TU WIEN Informatics

# Rule Extraction and Feature Attribution for Explainable Reinforcement Learning

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Data Science**

eingereicht von

**Alexander Stieger, BSc.**
Matrikelnummer 01629792

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo Sauter
Mitwirkung: Dipl.-Ing. Christian Stippel

Wien, 31. März 2025

| Alexander Stieger | Thilo Sauter |
|---|---|

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

# Rule Extraction and Feature Attribution for Explainable Reinforcement Learning

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Data Science**

by

**Alexander Stieger, BSc.**
Registration Number 01629792

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo Sauter
Assistance: Dipl.-Ing. Christian Stippel

Vienna, March 31, 2025

| Alexander Stieger | Thilo Sauter |
|---|---|

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

# Erklärung zur Verfassung der Arbeit

Alexander Stieger, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 31. März 2025

Alexander Stieger

# Danksagung

In erster Linie möchte ich mich bei meinem Betreuer, Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo Sauter bedanken. Seine Expertise bildet die akademische Grundlage für diese Arbeit und ich schätze seine Unterstützung meiner Forschung sehr.

Weiters bin ich Christian Stippel, MSc. zutiefst dankbar, dessen Betreuung und Engagement wirklich besonders war. Sein aufschlussreiches Feedback, seine wertvollen Einsichten und sein Engagement spielten eine entscheidende Rolle bei der Entwicklung dieser Arbeit. Seine Unterstützung ging weit über meine Erwartungen hinaus, und ich bin zutiefst dankbar für seine maßgebliche Mitwirkung.

Darüber hinaus danke ich auch Dipl.-Ing. Dr. Aleksey Bratukhin für sein Engagement und seine Beteiligung an diesem Projekt. Sein großzügiger Einsatz von Zeit, Fachwissen und Unterstützung wird von mir sehr geschätzt und war essenziel für den erfolgreichen Abschluss dieser Arbeit.

Schließlich möchte ich meiner Lebensgefährtin Emily, meinen Eltern Anna und Peter sowie meinen Freunden und meiner Familie für ihre Geduld, Ermutigung und Unterstützung danken.

# Acknowledgements

First and foremost, I extend my sincere thanks to Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo Sauter for serving as my official supervisor. His expertise provided the academic foundation for this work, and I greatly appreciate his willingness to support my research.

I am incredibly grateful to Christian Stippel, MSc., whose supervision and dedication were truly special. His insightful feedback, meaningful insights, and commitment played a crucial role in the development of this thesis. His support went far beyond expectations, and I am deeply thankful for his invaluable contributions.

I also wish to thank Dipl.-Ing. Dr. Aleksey Bratukhin for his dedication and involvement in the project. His generous commitment of time, expertise, and support was greatly appreciated and important in successfully completing this work.

Finally, I would like to express gratitude to my partner, Emily, my parents, Anna and Peter, as well as my friends and family for their patience, encouragement, and unwavering support throughout this journey.

# Kurzfassung

Reinforcement Learning (RL) zeigt großes Potenzial, Heizungs-, Lüftungs- und Klimatechnik (HLK) durch verbesserte Energieeffizienz und Anpassungsfähigkeit zu optimieren. Allerdings gelten RL-Modelle oft als Black-Box-Systeme, was ihre Anwendbarkeit in systemkritischen und stark reglementierten Systemen einschränkt. Diese Arbeit beschäftigt sich mit Erklärbarkeit in RL-basierter Regelungstechnik. Hierfür werden zwei verschiedene Ansätze zur Verbesserung der Nachvollziehbarkeit verwendet: Erstens wird ein Stellvertretermodell entwickelt, welches das Black-Box-Policy-Modell aus dem RL-Trainingsprozess durch einen White-Box-Entscheidungsbaum ersetzt. Dabei werden Vorhersageunsicherheiten und Feature Attribution in den klassischen CART-Algorithmus (Classification and Regression Trees) integriert, um dessen Genauigkeit zu erhöhen. Zweitens werden Rule Extraction und Feature Attribution getestet, um relevante Merkmale zu identifizieren und so eine Feature Selection zu ermöglichen.

Die experimentellen Ergebnisse zeigen, dass gewichtete Entscheidungsbäume – basierend auf Vorhersageunsicherheiten oder Feature Attribution– zwar das Verhalten der Modelle beeinflussen, in den meisten Fällen jedoch keine Leistungssteigerung gegenüber dem Standard-CART-Ansatz erzielen. Wird jedoch nur ein sehr flacher Entscheidungsbaum extrahiert, so lösen die daraus resultierenden Stellvertretermodelle mit Attributions- und Unsicherheitsgewichtung die Regelungsaufgabe konsistenter als mit CART extrahierte Entscheidungsbäume.

Experimente zur Feature Selection verdeutlichen die Limitationen einzelner Methoden auf: Entscheidungsbaum-basierte und Feature Attribution-basierte Selektionsmethoden schneiden in unterschiedlichen Umgebungen gut ab. Ein kombinierter Ansatz, der beide Methoden nutzt, zeigt sich robuster und wirksamer über mehrere Umgebungen hinweg.

Die Anwendung von Feature Attribution und Rule Extraction auf ein bestehendes RL HLK-Steuerungssystem identifiziert $CO_2$-Werte als dominanten Prädiktor für Steuerentscheidungen, was deren Rolle bei der Belegungsabschätzung unterstreicht. Ohne direkten Zugriff auf die Umgebung konnten wir die Anzahl benötigter Sensoren um über 70% reduzieren. Das reduzierte Modell agiert dennoch sehr vergleichbar mit dem Ursprungsmodell, selbst ohne Zugriff auf den vollständigen Sensorensatz.

# Abstract

Reinforcement learning (RL) has shown significant potential in optimizing Heating, Ventilation, and Air Conditioning (HVAC) control systems by improving energy efficiency and adaptability. However, RL models are often considered black-box systems, limiting their applicability in safety-critical and regulatory environments. This thesis addresses the challenge of explainability in RL-based control by investigating two different approaches to explainability in RL: First, by building surrogate models that can replace the black-box policy model from the RL training process with a white-box decision tree. Here, we introduce certainty levels and feature attribution into the standard Classification and Regression Trees (CART) algorithm to enhance their performance. Second, by testing rule extraction and feature attribution, we identify the most relevant features, allowing for feature selection.

Experimental results indicate that, while certainty-based or feature attribution-based weighting schemes influence decision tree behavior, they do not improve performance over standard CART in most cases. However, when extracting only very shallow decision trees, the resulting attribution- and uncertainty-weighted surrogate models solve the environment more consistently than decision trees extracted with CART.

Feature selection experiments highlight the limitations of single-method approaches, with decision tree-based and feature attribution-based selection performing well in different environments. A combined approach leveraging both methods demonstrates improved robustness and performance across multiple environments.

Applying feature attribution and rule extraction to a pre-existing HVAC control system reveals $CO_2$ levels as the dominant predictor for control decisions, aligning with their role in occupancy inference. Without access to the environment, we were able to reduce the sensors required by over 70%. We show that the model performs very similarly, without having access to the full sensor set.

# Contents

CHAPTER 1

# Introduction

Heating, Ventilation, and Air Conditioning (HVAC) systems are crucial in modern buildings, managing the quality of air, heat, and energy use to maintain efficiency and comfort. As the need for thermal comfort increases, HVAC devices have become critical, now accounting for almost half of building energy usage and around 10-20% of total energy consumption in developed countries [PLOP08]. Improving HVAC efficiency can therefore help to reduce costs and environmental impact.

Model Predictive Control (MPC) has become a widely used method for addressing the complex and dynamic requirements of HVAC systems [TRR+19]. MPC offers a structured model-based approach by predicting successive system states and optimizing control actions over a defined time horizon. This makes MPC particularly capable of handling multivariate systems [MRRS00]. Moreover, MPC shows significant potential to reduce both costs and energy usage in HVAC systems, aided by increasing processing capacity in building automation systems and the growing availability of building data [THR22]. However, MPC performance is highly dependent on its underlying model, which must capture the complex behavior of the HVAC environment accurately [KK16].

Hence, machine learning has become an increasingly important tool because of its ability to learn directly from data, without requiring a model of the environment. In particular, Reinforcement Learning (RL) has emerged as a compelling model-free alternative for control tasks. Unlike MPC, RL is able to learn control policies directly from interactions with the environment. Through an iterative process of reward-based feedback, RL agents can dynamically modify their strategies. The model-free and data-driven nature of RL has made it a popular alternative to MPC, especially for systems where accurate environment modeling is difficult or impractical [Rec19].

This enables agents to reduce energy consumption and improve control without relying on pre-existing models. RL has shown its effectiveness in various fields, including gaming [MKS+15], natural language processing [WHL+23], and is increasingly being

explored for complex control systems such as HVAC, where it has shown the potential to improve energy efficiency and operational performance [SSS+24, WWZ17].

Recent work in RL has focused mainly on improving performance, computational efficiency, and scalability of models. However, these developments often come at the cost of interpretability. RL methods work best with black-box neural networks as basis for their internal workflow. Using neural networks for internal decision-making operations means that the decision-making process itself becomes difficult to interpret. Although numerous methods have been suggested to enhance the interpretability of RL, such efforts have not yet been able to fully address the challenges of deploying these models, limiting their full potential in safety-critical applications [ADT95, HCDR21].

Moreover, for RL to be successfully integrated into more demanding interactive tasks, including self-driving vehicles, distributed sensor networks, agile robotic systems, and arguably even HVAC, it must demonstrate both high safety and reliability. Failure in such systems can have severe societal and economic consequences, potentially even leading to loss of human life [Rec19].

Understanding the rationale behind the decisions of an RL model is essential to building trust and ensuring safety in real-world deployments. Despite achieving high accuracy, the inherent black-box nature of RL can lead to unpredictable or unsafe actions, which can prevent industry adoption due to the challenges it poses for risk assessment. To address this problem, recent research has been focusing on explainable artificial intelligence (XAI) techniques to make the decision-making process more transparent. These techniques help users understand the choices of the model and build confidence in the predictions of the model [KKP+24]. Consequently, improving the interpretability and performance of RL systems is vital to their successful integration into safety-sensitive environments.

To address these challenges, we pose the following research questions. They explore how certainty levels and feature attribution can be systematically incorporated into rule extraction for reinforcement learning and how these technologies can be used for feature selection in real-world control applications such as HVAC systems.

The first research question addresses the challenge of incorporating certainty into explainability techniques. This added layer of certainty has the potential to improve the performance of rule extraction.

> **How can certainty levels be incorporated into rule extraction for deep reinforcement learning? How does this affect the performance and interpretability of the resulting tree?**

This question explores how certainty levels can be effectively integrated into rule extraction frameworks and how this integration influences the resulting decision trees' transparency and predictive performance. By using the RL Agents' confidence associated with each decision as a weight in the tree-splitting process, our method aims to provide more accurate

and reliable explanations compared to traditional algorithms like CART [BFOS84] or VIPER [BPSL18].

The thesis then seeks to improve the performance of rule extraction further by integrating feature attribution techniques with rule extraction. Feature attribution is valuable for identifying the most influential inputs in a model's decision-making process. However, there is little research on how these techniques can be used in combination with rule extraction to provide a more comprehensive explanation. This leads to the second research question:

> **How can feature attribution be incorporated into rule extraction for deep reinforcement learning? How does this affect the performance and interpretability of the resulting tree?**

This investigation examines how feature attribution can be systematically incorporated into rule extraction processes and evaluates its impact on the clarity and effectiveness of the resulting decision trees. Feature attribution techniques are valuable for identifying the most influential inputs in a model's decision-making process. However, they rarely provide a structured way to summarize how these features translate into decisions. Our method bridges this potential gap by integrating feature attribution directly into the rule extraction process. By providing an understanding of how important features are to the Agent at a given state, it aims to improve the resulting rules in the tree-splitting process similar to certainty.

Finally, we explore the technologies of the previous research questions and how they can be used for feature selection in deep reinforcement learning. Effective feature selection is essential for improving the simplicity, performance, and interpretability [GE03] of DRL models, enabling more efficient and reliable decision-making. The final research question is:

> **To what extend can rule extraction and feature attribution be used as feature selection methods in deep reinforcement learning?**

This research explores how rule extraction and feature attribution can compute feature importance levels and, therefore, be leveraged as feature selection methods in deep reinforcement learning. Effective feature selection can improve both the performance and the interpretability of DRL models, improving decision-making efficiency and reliability. We demonstrate the practical benefits of combining performance and interpretability in DRL, highlighting how explainability techniques can streamline the learning process and improve model generalization.

This thesis aims to answer the research questions and therefore the challenges by converting RL models into a surrogate model while minimizing performance trade-offs. More specifically, we are building a decision tree as a surrogate model because of its white-box

nature. To do this more efficiently than traditional rule extraction, we introduce two novel rule extraction approaches: one that uses certainty levels and the other that computes feature attribution, incorporating both of them into the decision tree-splitting algorithm. Furthermore, we evaluate how rule extraction and feature attribution can be leveraged for feature selection, identifying the most relevant inputs to enhance model efficiency and robustness. The proposed techniques are first tested on benchmark RL tasks to assess their validity in controlled environments before being applied to a simulated HVAC system.

The contributions of this thesis are summarized as follows:

- Introduced a new approach to rule extraction, leveraging the certainty levels of the RL agent
- Introduced a novel combination of feature attribution and rule extraction by utilizing feature attribution weighted decision trees.
- Application of rule extraction and feature attribution to DRL for feature selection
- Introduced a combination of rule extraction and feature attribution to DRL for feature selection

The remainder of this thesis is structured as follows. Chapter 2 reviews previous and related work to provide more context. Chapter 3 addresses two separate architectures of this thesis: the methodology used to interact with Gymnasium [TKT+24] control problems and the architecture of an HVAC control model aimed at optimizing energy and comfort. Chapter 4 discusses the state-of-the-art on pedagogical rule extraction and feature attribution, as well as the implementation of them in this paper. In Chapter 5 we introduce the novel approaches to rule extraction using either certainty levels or feature attribution as weights. In Chapter 6 integrative approaches are being applied and evaluated on Gymnasium control problems. Chapter 7 shows how a combination of rule extraction and feature attribution works best on the environments. Finally, in Chapter 8 we summarize our findings and discuss future work.

CHAPTER 2

# Related Work

The challenges of deploying RL in risk-sensitive, complex and dynamic environments, such as HVAC systems, show the importance of improving reliability and transparency. To address this, explainable RL (XRL) approaches seek to make learned policies more transparent. Two major XRL strategies are:

- rule extraction – deriving human-readable rules from trained models
- and feature attribution - quantifying the importance of each input for the model's decisions.

Rule extraction has been applied to standard control problems to interpret RL policies. A prominent example is the VIPER algorithm by Bastani et al. [BPSL18], which first trains a complex model and then extracts a decision tree policy via imitation learning. Vos and Verwer [VV24] showed good performance of the VIPER's surrogate model when applied to Gymnasium [TKT+24], formerly known as OpenAI's Gym [BCP+16], control problems. For example, on the CartPole balancing task, a small decision tree can be learned to replicate the optimal agent's behavior, yielding simple rules (e.g. "if pole angle $>$ X, push left, else push right") that explain the controller's decisions. Such distilled rule-based policies often retain near-optimal performance. These approaches demonstrate that even on standard control benchmarks, it is possible to replace black-box neural policies with white-box rule-based policies without a large loss in reward [VV24]. Overall, rule extraction in Gymnasium tasks serves as a proving ground for interpretable RL, illustrating in controlled settings how complex decision policies can be approximated by concise sets of rules. However, the VIPER algorithm requires access to the environment for extracting a surrogate model, which can hold it back in real-world applications.

Engelhardt, Lange, and Wiskott propose a sample-based rule extraction approach to enhance explainability in RL systems [ELWK23]. By observing agent actions and

environmental states, they do not require access to the environment during learning, but can instead learn from state-action pairs. They also extract a decision tree from the state-action relationships, but instead of using VIPER they are using CART [BFOS84], which does not require access to the environment for rule extraction. Nonetheless, according to their implementation they use specific seeds for randomness(16 to 20), without specifying on why they used those seeds, potentially leading to unlikely outcomes. Liu, Schulte, and Zhu present a framework aimed at enhancing interpretability in deep reinforcement learning (DRL) by employing Linear Model U-Trees (LMUTs) to approximate Q-function neural networks [LSZL19]. This approach enables a more transparent understanding of agent strategies by supporting rule extraction and feature influence analysis. However, it may be overly complex when the objective is merely to replicate system behavior, as Q-function approximation is unnecessary for straightforward imitation tasks. Both Engelhard et al.'s and Lui et al.'s approaches were only tested on simple classical control problems, which leaves their potential of scalability to more complex, high-dimensional environments open.

Rule extraction has been actively explored for building HVAC systems to create interpretable and practical control strategies. For instance, Cho and Park developed an explainable RL framework for a building cooling system by first training a Deep Q-Network and then using a decision tree to extract simple "if-then" rules that approximate the network's policy [CLO+23]. The extracted rules achieved significant energy savings close to the black-box optimum while being easy for practitioners to understand. This demonstrates that even for complex HVAC dynamics, we can derive human-readable control policies (e.g., decision trees or sets of rules) that closely mimic the performance of deep RL controllers while revealing the logic in terms of familiar HVAC variables (like temperatures, occupancy, etc.). Additionally, this is valuable for building operators who require trust and insight into automated control recommendations. Although this study shows the potential of rule extraction in the domain of HVAC, it only test the success and performance on a specific system involving a geo-thermal heat pump and ice thermal storage. Further tests and evaluations for the effectiveness of rule extraction in more broad use cases are needed.

Another XAI method is feature attribution, which assigns each input feature (sensor or state variable) a score indicating its influence on the RL agent's decision. In the HVAC context, feature attribution can highlight which factors (e.g. room temperature, occupancy, outdoor weather) drive the control actions at a given time. Feature attribution methods have also been tested on classic control problems and game environments, where they help interpret what the agent has learned. In image-based RL domains like Atari games, saliency maps highlight which parts of the input frame an agent is focusing on [GKDF18]. Greydanus et al. showed that visualizing an Atari agent's learned policy via saliency revealed, for instance, that a Pong agent attends to the ball rather than distractor objects. In continuous-state control tasks gradient-based attributions can similarly identify which state variables are most critical. Indeed, saliency-based explanations in RL generally aim to "highlight the features of the input state that are

most relevant for the agent in taking an action." [PVG+19] Recent studies evaluate a range of such methods (e.g. vanilla gradients, Integrated Gradients, perturbation-based masks) on standard RL benchmarks to determine how reliably they pinpoint important features [Hel]. Nonetheless, feature attribution is emerging area for HVAC-RL. Recent work has started to incorporate these methods. For example, Jiménez-Raboso et al. combined multiple XAI techniques (surrogate decision tree models, Shapley value analysis, and counterfactual scenarios) to explain a deep RL controller for a building HVAC system [JRMCN+23]. Their approach could identify the most influential state features for the HVAC control policy, e.g., highlighting that zone humidity and time of day were key determinants for a ventilation action. Nonetheless, the implications and uses of the computed XAI method still need to be explored and tested to see if this could lead to feature reduction or other performance improvements.

Furthermore, feature attribution can also be used as feature importance metrics: by aggregating these local attributions across many states or episodes, one can derive a ranking of features by their overall impact on the policy. For example, if integrated gradients repeatedly assign high importance to a building's indoor temperature across various scenarios, it indicates that temperature is a critical driver in the policy's decisions. In machine learning literature, feature attribution methods assign scores to input features, the absolute value of which informally represents their importance to the model prediction [ZBRS22]. This principle carries into RL: saliency maps and input-gradient methods highlight important state variables, effectively producing a weighted importance measure for each feature at each decision point. One advantage of attribution methods is their fine-grained, instance-specific insight – they can explain why a particular action was chosen in a particular state by showing which features were pivotal. This is useful for debugging or improving policies: If an agent occasionally makes an odd decision, analyzing the attributions of that timestep might reveal that it overemphasized a certain input. Over time, patterns in attributions can be used to infer global importance. Permutating features is a way to validate attribution-based importance: removing or randomizing a highly attributed feature should greatly degrade performance if the attributions are accurate [HLA22] [Hel]. In deep RL for control, these methods can thus serve as an importance estimator to find the most influential state inputs. However, attributions depend on the model and state – they reflect a correlation with the learned policy, which might latch onto spurious features if the training data is biased.

