near-optimal performance across varying scenarios.

# Chapter 6

## Conclusion

The primary objective of this work is to investigate the feasibility of the application of Weightless Neural Networks (WNNs) and reinforcement learning in controlling solar tracking systems. A closed-loop control algorithm based on the reinforcement learning (RL) paradigm was proposed, leveraging weightless neural networks for its implementation. Additionally, an environment simulator was developed to evaluate the performance of the proposed and benchmarked algorithms under realistic conditions.

The environment simulator models the behavior of a single PV array subjected to weather conditions derived from a weather station’s measurements. Data was sourced from the publicly available dataset published by INMET and enriched with astronomical models to estimate the plane-of-array (POA) irradiance. The methodology for constructing the environment simulator is detailed in Section 4.2, while the data preparation process is outlined in Section 5.1.

Four RL algorithm variants were explored using n-Tuple-based models, including three developed on the work Reinforcement Learning with Weightless Neural Networks [7] and the Weightless Q-Learning Neural Network (WQNN), which was introduced during this research and published in ESANN [9]. The algorithms were benchmarked against the state-of-the-art astronomical tracking model and a static horizontal system. The results were categorized into two groups for analysis: state-value models and policy search models.

The findings indicate that state-value models significantly outperformed policy search approaches. Among the state-value models, the WQNN achieved the highest overall performance, while n-Tuple SARSA demonstrated notable consistency and reliability, appearing in 60% of the top 10 configurations. This highlights the robustness of n-Tuple SARSA across varying parameter settings.

Although neither model exceeded the total energy return of the astronomical tracking model on the evaluation dataset, they achieved comparable results under realistic conditions. Both state-value methods, particularly WQNN and n-Tuple

SARSA, provided a 10% irradiance boost over static systems. Furthermore, they exhibited enhanced performance under cloudy conditions, demonstrating their practicality in real-world applications without requiring strong assumptions about environmental conditions.

These results underscore the efficacy of state-value models, particularly WQNN and n-Tuple SARSA, in achieving near-optimal performance across diverse scenarios. Their simplicity and flexibility make them attractive as plug-and-play solutions for solar tracking systems.

## 6.1 Future Work

Future research offers numerous opportunities to refine and extend the present study. Key directions include developing a physical PV testing system, investigating the energy consumption of the algorithms, evaluating performance under a wider range of environmental conditions, and comparing alternative closed-loop algorithms, such as reinforcement learning models based on weighted Artificial Neural Networks (ANNs).

The implementation of a real-world PV testing system is a critical next step, as it would allow for direct performance evaluation without relying solely on theoretical assumptions. Such a system would provide empirical evidence to validate the models in practical applications, potentially leading to commercial deployment and further reinforcing the credibility of the virtual simulator developed in this work.

While this study did not address the energy efficiency of the evaluated algorithms, this aspect warrants further exploration. Weightless neural networks have lower energy consumption compared to conventional networks and are particularly suited for deployment on smart chips, an important consideration for sustainable technologies.

Additionally, assessing system performance across diverse weather conditions—by incorporating datasets from various regions, applying metrics such as mean squared error, and simulating physical obstacles—would rigorously test the robustness and adaptability of the proposed approaches. Finally, with a comprehensive testing framework established, further investigation of alternative control algorithms, including those integrating weighted ANNs or hybrid methods like the ClusRegression WiSARD [34], may provide valuable insights for enhancing overall system performance.

In summary, the methodologies and findings presented here lay a strong foundation for future advancements in solar tracking systems, with significant potential to optimize renewable energy technologies.

# References

- [1] UNITED NATIONS, D. O. E., DEVELOPMENT, S. A. S. “Transforming our world: the 2030 Agenda for Sustainable Development”. 2015. Disponível em: <<https://sdgs.un.org/2030agenda>>.
- [2] MUSTAFA, G. E. G., SIDAHMED, B. A. M., NAWARI, M. O. “The Improvement of LDR Based Solar Tracker’s Action using Machine Learning”. In: *2019 IEEE Conference on Energy Conversion (CENCON)*, pp. 230–235, 2019. doi: 10.1109/CENCON47160.2019.8974834.
- [3] “Top 10 PV module suppliers in 2022 shipped 245GW”. <https://www.pv-tech.org/top-10-pv-module-suppliers-in-2022-shipped-245gw/>. Accessed: 2025-02.
- [4] WALD, L. “Solar radiation energy (fundamentals).” 01 2007.
- [5] CHENG, H.-Y., YU, C.-C., HSU, K.-C., et al. “Estimating Solar Irradiance on Tilted Surface with Arbitrary Orientations and Tilt Angles”, *Energies*, v. 12, n. 8, 2019. ISSN: 1996-1073. doi: 10.3390/en12081427. Disponível em: <<https://www.mdpi.com/1996-1073/12/8/1427>>.
- [6] SUTTON, R. S., BARTO, A. G. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA, A Bradford Book, 2018. ISBN: 0262039249.
- [7] KATOPODIS, R. F. *Reinforcement Learning with Weightless Neural Networks*. Federal University of Rio de Janeiro, Rio de Janeiro, 2022.
- [8] BACELLAR, A., SUSSKIND, Z., VILLON, L., et al. “Distributive Thermometer: A New Unary Encoding for Weightless Neural Networks”. In: *ESANN*, pp. 31–36, 01 2022. doi: 10.14428/esann/2022.ES2022-94.
- [9] SOUZA, G. S., FRANÇA, P. M. V., FRANÇA, F. G. “Sun Tracking using a Weightless Q-Learning Neural Network”. In: *31st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2023.

