Machine Learning with Applications 22 (2025) 100795

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Evaluating the effectiveness of LLMs for explainable deep reinforcement learning

Ayoub Belouadah *, Marcelo Luis Ruiz-Rodríguez, Sylvain Kubler, Yves Le Traon

SnT, University of Luxembourg, 6 Rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg

ARTICLE INFO

Keywords:
Explainable reinforcement learning
Large Language Models
Textual explanations
Chain-of-Thought
Monte-Carlo Tree Search
Supervised fine-tuning
LLM-as-a-judge

ABSTRACT

Understanding the decision-making of reinforcement learning (RL) agents is essential for real-world deployment. Existing eXplainable RL (XRL) techniques, such as feature attribution and policy visualization, provide insight but remain inaccessible to non-experts. Large Language Models (LLMs) offer a natural-language alternative, yet often lack logical consistency and alignment with agent goals. This study benchmarks three explanation generation methods: Chain-of-Thought (CoT) prompting as the standard baseline used in prior work, Monte Carlo Tree Search (MCTS) augmentation, and supervised fine-tuning (SFT) across various models. Evaluations using Soundness and Fidelity show that CoT frequently produces reasoning errors, whereas MCTS improves quality for larger models (avg. +23% Soundness, +17% Fidelity), while SFT yields greater and more consistent gains for smaller ones (+58% Soundness, +52% Fidelity), underscoring the need to align methods with model capacity. An LLM-as-a-Judge framework further validates these findings, showing strong agreement with human assessments (weighted Cohen's $\kappa = 0.77$, Spearman $\rho = 0.88$), supporting scalable and reliable assessment of textual explanations.

1. Introduction

Deep Reinforcement Learning (DRL) (Lillicrap et al., 2015; Sutton & Barto, 2018) has achieved remarkable success across a variety of tasks, like robotics and game-playing agents (Evans et al., 2023; Mnih et al., 2015; OpenAI et al., 2019), recommender systems and maintenance scheduling (Gupta et al., 2023; Kamrani et al., 2025; Ruiz-Rodríguez et al., 2025; Ziaei & Ranjbar, 2023). More broadly, AI systems have been applied in diverse domains such as healthcare, sustainability, and energy systems, demonstrating their growing role in data-driven decision-making (Ahmad et al., 2025; Ashraf et al., 2023). Despite these advances, AI, and more specifically, DRL systems, often operate as "black boxes", leaving both practitioners and non-experts uncertain about how decisions are made. Explainable Reinforcement Learning (XRL) (Milani et al., 2024; Qing et al., 2022) aims to bridge this gap by clarifying the behaviors, policies, and objectives of RL agents. While existing techniques such as feature importance attribution (Beechey et al., 2023; Sundararajan et al., 2017) and policy visualization (Atrey et al., 2020; Greydanus et al., 2018) are helpful for AI specialists, they often fail to reach broader audiences without further simplification.

Large Language Models (LLMs) (Touvron et al., 2023; Zhao et al., 2025) offer a promising approach for generating accessible natural-language explanations. They can translate complex outputs from XRL and broader Explainable AI (XAI) methods, such as feature importance, into simplified textual summaries (Mavrepis et al., 2024; Zytek et al., 2024b). However, applying LLMs to explain RL agents introduces unique challenges: explanations must faithfully represent the agent's actual behavior and decision-making factors, avoiding misleading interpretations, and remain consistent with the environment's dynamics by clearly linking actions to the agent's objectives. To the best of our knowledge, this paper provides the first empirical evaluation of LLM-generated explanations for DRL agents, considering how explanation quality is affected by the LLM scale, explanation strategy, and environment complexity. While existing studies typically focus on a single LLM evaluated within a single environment (Ameur et al., 2024; Metzger et al., 2023; Zhang et al., 2023), often relying on predefined queries and human interaction to correct or mitigate errors, this study conducts a comprehensive comparison study of distinct reasoning strategies: baseline Chain-of-Thought (CoT) prompting, adaptation of Monte Carlo Tree Search (MCTS), and supervised fine-tuning (SFT). Unlike prior methods using surrogate or distilled models (Zhang et al., 2023), or reward decomposition methods (Metzger et al., 2023), our approach employs feature attribution vectors extracted directly from the DRL agent's policy, providing faithful, fine-grained insight into the agent's

* Corresponding author.

E-mail addresses: ayoub.belouadah@uni.lu (A. Belouadah), marcelo.ruiz@uni.lu (M.L. Ruiz-Rodríguez), sylvain.kubler@uni.lu (S. Kubler), yves.letraon@uni.lu (Y. Le Traon).

https://doi.org/10.1016/j.mlwa.2025.100795
Received 1 September 2025; Received in revised form 6 November 2025; Accepted 11 November 2025
Available online 14 November 2025

internal reasoning. The quality of LLM-generated explanations is evaluated through manual validation using two metrics: Soundness (logical consistency) and Fidelity (alignment with feature attribution). Furthermore, we validate our evaluation framework through a quantitative "LLM-as-a-Judge" approach (Gu et al., 2024), demonstrating strong agreement with manual annotations, thereby presenting a scalable path toward automating explanation evaluation in XRL.

Section 2 reviews related work in XRL, with a focus on the emerging use of LLMs for generating explanations in DRL settings. Section 3 outlines the research methodology used to qualitatively and quantitatively evaluate LLM-based explanation approaches, namely CoT, MCTS, and SFT. Section 4 details the experimental setup, results, and key findings. Section 5 discusses the implications and limitations of our results and finally, Section 6 concludes our study and proposes directions for future research. To summarize, this paper makes the following key contributions:

- Comprehensive evaluation of LLM-based explanation generation strategies for DRL agents, benchmarking three distinct methods: CoT, MCTS, and SFT across multiple models and environments.
- Introduction of a qualitative and quantitative evaluation framework that measures explanation quality using two complementary metrics: Soundness (logical consistency) and Fidelity (alignment with feature attribution), supported by both expert and automated "LLM-as-a-Judge" validation.
- Qualitative and quantitative analysis of judgment reliability, identifying systematic biases and error patterns in LLM-based evaluation and outlining paths toward more robust, scalable assessment of explanation quality in XRL.

2. Related work

Section 2.1 reviews state-of-the-art methods developed within the field of XRL. Section 2.2 discusses the emerging role of LLMs in improving explainability across broader AI systems. Section 2.3 introduces recent advances in LLMs as natural language evaluators, highlighting how language models are increasingly used as automatic judges for assessing text quality and Section 2.4 explores recent work that combine LLMs with RL to generate natural language explanations.

2.1. Explainable reinforcement learning (XRL)

XRL seeks to enhance transparency of RL agents by providing human-understandable justifications for their actions. Qing et al. (2022) categorized existing work into four types of methods:

Model-explaining: these methods aim to reveal or restructure the internal logic of an RL agent to make its behavior more interpretable. Two main approaches exist: *(i) Self-explainable models:* These involve designing RL algorithms that are inherently interpretable, typically by learning policies represented as decision trees (Rodriguez et al., 2020; Topin et al., 2021), first-order logic (Jiang & Luo, 2019; Payani & Fekri, 2020), or by distilling complex models into simpler, interpretable forms (Bastani et al., 2018; Milani et al., 2022); *(ii) Explanation-generating models:* These apply auxiliary mechanisms to generate explanations after training. Examples include counterfactual explanations (Olson et al., 2021; Stein, 2021), which highlight how small input changes alter outcomes; instruction-based approaches (Fukuchi et al., 2017), which generate structured reasoning steps; and natural language explanations in response to user queries (Boggess et al., 2022).

Reward-explaining: These methods aim to clarify how the reward function influences the agent's behavior by identifying the most impactful factors. They can be grouped into: *(i) Reward decomposition:* these methods break down the reward into interpretable components, allowing analysis of each component's contribution to the agent's policy (Juozapaitis et al., 2019; Towers et al., 2024); *(ii) Reward shaping:* these methods refine or modify the reward function to enhance its interpretability and provide more meaningful learning signals (Ashwood et al., 2022; Mirchandani et al., 2021; Rezazadeh et al., 2023).

State-explaining: These methods aim to clarify how specific environment states influence the agent's behavior, typically through post-hoc explanations. Methods such as saliency maps (Greydanus et al., 2018), SHAP (Beechey et al., 2023), and Integrated Gradients (IG) (Sundararajan et al., 2017) highlight the most relevant features in an observation. Other methods, like Layer-wise Relevance Propagation (Huber et al., 2019) and attention mechanisms (Itaya et al., 2021; Mott et al., 2019) identify which inputs are most influential across different network layers.