For the field of HVAC, a recent study demonstrated the use of integrated gradients to interpret a neural network for the prediction of HVAC energy [WWGL22]. The study quantified the importance of each feature by computing the output gradients with respect to the input features; this enabled effective feature selection and dimensionality reduction. This approach improved the interpretability of the model and reduced computation time without sacrificing predictive accuracy. Wang et al. highlighted that gradient-based explanations can enhance the explainability and efficiency of RL-based HVAC control systems, proving their successful adoption in real-world, safety-critical environments [WWGL22]. In contrast, while this thesis also applies feature attribution

for feature selection and dimensionality reduction, it instead applies it to RL-based HVAC control. In addition, we introduce rule extraction to make RL-derived control policies more robust and interpretable.

CHAPTER 3

# System Architecture

This chapter addresses two types of environments used in this thesis: the Gymnasium [TKT+24], formerly known as OpenAI's Gym [BCP+16], based control problems for RL agents and an HVAC control model aimed at energy and comfort optimization.

The standardized Gymnasium control framework makes training, testing, and evaluation possible for our novel approaches in standardized, easy to use and well-researched environments. It allows for the interaction between the agents and the Gymnasium environment, enabling the agent to observe state transitions, receive rewards, and select actions based on its learned policy. Two different RL agents, Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO), are applied to the control problems.

In contrast, the architecture of the HVAC system focuses not on the construction or testing of the RL components within this work, but rather on applying the successful approaches to the framework and methodology proposed by Bratukhin et al. [BSS+24]. In this framework, all elements of the HVAC environment and control model are pre-defined. The architecture is based on a data-driven RL algorithm designed to optimize energy usage while maintaining indoor comfort based on a multisensor setup in an office building environment. In this context, our work does not aim to alter the model's architecture or develop new control algorithms, but rather, we apply XAI methods to an existing approach without having direct access to the environment or the possibility to rerun the HVAC experiment.

This dual approach allows for the exploration and evaluation of RL applications on benchmarks and shows its application in existing practical domains. While the Gymnasium control problems enable a controlled examination of RL algorithm performance, the HVAC efficiency model demonstrates RL's potential for real-world impact in energy management.

## 3.1 Gymnasium Control Framework

To evaluate and benchmark control algorithms effectively, researchers like to use standardized environments where the dynamics are well-defined and already well-researched. This allows for a clear assessment of algorithm performance and robustness, as well as the possibility to benchmark with other research. One very widely used framework in reinforcement learning and control problem research is Gymnasium [TKT+24]. Gymnasium provides a collection of simulation environments for testing and developing algorithms. These environments are essential tools, enabling us to experiment in controlled settings that mimic various dynamic challenges. Among the suite of environments available in Gymnasium, this paper experiments on three well-known problems: CartPole-v1(CartPole), MountainCar-v0 (MountainCar), and LunarLander-v3(LunarLander), each offering unique challenges for the RL algorithms and our approaches.

### 3.1.1 Environments

The **CartPole** problem is a classic benchmark in control theory and reinforcement learning [BSA83], often used to illustrate the fundamental principles of dynamic control tasks. In this environment, a pole is connected to a cart that can move horizontally left or right, as displayed in Figure 3.1. The objective is to apply either a left or right force to the cart to keep the pole balanced upright as long as possible. Rewards are provided for each time step that the pole remains upright, incentivizing the agent to balance the pole over extended periods. While the task may appear straightforward, CartPole requires careful balancing of the pole's dynamics by making timely corrective actions and thus captures essential aspects of control challenges, such as stability maintenance and real-time decision-making. Although simple in design, CartPole has become a foundational problem in reinforcement learning research due to its simplicity and interpretability.

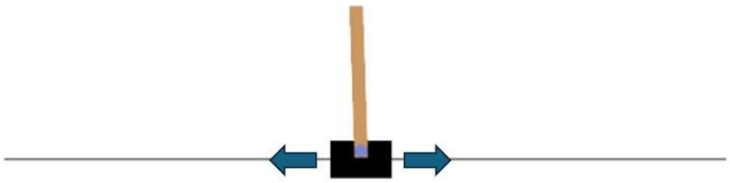


Figure 3.1: CartPole environment in Gymnasium

In contrast to CartPole, which focuses on balancing, the **MountainCar** problem emphasizes overcoming momentum-based challenges, as illustrated in Figure 3.2. The MountainCar environment consists of an underpowered car positioned in a valley between two hills, with the objective to drive the car to the top of the right hill. Due to insufficient engine power to ascend directly, the car must build momentum by oscillating between the hills. The agent can either accelerate left, right, or not accelerate at all and needs to learn

a control policy that utilizes the car's momentum effectively. By penalizing every time step, rewards are structured to minimize the number of time steps needed to reach the goal, encouraging the agent to find an efficient strategy. MountainCar is valuable because it is a non-trivial control problem where suboptimal or greedy actions may prevent task completion. The need to account for accumulated momentum and delayed reward makes RL a suitable approach because RL can balance between immediate and future rewards.

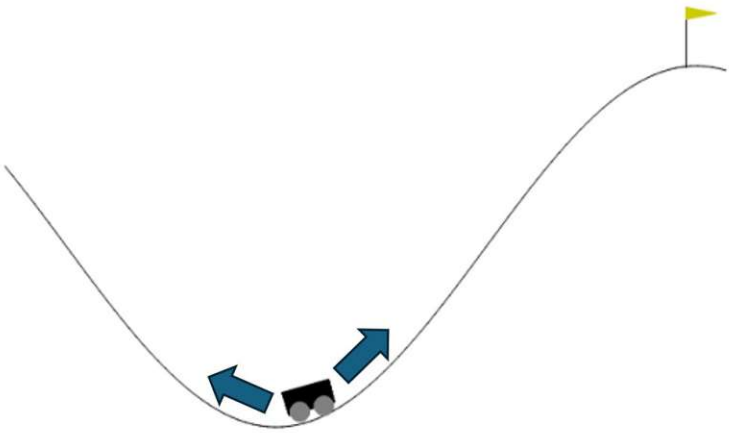


Figure 3.2: MountainCar environment in Gymnasium

The Discrete **LunarLander** is the most complex environment of the Gymnasium project that we use. It simulates a spacecraft that needs to be landed on a designated landing pad. The agent can fire the lander's main engine or either side orientation engine to guide it safely to the specified location, as shown in Figure 3.3. Rewards are provided for maintaining stable trajectories, minimizing fuel usage, and achieving a successful landing. Penalties are incurred for crashing or exiting the simulation boundaries. LunarLander challenges the agent to balance precision and efficiency, making it a valuable environment for testing policies that require fine-grained control and decision-making under constraints. This problem exemplifies tasks where high-dimensional continuous actions are necessary, emphasizing the importance of balancing competing objectives in real time.

These environments in Gymnasium offer standardized control tasks that help evaluate algorithms in scenarios with different properties. Successful performance in these benchmark environments provides a reasonable indication of an algorithm's ability to handle more complex and high-stakes real-world control applications. However, while these environments provide an accessible starting point for testing performance, real-world control tasks pose additional challenges related to uncertainty, safety constraints, and the need for interpretability in decision-making processes.

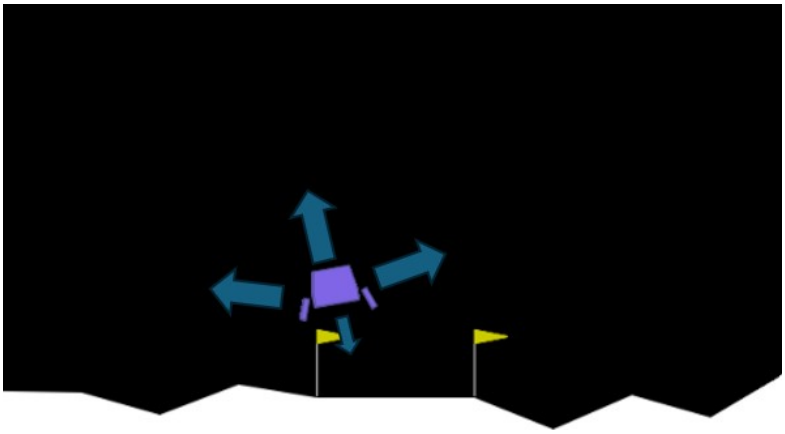


Figure 3.3: LunarLander environment in Gymnasium

## 3.2 HVAC Framework and Efficiency Model

The proposed HVAC automation framework [BSS+24] integrates two main models: a RL model for optimizing HVAC efficiency and a supervised learning model dedicated to predictive maintenance. This thesis focuses exclusively on the HVAC efficiency model implemented by Stippel et al. [SSS+24] as an online HVAC optimization model. The predictive maintenance model, which employs supervised learning to determine the time of failure of particular components and schedule maintenance, remains outside the scope of this work.

The HVAC efficiency model uses PPO to optimize energy consumption based on real-time environmental data. The model dynamically configures the HVAC system settings in order to minimize energy consumption while maintaining thermal comfort. The RL model operates within the broader system architecture described by Bratukhin et al. [BSS+24], where it controls a dynamic system-in-the-loop simulation that continuously interacts with the building's environment through control commands.

The training phase of the model occurs in the TRNSYS18 [Kle88] building simulation environment, which replicates a multi-zone office building. The PPO agent uses 37 sensory data points that represent a range of environmental and operational dynamics, such as temperature, humidity, $CO_2$ levels as well as system variables, like airflow rates and power consumption described in Table 3.1 [SSS+24].

The RL model's optimization is guided by a reward function that weighs different aspects, such as energy consumption and user comfort. Preliminary simulations show that this model can outperform traditional control methods, achieving significant energy savings over time, which underscores the potential of RL-based automation to enhance HVAC efficiency in real-world settings [SSS+24].

However, such an extensive set of sensors may not always be available or affordable, necessitating a focus on identifying the most relevant sensors that significantly impact

Table 3.1: State Space for Multi-Zone HVAC Control [BSS+24]

| Parameter | Unit | Description |
|---|---|---|
| T_Amb | °C | Outdoor Temperature |
| h_Amb | % | Outdoor Humidity |
| CO2_ODA | ppm | $CO_2$-Level of Outdoor Air |
| T_OP_i | °C | Operational Temperature in Zone i |
| h_Z_i | % | Relative Humidity in Zone i |
| CO2_Z_i | ppm | $CO_2$-Level in Zone i |
| VFR_Z_i | $m^3/h$ | Volume Flow Supply and Extract Air in Zone i |
| qH_Z_i | $W/m^2$ | Specific Heating Demand in Zone i (Radiators) |
| Q_Z_i | W | Power Consumption in Zone i |

RL performance.

## 3.3 RL Agents

In this work, we deploy two popular reinforcement learning algorithms, DQN and PPO, both implemented using the Stable-Baselines3 (SB3) library [RHG+21]. SB3 provides reliable, well-tested, and easy-to-use implementations of RL algorithms which are also easily integrated in the Gymnasium Environments. The DQN and PPO algorithms were selected due to their proven effectiveness in handling control tasks [ELWK23] [LSZL19], offering a robust basis across different control problems.

### Deep Q-Networks

The Deep Q-Network algorithm, introduced by Mnih et al. [Mni13] [MKS+15], combines classical Q-learning [WD92] with deep neural networks to approximate the action value function. SB3 DQN algorithm uses the core stabilizing techniques from the foundational paper: experience replay and a target network. Specifically, it maintains a separate target Q-network (with parameters $\theta^-$) that is updated periodically. The DQN loss function in SB3 follows the standard Bellman residual of the original DQN introduced by Mnih et al. [MKS+15]

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \tag{3.1}$$

where $L_i(\theta_i)$ is the squared Bellman error, with $s, a, r, s'$ representing the state, action, reward, and next state. $Q(s, a; \theta_i)$ is the predicted Q-value under the current network, while the target is composed of the reward $r$ and the discounted maximum future Q-value $\max_{a'} Q(s', a'; \theta_i^-)$ from the target network. The loss is computed over samples drawn uniformly from the experience replay buffer $\mathcal{U}(D)$.

### Proximal Policy Optimization

SB3's PPO implementation follows the PPO-Clip variant introduced by Schulman et al. (2017) [SWD+17]. Stable Baselines' PPO has the clipped surrogate objective: after each policy update, the change in the policy is constrained by clipping the policy probability ratio. The main objective proposed by Schulman et al. [SWD+17] is:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right], \tag{3.2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ is the policy ratio and $\epsilon$ is the clip range

The clipping term, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$, prevents large deviations from the old policy, thus maintaining a well-defined trust region without requiring a complex optimization process. PPO alternates between sampling data through interaction with the environment and optimizing the objective function using stochastic gradient ascent.

CHAPTER 4

# Explainable Reinforcement Learning

Although RL has shown great promise in a variety of domains, its applicability to real-world systems is often limited by its lack of interpretability. This paper discusses two XAI methods that might help shed light into the black-box nature of RL models: pedagogical rule extraction and feature attribution.

## 4.1 Pedagogical Rule Extraction

Rule extraction techniques aim to extract understandable rules or decision-making logic from a trained neural network. As defined by Craven, the objective of rule extraction is to "produce a description of the network's hypothesis that is comprehensible yet closely approximates the network's predictive behavior," given both the trained neural network and the data on which it was trained [Cra96].

They can generally be classified into three distinct types: Pedagogical, Decompositional and Eclectic [AAES+23]. Pedagogical rule extraction algorithms "view rule extraction as a learning task where the target concept is the function computed by the network and the input features are simply the network's input features" [CS94] On the other hand, Decompositional methods, "operate on the neuron" level, while "Eclectic methods" are a combination of decompositional and pedagogical methods" [AAES+23].

This work focuses exclusively on pedagogical rule extraction. This approach aligns with our goal of extracting interpretable rules without the need to analyze the internal structure of the network since pedagogical methods treat the neural network as a black box, which is particularly advantageous for HVAC systems because it can be applied independently of the architecture or type of the original model. Using pedagogical methods, we can provide insights into model decisions that are understandable to

operators and stakeholders without requiring specialized knowledge of the model's internal workings, thereby facilitating more transparent and trustworthy decision-making in HVAC applications.

In pedagogical rule extraction, decision trees are commonly used because of their straightforward and interpretable structure. As Huysmans et al. write, "decision trees are usually constructed directly from the available training observations and are therefore not considered to be rule extraction techniques in the strict sense of the word" [HBV06]. However, when they are applied for pedagogical rule extraction purposes, decision trees can be used to approximate the behavior of a black-box model by training on data points labeled with the target values predicted by the original model rather than the targets [HBV06]. This allows the tree to learn a surrogate model that closely aligns with the predictions of the underlying network while maintaining interpretability.

One of the most widely used methods for constructing decision trees is the Classification and Regression Trees algorithm (CART), introduced by Breiman et al. [BFOS84]. The CART algorithm builds binary trees by recursively splitting the data based on features that result in the highest information gain, thus creating a hierarchical structure that captures decision rules in an interpretable form. Each node represents a decision based on a specific feature, and each branch corresponds to an outcome that further partitions the data, eventually leading to terminal nodes with predicted outcomes. CART can be applied to both classification and regression tasks, making it highly versatile for extracting interpretable rules across various applications [BFOS84].

The quality of the splits in the trees is assessed using the Gini index for classification and squared error (SE) for regression,

$$Gini = 1 - \sum_{i=1}^{k} p_i^2 \qquad\qquad SE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \,, \tag{4.1}$$

where $p_i$ represents the proportion of observations in class $i$, $k$ represents the number of classes, and $y_i$ and $\hat{y}_i$ denote the actual and predicted values, respectively [BFOS84].

An easily reproducible application of a pedagogical rule extraction was made by Engelhardt et al. [ELWK23], where they tested a simple sample-based rule extraction algorithm on classical control problems from OpenAI's Gym environment [BCP+16]. By storing the labeled samples automatically created in the learning step of the DRL, they translate the problem into a simple supervised learning setting. Finally, they solved the supervised learning task with the CART Algorithm [BFOS84] and were able to extract a very simple and understandable decision tree, as seen in Figure 4.1 for the CartPole control problem [ELWK23].

While pedagogical rule extraction has proven successful on this rather simple problem, it has been shown to lack interpretability on more complex problems due to the high numbers of rules extracted [HBV06]. Therefore, there is a need to make rule extraction

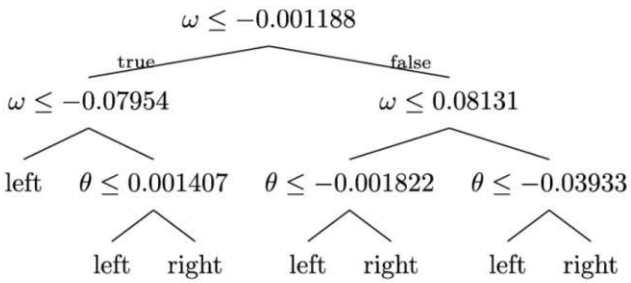



Figure 4.1: Extracted Decision Tree CartPole [ELWK23]

with Velocity $\omega$ and Pole angle $\theta$

more efficient, either by limiting the number of rules or requiring a certain performance. In other words, given a set number of rules, what is the performance a rule extraction algorithm can achieve, or, given a set minimum required performance, what is the simplest tree the rule extraction algorithm can build?

## 4.2 Feature Attribution

While rule extraction techniques aim to generate explicit decision rules that approximate a model's behavior, feature attribution methods take a complementary approach by identifying the most influential features contributing to the model's predictions. Rather than producing an alternative rule-based model, FA methods "compute the attribution of each input feature to represent its importance", providing a ranked perspective on how each feature impacts the predictions [ZBRS22]. This approach can be particularly valuable for complex models where rule extraction alone may not yield sufficiently interpretable results.

FA techniques offer several advantages for interpretability in HVAC applications and other control tasks. By highlighting the primary factors influencing the model's decisions, FA allows operators to gain insights into which variables drive critical predictions, such as energy efficiency adjustments or fault detection. For example, in an HVAC system, FA might reveal that specific environmental variables, such as outdoor temperature or indoor occupancy levels, are consistently more impactful in certain control decisions. Understanding these influences helps stakeholders ensure that the model aligns with domain-specific expectations and requirements, thereby increasing trust and transparency in the system's operations.

FA in neural networks using gradient-based methods was pioneered by Simonyan et al. (2013) [SVZ13], who introduced the gradient method (Grad) for computing feature-wise partial derivatives of the output with respect to the input features. Initially applied to image data, this approach calculates the sum or maximum of absolute gradient values across color channels, producing visual representations known as saliency maps (Saliency).

These maps highlight the most influential regions in an image by showing which pixels have the greatest impact on the model's output, providing insight into how the model interprets different parts of the input. This approach "indicates which features need to be changed the least to affect the score of a particular class the most" [KK21].

Gradient-based feature attribution methods, like the simple gradient method, can also be used on a global scale to estimate feature importance across the entire input space. By averaging or aggregating the absolute gradient values of each feature across many input samples, this approach provides a global measure of feature importance, helping identify which features are consistently influential in model predictions. This method has been particularly useful in identifying key factors in complex models across various domains. For instance, Ancona et al. discuss using aggregated gradients to prove reliable feature importance scores across samples, thereby capturing a broader understanding of which features contribute most significantly to the models' behavior [ACOG17].