- [10] *Global Energy Review 2021*. Relatório técnico, IEA, 2021. Disponível em: <<https://www.iea.org/reports/global-energy-review-2021>>.
- [11] SHEN, W., CHEN, X., QIU, J., et al. “A comprehensive review of variable renewable energy levelized cost of electricity”, *Renewable and Sustainable Energy Reviews*, v. 133, pp. 110301, 2020. ISSN: 1364-0321. doi: <https://doi.org/10.1016/j.rser.2020.110301>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S136403212030589X>>.
- [12] BENZEKRI, A., AZRAR, A. “FPGA-Based Design Process of a Fuzzy Logic Controller for a Dual-Axis Sun Tracking System”, *Arabian Journal for Science and Engineering*, v. 39, n. 8, pp. 6109–6123, Aug 2014. ISSN: 2191-4281. doi: 10.1007/s13369-014-1213-5. Disponível em: <<https://doi.org/10.1007/s13369-014-1213-5>>.
- [13] MOHAMAD, A., MHAMDI, H., AMIN, N., et al. “A review of automatic solar tracking systems”, *Journal of Physics: Conference Series*, v. 2051, n. 1, pp. 012010, oct 2021. doi: 10.1088/1742-6596/2051/1/012010. Disponível em: <<https://dx.doi.org/10.1088/1742-6596/2051/1/012010>>.
- [14] VIEIRA, R., GUERRA, F., VALE, M., et al. “Comparative performance analysis between static solar panels and single-axis tracking system on a hot climate region near to the equator”, *Renewable and Sustainable Energy Reviews*, v. 64, pp. 672–681, 2016. ISSN: 1364-0321. doi: <https://doi.org/10.1016/j.rser.2016.06.089>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1364032116303148>>.
- [15] VILLON, L. A. Q., SUSSKIND, Z., BACELLAR, A. T. L., et al. “A WiSARD-based conditional branch predictor”. In: *ESANN*, 2022.
- [16] AL-ROUSAN, N., ISA, N. A. M., DESA, M. K. M. “Advances in solar photovoltaic tracking systems: A review”, *Renewable and Sustainable Energy Reviews*, v. 82, pp. 2548–2569, 2018. ISSN: 1364-0321. doi: <https://doi.org/10.1016/j.rser.2017.09.077>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1364032117313266>>.
- [17] PHIRI, M., MULENGA, M., ZIMBA, A., et al. “Deep learning techniques for solar tracking systems: A systematic literature review, research challenges, and open research directions”, *Solar Energy*, v. 262, pp. 111803, 2023. ISSN: 0038-092X. doi: <https://doi.org/10.1016/j.solener.2023.111803>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0038092X23004280>>.

- [18] PATARO, I. M., CUNHA, R., GIL, J. D., et al. “Optimal model-free adaptive control based on reinforcement Q-Learning for solar thermal collector fields”, *Engineering Applications of Artificial Intelligence*, v. 126, pp. 106785, 2023. ISSN: 0952-1976. doi: <https://doi.org/10.1016/j.engappai.2023.106785>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197623009697>.
- [19] ALAMRO, H., ALQAHTANI, H., ALOTAIBI, F. A., et al. “Deep reinforcement learning based solution for sustainable energy management in photovoltaic systems”, *Optik*, v. 295, pp. 171530, 2023. ISSN: 0030-4026. doi: <https://doi.org/10.1016/j.ijleo.2023.171530>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0030402623010288>.
- [20] TAŞER, A., KOYUNBABA, B. K., KAZANASMAZ, T. “Thermal, daylight, and energy potential of building-integrated photovoltaic (BIPV) systems: A comprehensive review of effects and developments”, *Solar Energy*, v. 251, pp. 171–196, 2023. ISSN: 0038-092X. doi: <https://doi.org/10.1016/j.solener.2022.12.039>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0038092X22009197>.
- [21] ZHAO, J., WANG, A., GREEN, M. A., et al. “19.8% efficient “honeycomb” textured multicrystalline and 24.4% monocrystalline silicon solar cells”, *Applied Physics Letters*, v. 73, n. 14, pp. 1991–1993, 10 1998. ISSN: 0003-6951. doi: [10.1063/1.122345](https://doi.org/10.1063/1.122345). Disponível em: <https://doi.org/10.1063/1.122345>.
- [22] IBRAHIM, H., ANANI, N. “Variations of PV module parameters with irradiance and temperature”, *Energy Procedia*, v. 134, pp. 276–285, 2017. ISSN: 1876-6102. doi: <https://doi.org/10.1016/j.egypro.2017.09.617>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1876610217347628>. Sustainability in Energy and Buildings 2017: Proceedings of the Ninth KES International Conference, Chania, Greece, 5-7 July 2017.
- [23] REDA, I., ANDREAS, A. *Solar Position Algorithm for Solar Radiation Applications*. Relatório Técnico NREL/TP-560-34302, National Renewable Energy Laboratory, U.S. Department of Energy Laboratory, 1617 Cole Boulevard Golden, Colorado, January 2008.
- [24] ANDERSON, K., MIKOFSKI, M. *Slope-Aware Backtracking for Single-Axis Trackers*. Relatório Técnico NREL/TP-5K00-76626, National Renewable

Energy Laboratory, U.S. Department of Energy Laboratory, 1617 Cole Boulevard Golden, Colorado, July 2020.