Task-explaining: These methods break down the RL task to provide multi-level interpretability. A common method is Hierarchical RL (HRL), where a high-level controller delegates sub-tasks to lower-level policies. Existing work includes top-down multi-level task decomposition (Nangue Tasse et al., 2020; Peng et al., 2022; Shu et al., 2017) and simpler two-level decompositions (Jiang et al., 2019; Lyu et al., 2019).

While these approaches offer valuable insights, they also present practical limitations. Model-explaining methods often require pre-defined queries, surrogate models, or constrained architectures, which may involve retraining agents for interpretability. Reward-explaining techniques depend on designing or decomposing custom reward functions which are often impractical in complex environments. State-explaining methods produce technical outputs like saliency maps or feature importance scores that are difficult for non-experts to interpret without further explanation. Task-explaining approaches rely on hierarchical RL or multi-level task decomposition, making them less compatible with standard RL setups. As a result, many XRL methods struggle to deliver clear, context-aware justifications accessible to non-experts. This gap highlights the promise of LLMs, which can translate the outputs of XRL techniques into natural-language explanations grounded in an agent's actual behavior.

2.2. LLMs for explainable AI

Recent years have witnessed a growing adoption of LLMs and Generative AI to make AI systems more explainable and trustworthy (Ali et al., 2025). Building on this progress, recent works leverage LLMs to convert XAI outputs into concise, human-readable explanations. For instance, Feldhus et al. (2023) and Slack et al. (2023) enable interactive question-answering using SHAP scores. Martens et al. (2025) uses narrative-driven summaries based on SHAP and counterfactuals, while Kroeger et al. (2023) apply CoT prompting to clarify black-box predictions without requiring feature attributions. Others, like Zytek et al. (2024a), propose dual LLM-based systems to generate and score narrative explanations.

Other studies adopt another approach by leveraging *domain knowledge* or *specialized resources* to improve clarity. For example, Abu-Rasheed et al. (2024) and Hu et al. (2024) illustrate how integrating knowledge graphs as additional context improves explanation quality in fields such as recommendation and product design, while Spitzer et al. (2024) augments LLMs with Retrieval Augmented Generation to explain deep learning models to decision-makers. Similarly, Attai et al. (2024) and Mekrache et al. (2024) demonstrate LLM-driven explanations of SHAP or LIME outputs in domain-specific scenarios such as anomaly detection and healthcare, helping non-experts interpret critical decisions. Mavrepis et al. (2024) go a step further by extending the LLM-driven explanation framework to include other XAI outputs such as Grad-CAM and Partial Dependence Plots. In the work of Zytek et al. (2024b), evaluation metrics are proposed aimed at assessing the logical consistency and completeness of these explanations. Despite promising results, challenges remain, particularly around *avoiding hallucinations* and ensuring *faithful* model interpretations across diverse use cases.

2.3. LLMs-as-judges: automatic text evaluation

Recent studies demonstrate that LLMs can be used as dependable automatic judges for Natural Language Generation (NLG) outputs such as summaries, dialogues, and Q&A responses. For instance, Wang et al. (2023) show that ChatGPT achieves strong correlation with human judgments in natural language evaluation, Zheng et al. (2023) use GPT-4 as a judge in MT-Bench to assess conversational quality with near human-level agreement, and Badshah and Sajjad (2024) extend this paradigm using multiple LLMs-as-judges, substantially improving reliability and consistency with human ratings. This emerging LLM-as-a-Judge paradigm provides a methodological basis for extending automated evaluation to DRL explanation quality, reducing reliance on costly human annotation while maintaining interpretive consistency.

2.4. LLMs for XRL

With the advent of generative AI, LLMs are increasingly used as explanation generation tools for DRL agents across various domains. For example, Pandya et al. (2024) used LLMs to generate concise textual descriptions of multi-agent strategies based on transitions between key “landmark” states, helping users align with a robot’s intended collaborative policy. Zhang et al. (2023) propose a model-agnostic approach where a policy is first distilled into a decision tree, from which a local behavior representation is extracted and used to prompt an LLM while also supporting interactive user queries such as clarifications and counterfactuals. In the networking domain, Ameur et al. (2024) apply LLMs within a composable XRL framework to interpret DRL decisions in 6G network slicing. By combining LLMs with prompt engineering, they generate user-aware explanations tailored to diverse stakeholder profiles. Metzger et al. (2023) leverage the output of *XRL-Dine* (Feit et al., 2022), a reward-decomposition method, to enhance an LLM in explaining service-oriented systems. Similarly, Kim et al. (2025) proposed *TalkToAgent* where a combination of LLM agents coordinate with each other to map user queries into clear explanations of the RL agent’s decisions. Another line of work uses LLMs to shape or guide DRL training by generating rewards or corrective actions aligned with human preferences. For instance, Kwon et al. (2023) and Masadome and Harada (2025) incorporate LLMs as *reward designers* that generate and reshape the reward function based on human feedback, while Tan et al. (2025) extend this idea through *ULTRA*, where the LLM identifies critical states and suggests explanation-based actions and rewards, yielding policies that are both more interpretable and preference-aligned. Additionally, Yildirim et al. (2025) introduce *HighwayLLM*, which integrates an LLM with a pretrained RL driving agent to enhance transparency in autonomous navigation, the LLM predicts future waypoints and also generates natural language rationales explaining its decisions for every prediction, improving interpretability alongside control performance. Moreover, a recent study by Lu et al. (2025) explores the capacity of LLMs to mentally model reinforcement learning agents by reasoning over their interaction histories with the environment. The authors introduce the task of agent mental modeling and propose specific evaluation metrics to assess whether LLMs can infer an agent’s decision-making behavior solely from observed trajectories. Their results indicate that current LLMs struggle to robustly reconstruct agent intentions or policies without additional inductive biases or training, highlighting critical limitations in using general-purpose LLMs for behavior explanation in RL. Collectively, these works highlight the growing potential of LLMs to bridge the explainability gap in DRL systems by producing adaptive natural language explanations aligned with user expectations.

While recent studies have begun using LLMs to generate explanations for DRL agents, most efforts to date have focused on individual methods or case studies, without systematically assessing the reasoning capabilities of the LLMs themselves. To our best knowledge, comprehensive benchmarks that compare multiple LLMs on their capacity to reason about DRL decisions, especially in environments requiring causal and physical reasoning are still lacking. Our research work addresses this gap by evaluating multiple LLMs across two physics-based environments of varying complexity: Cartpole and Lunar Lander. We analyze how explanation quality evolves with increasing environment difficulty and progressively improve LLM performance through CoT prompting, MCTS refinement, and SFT. Our findings reveal the most effective explanation strategies across model scales and expose the limits of current LLMs in complex reasoning. Unlike prior work relying mainly on human evaluation, we propose a quantitative “LLM-as-a-Judge” framework that aligns with manual assessments and enables scalable, automated evaluation.

3. Methodology for integrating and evaluating LLM-based explanation generation methods

This section outlines our methodology for assessing the quality of explanations produced by different LLM-based explanation generation methods. We first run a DRL agent in a selected environment to generate `{state}-{action}` pairs, as illustrated through the stage denoted by “RL System” in Fig. 1. These pairs are then combined with feature attributions computed using Integrated Gradients (see “Features” stage). Next, we construct prompts by combining these inputs with contextual information about the environment such as core mechanics, objectives, and common pitfalls (see “Environment Context” stage). These prompts are passed to various explanation methods (CoT, MCTS, SFT), each applying different reasoning strategies to generate textual explanations for the same input, as indicated in their respective method blocks. Finally, the resulting explanations are evaluated using two metrics: Soundness and Fidelity, both via human annotation and through our “LLM-as-a-Judge” framework (see the “Evaluation” stage).