However, while the gradient method effectively distinguishes important and unimportant features on a global scale, it is limited as an attribution method for effect decomposition [KW24]. Effect decomposition aims to attribute specific portions of the output to individual features or feature combinations, isolating their unique contributions. The simple gradient methods show an inconsistent attribution and, thus, a strong model dependency. This limitation has driven the development of alternative methods that yield more stable and meaningful decompositions of feature effects [KW24].

One such method is **Input** $\times$ **Gradient**, which was introduced to address some of the interpretability issues of the simple gradient method while maintaining computational simplicity. Rather than relying solely on the partial derivative of the model's output, this approach multiplies the gradient by the corresponding input feature value to produce a saliency score:

$$\text{IxG}(x_i) = x_i \cdot \frac{\partial F(x)}{\partial x_i}, \tag{4.2}$$

where $F$ represents the model's output, $x_i$ denotes the input feature, and $\frac{\partial F(x)}{\partial x_i}$ is the gradient of the output with respect to the input feature [ACOG19].

This approach highlights features that have both a large input value and a significant gradient, effectively weighting the influence of each feature by its magnitude. In contrast to the simple gradient method, Input $\times$ Gradient can better capture the proportional importance of features, making it particularly useful for models where inputs contribute to predictions in a linear or categorical way [KW24].

While Input $\times$ Gradient improves upon the simple gradient method, it can still be sensitive to the choice of input values, particularly when the input features have large variations. To address some of this sensitivity to noise and local variations, **Integrated Gradients** was introduced as an alternative attribution method. Proposed by Sundararajan et al.

(2017) [STY17], Integrated Gradients computes the cumulative effect of each input feature along a continuous path from a baseline input (such as a zero vector) to the actual input. This integration over a path helps produce smoother, more interpretable attributions, providing a holistic view of each feature's contribution to the model's output.

Integrated Gradients calculates FA by integrating the gradients of the model's output with respect to its inputs as they transition from a baseline to the actual input:

$$\mathrm{IG}(x_i) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha \,, \tag{4.3}$$

where $F$ represents the model's output, $x_i$ denotes the input feature, and $x_i'$ is the baseline [STY17].

Figure 4.2 shows an example of the gradient of the input feature applied to a random state of a DQN agent trained to the CartPole problem. While the CartPole has a center position, the angle the pole is tilted quite strongly. The feature attribution computed on the right-hand side shows that for making a decision in this very instance, the agent deems the feature Angle more important than the other features. Although this method provides us with some insight into the inner workings of the model, it does not address potential risks or ensure certain outcomes.

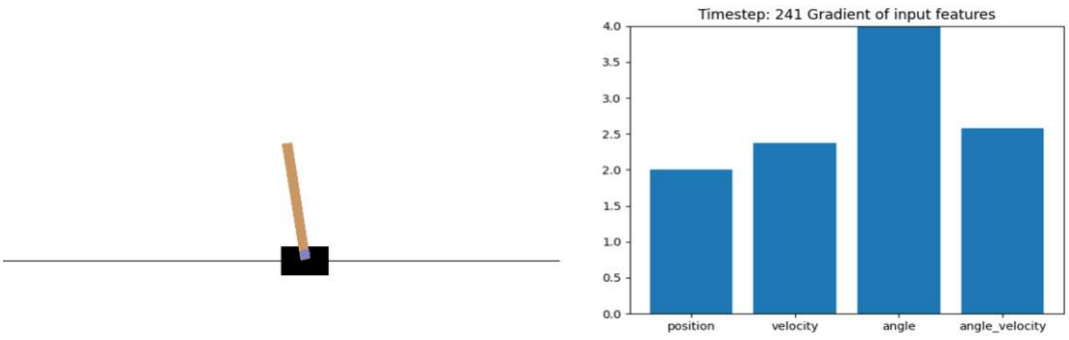



Figure 4.2: Feature Attribution CartPole

## 4.3 Feature Attribution and Rule Extraction for Feature Importance

FA can also be used as a feature importance metric [ACOG17]. Formally, for the feature attribution method Input × Gradient the feature importance is computed by averaging over the mean feature attributions as

$$\mathrm{FI}_{\mathrm{FA}}(x_i) = \frac{1}{N} \sum_{j=1}^{N} \left| \frac{\partial F(x^{(j)})}{\partial x_i} \right|,$$

where $N$ is the total number of samples.

On the other hand, in rule extraction, a decision tree is trained on data labeled with the model's predictions. The resulting tree encapsulates the decision-making process in a hierarchical structure. Like feature attribution, rule extraction can also be used for feature selection. Besides yielding human-readable policies, rule extraction methods inherently perform a form of feature selection. The simplified rules distilled from a deep RL agent typically involve a subset of the original input features – those most salient for decision-making. For instance, if a decision tree policy extracted from a HVAC controller uses only outdoor temperature and room occupancy in its splits, we learn that other inputs (e.g. humidity) were less influential on the control logic. In this way, rule extraction serves as a global feature importance measure for the policy. The presence of a feature in the extracted rules (especially high in a decision tree) implies it had a strong effect on the agent's actions, whereas omitted variables likely had minimal impact. Determining a feature's contribution to decision-making can be quantified through measures like conditional entropy, which captures how much information a feature provides when partitioning data. More precisely, the feature importance is determined by summing the reductions in impurity achieved at each node where the feature is used to split the data. [NAPS11] [BH21].

The impurity reduction is weighted by the number of samples reaching the node, yielding an expression of feature importance as

$$\mathrm{FI}_{\mathrm{DT}}(x_i) = \sum_{t \in T_i} \frac{n_t}{N} \Delta I_t,$$

where $T_i$ is the set of nodes splitting on $x_i$, $n_t$ represents the number of samples at node $t$, $N$ is the total number of samples, and $\Delta I_t$ is the decrease in impurity at node $t$. This calculation accounts for both the frequency of splits involving a feature and the effectiveness of those splits in reducing uncertainty.

To summarize, both approaches yield complementary perspectives on feature importance. Feature attribution focuses on the sensitivity of the output with respect to each input, while rule extraction quantifies feature importance based on the structure and performance of an interpretable model. The integration of these methods can provide a robust framework for understanding which features most significantly affect a model's decisions.

CHAPTER 5

# Integrative Approaches

This chapter introduces two novel rule extraction methods that aim to improve upon traditional rule extraction. By integrating advanced mechanisms such as feature attribution and model certainty into the tree construction process, these approaches aim to use this additional information to improve performance while leveraging the explainability of decision trees. The novel approaches discussed are Feature Attribution Weighted Decision Trees (FAWDT) and (Un-)Certainty Weighted Decision Trees.

## 5.1 Feature Attribution Weighted Decision Trees

This work proposes a novel framework for explainability that integrates rule extraction with feature attribution. Feature attribution values provide insight into the factors driving an RL agent's decisions, enabling the construction of decision trees that closely reflect the agent's reasoning.

Traditional decision trees partition the input space on the basis of the inequalities in the labels predicted by the RL agent. This process is shown on the left-hand side of Figure 5.1, where the algorithm chooses a split point that optimally separates the data points according to their labels, depicted as circles or squares. The dotted line in the figure represents the decision boundary generated by this traditional splitting process, which is solely informed by the distribution of target labels.

In contrast, the right-hand side of Figure 5.1 illustrates the proposed feature attribution-weighted splitting process. Here, the split is chosen not only by the distribution in the output labels but also by the feature attribution values extracted from the RL model. These values quantify the relative importance of individual features for specific outputs. By including these importance scores, the split point is changed to potentially better reflect the significance of the features in the model's given decisions. As shown, this adjustment can lead to a decision boundary that differs from the one produced by

traditional methods, as it can account for both the predictive behavior of the RL agent and the importance context provided by the feature attribution values.

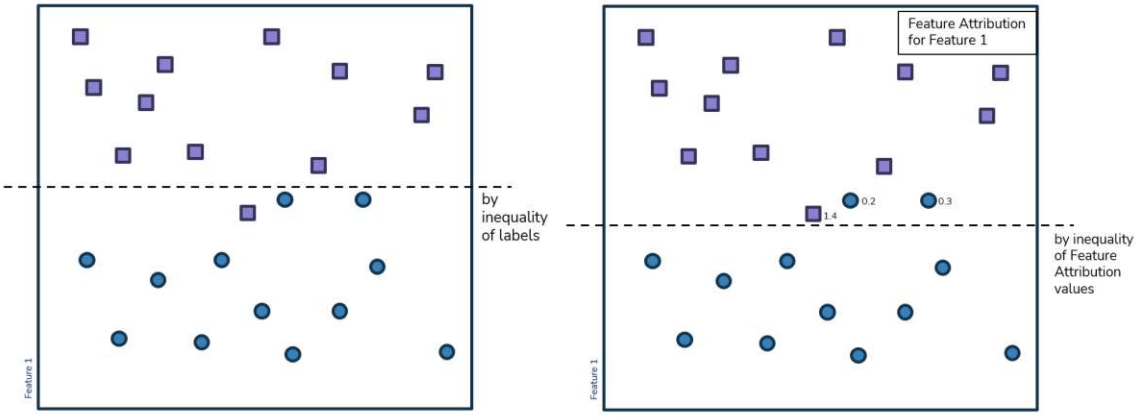



Figure 5.1: Conventional Split (left) vs Feature Attribution Weighted Split (right)

The incorporation of feature attribution values tries to capture a more nuanced representation of the model's behavior. Instead of treating all features equally during the splitting process, the weighted approach emphasizes features with greater importance. The aim is to maintain the high interpretability of the resulting rules while increasing the performance, as the extracted decision tree aligns more closely with the factors that the RL agent uses to make predictions.

### 5.1.1 Splitting Mechanisms

While there are also other cart splitting criteria, the Gini impurity is the common criterion used for CART to measure the impurity of a split [BFOS84]. The objective is to find splits that minimize impurity, thus enhancing the homogeneity of the resulting subsets. The following subsections compare the traditional Gini split with a novel approach where the Gini impurity is weighted by feature attribution values derived from an RL model.

#### Traditional Gini Split

As discussed, the Gini impurity is calculated as:

$$Gini = 1 - \sum_{i=1}^{k} p_i^2 \tag{5.1}$$

with $p_i$ representing the proportion of observations in class $i$ and $k$ representing the number of classes [BFOS84]. For creating a split into left ($S_L$) and right ($S_R$) subsets, the Gini index of the split is computed as the weighted average of the Gini impurities of the subsets:

$$\text{Gini}_{\text{split}} = \frac{|S_L|}{|S|} \cdot \text{Gini}(S_L) + \frac{|S_R|}{|S|} \cdot \text{Gini}(S_R), \tag{5.2}$$

where $|S_L|$ and $|S_R|$ are the sizes of the left and right subsets, respectively, and $|S| = |S_L| + |S_R|$ is the size of the original dataset.

#### Feature Attribution Weighted Decision Tree: Gini Split

The proposed novel Feature Attribution Weighted Gini (FAW-Gini) modifies the previously introduced "traditional" Gini impurity by incorporating feature attribution values to weigh the contributions of each data point. Unlike the traditional Gini impurity, which treats all features equally, the FAW-Gini emphasizes features that are most influential for a given decision. The weighted Gini impurity for a dataset $S$ is defined as:

$$\text{FAW-Gini}(S, f) = 1 - \sum_{i=1}^{k} \left( \frac{A_{i,f}}{A_f} \right)^2, \tag{5.3}$$

where $A_{i,f}$ is the total feature attribution for feature $f$ across samples belonging to class $i$, and $A_f$ is the total feature attribution for $f$ across all samples in $S$. For the split into the left ($S_L$) and right ($S_R$) subsets, the Feature Attribution Weighted Gini index of the split is computed as:

$$\text{FAW-Gini}_{\text{split}} = \frac{|S_L|}{|S|} \cdot \text{FAW-Gini}(S_L, f) + \frac{|S_R|}{|S|} \cdot \text{FAW-Gini}(S_R, f), \tag{5.4}$$

where the feature attribution weights adjust the impurity calculation based on the relative importance of feature $f$ for each subset.

### 5.1.2 Pseudo-code for Tree Creation with FAW-Gini Splitting

The following pseudocode outlines the interaction between the FAW-Gini splitting criterion and the decision tree creation process:

**Algorithm 5.1:** Decision Tree Creation with FAW-Gini Splitting Criterion

**Input:** Dataset $X$, Labels $y$, Feature Attribution Matrix $F_{\text{Attributions}}$, Parameters: max_depth, min_samples_split, max_features
**Output:** Trained Decision Tree tree

**Recursive Tree Construction:**
**Function GrowTree($X$, $y$, $F_{\text{Attributions}}$, depth)**
  **if** *Stopping criteria met (max_depth, homogeneous labels, or too few samples)* **then**
    **return** Leaf node with the most common label in $y$
  **end**
  Compute the best feature and split value for splitting using `FAW-Gini Split`
  Split data into left and right subsets based on the split value
  Recursively call GrowTree for left and right subsets:
    Left subtree: GrowTree(left_X, left_y, left_F_Attributions, depth+1)
    Right subtree: GrowTree(right_X, right_y, right_F_Attributions, depth+1)
  **return** A decision node with the best feature, split value, left subtree, and right subtree
**Function FAW-Gini Split($X$, $y$, $F_{\text{Attributions}}$, features)**
  For each feature and split value:
    Split data into left and right subsets based on the split value
    Compute the FAW-Gini impurity for the split
  Select the feature and split value with the lowest FAW-Gini impurity
  **return** Best feature and split value

However, in practice, our implementation is a little more complex, since we have implemented an incremental computation for the FAW-Gini values to improve computational efficiency. This incremental computation enables us to only update to the impurity score in every step, avoiding the need to recompute it from scratch for every potential split.

The pseudo-code assumes a complete and dense dataset. To extend the method to sparse data with missing values, pre-processing would need to be performed to ensure compatibility with the method. Furthermore, the dimensions of matrix $F_{\text{Attributions}}$ need to be the same as the dimensions for the dataset $X$: each row corresponds to a sample, and each column represents the attribution score of a specific feature.

In practice, we also scaled the values of the FA using a Min MAX scaler between scaling the values between 0 and 1. Even if this might not be necessary in theory it makes the feature attribution values easier to handle and to interpret.

### 5.1.3 Feature Attribution Methods Applied

When evaluating the performance of the FAWDT approach, we test two different feature attribution methods: "Input × Gradient," and "Integrated Gradients" as introduced in

Section 4.2. These methods will be used to compute the previously mentioned feature attribution matrix $F_{\text{Attributions}}$.

- **Input × Gradient:** multiplies the gradient with the corresponding input feature, weighing the influence of each feature by its importance. Input × Gradient is particularly useful for models where inputs contribute to predictions in a linear or categorical way [KW24].
- **Integrated Gradients (IG):** This computing feature attributions by integrating the gradients of the model's output with respect to the input along a straight line from a baseline to the actual input. This integration over a path helps make it less sensitive to noise and local variations [STY17].

By testing both of these two feature attribution methods in the FAWDT framework, we investigate whether different methods for computing feature importance lead to variations in the resulting decision trees. The comparison will also help us find the better performing feature attribution for this task, to make FAWDT as competitive as possible compared to standard rule extraction.

## 5.2 (Un-)Certainty Weighted Decision Trees

Similarly to FAWDT, the Certainty Weighted Decision Tree (CWDT) framework suggests a novel method for rule extraction. By incorporating certainty levels, CWDT gains different decision boundaries to CART. These boundaries reflect the regions where the RL agent exhibits high confidence more precisely in its predictions. The method extends the principles of FAWDT by replacing feature attribution values with certainty scores, computed directly from the RL agent's model outputs.

### 5.2.1 Certainty Computation for RL Models

The certainty $c_k$ for a given sample $k$ is derived from the RL agent's predictions. Certainty quantifies the confidence of the model in its decision, represented as the maximum probability of the predicted action or class. The computation of $c_k$ depends on the type of RL algorithm used.

#### Certainty in DQN

For DQN models, we compute the certainty from the $q_{\text{values}}$ predicted by the policy network, which represent the expected future rewards for each action. The certainty from DQN is computed as follows:

1. Obtaining the $q_{\text{values}}$ for the given observation $\text{obs}_k$ from the agent's policy network:

$$q_{\text{values}} = \text{agent.policy.q\_net}(\text{obs}_k)$$

2. Computing the softmax probabilities over the Q-values:

$$p_i = \frac{\exp(q_i)}{\sum_j \exp(q_j)} \quad \forall i \in \text{actions}$$

3. Taking the maximum softmax probability gives us the certainty $c_k$:

$$c_k = \max(p_1, p_2, \ldots, p_n)$$

**Certainty in PPO**

For PPO models, we derive the certainty from the policy distribution produced by the agent. The policy distribution provides probabilities for each action, from which the certainty is extracted as follows:

1. Obtaining the policy distribution for the given observation $\text{obs}_k$ from the agent:

$$\text{dist} = \text{agent.policy.get\_distribution}(\text{obs}_k)$$

2. Extracting the probabilities of the distribution:

$$p_i = \text{dist.distribution.probs}[i] \quad \forall i \in \text{actions}$$

3. Taking the maximum probability, gives us the certainty $c_k$:

$$c_k = \max(p_1, p_2, \ldots, p_n)$$

### 5.2.2 Integration into Decision Tree-Splitting

Once the certainty values $c_k$ are computed for each sample, they are incorporated into the Gini impurity calculation, resulting in the Certainty-Weighted Gini (CW-Gini):

$$\text{CW-Gini}(S) = 1 - \sum_{i=1}^{k} \left(\frac{c_i}{C}\right)^2, \tag{5.5}$$

where $c_i$ is the certainty score for class $i$ which represents the model's confidence in predicting class $i$, and $C$ is the summed-up certainty across all samples.

Just as for FAWDT, for each potential split, the CW-Gini impurity is calculated for the resulting left ($S_L$) and right ($S_R$) subsets. Then the split with the lowest weighted impurity is selected. This ensures that the splits prioritize the subsets where the model is most confident.

By recursively applying this mechanism, the tree-building process generates decision boundaries that approximate the agent's behavior and reflect the confidence underlying its decisions.

### 5.2.3 Uncertainty Weighted Decision Trees

The Uncertainty Weighted Decision Tree (UWDT) functions exactly the same way as CWDT, but instead of using the certainty, they use the complement of the certainty scores derived from the model's predictions. The complement probability or uncertainty $u_k$ for a given sample $k$ is calculated as:

$$u_k = 1 - c_k, \tag{5.6}$$

where the complement of the model's certainty score $c_k$ is taken. This is then used as an Uncertainty Weighted Decision Tree approach. Using the complement of certainty, $u_k = 1 - c_k$, emphasizes regions where the model has lower confidence, allowing the decision tree to focus on the values around the decision boundaries. By prioritizing splits based on uncertainty, the UWDT framework can better address cases where the model's confidence is low, ensuring that the extracted rules capture and manage these areas of ambiguity effectively.

CHAPTER 6

# Experimentation and Results: Integrative Approaches

The experimentation process was conducted on control tasks in Gymnasium [TKT+24] environments, specifically CartPole, MountainCar, and LunarLander, using DQN and PPO algorithms implemented via the Stable-Baselines3 [RHG+21] library.

### 6.0.1 Evaluation Method

The decision trees extracted by the different rule extraction approaches will be evaluated by varying several parameters across different runs. The parameters analyzed are:

- **Gymnasium Control Problem:** We conduct experiments and evaluations on three control tasks: CartPole, MountainCar, and LunarLander. Each task presents distinct challenges to the RL models—CartPole focuses on balancing, MountainCar on momentum-based control, and LunarLander requires precision in navigating and landing. These tasks are discussed in detail in Section 3.1.1.
- **RL Model:** We implement two reinforcement learning algorithms, Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN), to analyze the performance of rule extraction approaches across types of RL models. The RL models are discussed in detail in Section 3.3.
- **Training Episodes:** The number of training episodes defines how often the trained RL model interacts with the environment, producing the dataset that the decision tree uses to approximate the RL policy. A higher number of episodes generally provides more training data, potentially enhancing the stability and accuracy of the surrogate decision tree model. While more training data is easily generated

within the Gymnasium environments, in real-world applications, data collection may be more limited or costly, posing a potential constraint.