- [25] KAMPHUIS, N., GUEYMARD, C., HOLTZAPPLE, M., et al. “Perspectives on the origin, derivation, meaning, and significance of the isotropic sky model”, *Solar Energy*, v. 201, pp. 8–12, 2020. ISSN: 0038-092X. doi: <https://doi.org/10.1016/j.solener.2020.02.067>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0038092X20301948>>.
- [26] LOUZENHISER, P., MANZ, H., FELSMANN, C., et al. “Empirical validation of models to compute solar irradiance on inclined surfaces for building energy simulation”, *Solar Energy*, v. 81, n. 2, pp. 254–267, 2007. ISSN: 0038-092X. doi: <https://doi.org/10.1016/j.solener.2006.03.009>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0038092X06000879>>.
- [27] NIKHIL KUSHWAHA, V. K. Y., SAHA, R. “Effect of partial shading on photovoltaic systems performance and its mitigation techniques-a review”, *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, v. 45, n. 4, pp. 11155–11180, 2023. doi: 10.1080/15567036.2023.2254731. Disponível em: <<https://doi.org/10.1080/15567036.2023.2254731>>.
- [28] KUMBA, K., UPENDER, P., BUDUMA, P., et al. “Solar tracking systems: Advancements, challenges, and future directions: A review”, *Energy Reports*, v. 12, pp. 3566–3583, 2024. ISSN: 2352-4847. doi: <https://doi.org/10.1016/j.egyr.2024.09.038>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2352484724006164>>.
- [29] ALLINSON, N. M., KOLCZ, A. R. “N-Tuple Neural Networks”. In: Ellacott, S. W., Mason, J. C., Anderson, I. J. (Eds.), *Mathematics of Neural Networks: Models, Algorithms and Applications*, pp. 3–14, Boston, MA, Springer US, 1997. ISBN: 978-1-4615-6099-9. doi: 10.1007/978-1-4615-6099-9\_1. Disponível em: <[https://doi.org/10.1007/978-1-4615-6099-9\\_1](https://doi.org/10.1007/978-1-4615-6099-9_1)>.
- [30] KOLCZ, A., ALLINSON, N. M. “N-tuple Regression Network”, *Neural Networks*, v. 9, n. 5, pp. 855–869, 1996. ISSN: 0893-6080. doi: [https://doi.org/10.1016/0893-6080\(95\)00116-6](https://doi.org/10.1016/0893-6080(95)00116-6). Disponível em: <<https://www.sciencedirect.com/science/article/pii/0893608095001166>>.

- [31] XAVIER, P. M., GREGORIO, M. D., FRANÇA, F. M. G., et al. “Detection of elementary particles with the WiSARD n-tuple classifier”. In: *ESANN*, 2020.
- [32] DE GREGORIO, M., GIORDANO, M. “An experimental evaluation of weightless neural networks for multi-class classification”, *Applied Soft Computing*, v. 72, pp. 338–354, 2018. ISSN: 1568-4946. doi: <https://doi.org/10.1016/j.asoc.2018.07.052>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S156849461830440X>>.
- [33] CARNEIRO, H. C. C., PEDREIRA, C. E., FRANÇA, F. M. G., et al. “The Exact VC Dimension of the WiSARD n-Tuple Classifier”, *Neural Computation*, v. 31, n. 1, pp. 176–207, 01 2019. ISSN: 0899-7667. doi: [10.1162/neco\\_a\\_01149](https://doi.org/10.1162/neco_a_01149). Disponível em: <[https://doi.org/10.1162/neco\\_a\\_01149](https://doi.org/10.1162/neco_a_01149)>.
- [34] LUSQUINO FILHO, L. A., OLIVEIRA, L. F., FILHO, A. L., et al. “Extending the weightless WiSARD classifier for regression”, *Neurocomputing*, v. 416, pp. 280–291, 2020. ISSN: 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2019.12.134>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S092523122030504X>>.
- [35] ERBS, D., KLEIN, S., DUFFIE, J. “Estimation of the diffuse radiation fraction for hourly, daily and monthly-average global radiation”, *Solar Energy*, v. 28, n. 4, pp. 293–302, 1982. ISSN: 0038-092X. doi: [https://doi.org/10.1016/0038-092X\(82\)90302-4](https://doi.org/10.1016/0038-092X(82)90302-4). Disponível em: <<https://www.sciencedirect.com/science/article/pii/0038092X82903024>>.