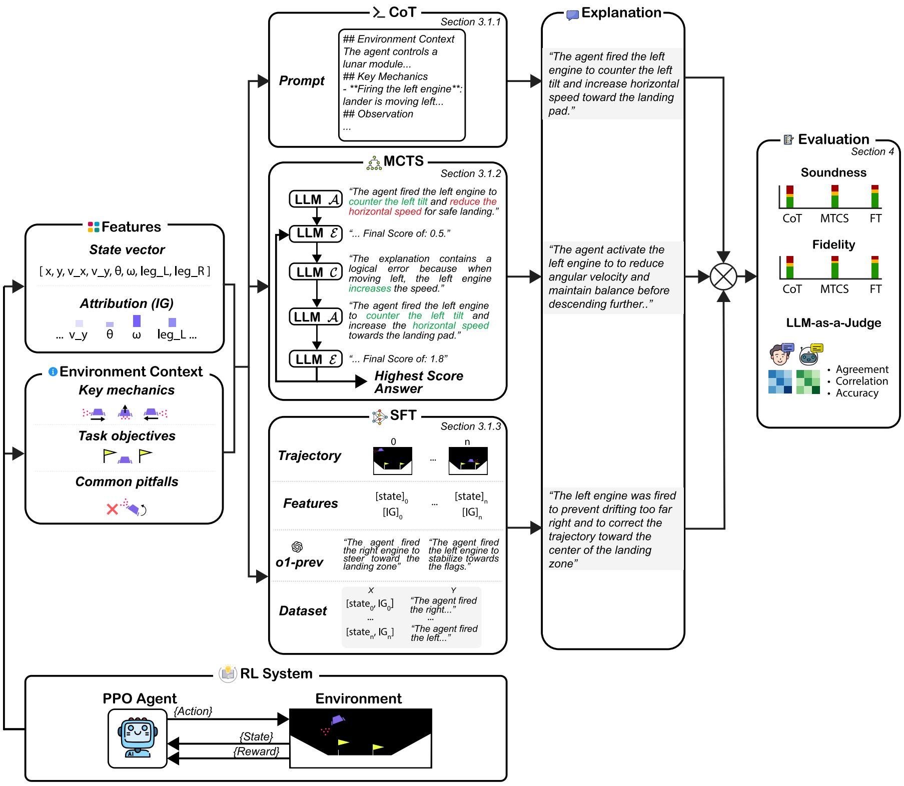
This study considers three explanation strategies that build upon one another in terms of reasoning depth and computational complexity. CoT serves as the baseline, generating single-pass textual explanations directly from the LLM. MCTS extends this baseline by introducing iterative self-critique and refinement steps, allowing the model to explore multiple reasoning paths before selecting the best explanation. Finally, SFT incorporates these reasoning patterns within the model weights, producing CoT-style explanations in a single forward pass but with improved logical alignment and fidelity. This progression from CoT to MCTS to SFT defines the core structure of the proposed strategy and clarifies how the methods complement one another. Section 3.1 details the three LLM-based approaches employed for generating and refining explanations. Section 3.2 presents the evaluation framework used to assess explanation quality.

3.1. LLM-based explanation generation approaches

Section 3.1.1 presents CoT prompting as the baseline method. Section 3.1.2 introduces the enhancement of LLMs through self-refinement, critique, and evaluation using MCTS. Finally, Section 3.1.3 describes the use of SFT and the dataset generation process.

3.1.1. Chain-of-Thought (CoT) prompting

CoT prompting is used as baseline, where the LLM is asked to explain why a specific action was chosen in a given state, using contextual cues and structured guidance. The prompt includes an overview of the environment and the agent’s goal, details on how actions affect state transitions, the specific `{state}-{action}` pair to be explained, and a feature attribution vector indicating the influence of each state feature on the decision. The prompt also incorporates instructions to help the model avoid common reasoning errors and ensure consistency with the agent’s actual behavior. An example prompt is provided in Appendix A.1.



Environment Context
The agent controls a lunar module...
Key Mechanics
- **Firing the left engine**: lander is moving left...
Observation
...

"The agent fired the left engine to counter the left tilt and reduce the horizontal speed for safe landing."

"... Final Score of: 0.5."

"The explanation contains a logical error because when moving left, the left engine increases the speed."

"The agent fired the left engine to counter the left tilt and increase the horizontal speed towards the landing pad."

"... Final Score of: 1.8"

"The agent fired the right engine to steer toward the landing zone"

"The agent fired the left engine to stabilize towards the flags."

"The agent fired the right..."

"The agent fired the left..."

"The agent fired the left engine to counter the left tilt and increase horizontal speed toward the landing pad."

"The agent activate the left engine to to reduce angular velocity and maintain balance before descending further.."

"The left engine was fired to prevent drifting too far right and to correct the trajectory toward the center of the landing zone"

Fig. 1. Methodology for generating and evaluating natural language explanations of RL agent policies based on LLM.

3.1.2. Monte-Carlo Tree Search (MCTS)

To enhance the reasoning process of LLMs beyond CoT prompting, we integrate MCTS for iterative explanation refinement. MCTS has traditionally been used in decision-making and game-playing scenarios, where a rollout simulates a sequence of environment interactions to evaluate actions. Following recent adaptations of MCTS for guiding LLM reasoning in complex generation tasks (Zhang et al., 2024), we adapt this approach for the explanation generation setting.

In our adaptation, a rollout corresponds to prompting the LLM to refine an existing explanation based on a critique of its prior version. The search begins with an initial explanation, which serves as the root of the search tree. Each node in the tree represents a possible explanation, and child nodes are generated through critique and refinement steps. Each new version is evaluated by an evaluator (in our case an LLM), which assigns a reward score based on Soundness and Fidelity, along with an auxiliary score reflecting linguistic fluency (rated from 0 to 2). This fluency bonus is included in the final reward to promote clearer and more human-readable explanations. The search proceeds by selecting nodes to expand using the Upper Confidence Bound for Trees (UCT) criterion, balancing exploration of new reasoning paths and exploitation of promising ones. After a fixed number of rollouts, the explanation with the highest aggregated reward is returned as the final output.

To formalize the components of the MCTS algorithm, let $\mathcal{A}$ denote the LLM used to generate and refine explanations, $\mathcal{C}$ denote the LLM used to generate a critique of an explanation, and $\mathcal{E}$ denote the LLM used to evaluate the quality of an explanation. These three components interact during each rollout: $\mathcal{A}$ proposes an initial explanation, $\mathcal{C}$ generates a critique, $\mathcal{A}$ uses the critique to refine the explanation, and $\mathcal{E}$ assigns a reward to guide the search process. The overall search procedure is presented in Algorithm 1. Details of the node selection, reward aggregation, backpropagation, and prompt templates used by each component are provided in Appendix B.

Algorithm 1: MCTS for Explanation Refinement

```
Input: CoT Prompt $P$, Maximum rollouts $N$
Output: Best explanation $A^*$
Initialize root node $n_0$: $n_0.answer \leftarrow \mathcal{A}(P)$;
for $i = 1$ to $N$ do
    $n \leftarrow$ SelectNode($n_0$) ; // The node with the highest UCT score

    $n.rewards \leftarrow n.rewards \cup \mathcal{E}(n.answer)$ ; // Evaluate current answer and append score to node's rewards
    $c \leftarrow \mathcal{C}(n.answer)$ ; // Generate critique
    $a' \leftarrow \mathcal{A}(n.answer, c)$ ; // Refine answer using critique
    Create child node $n'$ with $n'.answer \leftarrow a'$
    $n'.rewards \leftarrow n'.rewards \cup \mathcal{E}(n'.answer)$
    Add $n'$ as child of $n$
    Backpropagate from $n'$ to $n_0$ ; // Update visit counts from $n'$ to all parent nodes
$A^* \leftarrow$ Answer from node with highest aggregated reward in the tree
return $A^*$
```

3.1.3. Supervised fine-tuning (SFT)

We also explore the effectiveness of fine-tuning on explanation quality. To collect the data, we instantiate the environments with randomized initial states (e.g., varying positions, angles, or velocities) to obtain multiple trajectories. From these, we extract `{state}-{action}` transitions, filtering out uninformative cases (e.g., end-of-episode states where actions no longer influence behavior meaningfully). For each `{state}-{action}` pair in the dataset we compute its corresponding attribution vector, forming the core elements in the CoT prompt used in 3.1.1. All the constructed prompts are passed to an LLM to generate the corresponding explanations. After manual verification, the final fine-tuning dataset consists of 500 examples per environment, with balanced action distributions based on the available action space.

Table 1
Scoring criteria for Soundness and Fidelity.

	Score	Definition	Example
Soundness	2	The explanation accurately describes how the action affects the relevant state features, and logically connects this to the agent's objective. No factual or logical errors are present.	"*The agent fires the left engine to reduce left tilt and push the lander toward the center pad, helping it land safely.*"
	1	The explanation makes no direct contradictions but may be vague, incomplete, or weakly connected to the goal.	"*The left engine helps stabilize the lander.*" (True, but unclear how or why that helps.)
	0	The explanation includes factual errors (e.g., incorrect action effects) or contradicts the agent's objective.	"*The left engine is fired to counter right tilt.*" (Wrong effect) or "*The agent pushes left to move away from the pad.*" (Contradicts goal)
Fidelity	2	Clearly links all top contributing features to the decision.	"*The agent fires the left engine to reduce left tilt and leftward rotation.*" (Angle and Angular velocity were the most contributing features)
	1	Mentions some features but omits or vaguely addresses others.	"*The agent fires the left engine to reduce left tilt.*" (Missing a reference to the Angular velocity e.g rotation)
	0	The explanation omits all top features, mentions irrelevant ones, or is logically incorrect (i.e., Soundness = 0).	"*The decision was based on vertical speed.*" (when it had no impact).

Table 2
Agreement metrics.