- **Decision Tree Max Depth:** The decision tree's maximum depth parameter controls the complexity of the surrogate model. By limiting the depth, we aim to balance interpretability and fidelity. A shallow tree produces a simpler, more interpretable model but may lack the complexity to solve tasks in complex environments. In contrast, a deeper tree may closely mimic the RL model's policy, being able to capture subtle decision-making nuances. Nonetheless, this comes at the cost of reduced interpretability due to an increased number of rules and complexity. Exploring various depths provides insight into the trade-offs between simplicity and performance in approximating RL behavior.

- **Feature Attribution Method:** For the FAWDT Framework we test the performance of two different feature attribution techniques: Integrated Gradients and Input $\times$ Gradient. Differences in the calculation of these methods can influence the construction and effectiveness of the resulting decision trees. The feature attribution methods are discussed in detail in Section 4.2.

- **Weight Exponent:** The weight exponent determines how feature attribution values are adjusted after normalization. Unlike CWDT and UWDT, which inherently assign weights between 0 and 1 based on probabilities, FAWDT applies a MinMax scaler to normalize the feature attribution values to the range $[0, 1]$. The weight exponent modifies these normalized values to control the degree of variation in the weights, by taking the normalized weight to the power of the tested weight exponent:

  - A weight exponent of 0.5 results in the square root of the weights, reducing the impact of outliers and bringing the values closer together.
  - A weight exponent of 2 squares the weights, increases differences and emphasizing the disparity between feature attributions.

  These adjustments influence how strongly certain features are prioritized during the splitting process. If all weights are equal (i.e., no variation is applied), FAWDT behaves identically to a traditional CART decision tree. Evaluating different weight exponents will provide insights into how to optimize the performance of FAWDT.

We use three different metrics for evaluation:

- **Fidelity to the RL Model:** Fidelity "describes how well the rules mimic the behavior of a neural network" [Zho04].A High fidelity indicates that the decision tree reliably mimics the RL policy's decisions.This also shows how useful a surrogate model is for interpreting the learned behavior of the RL model. Mathematically,

fidelity is calculated as the proportion of matching actions between the RL model and the decision tree model:

$$\text{Fidelity} = 1 - \Pr_{x \in X}[f_R(x) \neq f_N(x)]$$

where $X$ is the input dataset, $x \in X$ is an instance from the dataset, $f_R(x)$ is the output of the extracted rule-based model, $f_N(x)$ is the output of the original model, and $\Pr_{x \in X}$ is the probability over the dataset instances [Zho04]. When calculating and expressing fidelity in the following sections, we always computed it on a test set separate from the training data. This metric captures the accuracy of the decision tree in mirroring the RL model, with higher fidelity indicating a closer approximation of the RL policy.

- **Decision Tree Model Score:** The decision tree's performance on the control problem is evaluated by measuring its score when interacting in the environment (e.g., CartPole). This serves as a metric for how well it generalizes the RL model's policy. This score is calculated as the total reward accumulated during a single run of the control problem, as defined within the Gymnasium environment. It directly reflects the surrogate model's ability to achieve the control task objectives. The decision tree model score also shows how well the decision tree would perform if the RL-Agent were to be discarded and the understandable surrogate decision tree model was used instead.
- **Percentage of Solved Cases:** Another metric to evaluate the decision tree in the Gymnasium environment is the percentage of cases solved. It measures the percentage of solved cases based on a predefined threshold defined by Gymnasium for each environment. A control problem is considered solved if the average score (or reward) across evaluation runs exceeds this threshold. This metric can be applied to both the RL model and the decision tree surrogate. For additional granularity, we also calculated the percentage of individual evaluation runs where the score surpasses the threshold. This metric is particularly useful for measuring performances across control tasks, since the scores and rewards differ.

These evaluation criteria enable a thorough analysis of the effectiveness of both the RL model and the surrogate model. By examining fidelity and performance across varied parameters, our approaches might perform better than the status quo.

## 6.1 Results of RL Agents on Gymnasium Environments

Each RL model was trained 50 times on different random seeds per reinforcement learning method and environment. With every agent having a maximum of 2,000,000 learning timesteps while employing regular evaluation callbacks to check if the agent already sufficiently solves the environment. The training was terminated early if the agent

consistently solved the task, defined as achieving a score above the predefined Gymnasium threshold for that environment on all 10 consecutive evaluation runs employed during the callback. This criterion will be referred to as "Reached Learning Threshold" and marks the point where the agent demonstrates reliable success during evaluation. To verify the robustness of these results, an additional "Evaluation Over Threshold" metric was introduced, which tests the agent's performance after training on significantly more runs to ensure that it behaves robustly.

To provide a clearer comparison between DQN and PPO performances in each control environment the following metrics were computed:

- The percentage of agents that achieved a mean score across all evaluation runs above the threshold in the evaluation set. In other words, if the total mean of the evaluation set is over the threshold.
- The mean percentage of episodes solved in the evaluation set.
- The percentage of agents who "Completed" training (Cmptd), that is, the agent successfully solved the Gymnasium environment on ten consecutive callback runs during training.
- The "Mean Over Threshold" but only for the agents which "Completed" training and therefore solved ten consecutive runs during training callbacks.
- The "Episodes Solved" but only for the agents which "Completed" training and therefore solved ten consecutive runs during training callbacks.
- The average timesteps needed for the agents to "Complete" training and therefore solve ten consecutive runs during training callbacks.

The trained agents were tested on 100 evaluation runs after training, shown in Table 6.1.

Table 6.1: Performance Summary of DQN and PPO in Gymnasium Environments

| | **CartPole** | | **LunarLander** | | **MountainCar** | |
|---|---|---|---|---|---|---|
| | DQN | PPO | DQN | PPO | DQN | PPO |
| **Mean Over Threshold** | 94% | 98% | 44% | 30% | 8% | 26% |
| **Episodes Solved** | 96% | 98% | 47% | 67% | 37% | 26% |
| **Completed training** | **100%** | **100%** | **52%** | **80%** | **16%** | **28%** |
| **Mean Over Threshold (Cmptd)** | 94% | 98% | 85% | 38% | 50% | 86% |
| **Episodes Solved (Cmptd)** | 96% | 98% | 82% | 74% | 86% | 75% |
| **Timesteps Needed (Cmptd)** | 780K | 23K | 1,568K | 847K | 1,451K | 1,624K |

The table highlights key differences in algorithm performance across tasks. In the CartPole environment, both DQN and PPO achieved a 100% completion rate in solving the task within 10 consecutive evaluation runs over the learning period. PPO reached

this milestone significantly faster, requiring only 23,000 timesteps on average compared to DQN's 780,000. Nevertheless, for PPO, there has been one random seed (2 Percent) where the training was completed, but on the following 100 evaluation runs, their mean did not reach the required threshold.

LunarLander shows very diverse results. While PPO is able to solve more episodes with 67% over all evaluation, DQN sees to achieve more stable models with 44% of all cases having a mean over the threshold. PPO reached the learning threshold of 10 solved runs in 80% of runs compared to 52% for DQN. The low "Mean Over Threshold" scores show that while PPO solves more runs overall. However, when it fails, its score seems to be significantly lower, pushing the mean down. The percentage of episodes solved in the agents which completed training is very comparable between PPOs 74% and DQNs 82%.

MountainCar proves to be a significantly more challenging environment for both algorithms, with only 16% of DQN runs and 28% of PPO runs achieving 10 consecutive successful evaluations. However, among completed runs, PPO demonstrates greater robustness compared to its performance in LunarLander, with 86% of completed PPO runs maintaining a mean score above the threshold. DQN, in contrast, achieves 50% in this category. Learning in this environment also takes considerably longer, particularly for PPO, which requires 1,624,000 timesteps on average compared to DQN's 1,451,000. Despite this, PPO appears to offer more stable solutions once the environment is successfully learned.

These results demonstrate PPO's higher performance, especially in terms of learning efficiency. Nonetheless, its performance exhibits greater variance in the case of LunarLander, with some greater failures leading to significantly lower scores. Meanwhile, DQN performed reasonably well in the CartPole and LunarLander environments. MountainCar remains the most difficult environment for both algorithms, with low completion rates and long training times.

## 6.2 Agent Selection for Rule Extraction

The next step is to determine the set of agents on which to perform rule extraction. For each environment and the reinforcement learning method, all parameters stayed the same, except for the random seed used for both training the agent and initializing the environment. Therefore, depending on their random seed, not all trained agents are suitable for extracting decision trees, as some agents failed to learn a policy that performs well in the environments. These agents do not provide a reliable basis for further analysis. Therefore, agents were filtered based on their performance using three different criteria:

1. **All Agents**: Using all trained agents, regardless of their performance. This maximizes variance in training samples but may introduce misleading patterns from unsuccessful agents.

2. **Completed Agents**: Filtering agents that successfully solved their environment by achieving 10 consecutive successful evaluations. This ensures that only functional policies are learned, but significantly reduces the number of available training samples, especially in more difficult environments.

3. **Over Threshold Agents**: Selecting agents that, during evaluation, achieved a mean score above the environment-specific threshold. This balances ensuring policy quality while retaining more training samples than the completed-only approach.

The evaluation of these filtering methods was conducted by measuring the scores of resulting CART decision trees with a maximum depth of 5 and 100 training episodes. Table 6.2 summarizes the number of agents remaining for each game and filter, as well as the average score and the percentage of episodes solved. The number of agents remaining under each filtering criterion was also examined, as having a larger sample size is beneficial for generalization.

Table 6.2: Results of CART for Environments and Filters

| filter | game | # PPO | # DQN | mean score | percent solved |
|---|---|---|---|---|---|
| All Agents | CartPole | 50 | 50 | 421 | 72% |
| | MountainCar | 50 | 50 | -149 | 29% |
| | LunarLander | 50 | 50 | -341 | 1% |
| Completed | CartPole | 50 | 50 | 421 | 72% |
| | MountainCar | 14 | 8 | -110 | 74% |
| | LunarLander | 40 | 26 | -341 | 1% |
| Over Threshhold | CartPole | 49 | 47 | 427 | 74% |
| | MountainCar | 13 | 4 | -107 | 81% |
| | LunarLander | 15 | 22 | -406 | 0% |

The results indicate that including all agents provides the largest dataset but results in poor overall performance, particularly in more complex environments such as LunarLander and MountainCar, where the mean scores and percentages of episodes solved are notably low. Filtering based on completed agents significantly improved these metrics; for example, in MountainCar, retaining only agents that successfully solved the task raised the mean score from -149 to -110 and increased the solution rate from 29% to 74%. The over-threshold filter further refines the selection by enforcing a stricter performance criterion, which slightly boosts scores in CartPole and MountainCar but at the cost of a reduced number of agents.

Given these trade-offs, the Completed Agents filtering method was selected for further analysis. This approach produces significantly higher mean scores for CartPole and MountainCar compared to using all agents, and it retains a larger pool of agents than the over-threshold filter, thereby increasing the reliability of the extracted policies.

## 6.3 Impact of certainty-based weighting on Decision Boundaries

This section examines how certainty-based weighting affects decision tree behavior and decision boundaries. Weighted decision trees adjust the influence of individual samples based on model certainty, leading to different partitioning of the feature space. CWDTs prioritize instances where the model is more confident in its predictions, reinforcing patterns where certainty is high. In contrast, UWDTs emphasize samples near the decision boundary in order to refine the model in regions where classification is more ambiguous.

Figure 6.1 visualizes the decision boundaries of a standard CART decision tree and a CWDT model.

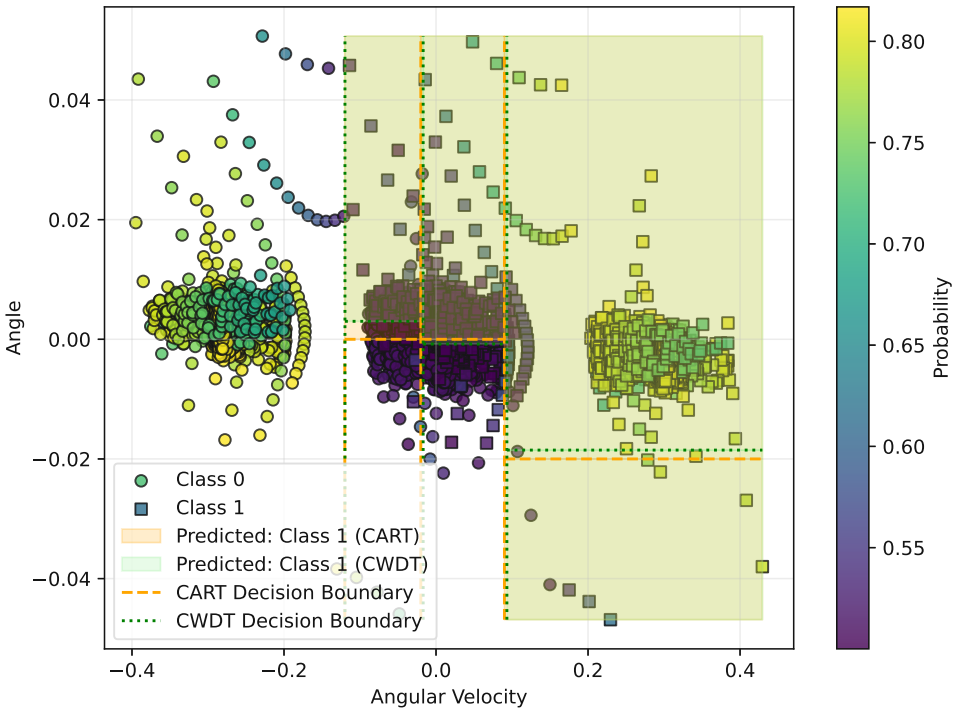



Figure 6.1: Decision boundaries of CART and CWDT models

The color gradient represents the certainty values, with lower certainty observed near the decision boundary. This behavior aligns with expectations, as the model is naturally less confident when making decisions in ambiguous regions. The decision boundary itself shifts when using CWDT, and additional splits appear, indicating that certainty-based

weighting influences how the tree partitions the feature space. However, judging the overall effectiveness of this change purely from visualization is not possible.

To highlight the differences between these two weighting schemes, Figure 6.2 presents a zoomed-in version of the decision boundary comparison. This visualization provides better insight into how CWDT modifies the structure of the tree, with noticeable shifts in the decision boundaries compared to the standard CART model. The changes introduced by UWDT would likely contrast with CWDT by emphasizing regions where uncertainty is highest, potentially leading to different splitting strategies.

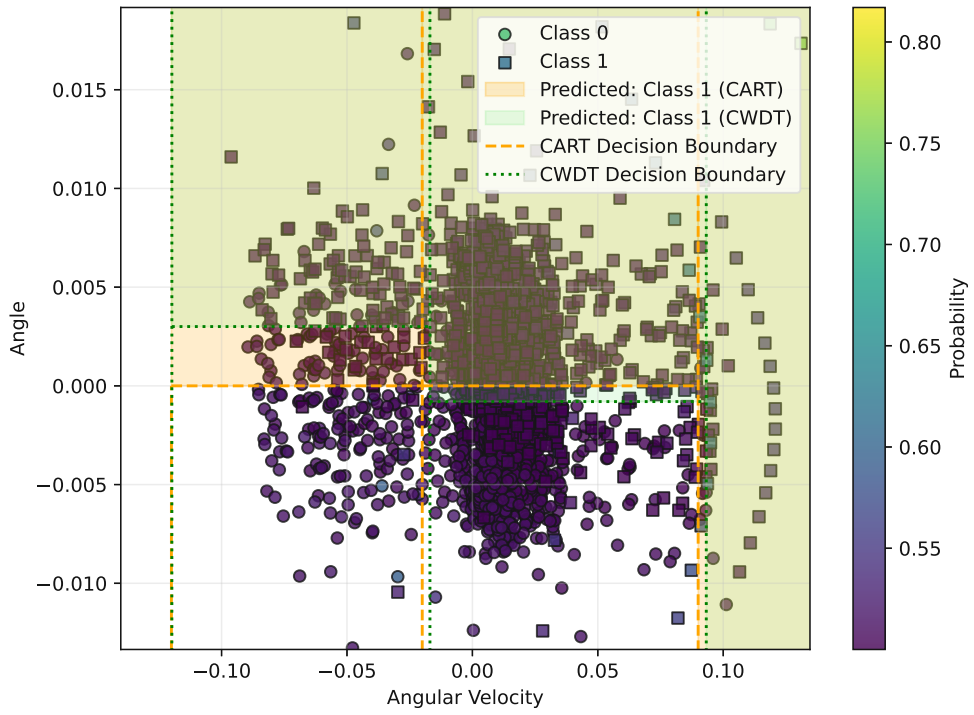



Figure 6.2: Zoomed-in view of decision boundaries

Finally, the effectiveness of CWDT and UWDT in improving decision tree performance will be evaluated in the next sections. The impact of prioritizing certainty versus uncertainty will be tested using performance metrics to determine which approach yields better results in different environments.

## 6.4 FAWDT Attribution Methods

As discussed in Section 4.2, the FAWDT algorithm was tested using two different feature attribution methods: Integrated Gradients and Input x Gradient. The aim is to evaluate the different feature attributions effectiveness for rule extraction.

To assess the impact of feature attribution, we compared performance for the CartPole problem using both DQN and PPO agents under varying decision tree depths (2, 3, 4, and 5). One hypothesis is that, for shallower trees, the additional information provided by feature attributions could help compensate for their limited model capacity by highlighting the most critical features.

Figure 6.3 shows the percentage of solved cases across different decision tree depths (2, 3, 4, and 5) for DQN-based FAWDT using Integrated Gradients and Input x Gradient. The results indicate that Integrated Gradients outperform Input x Gradient at depths 2,3, and 5. Interestingly, for depth 2, it even performs outperforms CART. However, all approaches remain within the 95% confidence interval, indicating that these differences are not statistically significant. As expected, overall scores improve as the decision tree depth increases, suggesting that deeper trees provide greater capacity to approximate the policy and enhance prediction accuracy.

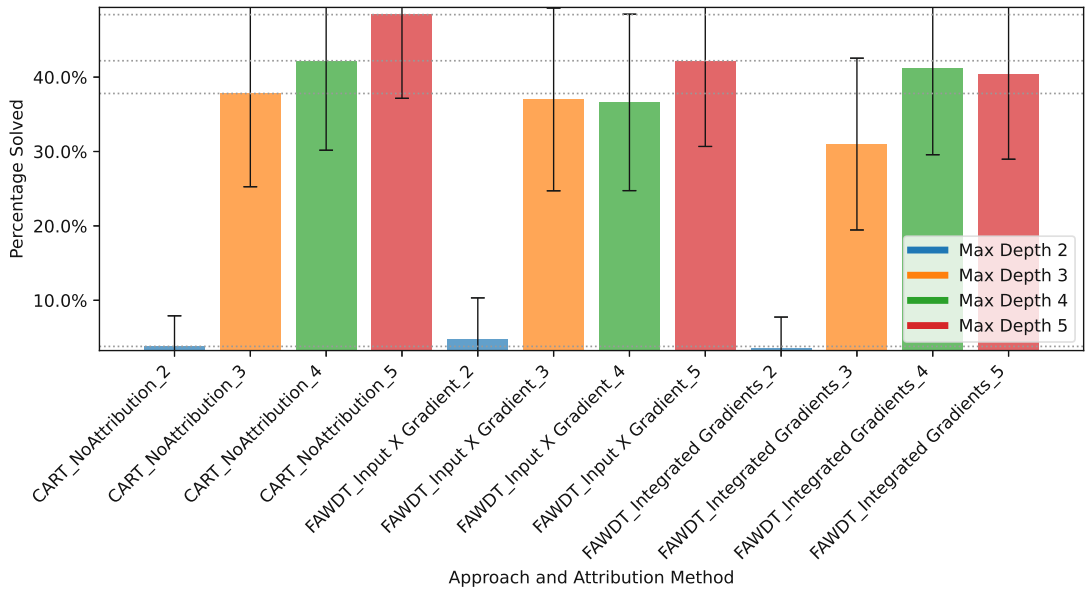



Figure 6.3: Percentage of solved cases for DQN-based FAWDT across different decision tree depths and feature attribution methods

Figure 6.4 illustrates the percentage of solved cases for the PPO-Agent. In contrast to the DQN results, all approaches for depths 3, 4, and 5 achieve significantly higher performance, with solved rates exceeding 80%. *Input x Gradient* achieves the best performance at depth 5, solving nearly 100% of cases, outperforming *Integrated Gradients*.

Interestingly, for depth 2, none of the three approaches, including CART, are able to solve the CartPole task, indicating that this depth is insufficient to represent the policy learned by the PPO agent.

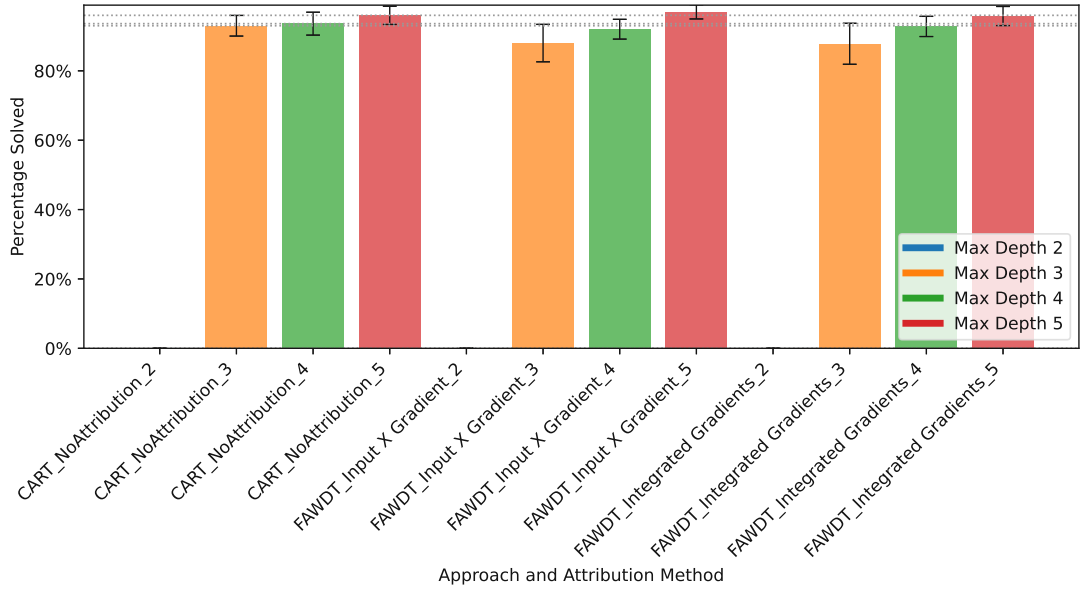



Figure 6.4: Percentage of solved cases for PPO-based FAWDT across different decision tree depths and feature attribution methods

Given that the results from both feature attribution methods are consistently aligned across most experiments and consistently underperform compared to the baseline CART algorithm, it is worth noting that the DQN agent requires more complex decision trees to achieve high performance than the PPO agent. This observation highlights the PPO agent's ability to generate policies that are more easily approximated by decision trees at shallower depths. For the remaining evaluations of the FAWDT algorithm, Input x Gradient will be used as the primary attribution method due to its computational efficiency and consistent performance.

## 6.5 Overfitting Depth vs Episodes

In this section, we analyze the CART algorithm performance on the CartPole environment by comparing decision tree maximum depths against the number of training episodes. The objective is to investigate how different constraints—either low maximum depth or limited training episodes—impact the performance of the decision tree. This analysis focuses on DQN because, as observed earlier, the rule extraction algorithms require deeper trees to solve the DQN Agent's CartPole environment effectively. Consequently, it is more meaningful to analyze the relationship between depth and training episodes within this context.

Figure 6.5 illustrates the average performance scores for different training episode counts across various maximum depths. It shows the optimal number of training episodes for each maximum depth. At low maximum depths (e.g., 3 or 4), fewer training episodes (such as 50) yield the best performance, likely due to the model's simplicity, which limits its ability to leverage extensive training data. Conversely, as the maximum depth increases, the model can utilize more training episodes effectively, leading to improved scores.

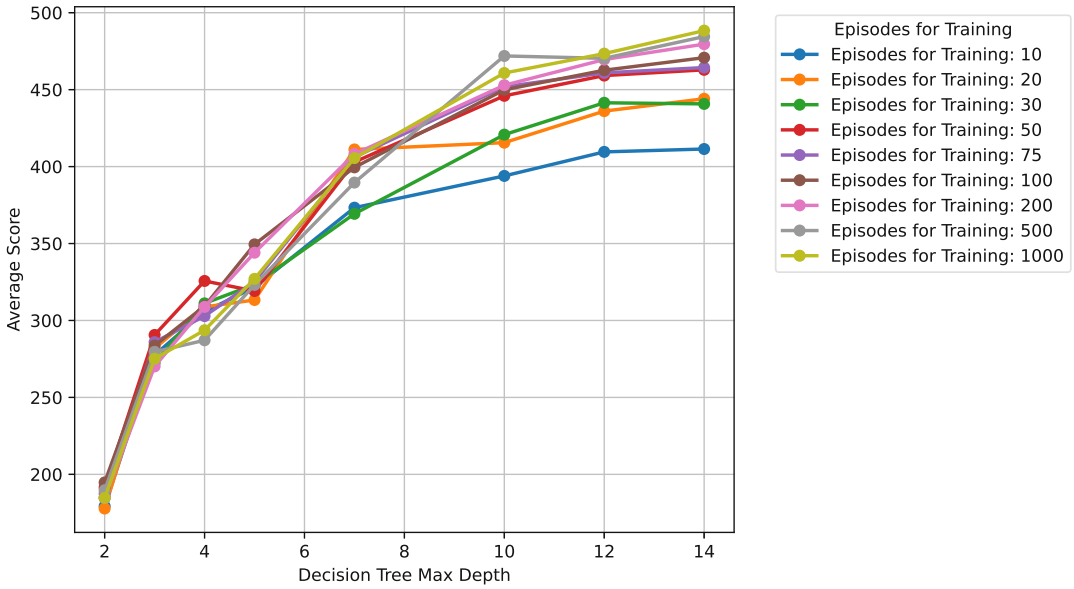



Figure 6.5: Average scores for different numbers of training episodes at varying decision tree maximum depths in the CartPole environment

For all training episode counts, the average score greatly increases with greater maximum depth up to a certain point. After this point, the scores begin to plateau and only increases slightly. This saturation point occurs earlier for lower training episode counts, while for higher counts—such as 1,000 episodes—the scores do not plateau even at a maximum depth of 14. This suggests that deeper decision trees can handle more complex information, whereas shallow trees cannot capitalize on additional training episodes.

Figure 6.6 provides a complementary perspective by showing the average performance scores for different maximum depths at varying numbers of training episodes. For low maximum depths (3, 4, and 5), increasing the number of training episodes leads to a decrease in performance after a certain peak, indicating clear signs of overfitting. For depth 2, the outcomes appear rather random, as the decision tree of such a limited depth is unable to approximate the underlying policy effectively, leading to inconsistent performance. This shows that simple models not only fail to utilize the additional information but may also deteriorate in performance with excess data.

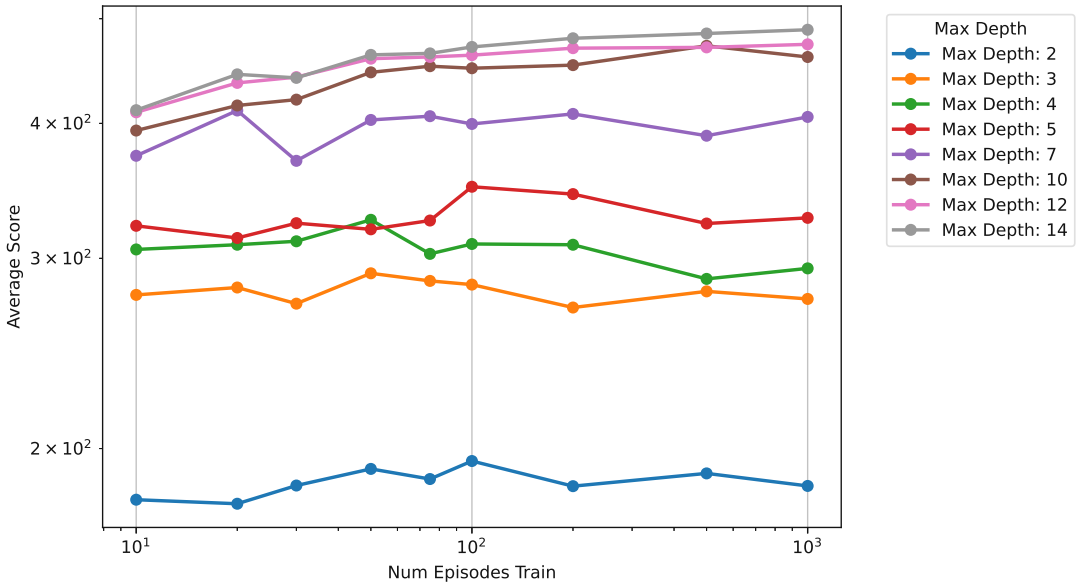



Figure 6.6: Average scores for different decision tree depths at varying numbers of training episodes in the CartPole environment

In contrast, deeper decision trees (e.g., 10,12 or 14) improve in performance with the additional information gained. This behavior confirms that higher-depth decision trees can better handle large amounts of training data without overfitting, while lower-depth trees are prone to diminishing returns and even performance declines as training episodes increase.

In summary, the results indicate that the optimal configuration depends on the chosen constraints. For low maximum depths, fewer training episodes yield better results due to the model's limited capacity and vulnerability to overfitting. However, if complexity is not a concern, increasing the number of training episodes for deeper trees can significantly enhance performance without risking overfitting.

## 6.6 Weight Exponents

In this section, we examine the impact of different weight exponents on the performance of FAWDT to find which weights achieve the best performance. Unlike CWDT and UWDT, which automatically assign weights between 0 and 1 based on probabilities, FAWDT applies a MinMax scaler to normalize its weights to the same range. This normalization ensures that the weights lie between 0 and 1, but the values still differ based on the chosen feature attribution scores.

If all weights were equal, FAWDT would function identically to a traditional CART decision tree, highlighting the importance of determining how much weights should vary

to achieve optimal performance. The degree of variation can be controlled by applying different exponents to the weights. For example, an exponent of 0.5 takes the square root of the weights, bringing their values closer together and reducing the impact of outliers. In contrast, an exponent of 2 squares the weights, exaggerating differences, and pushing the values further apart.

Exploring exponents between 0.1 and 10 allows us to balance these effects and test their impact on model performance. A lower exponent smooths the weights, potentially making the model more robust but less sensitive to specific feature attributions. On the other hand, a higher exponent sharpens the distinctions between features, giving more weight to certain attributes but potentially making the model more prone to overfitting.

For this analysis, we used a maximum depth of 5 with 100 episodes for learning. This configuration represents a scenario with minimal constraints, avoiding overfitting while achieving a comparably high performance, as seen in Section 6.5.

Figure 6.7 shows the average performance scores for different weight exponents for each approach learned on a PPO Agent. For FAWDT and UWDT, using a weight exponent of 1 (i.e., no transformation) results in the best performance. Both lower exponents (closer to 0.1) and higher exponents (closer to 10) lead to a decrease in performance, indicating that these approaches perform best when the weights remain proportional to their original probability values. Notably though a very high exponent for the FAWDT approach results in significantly lower scores, potentially overweighing the feature attribution values to a degree where it is counterproductive for the rule extraction. CWDT also have lower scores with increasing exponent, even if only by a small margin.

The very different results for DQN agents in Figure 6.7 show much lower scores. Here both CWDT and UWDTs performances improve with increasing weight exponents. FAWDT shows a similar pattern to their PPO counterparts, with high exponent performing poorly. It is noticeable that not the performance, but also the variance is much higher for DQN-based Agents. Therefore, it is harder to extract significant results and relevant learnings from the DQN-based Agents.

For further testing, we do not want to take an exponent close to 0 since then we would have the same results and method as CART. Since the results are very inconclusive and, we will continue with a weight exponent of 1 for further testing due to its simplicity and good performance on PPO-based agents.

## 6.7 Comparing Approaches

In this section, we compare the performance of our different decision tree approaches: CART, CWDT, UWDT, and FAWDT. First they will be tested in the CartPole environment and then in the MountainCar and LunarLander environments. For these tests we will use the maximum depths of 2,3 and 5 and we will use 100 episodes for training.

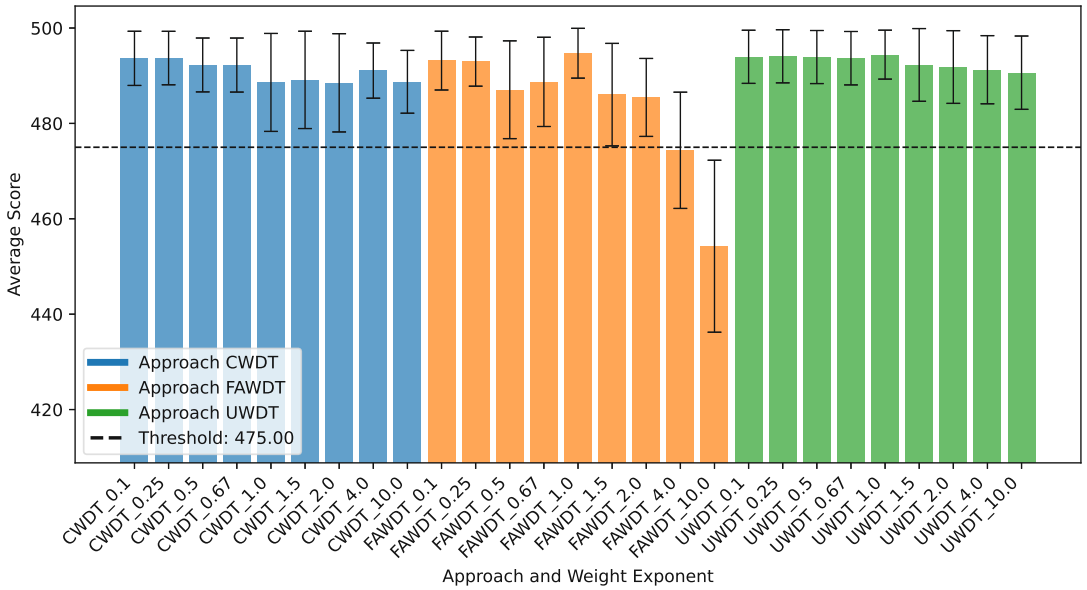



Figure 6.7: Effect of weight exponent on FAWDT, CWDT, and UWDT performance for a PPO agent

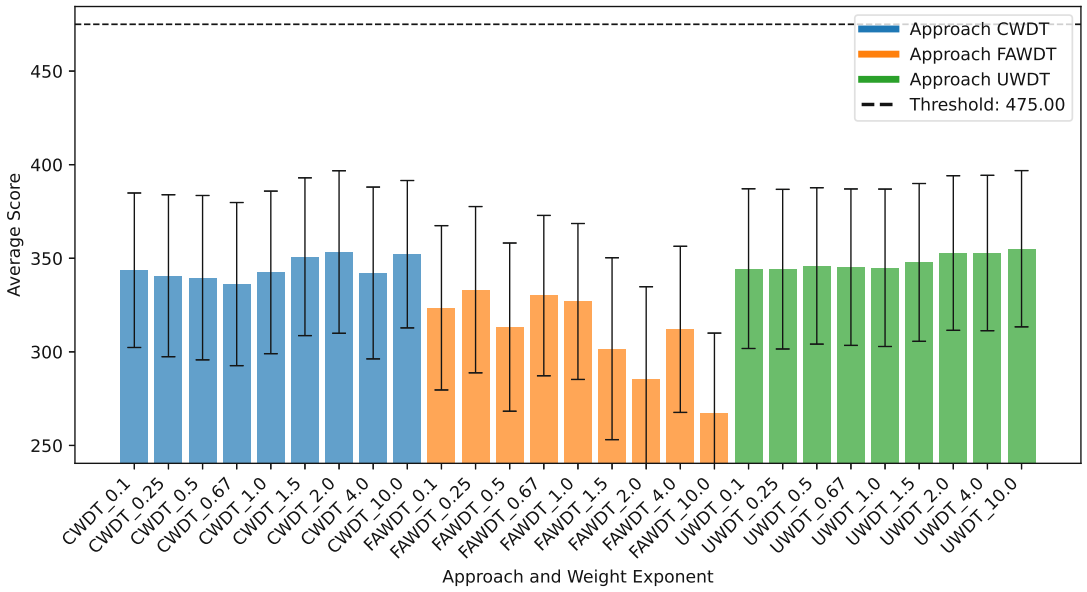



Figure 6.8: Effect of weight exponent on FAWDT, CWDT, and UWDT performance for a DQN agent

### 6.7.1 CartPole

In Figure 6.9, the fidelity for each approach is shown for rules learned on PPO and DQN Agents across the different depths with 95% confidence intervals. Fidelity measures how closely the decision tree replicates the actions of the reinforcement learning agent. The results indicate that fidelity improves consistently as the maximum depth increases, reflecting the fact that deeper trees are better at approximating the policy of the RL agent.

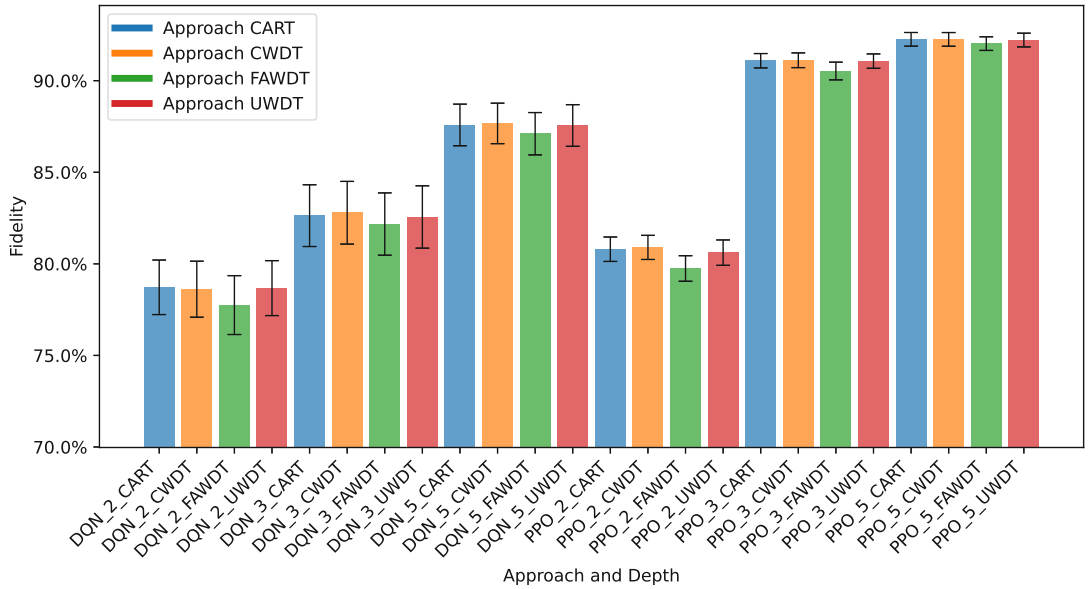



Figure 6.9: Fidelity by approach and maximum depth in the CartPole environment with 95% confidence intervals

The CART, CWDT, and UWDT approaches have nearly identical fidelity values, ranging from 74% at depth 2 for DQN to 94% at depth 5 for PPO. This similarity suggests that both CWDT and UWDT closely mirror CART's behavior despite their additional weighting mechanisms. FAWDT, on the other hand, consistently shows slightly lower fidelity across all depths, though it remains within the confidence intervals.