Scope	Metric	Definition
Per-Explanation	Weighted Kappa	Measures how well judge LLMs and expert scores agree while accounting for chance and the degree of disagreement
	Accuracy	Percentage of explanations for which judge LLMs and expert scores exactly matched.
Per-Model	Spearman Correlation	Rank-order correlation: Measures whether judge LLMs preserved the relative ranking across models (e.g., if experts rate model X higher than Y and Z, do judge LLMs reproduce the same ordering?)
	Pearson	Tests the linear relationship between judge LLMs averages and expert averages across models.
	MAE	Average absolute difference between judge LLMs and experts scores across models.

Each training example consists of a minimalist input prompt: `{state, selected action, feature attributions}`, paired with the corresponding explanation as the output. At inference, the models are queried in a zero-shot fashion with the same input prompt used in training.

3.2. *Evaluation framework*

Sections 3.2.1 and 3.2.2 outline the qualitative and quantitative methods used to evaluate the quality of the generated explanations, along with the metrics applied in each case.

3.2.1. *Qualitative evaluation*

We adopt evaluation metrics inspired by the work of Zytek et al. (2024b) to assess the quality of LLM-generated explanations. Given that our work focuses on evaluating the ability of LLMs to generate *correct* explanations, we adopt two metrics that reflect different aspects of correctness: **(i) Soundness:** measures whether the explanation is logically correct with respect to the environment mechanics and whether it correctly explains how the action helps the agent pursue its goal in the current state; **(ii) Fidelity:** Evaluates whether the explanation emphasizes the right reasons, specifically, the features that most strongly influenced the agent's decision, as identified by the attribution vector. A high fidelity explanation must incorporate these features meaningfully into the reasoning. For instance, if angle and angular velocity had the largest contribution in the attribution vector, a high-fidelity explanation might say: "*The agent fired the left engine to correct the left tilt and rotation.*" In contrast, mentioning only one of them, or citing unrelated factors, would indicate lower Fidelity.

Table 1 summarizes the scoring criteria for each metric. Notably, Fidelity complements Soundness: while Soundness assesses whether the reasoning is logically valid and consistent with the environment's mechanics, Fidelity evaluates whether it is grounded in the correct input features. Compared to commonly used metrics such as fluency, context-awareness, or helpfulness, our chosen metrics are less subjective and can be more reliably assessed by experts, thus enabling a more consistent and objective evaluation of LLM reasoning capabilities.

3.2.2. *Quantitative evaluation*

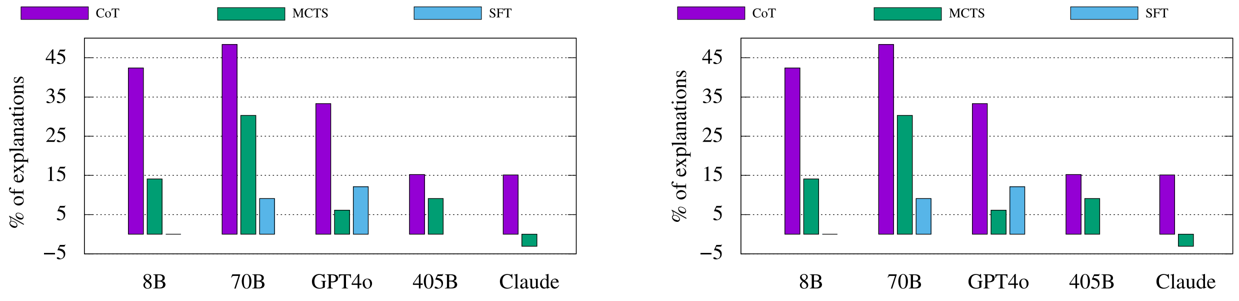
To automate the evaluation of explanations, we adopt a quantitative approach based on the *LLM-as-a-Judge* paradigm, similar to Badshah and Sajjad (2024) have proposed. In our setup, three LLMs (judges) were independently prompted to evaluate explanations based on Soundness and Fidelity. Each model provides a score per explanation and per metric; final scores are determined via majority voting, with the maximum score selected in case of a tie, the input prompt is described in Appendix A.5. The scores from the judge LLMs are then compared against manually assigned expert scores (i.e., against the qualitative evaluation defined in Section 3.2.1) to assess alignment. The level of agreement is measured using several statistical metrics, summarized in Table 2, which are grouped into two categories: **(i) Per-Explanation:** assess agreement between LLM and expert scores on an individual explanation level. They aim to answer the question: *"How similar are the individual explanation scores assigned by the judge LLMs and human experts?"*; **(ii) Per-Model:** assess how well the LLMs preserve the overall model-level trends identified by experts. They aim to answer the question: *"Does the judge LLMs reproduce similar ranking and average evaluation of models as human experts?"*. Further details on the metrics are described in Appendix E.

4. Experimental results and analyses

This section presents and discusses the experimental findings. Section 4.1 outlines the experimental setup, including the RL environments and models used. Section 4.2 reports and analyzes the results regarding the quality of the generated explanations. Finally, Section 4.3 presents the results of the LLM-as-a-Judge evaluation and analyzes their alignment with expert annotations.

4.1. *Experimental setup*

Experiments were conducted on two widely used 2D physics-based RL environments: Lunar Lander and Cartpole. **Lunar Lander** simulates a lander that must descend and touch down safely using side and main thrusters. The agent has four discrete actions: firing the left, right, or main engine, or doing nothing. Observations include the lander's



(a) Increase in unsound explanations (score 0) from Cartpole to Lunar Lander.

(b) Increase in unfaithful explanations (score 0) from Cartpole to Lunar Lander.

Fig. 2. Increase in unsound and unfaithful explanations from Cartpole to Lunar Lander across models and explanation methods. Higher bars reflect greater performance drops under increased task complexity.

position, velocities in x and y, angle, angular velocity, and leg contact indicators. The agent is rewarded for moving toward the landing pad and contacting the ground, and is penalized for fuel usage and crashing. The objective is to learn a landing policy by maximizing the cumulative reward; **Cartpole** is a classical control task where the agent balances a pole on a moving cart. It chooses between two actions: applying a left or right force to the cart. Observations include the cart's position and velocity, and the pole's angle and angular velocity. The agent receives a positive reward for each timestep that the pole remains balanced. The episode ends if the pole falls or the cart moves outside the boundaries. The objective is to maximize the number of timesteps that the pole remains balanced. In both environments, agents are trained using the Proximal Policy Optimization (Schulman et al., 2017) algorithm as implemented in Stable-Baselines3 (Raffin et al., 2021).

A diverse set of open- and closed-source LLMs is evaluated: DeepSeek R1 and o1-preview (reasoning models); GPT-4o, Claude 3.5 Sonnet and LLaMA 3.1 405B (large, general purpose models); LLaMA 3.1 8B and 70B (small to mid-scale models). The three explanation generation methods (CoT prompting, MCTS augmentation, SFT) are tested on a set of 15 representative states per environment and the scores are averaged across 3 different runs. Feature attribution vectors are computed using the Integrated Gradients algorithm, though our method is compatible with alternative attribution methods such as SHAP or LIME. Additionally, LoRA (Low-Rank Adaptation) (Hu et al., 2022) is used as the SFT strategy on LLaMA 3.1 8B and 70B. GPT-4o is also fine-tuned using OpenAI's fine-tuning API. We use OpenAI's o1-preview model to generate the explanations required to construct the fine-tuning dataset. Hyperparameters and training details are provided in Appendix C. Additionally, we run MCTS for 4 rollouts (iterations) with a greedy selection strategy of the next node based on their UCT values. The training and test times are reported in Appendix D, which shows that MCTS incurs a higher inference cost proportional to the number of rollouts, while SFT restores single-call efficiency after fine-tuning, and CoT remains the cheapest per-state method.

For the quantitative evaluation, GPT-4o, Claude 3.5 Sonnet, and DeepSeek R1 are used as the judge LLMs with majority voting to aggregate the final scores.

4.2. *Experimental results*

Section 4.2.1 analyzes the influence of the RL environment's complexity by comparing model performance between Cartpole and Lunar Lander. Section 4.2.2 analyzes the effectiveness of the three explanation generation methods. Section 4.2.3 analyzes the importance of different MCTS components and their effect on improving the explanations quality.