This is expected because while CART aims solely to mimic the RL agent's actions, the integrative approaches (CWDT, UWDT, and FAWDT) introduce feature-based or uncertainty-based weights to optimize performance for the control problem, potentially sacrificing some fidelity. It is particularly noteworthy that CWDT and UWDT perform so similarly to CART despite their additional complexity, which implies that they successfully using a different splitting criterion, without compromising fidelity.

Furthermore, the fidelity of decision trees derived from DQN agents is consistently lower than that of PPO agents. This pattern suggests that the policies learned by DQN agents are less amenable to approximation by decision trees. For PPO agents, a maximum

depth of 3 suffices to achieve a fidelity exceeding 90%, indicating that the PPO policy structure can be effectively captured by relatively shallow trees. In contrast, DQN agents show a marked improvement in fidelity when the decision tree depth is increased from 3 to 5, suggesting that the additional complexity provided by deeper trees is necessary to approximate their decision-making process accurately. These findings imply that, while PPO policies can be efficiently represented with moderate model capacity, DQN policies require more complex tree structures to bridge the gap in fidelity.

For PPO-trained agents, a depth of 3 is sufficient to achieve over 90% fidelity, with only marginal improvement when increasing to depth 5. This indicates that a moderate tree depth is adequate for capturing PPO policies with high accuracy.

Figure 6.10 presents the average performance scores achieved by each approach across the same depths, with 95% confidence intervals shown. The threshold score of 475 required to solve the game is included as a reference. The average score represents the overall success of the decision tree in the CartPole task. Similar to fidelity, the scores increase with greater depth across all approaches. However, in contrast to the fidelity results, there are more noticeable differences between the approaches in terms of average score.

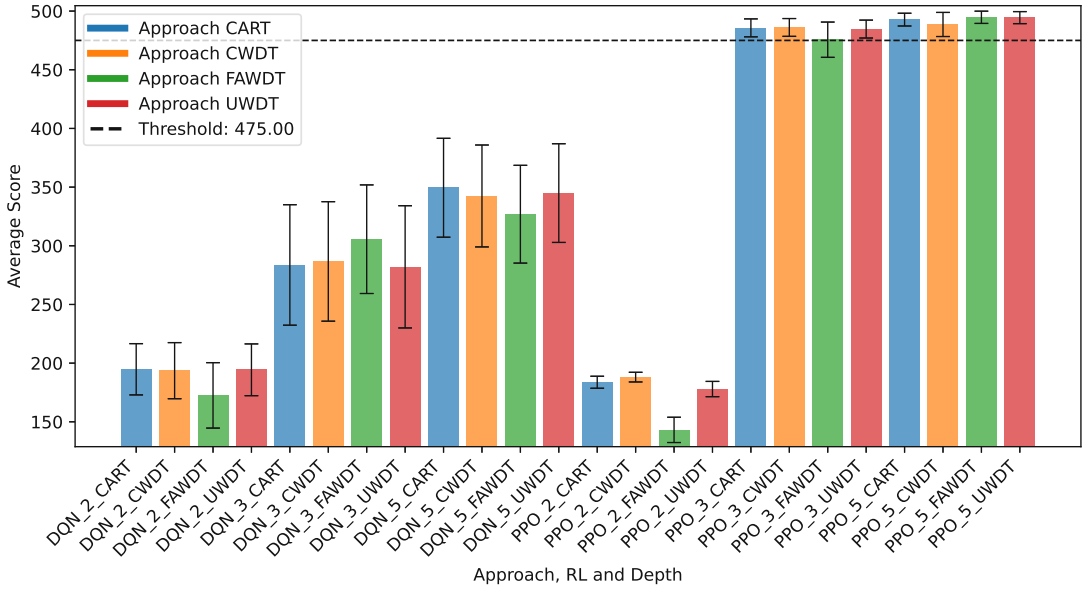



Figure 6.10: Average scores by approach and maximum depth in the CartPole environment with 95% confidence intervals

PPO-based rules achieve higher scores at depths 3 and 5, while rules extracted from DQN policies at depth 2 perform slightly better. This could be due to an instance where the extracted rules of depth 2 from DQN successfully solved the CartPole problem.

CART, CWDT, and UWDT generally exhibit similar performance, reinforcing the observation that weighting mechanisms in CWDT and UWDT do not significantly impact

performance in this environment. However, FAWDT achieves lower scores in most depths, indicating that feature attribution weighting might not be effective for this task.

Overall, no significant consistent performance difference is observed between the rule extraction approaches themselves. Instead, the results appear to be more dependent on the underlying RL agent being mimicked, with PPO-based policies leading to better overall performance in this environment.

### 6.7.2 Performance on all games

To extend the analysis beyond CartPole, the performance of decision tree models is evaluated across all three environments: CartPole, MountainCar, and LunarLander. The results are examined separately for models trained on DQN and PPO agents, comparing the four rule extraction approaches all with a maximum decision tree depth of five.

Figure 6.11 presents the percentage of solved runs for each environment and agent type. The results highlight clear discrepancies in the solvability of different environments. Notably, DQN-trained agents fail to solve LunarLander across all tested methods, while PPO-based agents very rarely succeed. This suggests that decision tree models struggle to approximate the complex control policy required for a successful landing. Additionally, no statistically significant differences are observed between methods within the 95% confidence intervals.

In the MountainCar environment, both DQN and PPO-based models achieve a higher percentage of solved runs compared to CartPole models trained on DQN but remain below the performance of CartPole models trained on PPO. Interestingly, MountainCar exhibits a slight advantage for DQN-based models over PPO-based models, though the differences are not statistically significant.

Figure 6.12 presents the fidelity of decision trees across environments. Unlike the solved percentage metric, the differences between approaches are much smaller, and the results appear more robust, with narrower confidence intervals. This suggests that while the extracted rules may not always successfully solve the environment, they still approximate the RL agent's behavior reasonably well.

A particularly noteworthy finding is that, despite the near-zero success rate of LunarLander decision trees, their fidelity remains around 70%, indicating that they replicate the RL agent's decisions with moderate reliability even though those decisions do not consistently lead to solving the task. Fidelity is lowest in LunarLander (approximately 70%) but significantly higher in CartPole (ranging from 86% to 93%) and MountainCar (consistently above 95%), where decision trees appear to mimic their respective RL policies more effectively. Interestingly, MountainCar models exhibit higher fidelity than CartPole PPO models, even though they solve the environment less frequently. This discrepancy suggests that fidelity alone does not directly translate to task success.

These findings reinforce the observation that rule extraction performance depends more on the underlying RL agent than on the specific decision tree weighting approach. While

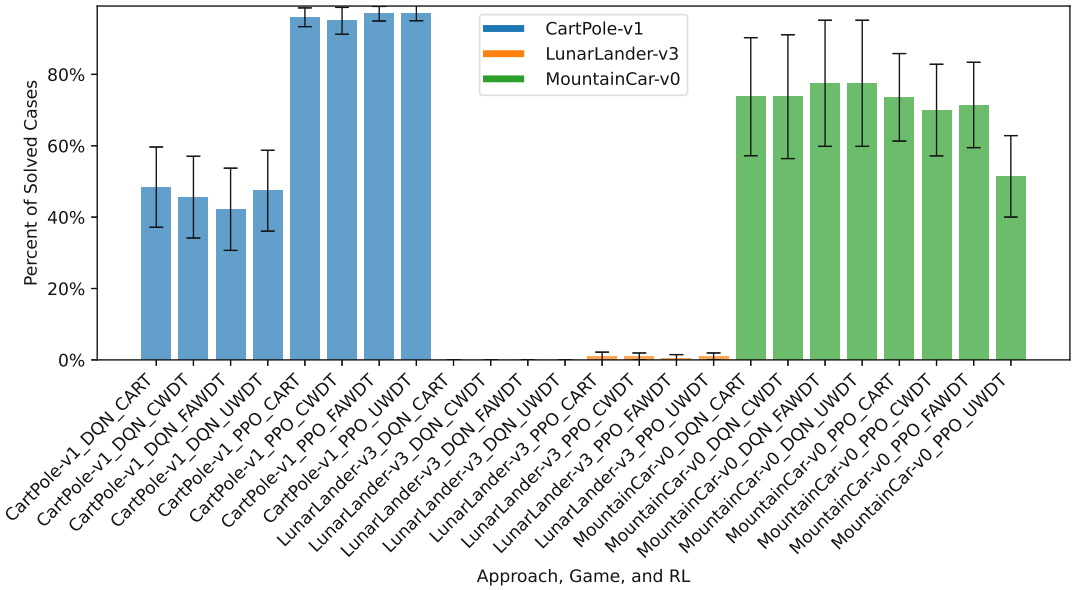



Figure 6.11: Percentage of solved runs across all environments, comparing different decision tree approaches for DQN and PPO-based policies

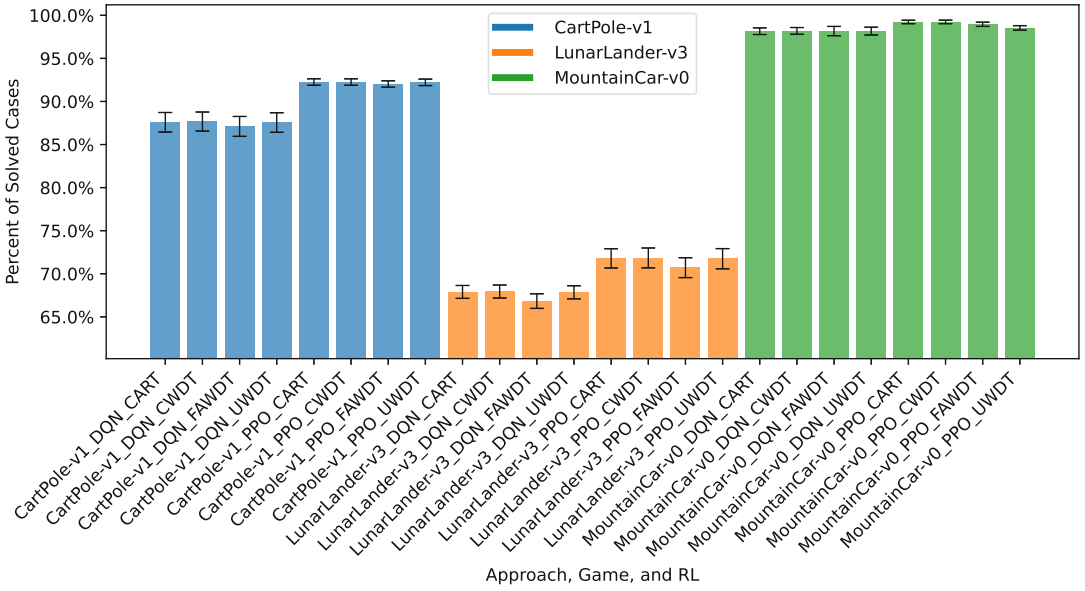



Figure 6.12: Fidelity of Decision Trees Across Environments and RL Policies

some environments allow for higher fidelity models, success in solving the environment remains highly dependent on the complexity of the task and the suitability of decision trees for representing its control strategy.

## 6.8 Statistical Testing Methodology

The following statistical testing method was employed to evaluate the significance of differences between paired observations. The statistical procedure followed the following sequence: First we assessed the normality of the data, then we evaluated if it is symetric, and last we applied an appropriate significance test selection based on the data properties. All statistical tests were done using the SciPy library [VGO+20].

Normality was examined using both the Shapiro-Wilk test [SW65] and the D'Agostino-Pearson test [DBDJ90]. In cases where normality was not established, the symmetry of the paired differences was examined. The skewness of the differences was computed, and an absolute skewness value below 0.5 was used as a criterion to classify the differences as symmetric [HZGM22].

Based on the results of the normality and symmetry assessments, an appropriate statistical test was selected. If normality was confirmed, a paired t-test [Stu08] was applied. If normality was not established but symmetry was present, the Wilcoxon Signed-Rank test [Wil92] was employed. If neither normality nor symmetry was confirmed, a non-parametric permutation test [Goo13] was performed.

A significance threshold of 0.05 was applied across all tests. If the resulting p-value was below this threshold, the null hypothesis of no difference was rejected. The direction of significance was determined by comparing the mean values of paired observations. If the mean of one condition exceeded the other, it was classified as superior.

All statistical tests were conducted under the assumption that the paired observations were exchangeable under the null hypothesis, ensuring that the analysis remained valid under different distributional properties.

## 6.9 Statistical Evaluation

To determine whether any of the weighted decision tree approaches provide statistically significant improvements over the standard CART model, a statistical analysis was conducted across all environments. The comparison was performed over various settings, including decision tree depths of 2, 3, and 5, and training episode counts of 10, 50, and 100. The goal of this analysis was to test whether any approach showed significant deviations from CART in terms of average score, fidelity, or the percentage of solved episodes.

Table 6.3 presents the test results testing if the distribution of the approaches is statistically significantly different from CART. The values are the mean across all variations. The p-values are shown in parentheses next to the mean. A lower p-value suggests stronger evidence that the approach deviates from CART, with a significance threshold typically set at 0.05. Since neither normality nor symmetry was present, we applied the parametric permutation test.

| | CART | CWDT | FAWDT | UWDT |
|---|---|---|---|---|
| average score | 26.6 | 23.7 (9.5%) | **4.4 (<0.1%)** | 27.1 (79.0%) |
| fidelity | 80.2% | 80.3% (36.2%) | **79.3% (<0.1%)** | **80.1% (<0.1%)** |
| percent solved | 29.2% | 29.4% (46.4%) | **27.8% (0.5%)** | **28.2% (0.2%)** |

Table 6.3: Mean Performance and p-Values Across Methods

The results indicate that, in all cases where a significant difference is observed ($p < 0.05$), the alternative approach performs worse than CART. Specifically, FAWDT and UWDT exhibit a statistically significant lower fidelity, suggesting that these weighting strategies mimic the RL model significantly worse than CART over all parameters. FAWDT also shows a significantly lower average score, reinforcing the notion that its weighting scheme may not be optimal for rule extraction. CWDT, on the other hand, does not show statistically significant differences from CART, implying that its weighting mechanism does not negatively impact performance but also does not provide a measurable improvement.

No approach shows a statistically significant improvement in any of the metrics, indicating that weighting strategies do not consistently enhance the ability of decision trees to reproduce the RL agent's success rate. The relatively high p-values for CWDT across all metrics suggest that it remains closest to CART in overall performance, while FAWDT and UWDT introduce modifications that reduce fidelity and overall effectiveness.

These findings suggest that, while weighting schemes influence decision tree behavior, they do not provide a universal improvement over the standard CART model.

However, one hypothesis is that for shallower trees, the additional information provided by feature attributions or certainty levels could help compensate for their limited model capacity by highlighting the most critical features. By testing various maximum depths for each approach and leaving all other parameters the same, we received the results in Table 6.4.

| | depth | CART | CWDT | FAWDT | UWDT |
|---|---|---|---|---|---|
| average score | 2 | -70.9 | -75.9 (14.2%) | **-99.6 (<0.1%)** | -68.8 (56.7%) |
| average score | 3 | 65 | 62.1 (27.6%) | **36.2 (<0.1%)** | 62 (21.3%) |
| average score | 5 | 85.7 | 84.8 (78.3%) | 76.7 (9.4%) | 88.2 (41.5%) |
| fidelity | 2 | 75.7% | 75.7% (47.0%) | **74.8%(<0.1%)** | **75.5% (<0.1%)** |
| fidelity | 3 | 80.9% | 81.0% (21.7%) | **80.0% (<0.1%)** | **80.8% (<0.1%)** |
| fidelity | 5 | 84.1% | 84.1% (26.0%) | **83.2% (<0.1%)** | **84.0% (<0.1%)** |
| percent solved | 2 | 1.0% | 1.2% (30.5%) | **2.4% (0.2%)** | **1.9% (0.1%)** |
| percent solved | 3 | 40.7% | 40.8% (74.8%) | **36.5% (<0.1%)** | **38.0% (<0.1%)** |
| percent solved | 5 | 46.0% | 46.1% (90.0%) | 44.5% (10.5%) | 44.8% (7.2%) |

Table 6.4: Performance and p-values across approaches and tree depths

Again, over all metrics and depth CWDT does not perform significantly different to CART. Similarly to Table 6.3, the average score of FAWDT is significantly lower than

CART for depth 2 and 3, while UWDT does not significantly differ from CART. Again, for both FAWDT and UWDT the fidelity is significantly lower for all decision tree depths. While both FAWDT and UWDT showed significantly lower percent solved over all parameters, when splitting it up by depth the trees behave very differently. For a depth of 3, both approaches still perform significantly worse than CART, for a depth of 5 there is no significance, but for a depth of 2 FAWDT and UWDT perform significantly better than CART.

This shows that for very shallow trees, given the environments and parameters, both novel decision tree-splitting criteria have a significantly higher percentage of solved cases compared to CART. However, while these extracted trees solve the environment more often than CART, their average score is still lower than the status quo CART algorithm.

CHAPTER 7

# Experimentation and Results: Feature Selection

This chapter presents the experimentation process and results related to feature selection for HVAC systems. Despite the lack of significant improvement in rule extraction, as observed in the previous chapter, both standard rule extraction and feature attribution remain viable as feature importance metrics. We employ Input × Gradient as the feature attribution method to assess the relevance of different input variables.

## 7.1 Feature Importance in Gymnasium

Feature importance is analyzed using decision trees (DT) and feature attribution (FA) with the Input × Gradient method. Table 7.1 presents a comparison of feature importance rankings for the CartPole environment, with the least important features highlighted. The results show considerable differences between FA and DT, with some agreement on key features. Angular velocity is the most important feature for DT and the second most important for FA. Conversely, the position variable ranks highest in FA but is the least important in DT. Notably, while position is the least important for DT and angle for FA, their combined importance ranking highlights velocity instead. These discrepancies suggest that FA and DT capture different aspects of feature relevance, potentially leading to different interpretations.

The high importance placed on the position variable from FA comes as a surprise since previous work has placed position as the least important feature [TIF22]. This could have many reasons. One reason could be a bias in the trained policy, so that the agent has trained with a policy that prioritizes staying centered on the track (rather than purely stabilizing the pole), the learned function may exhibit higher gradients for cart position. Another reason could be that if the at the end of learning agent is already very good

a keeping the pole upright. Therefore, if this feature is already in a region where the network is not changing much with respect to it, the gradients for these features might be very small compared to other gradients.

| Decision Tree | Feature Attribution | Sum of Both |
|---|---|---|
| Angular Velocity | Position | Angular Velocity |
| Angle | Angular Velocity | Position |
| Velocity | Velocity | Angle |
| Position | Angle | Velocity |

Table 7.1: Feature Importance Overview

Figure 7.1 illustrates feature importance values sorted by the sum of FA and DT rankings, while also comparing their significance across reinforcement learning (RL) algorithms, specifically DQN and PPO. PPO assigns significantly higher importance to angular velocity but lower importance to other features. DQN, in contrast, assigns slightly higher importance to velocity, whereas PPO and the sum of both algorithms emphasize angle more.

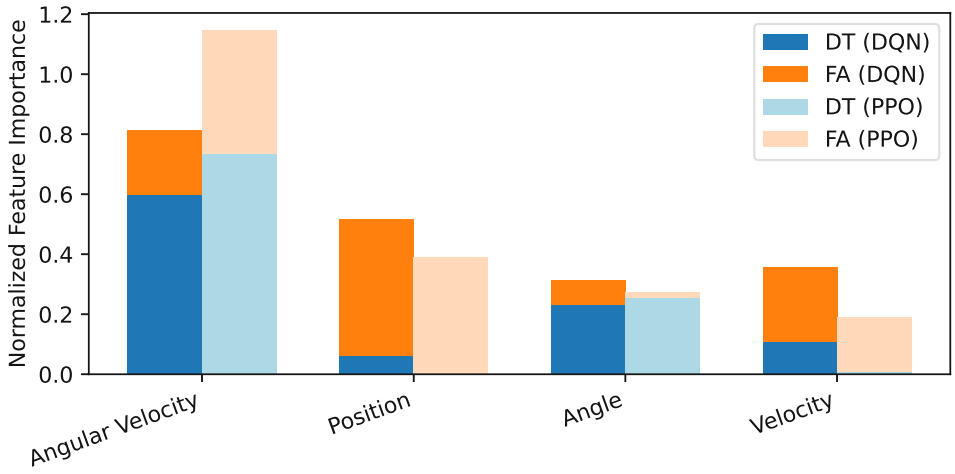



Figure 7.1: Feature importance for CartPole across FA, DT, and RL algorithms (DQN, PPO)

In the MountainCar environment (Figure 7.2), feature importance is assessed for its two available state variables: position and velocity. Given the limited number of features, feature selection is less meaningful in this case. However, DT identifies velocity as the most important feature, while FA assigns slightly higher importance to position. Despite minor

variations, both methods consistently consider both features highly relevant, without too much variation between PPO and DQN.

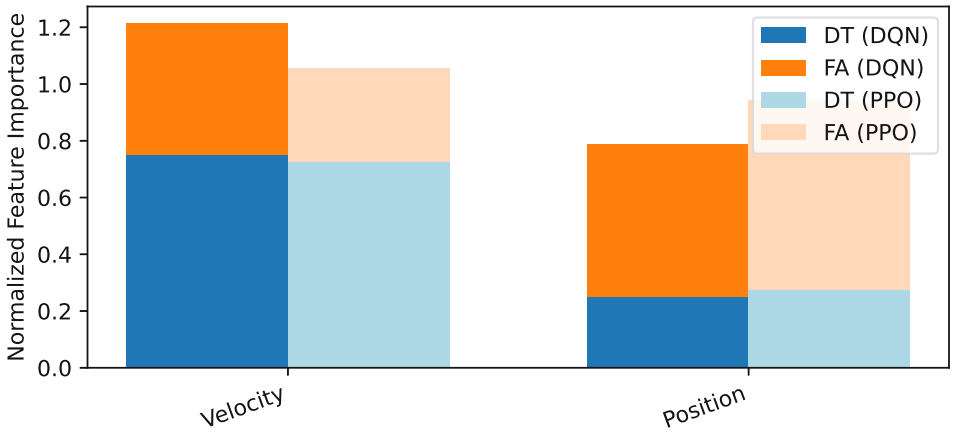



Figure 7.2: Feature importance for MountainCar based on FA, DT, and RL algorithms

Figure 7.3 presents the feature importance rankings for the LunarLander environment, showing how they differ across reinforcement learning methods (DQN and PPO). Y velocity emerges as the most important feature across all approaches, indicating its strong influence on the agent's decision-making. Beyond this, the rankings diverge, but stay more similar than in the Cartpole Case: DT assigns higher importance to angle, while FA places more emphasis on X velocity and X position, possibly capturing directional adjustments. Notably, leg contact features rank lower in all cases, indicating they contribute less to immediate decision-making.

Table 7.2 highlights different feature selection strategies by identifying the least important features according to decision trees, feature attribution (FA), and their combined ranking. Three selection options are considered: removing the two, three, or four least important features. For the worst two features, FA and the combined ranking identify Right Leg Contact and X Position, while DT ranks Left Leg Contact and X Position as least important, suggesting some disagreement on the relevance of leg contact sensors. When selecting the worst three features, DT and the sum ranking agree on excluding Left Leg Contact, Right Leg Contact, and X Position, whereas FA instead ranks Right Leg Contact, Left Leg Contact, and Angular Velocity as the least relevant. Expanding to four features, FA and the sum ranking remain aligned, including Angular Velocity as the next least important, while DT instead ranks X Velocity among the lowest. These differences highlight the varying perspectives on which features are least influential for the agent's decision-making, with DT assigning lower importance to velocity-related features, while FA de-emphasizes contact-related features. The performance of these

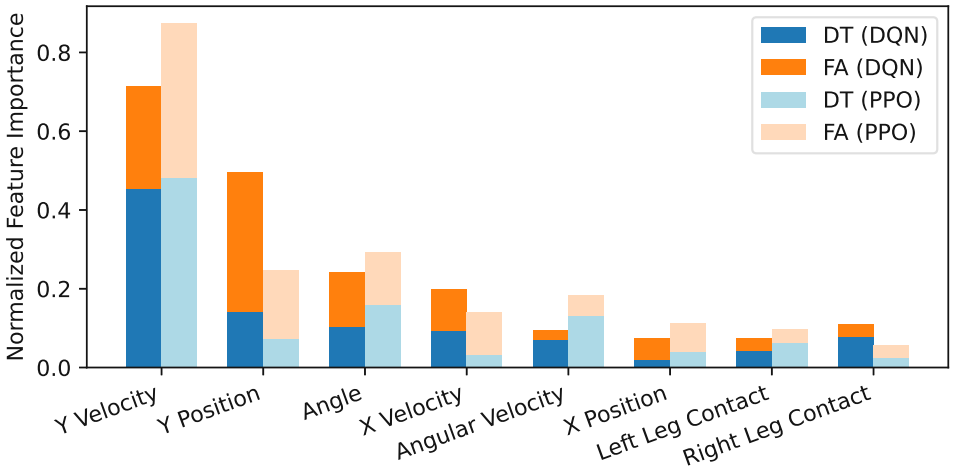



Figure 7.3: Feature importance for LunarLander across FA, DT, and RL algorithms

different approaches, and therefore finding the optimal feature subset, will be evaluated in the next chapter.

| Decision Tree | Feature Attribution | Combined: Sum of Both |
|---|---|---|
| Y Velocity | Y Velocity | Y Velocity |
| Angle | Y Position | Y Position |
| Y Position | Angle | Angle |
| Angular Velocity | X Velocity | X Velocity |
| X Velocity | X Position | Angular Velocity |
| Left Leg Contact | Angular Velocity | X Position |
| Right Leg Contact | Left Leg Contact | Left Leg Contact |
| X Position | Right Leg Contact | Right Leg Contact |

Table 7.2: Feature Importance Comparison

## 7.2 Results on Feature Selected Models

To evaluate the impact of feature selection, we trained new agents using the same random seeds while limiting access to selected feature subsets. Both DQN and PPO against were trained with all parameters, including random seed, left the same as their original agents counterparts, which had access to the full feature set. As a reference, each plot includes a dotted line representing the performance of original agents trained with full feature access.

Additionally, comparisons are made against models trained by omitting only the most important features. The percentage of solved episodes is stacked for PPO agents, making a perfect score 2 instead of 1. Confidence intervals are included, capturing variations across different seeds.

For CartPole (Figure 7.4), removing the least important feature according to DT (position) results in the highest performance, even slightly exceeding that of agents with full feature access (though within the 95% confidence interval). The second-best performing setup follows the sum of FA and DT importance, excluding velocity, with a slight average performance drop but still within CI. Excluding angle, as determined by FA, leads to the worst performance, as DQN agents fail to learn a successful policy, while PPO agents almost always solve the environment. Notably, even training with only the most important feature (angular velocity)—which is the only feature not excluded by any method—yielded better results than FA-based selection, though their confidence intervals overlapped.

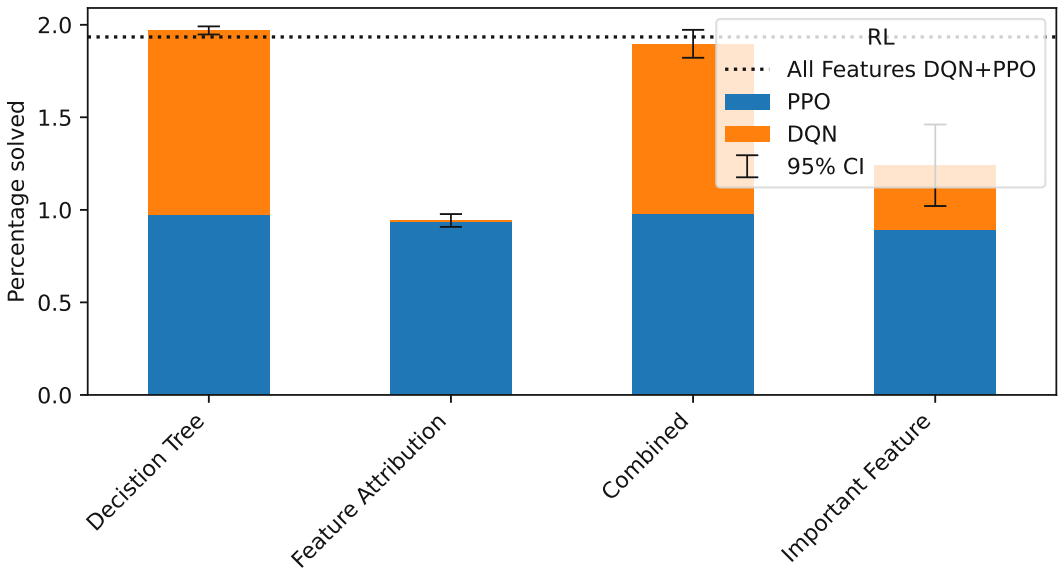



Figure 7.4: Performance comparison of CartPole agents with selected feature exclusions

For LunarLander (Figure 7.5), excluding the least important two features (Left Leg Contact and Right Leg Contact) based on the combined approach results in the highest average score, whereas excluding the least important feature identified by DT reduces the percentage of solved episodes to approximately 0.4. When three features are excluded, DT and the combined approach align, performing a little worse than FA-based exclusion. With four excluded features, FA and the combined approach again match, with PPO agents still solving 70% of episodes, while DQN agents fail entirely. Excluding the four least important DT features leads to neither DQN nor PPO solving the environment.

Moreover, training agents with only the two, three, or four most significant features also results in failure to solve the environment, demonstrating that the feature importance rankings provide a reasonable and meaningful assessment of feature relevance in the agent's decision-making process.

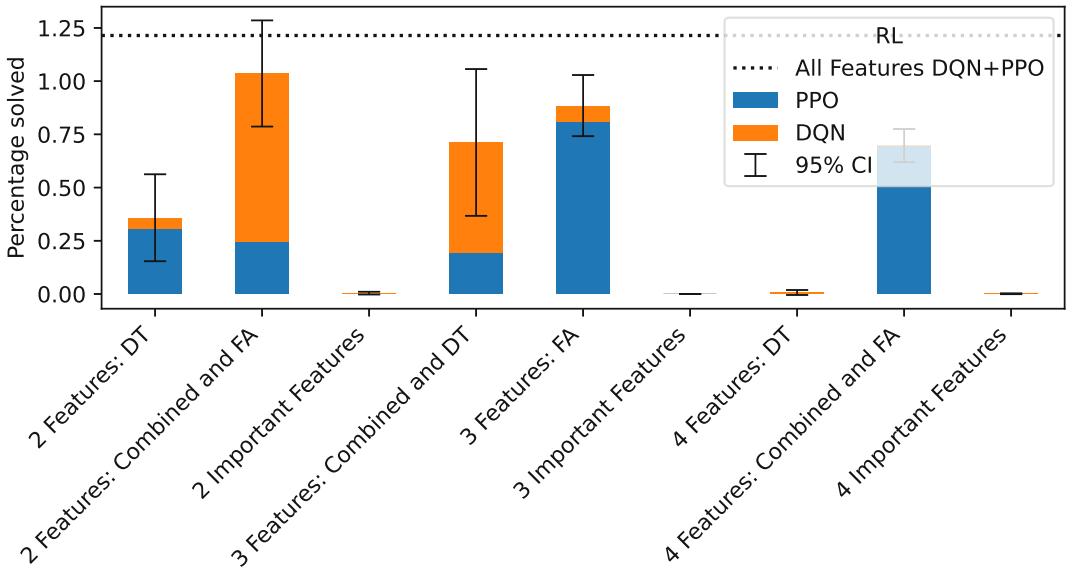



Figure 7.5: Performance comparison of LunarLander agents with selected feature exclusions

The results highlight significant differences in the effectiveness of feature selection methods across environments. In CartPole, the decision tree DT-based feature selection approach clearly outperforms feature attribution (FA), with DT-selected features leading to even better performance than using the full feature set. In contrast, FA-based selection performs worse, with some configurations failing entirely, despite including features deemed most important by FA. This suggests that as discussed in the previous section the FA feature importance for CartPole might be flawed.

For LunarLander, the pattern is reversed: agents trained after excluding FA-selected features consistently outperform those trained using DT-selected feature exclusion. Here, DT-based exclusion is significantly worse, reinforcing that DT importance rankings may not always generalize well across more complex environments. This contrast between environments suggests that no single feature selection method is universally optimal.

The combined approach, which integrates both FA and DT importance rankings, consistently yields strong results. While FA alone performs poorly in CartPole and DT struggles in LunarLander, the combined ranking approach achieves near-optimal performance across both cases. This indicates that leveraging multiple feature importance

metrics provides a more robust selection strategy, reducing the risk of excluding critical information and improving agent performance across diverse tasks.

### 7.2.1 Statistical Evaluation of Feature Selection Methods

To assess the effectiveness of different feature selection methods, we tested whether the performance distributions of the four approaches: DT, FA, combined, and important feature(s). For each pairing of approaches, we tested whether the two distributions were statistically significantly different from each other. Since normality was not established but the data was symmetric, we applied the Wilcoxon Signed-Rank tests. The results are described in Figure 7.6. Each of the four rounded yellow boxes represents a different approach, with the mean percentage of solved cases on the CartPole and LunarLander environments written below. The arrows point from the method performing worse to the method performing better. In the small boxes the p-value resulting from the Wilcoxon Signed-Rank test is plotted.

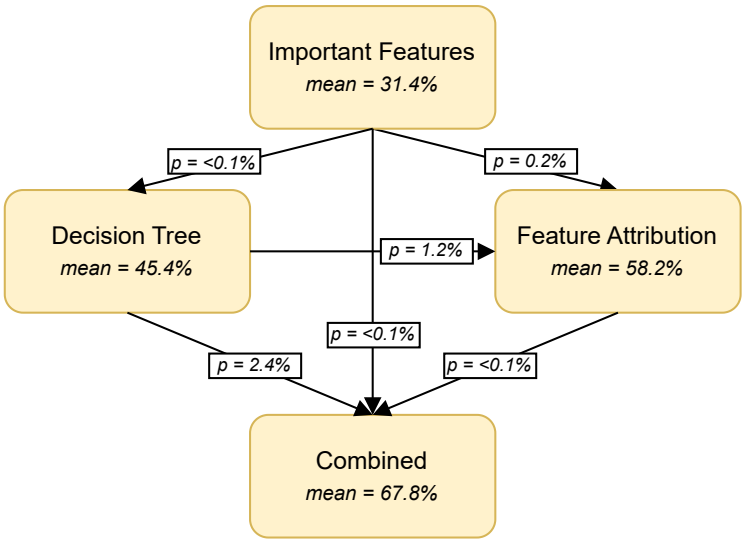



Figure 7.6: Wilcoxon Signed-Rank test results for pairwise comparison of feature selection methods

All test results are significant with p-values smaller than 5%. Among the methods, selecting the important features alone resulted in the lowest performance, with a mean success rate of 31.4%. This baseline shows that all selected feature section techniques do something reasonable, and do not just randomly select a feature. The decision tree-based approach has a mean percent solved of 45.4%, followed by feature attribution at 58.2%. The combined approach, integrating both DT and FA rankings, achieved the highest performance at 67.8%, highlighting its better performance in the two environments.

## 7.3 Feature Selection in HVAC

Following the evaluation of feature selection in reinforcement learning environments, we now extend our analysis to a real-world application: the HVAC control system introduced in section 3.2. Understanding which features influence HVAC decision-making is essential for improving energy efficiency and optimizing climate control strategies. Furthermore, using feature selection to completely remove a sensor can reduce system complexity, lower hardware and maintenance costs, and eliminate potential points of failure in real-world deployments.

To get the feature importance, we computed the sum of integrated gradients of the PPO model introduced by Stippels et al. [SSS+24] throughout the learning process. Figure 7.7 shows the top 10 features alongside the sum of the remaining 27 features. The plot indicates that $CO_2$ concentration is by far the most significant feature, followed by the energy usage $Q_{Z3}$ and the humidity $h_{Z2}$. This prominence is likely due to the strong correlation between $CO_2$ levels in different zones and the presence of people in the office. Elevated $CO_2$ concentrations often indicate occupied spaces, making it a reliable criterion for detecting human presence in enclosed environments. Since temperature regulation is unnecessary in unoccupied zones, $CO_2$ serves as an initial indicator for HVAC operation, enabling occupancy-based control followed by fine-tuned temperature regulation. $CO_2$-based occupancy inference has been extensively studied [JBLW+15], highlighting its effectiveness in optimizing HVAC performance.

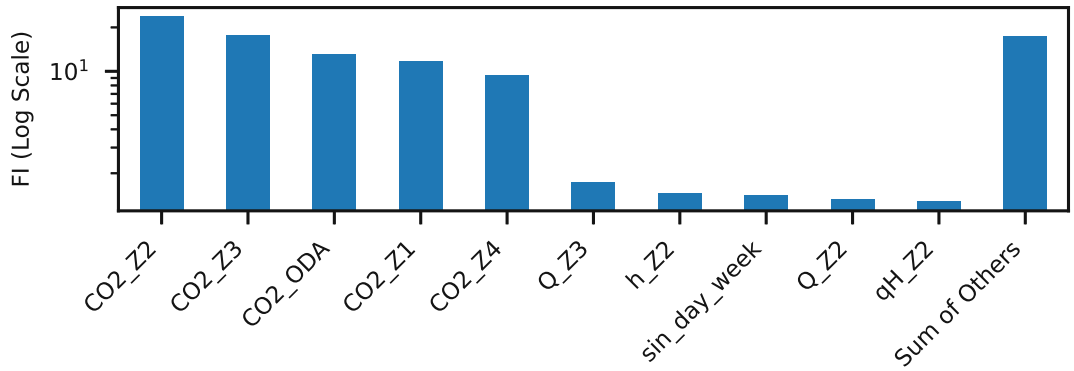
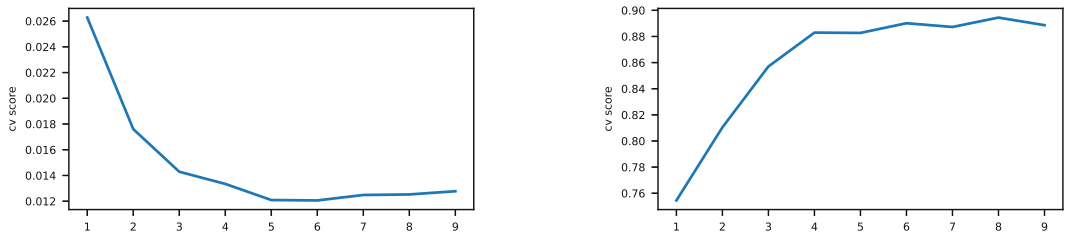



Figure 7.7: Feature Attribution: Top 10 Feature Importances based on Integrated Gradients

To gain more insights into the workings of the HVAC RL model, we applied rule extraction based on CART to obtain two decision trees: One regression tree that explains the HVAC control set points and one classification tree, which explains the categorical HVAC control strategy to use. For this, instead of learning on the whole online learning process of the HVAC RL model, we used only the last three years of data, during which the model's learning had largely stabilized. Here we used the earlier two years as a training set and

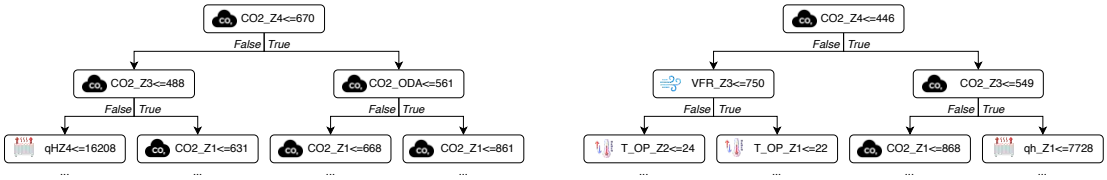
the last year as a test set. To decide how deep the extracted tree should be, we tested maximum decision tree depths from one to ten. This test was conducted on the test set data using cross validation. The results visualized in Figure 7.8 show that the trees perform best at depths 6 and 8. For further analysis, we will use the depth of 6 for both trees since shallower trees are more easily explained and the difference in performance is only minor.