4.2.1. *Environment complexity*

To understand how task complexity influences the explanation quality, we compare the models performance across two environments with different degrees of complexity: Cartpole (2 discrete actions, 4-dimensional observation space) and Lunar lander (4 discrete actions 8-dimensional observation space). The change in performance is quantified by the difference in percentage of unsound explanations (Soundness = 0) under increased task complexity. Let $S^0_{\text{env}} = \left\{x \in \mathcal{D}_{\text{env}} \mid S(x) = 0\right\}$ and $\mathcal{F}^0_{\text{env}} = \left\{x \in \mathcal{D}_{\text{env}} \mid F(x) = 0\right\}$ denoting, respectively, the sets of unsound and unfaithful explanations in environment $env \in \{\text{Cartpole}, \text{LunarLander}\}$, where $S(x)$ and $F(x)$ are the Soundness and Fidelity scores assigned to explanation x. We define the change in the proportion of low-quality explanations between environments as: $\Delta_{\text{unsound}} = \frac{|S^0_{\text{Lunar}}|}{|\mathcal{D}_{\text{Lunar}}|} - \frac{|S^0_{\text{Cartpole}}|}{|\mathcal{D}_{\text{Cartpole}}|}$ and $\Delta_{\text{unfaithful}} = \frac{|\mathcal{F}^0_{\text{Lunar}}|}{|\mathcal{D}_{\text{Lunar}}|} - \frac{|\mathcal{F}^0_{\text{Cartpole}}|}{|\mathcal{D}_{\text{Cartpole}}|}$. For example, if the model generates 30 unsound explanations out of 100 in Lunar Lander and 10 out of 100 in Cartpole, then $\Delta_{\text{unsound}} = \frac{30}{100} - \frac{10}{100} = 0.2$, indicating a 20% increase in unsound explanations when moving to the more complex environment.

As observed in Fig. 2(a), most models exhibit a clear drop in performance with increasing task complexity, particularly under CoT prompting. For instance, GPT-4o, LLaMA 70B, and LLaMA 8B show respective increases of 33%, 48%, and 42% in unsound explanations. In contrast, models like o1-preview and DeepSeek R1 show no variation ($\Delta_{\text{unsound}} = 0$), indicating that this level of complexity does not significantly challenge their reasoning. While this degradation is visible across all explanation generation methods, it is most pronounced with CoT prompting and less so for MCTS and SFT, suggesting that more structured or trained approaches may mitigate the impact of complexity. Similarly, Fig. 2(b) shows that Fidelity follows a comparable trend, reinforcing the conclusion that environment complexity significantly affects explanation quality across both metrics.

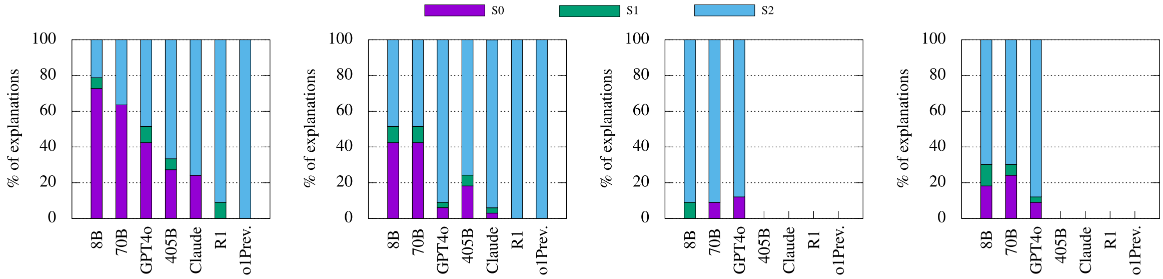
4.2.2. *Explanation generation methods*

Explanation quality can vary significantly depending on the method used to generate explanations. Table 3 presents the average Soundness and Fidelity scores for each model–method pair across both environments. To provide a more intuitive understanding of how explanation quality varies across methods, Figs. 3 and 4 and Figs. 5 and 6 provide a complementary visualization of the full distribution of explanation quality scores (0, 1, 2) for Soundness and Fidelity across models in Lunar Lander and Cartpole respectively.

CoT prompting yields the lowest overall explanation quality in both environments (see Figs. 3(a) and 5(a)), especially in Lunar Lander. Table 3 reveals that general-purpose models like GPT-4o, Claude 3.5, and LLaMA 3.1 405B reach Soundness scores as low as 1.05, 1.51, and 1.39, respectively. Fidelity scores follow a similar trend, dropping to 1.02 for GPT-4o and 1.36 for LLaMA 405B. These low values correspond to high rates of unsound and unfaithful explanations, ranging

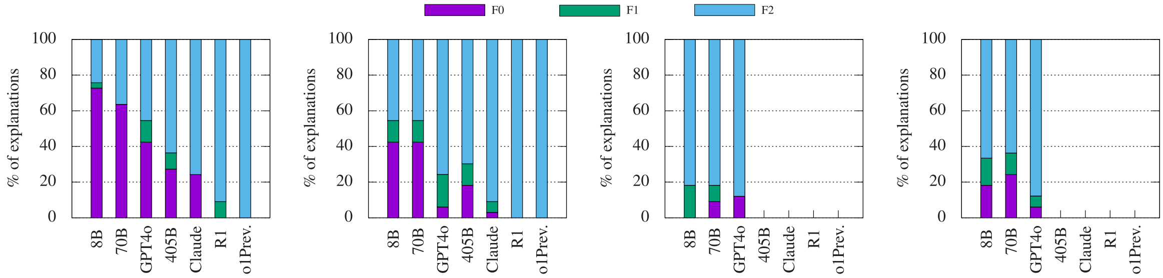
Table 3
Average Soundness (S) and Fidelity (F) scores for Lunar Lander and Cartpole explanations under CoT, MCTS, and Finetuned settings.

	Lunar Lander				Cartpole			
	CoT		MCTS		CoT		MCTS	
LLM	S	F	S	F	S	F	S	F
o1-preview	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00
Deepseek R1	1.93 ± 0.04	1.93 ± 0.04	2.00 ± 0.00	2.00± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00
Claude 3.5 Sonnet	1.51 ± 0.16	1.51 ± 0.17	1.90 ± 0.13	1.87 ± 0.11	1.81 ± 0.02	1.81 ± 0.02	1.90 ± 0.07	1.90 ± 0.07
LLaMA 3.1 405B	1.39 ± 0.18	1.36 ± 0.15	1.57 ± 0.11	1.51 ± 0.11	1.75 ± 0.08	1.72 ± 0.12	1.81 ± 0.08	1.81 ± 0.11
GPT-4o	1.05 ± 0.11	1.02 ± 0.08	1.84 ± 0.11	1.69 ± 0.04	1.78 ± 0.04	1.78 ± 0.04	1.90 ± 0.07	1.90 ± 0.07
LLaMA 3.1 70B	0.72 ± 0.04	0.72 ± 0.08	1.22 ± 0.24	1.18 ± 0.21	1.42 ± 0.02	1.42 ± 0.02	1.72 ± 0.12	1.72 ± 0.12
LLaMA 3.1 8B	0.48 ± 0.04	0.48 ± 0.04	1.00 ± 0.07	1.00 ± 0.17	1.09 ± 0.04	1.27 ± 0.04	1.27 ± 0.07	1.27 ± 0.07
Finetuned LLM	SFT		MCTS-SFT		SFT		MCTS-SFT	
GPT-4o	1.75 ± 0.08	1.75 ± 0.08	1.78 ± 0.18	1.72 ± 0.19	2.00 ± 0.00	2.00 ± 0.00	1.78 ± 0.04	1.81 ± 0.02
LLaMA 8B	1.90 ± 0.02	1.81 ± 0.01	1.48 ± 0.17	1.48 ± 0.11	2.00 ± 0.00	2.00 ± 0.00	1.84 ± 0.04	1.84 ± 0.04
LLaMA 70B	1.81 ± 0.01	1.72 ± 0.03	1.45 ± 0.07	1.39 ± 0.08	2.00 ± 0.00	2.00 ± 0.00	1.75 ± 0.02	1.75 ± 0.02



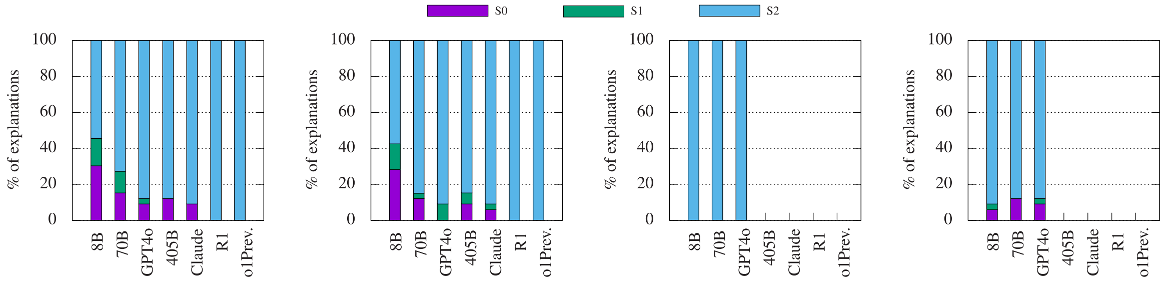
(a) CoT (b) MCTS (c) SFT (d) MCTS-SFT

Fig. 3. Distribution of Soundness scores for Lunar Lander.



(a) CoT (b) MCTS (c) SFT (d) MCTS-SFT

Fig. 4. Distribution of Fidelity scores for lunar lander.

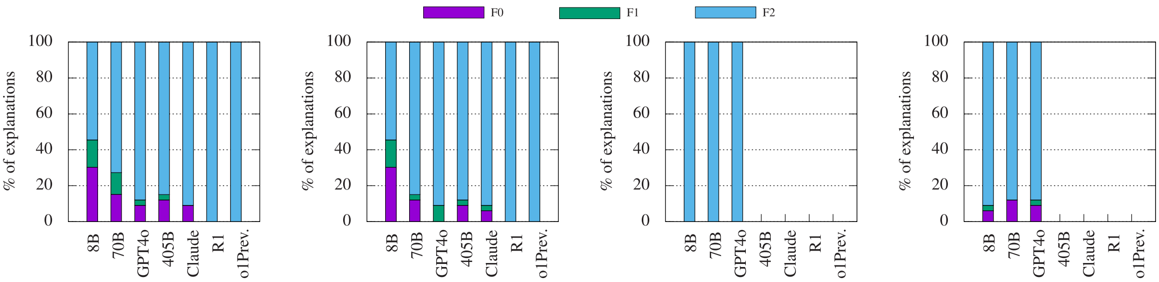


(a) CoT (b) MCTS (c) SFT (d) MCTS-SFT

Fig. 5. Distribution of Soundness scores for Cartpole.

from 27% to 42% as shown in Figs. 3(a) and 4(a). This underscores CoT's limitations in environments with complex dynamics, unlike Cartpole where scores were noticeably higher due to its simpler mechanics (see Figs. 5(a) and 6(a)). Smaller models, such as LLaMA 8B and 70B, performed worse under CoT across both tasks, with Soundness and Fidelity dropping to 0.48 and 0.72 (see Table 3), and error rates exceeding 63% in Lunar Lander (see Figs. 3(a) and 4(a)).

MCTS improves explanation quality for large models, reducing error rates in both Soundness and Fidelity to under 7% for GPT-4o and Claude 3.5 (see Figs. 3(b)–4(b) and Figs. 5(b)–6(b)). However, its effect is limited for LLaMA 8B and 70B with average Soundness scores remaining low (1.00 and 1.22 respectively shown in Table 3), likely due to their limited capacity to critique and evaluate explanations effectively, which is a crucial step in the MCTS process.



(a) CoT (b) MCTS (c) SFT (d) MCTS-SFT

Fig. 6. Distribution of Fidelity scores for Cartpole.

Table 4
Impact of varying MCTS components (Evaluator and Critic) on explanation quality in Lunar Lander and Cartpole.

Model	Configuration	Lunar Lander		Cartpole	
		Soundness	Fidelity	Soundness	Fidelity
LLaMA 8B	Same LLM	1.00 ± 0.07	1.00 ± 0.17	1.27 ± 0.07	1.27 ± 0.07
	+ R1 Evaluator	1.45 ± 0.02	1.45 ± 0.04	1.54 ± 0.02	1.36 ± 0.02
	+ R1 Critic	1.36 ± 0.07	1.27 ± 0.07	1.36 ± 0.04	1.36 ± 0.04
	+ R1 Evaluator & Critic	**2.00** ± 0.00	**1.83** ± 0.04	**1.81** ± 0.04	**1.90** ± 0.04
LLaMA 70B	Same LLM	1.22 ± 0.24	1.18 ± 0.21	1.72 ± 0.12	1.72 ± 0.12
	+ R1 Evaluator	1.90 ± 0.04	1.90 ± 0.04	1.81 ± 0.04	1.81 ± 0.04
	+ R1 Critic	1.27 ± 0.21	1.36 ± 0.21	1.63 ± 0.12	1.63 ± 0.12
	+ R1 Evaluator & Critic	**1.90** ± 0.04	**2.00** ± 0.00	**2.00** ± 0.00	**2.00** ± 0.00
GPT-4o	Same LLM	1.84 ± 0.11	1.69 ± 0.04	1.90 ± 0.07	1.90 ± 0.07
	+ R1 Evaluator	2.00 ± 0.00	1.81 ± 0.04	2.00 ± 0.00	2.00 ± 0.00
	+ R1 Critic	1.81 ± 0.08	1.81 ± 0.04	1.81 ± 0.07	1.81 ± 0.07
	+ R1 Evaluator & Critic	**2.00** ± 0.00	**2.00** ± 0.00	**2.00** ± 0.00	**2.00** ± 0.00

SFT proves most beneficial for LLaMA 8B and 70B. Due to their relatively small size, LoRA fine-tuning on a task-specific dataset significantly outperforms their CoT and MCTS variants, raising the rate of sound and faithful explanations above 80% in Lunar Lander (see Figs. 3(c)–4(c)) and reaching perfect scores (2.0) for both metrics in Cartpole (see Table 3). In contrast, GPT-4o gains little from SFT and even underperforms its MCTS variant in Lunar Lander. Given the model's scale and the dataset's small size, SFT appears largely unnecessary for Cartpole and only marginally effective for Lunar Lander.

MCTS-SFT (Figs. 3(d)–4(d) and Figs. 5(d)–6(d)) show that applying MCTS after fine-tuning consistently *decreases* performance- *in both environments*. Once the model has learned a direct mapping from `state + attributions` to explanation, additional refinement via external prompts (unseen during training) tends to introduce noise rather than improvements. Meanwhile, reasoning models such as o1-preview and R1 achieve near-perfect scores under both CoT and MCTS (see Table 3), showing minimal sensitivity to the method used or environment, with MCTS refining the small mistakes made by R1.

In summary, our experiments show that explanation scores are consistently higher in Cartpole than in Lunar Lander, reflecting Cartpole's lower complexity. Yet, the same overall pattern emerges in both environments: larger models gain the most from MCTS, while smaller ones benefit most from SFT.

4.2.3. MCTS component variations

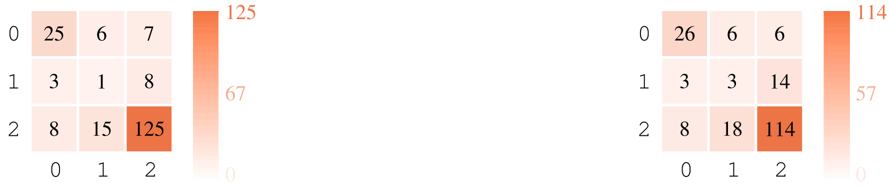
In addition to evaluating MCTS effectiveness, we study the importance of its components, specifically the *Critic* and *Evaluator*, to the quality of generated explanations. In the previous experiments, all three MCTS components: Generator/Refiner ($\mathcal{A}$), Critic ($\mathcal{C}$), and Evaluator ($\mathcal{E}$) were instantiated with the same LLM. In this set of ablation experiments, we isolate the effect of upgrading the $\mathcal{C}$ and $\mathcal{E}$ (using a stronger LLM instead) while keeping $\mathcal{A}$ fixed. Four MCTS configurations are evaluated for each model (LLaMA 3.1 8B, LLaMA 3.1 70B, and GPT-4o), using R1 as the upgraded LLM for $\mathcal{C}$ and $\mathcal{E}$:

1. Same LLM: all components use the same LLM;
2. Strong Evaluator: only the Evaluator is replaced with R1;
3. Strong Critic: only the Critic is replaced with R1;
4. Strong Evaluator & Critic: both components are replaced with R1.

Table 4 summarizes the performance changes across the two metrics. The results show that swapping the Evaluator with R1 leads to large improvements, especially for smaller models. In Lunar Lander, Soundness for LLaMA 8B rises by +45% and for LLaMA 70B by +56%, while Fidelity increases by a similar margin. In Cartpole, the gains are smaller but still noticeable (e.g., +21% in soundness for 8B). Additionally, having R1 as an Evaluator helps GPT-4o closing the small gap needed to reach perfect Soundness score (2), confirming the high impact of having a strong LLM as an Evaluator. Replacing the Critic only results in modest and sometimes even reduces performance compared to the baseline configuration. This is likely because, even with valuable feedback from a stronger Critic, a weak Evaluator can still steer the search toward suboptimal candidates. This interpretation is supported by the results of the final configuration, where upgrading both components to R1 lead to the best overall performance, with all models reaching perfect or near-perfect scores.