(a) Cross Validation for Regression Tree (b) Cross Validation for Classification Tree

Figure 7.8: Hyperparameter optimization max depth

Figure 7.9a illustrates the top three nodes of the regression tree for visualization purposes. The tree predominantly examines indoor $CO_2$ levels across various zones and outdoor $CO_2$ levels. These variables frequently appear in multiple branches, highlighting their crucial role in predicting the target variable. In addition to $CO_2$, the model also considers energy consumption and heating demand.




(a) Regression Tree of depth 3 (b) Classification Tree of depth 3

Figure 7.9b illustrates the top three nodes of the classification tree of depth 3. The decision tree identifies the appropriate control logic to apply, as with the regression tree, the concrete decision variable for the control logic is stored at the leaf nodes.

Like the regression tree, $CO_2$ levels are the most significant factor in the decision-making process. They appear frequently in multiple branches, which indicates that $CO_2$ levels are a strong predictor for determining the control strategy. We assume that the model learns to predict when thermal comfort is important, based on occupancy of the building, which can be determined with the $CO_2$ level. Additionally, the model considers volume flow rates, indoor temperature and heating demands.

In addition to the tree structure, and as an alternative to the feature attribution-based feature importance visualized in Figure 7.7, we compute the feature importance based on the classification and regression tree to quantify the relative contribution of each variable

to the model's overall performance and visualize the top 10 most influential features in Figure 7.10.

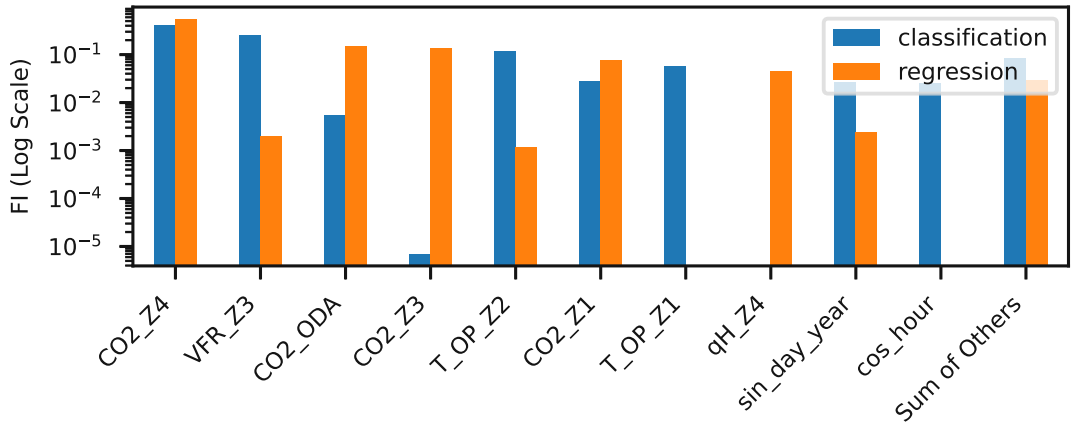



Figure 7.10: Top 10 Feature Importances with the sum of other 27 Features based on regression of depth 6 and classification tree of depth 6

As expected by the high position taken before in the decision trees and in line with the feature attribution, indoor and outdoor $CO_2$ levels rank as the most important features. However, Airflow rates, indoor temperatures, heating demands along with time and date also make it into the top ten most important features.

Table 7.3 shows the mean absolute error between the regression tree's predictions and the original model for the final year. The results show that the predicted setpoints differ from the models setpoints on average by around 0.78 °C for temperature, 3.88 %rH for humidity, and 12 % for the fraction of fresh air, indicating a reasonable level of accuracy overall. This demonstrates that the regression tree offers a viable and interpretable substitute for the black-box RL model in determining continuous control setpoints. Additionally, the classification tree reaches an accuracy of 90% when evaluated in the final year, strengthening the case for decision tree-based models as effective and interpretable surrogates for the RL model.

| Feature | MAE | Feature | MAE | Feature | MAE |
|---|---|---|---|---|---|
| T_Set_Z1 | 0.54 | Hum_Set_Z1 | 1.78 | T_Set_Z4 | 0.32 |
| T_Set_Z2 | 0.45 | Hum_Set_Z2 | 3.73 | Hum_Set_Z4 | 6.12 |
| T_Set_Z3 | 1.81 | Hum_Set_Z3 | 3.88 | f_Mix_RCA | 0.12 |

Table 7.3: Mean Absolute Error (MAE) for decision tree with all features: Hum_Set - Set Humidity in %rH, T_Set - Set Temperature in °C, f_Mix_RCA - Relative amount of fresh air. Z indicates the respective zone.

To check the performance of sensor selection for the HVAC RL model we took the 9 most

important sensors and replaced all other sensor values with their global mean. This was necessary because we could not train the model again, since we had no access to the environment. However, the RL model expected to receive values for the other sensors. The results for the metric setpoints are presented in Table 7.4.

The low error values indicate strong alignment with the original model's predictions, showing that the regression tree remains effective even with limited input data. This is further supported by the high classification accuracy of 97% for the control strategy in the final year, confirming that the model can reliably select the appropriate control strategy under the same constraints.

Table 7.4 shows the MAE for the predicted setpoints when only using the nine most important sensors, with all other inputs masked by their global mean over the seven-year period. This scenario evaluates how well the model performs without access to the full sensor set.

The results show that the predicted values differ from the models on average by around 0.75 °C for temperature, 3.43 %rH for humidity, and 4 % for the fraction of fresh air, indicating a reasonable level of accuracy overall. These values are consistently lower than their decision tree counterparts. Furthermore, they indicate a strong ability to mimic the original model, even with limited data. In addition, the limited model achieves a very high accuracy of 97% for choosing the control strategy.

| Feature | MAE | Feature | MAE | Feature | MAE |
|---|---|---|---|---|---|
| T_Set_Z1 | 0.80 | Hum_Set_Z1 | 2.86 | T_Set_Z4 | 0.57 |
| T_Set_Z2 | 1.23 | Hum_Set_Z2 | 3.75 | Hum_Set_Z4 | 2.31 |
| T_Set_Z3 | 0.42 | Hum_Set_Z3 | 4.80 | f_Mix_RCA | 0.04 |

Table 7.4: MAE for continuous target variables over the last year: Hum_Set – Set humidity in %rH, T_Set – Set temperature in °C, f_Mix_RCA – Relative amount of fresh air. Z indicates the zone.

By applying feature attribution and rule extraction, we were able to identify the most relevant inputs for the RL HVAC control model. Both techniques consistently highlighted $CO_2$ levels as the dominant input feature, confirming their strong correlation with occupancy. We used decision trees as surrogate models to demonstrate that a reasonable accuracy of the RL model can be achieved when using a white-box model. By limiting the model's input to the nine most important features, we were able to reduce the required number of sensors by more than 70%, while preserving the model's behavior with high fidelity.

CHAPTER 8

# Conclusion and Future Work

This thesis investigates two different approaches to improve the explainability in RL. First, it introduces novel methods that integrate either certainty levels or feature attribution into rule extraction and evaluates the performance of the resulting surrogate decision tree models. Second, it tests rule extraction and feature attribution for the use of Feature Selection and suggests a novel feature selection method by combining the two.

The experimental results indicate that incorporating certainty levels or feature attribution values in rule extraction does not enhance fidelity compared to standard CART. For most tree complexities tested, FAWDT and UWDT exhibit reduced performance on the underlying control problem, while CWDT perform similarly to CART without a significant difference. However, when extracting only very shallow trees, decision trees extracted with FAWDT or UWDT are able to solve the environments more consistently than CART.

Feature selection experiments highlight significant differences between selection methods in the two environments. Decision tree-based selection performs better in CartPole, whereas feature attribution-based selection is more effective in LunarLander. The combined approach, which integrates both FA and DT rankings, shows strong results across both environments. When testing the differences between groups with Wilcoxon Signed-Rank tests, all approaches show significant differences from each other. The baseline of selecting important features performed worst, decision tree-based selection placed third, feature attribution-based selection ranked second, and the combined approach achieved the best performance.

Applying these explainability techniques to HVAC control systems reveals $CO_2$ levels as the dominant predictor for control decisions, aligning with its role in occupancy inference. Rule-extracted decision trees further highlight the importance of $CO_2$ levels and can be used as a surrogate model instead of the RL model. Reducing the sensor set to the most

important features decreased the number of required sensors by more than 70%, while preserving the model's behavior with high fidelity.

This paper demonstrates that explainability techniques, such as feature attribution and rule extraction, enable sensor reduction and enhance model interpretability. By identifying critical features and extracting decision tree logic, these methods facilitate the deployment of efficient and transparent RL-based HVAC control systems.

The results of this thesis motivate further exploration of certainty-based and feature-attribution-based rule extraction methods to assess how the novel splitting criteria perform in other use cases. Alternative approaches, such as using different feature attribution techniques or weighting mechanisms, could be explored to improve performance. Furthermore, extending the evaluation to more complex HVAC environments with real-world deployment would provide deeper insight into the applicability and robustness of the proposed methods. Finally, while the combination of feature attribution and rule extraction shows very promising results, the evaluation could be extended to different domains and tested against an extended set of benchmarks.

# Overview of Generative AI Tools Used

I acknowledge the use of generative AI tools in the creation of this work. These tools were used for tasks such as code debugging, summarization, paraphrasing, and grammar checking through the whole thesis. These tools have been an important aid in my research and writing process, and their impact on the final product is acknowledged. Nonetheless, I have only used generative AI tools as an aid and my creative influence predominates in the present work.

Ai tools used:

- ChatGPT [Ope24]
- Writefull [noa25b]
- Grammarly [noa25a]

# List of Figures

# List of Tables

# Bibliography


[AAES+23] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Information Fusion*, 99:101805, November 2023.

[ACOG17] Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[ACOG19] Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Gradient-Based Attribution Methods. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 169–191. Springer International Publishing, Cham, 2019.

[ADT95] Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, December 1995.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, June 2016. arXiv:1606.01540 [cs].

[BFOS84] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification And Regression Trees*. Chapman and Hall/CRC, 1 edition, 1984.

[BH21] Nadia Burkart and Marco F. Huber. A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research*, 70:245–317, January 2021.

[BPSL18] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems*, 31, 2018.

[BSA83] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, September 1983. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[BSS+24] Aleksey Bratukhin, Christian Stippel, David Sengl, Markus Kobelrausch, and Thilo Sauter. Energy Efficiency Optimization and Preventive Maintenance of HVAC Systems Using Machine Learning. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, September 2024. ISSN: 1946-0759.

[CLO+23] Youngsik Choi, Xing Lu, Zheng O'Neill, Fan Feng, and Tao Yang. Optimization-informed Rule Extraction for HVAC system: A Case Study of Dedicated Outdoor Air System Control in a Mixed-Humid Climate Zone. *Energy and Buildings*, page 113295, June 2023.

[Cra96] Mark William Craven. *Extracting comprehensible models from trained neural networks*. The University of Wisconsin-Madison, 1996.

[CS94] Mark W. Craven and Jude W. Shavlik. Using Sampling and Queries to Extract Rules from Trained Neural Networks. In William W. Cohen and Haym Hirsh, editors, *Machine Learning Proceedings 1994*, pages 37–45. Morgan Kaufmann, San Francisco (CA), January 1994.

[DBDJ90] Ralph B D'agostino, Albert Belanger, and Ralph B D'Agostino Jr. A suggestion for using powerful and informative tests of normality. *The American Statistician*, 44(4):316–321, 1990. Publisher: Taylor & Francis.

[ELWK23] Raphael C. Engelhardt, Moritz Lange, Laurenz Wiskott, and Wolfgang Konen. Sample-Based Rule Extraction for Explainable Reinforcement Learning. In Giuseppe Nicosia, Varun Ojha, Emanuele La Malfa, Gabriele La Malfa, Panos Pardalos, Giuseppe Di Fatta, Giovanni Giuffrida, and Renato Umeton, editors, *Machine Learning, Optimization, and Data Science*, Lecture Notes in Computer Science, pages 330–345, Cham, 2023. Springer Nature Switzerland.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[GKDF18] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and Understanding Atari Agents. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1792–1801. PMLR, July 2018. ISSN: 2640-3498.

[Goo13] Phillip Good. *Permutation tests: a practical guide to resampling methods for testing hypotheses*. Springer Science & Business Media, 2013.

[HBV06] Johan Huysmans, Bart Baesens, and Jan Vanthienen. Using Rule Extraction to Improve the Comprehensibility of Predictive Models, 2006.

[HCDR21] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214:106685, February 2021.

[Hel] Rico Helmboldt. Evaluating Saliency Map Methods in Reinforcement Learning.

[HLA22] Tobias Huber, Benedikt Limmer, and Elisabeth André. Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents. *Frontiers in Artificial Intelligence*, 5, July 2022. Publisher: Frontiers.

[HZGM22] Georges Hatem, Joe Zeidan, Mathijs Goossens, and Carla Moreira. NORMALITY TESTING METHODS AND THE IMPORTANCE OF SKEWNESS AND KURTOSIS IN STATISTICAL ANALYSIS. *BAU Journal - Science and Technology*, 3(2), June 2022.

[JBLW+15] Ming Jin, Nikolaos Bekiaris-Liberis, Kevin Weekly, Costas Spanos, and Alexandre Bayen. Sensing by proxy: Occupancy detection based on indoor CO2 concentration. *UBICOMM 2015*, 14, 2015.

[JRMCN+23] Javier Jiménez-Raboso, Antonio Manjavacas, Alejandro Campoy-Nieves, Miguel Molina-Solana, and Juan Gómez-Romero. Explaining Deep Reinforcement Learning-Based Methods for Control of Building HVAC Systems. In Luca Longo, editor, *Explainable Artificial Intelligence*, pages 237–255, Cham, 2023. Springer Nature Switzerland.

[KK16] M. Killian and M. Kozek. Ten questions concerning model predictive control for energy efficient buildings. *Building and Environment*, 105:403–412, August 2016.

[KK21] Ioannis Kakogeorgiou and Konstantinos Karantzalos. Evaluating explainable artificial intelligence methods for multi-label deep learning classification tasks in remote sensing. *International Journal of Applied Earth Observation and Geoinformation*, 103:102520, December 2021.

[KKP+24] K. Kalaiselvi, K.Niranjana, V. Prithivirajan, K. Suresh Kumar, V.Syambabu, and Sathiya B. Machine Learning for Predictive Maintenance in Industrial Equipment: Challenges and Application. In *2024 4th Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–6, August 2024.

[Kle88] Sanford A Klein. TRNSYS-A transient system simulation program. *University of Wisconsin-Madison, Engineering Experiment Station Report*, pages 38–12, 1988.

[KW24] Niklas Koenen and Marvin N. Wright. Toward Understanding the Disagreement Problem in Neural Network Feature Attribution. In Luca Longo, Sebastian Lapuschkin, and Christin Seifert, editors, *Explainable Artificial Intelligence*, pages 247–269, Cham, 2024. Springer Nature Switzerland.

[LSZL19] Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 414–429, Cham, 2019. Springer International Publishing.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. Publisher: Nature Publishing Group.

[Mni13] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[MRRS00] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[NAPS11] Arundhati Navada, Aamir Nizam Ansari, Siddharth Patil, and Balwant A. Sonkamble. Overview of use of decision tree algorithms in machine learning. In *2011 IEEE Control and System Graduate Research Colloquium*, pages 37–42, June 2011.

[noa25a] Grammarly (https://www.grammarly.com/grammar), 2025.

[noa25b] Writefull (https://www.writefull.com/), 2025.

[Ope24] OpenAI. ChatGPT (multiple versions) (https://chat.openai.com), 2024.

[PLOP08] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and Buildings*, 40(3):394–398, January 2008.

[PVG+19] Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move:

Understanding agent actions using specific and relevant feature attribution. *arXiv preprint arXiv:1912.12191*, 2019.

[Rec19] Benjamin Recht. A Tour of Reinforcement Learning: The View from Continuous Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(Volume 2, 2019):253–279, May 2019. Publisher: Annual Reviews.

[RHG+21] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[SSS+24] Christian Stippel, Rafael Sterzinger, David Sengl, Aleksey Bratukhin, Markus Kobelrausch, Stefan Wilker, and Thilo Sauter. Online HVAC Optimization under Comfort Constraints via Reinforcement Learning. 2024.

[Stu08] Student. The Probable Error of a Mean. *Biometrika*, 6(1):1–25, 1908. Publisher: [Oxford University Press, Biometrika Trust].

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3319–3328. PMLR, July 2017. ISSN: 2640-3498.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv e-prints*, pages arXiv–1312, 2013.

[SW65] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, 1965. Publisher: Oxford University Press.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347.

[THR22] Saman Taheri, Paniz Hosseini, and Ali Razban. Model predictive control of heating, ventilation, and air conditioning (HVAC) systems: A state-of-the-art review. *Journal of Building Engineering*, 60:105067, November 2022.

[TIF22] Ahmad Terra, Rafia Inam, and Elena Fersman. BEERL: Both Ends Explanations for Reinforcement Learning. *Applied Sciences*, 12(21):10947, January 2022. Number: 21 Publisher: Multidisciplinary Digital Publishing Institute.

[TKT+24] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments, November 2024. arXiv:2407.17032 [cs].

[TRR+19] Mohamed Toub, Chethan R. Reddy, Meysam Razmara, Mahdi Shahbakhti, Rush D. Robinett, and Ghassane Aniba. Model-based predictive control for optimal MicroCSP operation integrated with building HVAC systems. *Energy Conversion and Management*, 199:111924, November 2019.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[VV24] Daniël Vos and Sicco Verwer. Optimizing Interpretable Decision Tree Policies for Reinforcement Learning, August 2024. arXiv:2408.11632 [cs].

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[WHL+23] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136, May 2023. Conference Name: IEEE/CAA Journal of Automatica Sinica.

[Wil92] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*, pages 196–202. Springer, 1992.

[WWGL22] Man Wang, Zhe Wang, Yang Geng, and Borong Lin. Interpreting the neural network model for HVAC system energy data mining. *Building and Environment*, 209:108449, February 2022.

[WWZ17] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep Reinforcement Learning for Building HVAC Control. In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, pages 1–6, New York, NY, USA, June 2017. Association for Computing Machinery.

[ZBRS22] Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. Do Feature Attribution Methods Correctly Attribute Features? *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9):9623–9633, June 2022. Number: 9.

[Zho04] Zhi-Hua Zhou. Rule extraction: Using neural networks or for neural networks? *Journal of Computer Science and Technology*, 19(2):249–253, March 2004.