4.3. LLM-as-a-judge evaluation

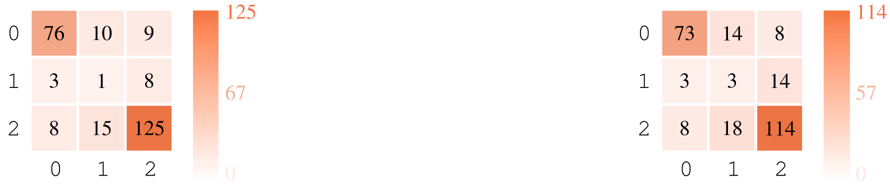
To understand how well LLMs can evaluate explanations, we compare their ratings with those of our manual evaluation for the Lunar Lander environment. The results of the quantitative evaluation, given in Table 5, indicate that LLMs as judges approximate manual evaluations reasonably well, both at the individual explanation and model level. Weighted Kappa scores (0.627 for Soundness, 0.611 for Fidelity) and an accuracy of over 72% indicate high agreement. At the model level, Spearman correlations (0.824 for Soundness, 0.765 for Fidelity) indicate that LLM judges preserve ranking and performance trends observed manually, while Pearson correlations (0.838 for Soundness, and 0.873 for Fidelity) show strong linear correlation in average scores. These findings are further supported by the low MAE values (0.212 for Soundness, 0.197 for Fidelity).



(a) Confusion matrix for Soundness. Predictions (y-axis) are LLM-as-a-Judge scores, ground truth (x-axis) is expert annotations.

(b) Confusion matrix for Fidelity. Predictions (y-axis) are LLM-as-a-Judge scores, ground truth (x-axis) is expert annotations.

Fig. 7. Confusion matrices for Soundness and Fidelity.



(a) Confusion matrix for Soundness. Predictions (y-axis) are LLM-as-a-Judge scores, ground truth (x-axis) is expert annotations.

(b) Confusion matrix for Fidelity. Predictions (y-axis) are LLM-as-a-Judge scores, ground truth (x-axis) is expert annotations.

Fig. 8. Confusion matrices for Soundness and Fidelity after data augmentation.

Table 5
Per-explanation and Per-model agreement between expert and LLM-judge evaluations.

Scope	Metric	Soundness	Fidelity
Per-Explanation	Weighted Kappa	0.627	0.611
	Accuracy	0.763	0.722
Per-Model	Spearman	0.824	0.765
	Pearson	0.838	0.873
	MAE	0.212	0.197

Table 6
Per-class precision, recall, and F1 scores comparing LLM-judge predictions to expert labels for Soundness and Fidelity.

Class	Soundness				Fidelity			
	Precision	Recall	F1	Support	Precision	Recall	F1	Support
0	0.69	0.66	0.68	38	0.70	0.68	0.69	38
1	0.05	0.08	0.06	12	0.11	0.15	0.13	20
2	0.89	0.84	0.87	148	0.85	0.81	0.83	140
Macro-F1	0.53				0.55			

4.3.1. *Class imbalance & error analysis*

Since our goal is to improve the explanation quality and our experiments showed that both MCTS and SFT improved upon CoT, the results from our manual validation was heavily skewed toward score 2 in 70%–75% of all cases:

$$\text{Soundness: } \{0 = 38,\ 1 = 12,\ 2 = 148\},$$

$$\text{Fidelity: } \{0 = 38,\ 1 = 20,\ 2 = 140\}.$$

This class imbalance inflates overall accuracy, as LLM judges that overpredict score 2 appear more correct simply due to the prevalence of high-scoring cases. To assess performance beyond raw accuracy, we examine confusion matrices (Fig. 7) and per-class precision/recall (Table 6).

Table 6 reports that LLM judges perform reliably when identifying sound explanations (score 2), achieving high precision (≈ 0.85–0.89) and recall (≈ 0.84), correctly classifying 125 out of 148 sound explanations (Fig. 7(a)) and 114 out of 148 faithful ones (Fig. 7(b)). However, only 1 out of 12 borderline cases (score 1) are correctly classified for Soundness, and 3 out of 20 for Fidelity, resulting in a recall ranging only between 8 and 15%. Notably, recall for unsound explanations is 66%, with over 18% of unsound explanations being confidently misclassified as sound. This overconfidence raises concerns about reliability in unsound cases.

To further investigate the overconfidence problem of the judge LLMs, we augmented our evaluation set with more unsound explanations (class 0). Specifically, we take originally correct explanations and introduce logical mistakes and reasoning errors that LLMs frequently make. Fig. 8 shows the new confusion matrices. For Soundness (Fig. 8(a)), out of 95 ground-truth unsound explanations, 76 are correctly classified as class 0, yielding a recall of 80%. The remaining misclassifications are split between 10 predicted as borderline and 9 predicted as sound. With the same trend being observed for Fidelity (Fig. 8(b)), this improvement suggests that LLMs can, in fact, exhibit a degree or reliability in detecting unsound and unfaithful explanations with an improved recall (≈ 80%) under a more balanced evaluation set.

4.3.2. *Qualitative analysis of misclassified score-1 explanations*

We analyzed explanations that were rated as partially sound or faithful (score 1) by human experts but misclassified by the judge LLMs. The goal was to understand why the judges failed to recognize partial correctness. Two main patterns were identified: (1) misinterpretation caused by differences in wording, and (2) overlooking vague or generic phrasing when the explanation was otherwise fluent.

Case 1: Misinterpretation due to wording differences. In several cases, the judge LLMs marked explanations as incorrect even though they were logically consistent with the mechanics of the environment. This happened when certain words were interpreted too literally. For example, the judges argued that phrases such as “assists rightward movement” contradicts the action effect that is “increases rightward movement” or that “corrects its right tilt” is the opposite of “counteracts the right tilt” even though they mean the same thing. Human evaluators naturally understood these as describing the correct physical effect, but the LLMs relied strictly on the wording provided in the environment description. These errors show that small differences in phrasing can lead to wrong judgments, particularly when the model lacks a flexible understanding of how actions influence motion and when its learned embeddings are not strong enough to capture semantic similarities. This reflects a **semantic misalignment** between the wording of the explanation and the specific terminology presented in the evaluation prompt.

Case 2: Partial correctness dominates vagueness. In these cases, explanations that included some accurate and well-phrased action–effect links were rated as fully correct (score 2) by the judge LLMs, even when other effects were described vaguely or omitted altogether. For example, an explanation might clearly state that "the action reduces horizontal speed to avoid overshooting the landing pad", a correct and specific observation while also including more generic phrases such as "the action stabilizes rotation". Human experts assigned these a score of 1 due to the incomplete coverage and lack of causal detail in parts of the explanation (How does the action stabilize rotation). However, the LLM judges appeared to be strongly influenced by the presence of a few correct effects and the overall fluency of the explanation. This led them to overlook the missing or imprecise reasoning regarding other important factors (e.g., vertical velocity, tilt angle). A similar pattern emerged in the fidelity evaluation, where LLMs inferred overall faithfulness based on the presence of some correct features, even when other influential variables were not mentioned. These findings suggest that LLM judges are susceptible to a fluent partial correctness bias, rewarding explanations for their clarity and partial accuracy, even if they fail to provide a complete and causally grounded justification.

Validation and further observations. To validate these observations, we modified the misclassified explanations and re-evaluated them. For *Case 1*, we replaced the ambiguous terms with the exact expressions used in the environment description. For *Case 2*, we removed the correct action–effect statements and kept only the vague parts of the explanations. After applying these two changes for the class 1 explanations, the judges' performance improved markedly: None of the Score-1 explanations were misclassified as unsound (0% false negatives), 85% were correctly classified as Score 1, and only 15% were over-estimated as Score 2, confirming that the observed errors were primarily driven by wording interpretation and fluency bias rather than misunderstanding of the reasoning itself.

To further examine the wording issue (Case 1), we compared the results across individual judge models. When the judge LLM belonged to the same family as the model that generated the explanation (for example, OpenAI model judging an OpenAI explanation or Claude judging a Claude explanation), the misinterpretation problem largely disappeared. This suggests that models from the same family may share similar linguistic patterns and embeddings, allowing them to interpret wording in a consistent way. However, since our evaluation relied on majority voting across three distinct judge LLMs, the disagreement introduced by the other two (typically from different providers) was enough to cause misclassifications. This finding indicates that cross-model differences in language representation can directly influence judgment consistency, especially for explanations that depend on subtle wording or stylistic nuances.

5. Lessons learned and limitations

Our study highlights that explanation strategies must be carefully aligned with both model capacity and deployment context. Two main lessons emerge:

- **Large models + MCTS:** It consistently delivers higher-quality explanations, particularly for larger models. However, its substantial computational cost limits applicability in real-time or resource-constrained settings;
- **Small models + SFT:** It offers an efficient and cost-effective alternative, showing strong gains for smaller models. Its main limitation is the dependence on high-quality training data.

These findings suggest that MCTS and SFT can be viewed as complementary strategies: one maximizes quality but at high computational expense, while the other emphasizes efficiency but relies on data availability. It is important to note that our work focuses exclusively on explanation quality. In practice, other dimensions, such as inference speed, energy consumption, and scalability, are equally relevant and should be part of future evaluations. Additionally, applying the proposed evaluation framework to real-world domains represents an important next step. While the framework operates independently of the environment or agent architecture and can be directly applied to real-world RL systems without architectural modification, doing so effectively requires a sufficient understanding of the environment's mechanics, agent objectives, and the overall context as highlighted in Fig. 1. This knowledge is essential both for constructing prompts that guide the LLM's reasoning (see "Environment Context" stage) and for evaluating explanation quality based on logical correctness (as summarized in Table 1). Supporting such domain transfer remains a key direction for future work. Beyond these specific findings, our results reveal a broader principle about the interplay between reasoning strategy and model capacity: explanation quality improves when the chosen strategy leverages the model's inherent strengths. Larger models benefit from search-based reasoning that refines intermediate thoughts (as in MCTS), while smaller ones require compact, data-driven alignment (as in SFT). This insight extends beyond our experiments and could inform the design of adaptive explanation pipelines in future XRL systems, where reasoning complexity, computational efficiency, and interpretability are jointly optimized. Similarly, while the use of "LLM-as-a-Judge" provides a scalable and promising approach to evaluation, challenges remain, especially in borderline cases where soundness or fidelity is difficult to assess. More standardized evaluation protocols will be essential to ensure robustness and reproducibility in future work.

Despite the insights provided, our experiments come with some limitations:

1. **Scope of environments:** We evaluated only two relatively simple environments (Cartpole and Lunar Lander). Although they pose non-trivial challenges for explanation generation for LLMs, they remain limited in observation space, dynamics, and task complexity. The generalizability of our conclusions to vision-based, multi-agent, or higher-dimensional tasks remains untested;
2. **Parameter tuning:** We did not perform extensive hyperparameter sweeps for MCTS (e.g., rollouts, exploration constant). Systematic tuning would likely influence results but was infeasible given the annotation burden of manual evaluation;
3. **Fine-tuning data scale:** Our SFT results, though promising, relied on relatively small datasets (500 examples per environment). This restricts claims regarding scalability and generalization to out-of-distribution states;
4. **Evaluation consistency:** Both human annotators and LLM judges face difficulties in ambiguous or partially correct cases. This introduces potential noise into the scoring process and highlights the broader challenge of achieving consistent evaluation in explainable reinforcement learning.

6. Conclusion and future work

In this work, we conducted a comprehensive evaluation of LLMs as both explanation generators and evaluators for DRL agents. We benchmarked several models under CoT prompting, enhanced them with MCTS, and fine-tuned a subset on a small dataset. Our results showed that CoT prompting alone is often insufficient for producing accurate and reliable explanations, particularly as environment complexity increases. MCTS substantially boosts performance for larger models, leveraging their stronger evaluation and critique capabilities, while smaller models benefit more from fine-tuning, even with limited data. Further analysis underscored the central role of the Evaluator in MCTS, where robust evaluation guidance drove improved explanation quality. We also investigated automated evaluation via the quantitative "LLM-as-a-Judge" approach. Results demonstrated strong agreement with manual evaluations, especially in model ranking and average scoring, paving the way for scalable and automated assessment pipelines in explainability research. Overall, our findings highlighted both the potential and the current limitations of LLMs for XRL, emphasizing the need to align explanation methods with model capacity and deployment constraints.

We plan to extend this work toward richer and more challenging domains (e.g., vision-based or multi-agent environments) to uncover new reasoning challenges and further test the proposed methods. Another promising direction integrates policy summarization with LLMs, enabling explanations of entire agent policies rather than individual states. On the evaluation side, we aim to develop more advanced prompting and evaluation techniques. Recent work on multi-step adaptive prompting (de Souza Loureiro et al., 2025) for automatic text evaluation shows how augmenting the standard CoT prompt with self-reflection to iteratively modify the original prompt can improve evaluation robustness, suggesting that similar strategies could be applied to judge models for handling ambiguous or partially correct explanations. Additionally, checklist-based evaluation frameworks (Lee et al., 2024) decompose the evaluation prompt into structured multi-criteria checks, guiding the judge to assess causal correctness, completeness, and fidelity in a more systematic way. Moreover, the analysis in Section 4.3.2 also showed that misinterpretation errors decreased when the judge and generator belonged to the same model family, suggesting that future evaluations could adopt weighted or priority voting schemes that account for linguistic alignment between models. Combining these techniques with refined scoring criteria and improved metrics could make LLM-based evaluation more transparent, consistent, and aligned with human judgment, helping to close the current gap in explanation assessment within XRL.

CRediT authorship contribution statement

Ayoub Belouadah: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Marcelo Luis Ruiz-Rodríguez:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Sylvain Kubler:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Yves Le Traon:** Conceptualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded in whole or in part by the Luxembourg National Research Fund (FNR), grant reference 16756339. For the purpose of open access, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

Appendix A. Prompt templates

This appendix includes the prompt templates used for our experiments, including CoT prompting, as well as the prompts used in the MCTS components (Evaluator, Critic, and Refiner).

A.1. Chain-of-Thought prompt (lunar lander example)

```
## Environment Context
The agent controls a lunar module in the Lunar Lander
     environment.
The goal is to land safely on a designated pad at the
     center. The agent was trained using the PPO
     algorithm,
optimizing the policy to maximize expected rewards.
A positive horizontal speed means that the lander is
     moving to the right.
A negative horizontal speed means that the lander is
     moving to the left.

## Key Mechanics
– **Firing the left orientation engine**:
    – If the lander is moving left (negative horizontal
         speed): Increases leftward movement.
    – If the lander is moving right (positive
         horizontal speed): Counteracts rightward
         movement.
    – Counteracts the **negative** tilt and **negative
         ** angular velocity.

## Common Pitfalls
– **Pitfall 1**: Assuming that firing the left
     orientation engine counter the positive tilt and
     the positive angular velocity.

## Observation
– {state}
– Action taken: {action}

## Features attribution:
{feature_attribution}

## Instructions
1. Analyze the current state: Consider the lander's
     position, speed, angle, and ground contact.
2. Analyze the feature contributions vector.
3. Determine the lander's position relative to the pad
     and its current movement direction.
4. Identify the lander's tilt direction and any
     rotation.
5. Explain why the agent chose the action {
     action_decode[action]} in this state.
   – Address how the action affects:
     a) Horizontal position and speed
     b) Vertical position and speed
     c) Angle and angular velocity
   – And how the action helps with the goal of reaching
         the pad and landing safely
6. **Review your explanation**:
   – Ensure consistency and logical flow and alignment
        with the feature attribution vector.
   – **Double–check that you have not violated any
        environment mechanics or fallen into common
        pitfalls.**

## Final Response
Provide a clear and concise explanation of the agent's
     decision. Focus on the final explanation only.
```

A.2. Critic prompt

```
Given the current observation: {},
The environment mechanics: {},
And the common pitfalls: {}
```

```
### Instructions:
Provide a detailed and constructive critique of the
     provided answer focusing on the following aspects:

1. Logical Consistency
   - Identify any logical inconsistencies or errors in
        reasoning.

2. Factual Accuracy
   - Point out any factual inaccuracies based on the
        environment mechanics provided.

3. Clarity and Completeness
   - Assess whether the explanation is clear and covers
        all necessary aspects without ambiguity.

Guidelines:
- Be Specific: Reference the exact parts of the answer
     that need improvement.
- Avoid Generalizations: Provide precise feedback
     rather than vague statements.
- Stay Objective: Focus solely on the content of the
     answer without critiquing the action choice.
- Do not critique the action itself or suggest another
     action.
- Use Structured Format under each heading for
     consistency.
```

A.3. Refinement prompt

```
# Instruction
Refine the answer based on the critique. Your refined
     answer should be a direct and concise solution to
     the problem.

## Additional guidelines
- Do not refer to or discuss the criticisms.
- Do not repeat the problem statement.
- Respond with only the answer.
```

A.4. Evaluator prompt

```
You are an evaluator specialized in assessing
     explanations based on specific environment
     mechanics and common pitfalls.
Your task is to assign an integer score to the provided
      answer based on its correctness and alignment
     with the environment mechanics.

Scoring system:
1) Soundness
- 2: The answer is completely correct, free from any
     logical inconsistencies or common pitfalls.
- 1: The answer is logically correct but contains some
     misleading or vague statements
- 0: The answer contains one or more common pitfalls or
       violates the mechanics.

2) Fidelity
- 2: The answer explicitly mentions at least the two
     most positively contributing features
- 1: The answer mentions only one positively
     contributing feature among and ignores the rest.
- 0: The answer fails to mention any positively
     contributing features or is logically incorrect (
     Soundness = 0)
```

```
Current observation: {}
Environment mechanics: {}
Common pitfalls: {}

Instructions:
1. Assess the provided answer based on the Environment
     Mechanics and Pitfalls.
2. Assign a score for both Soundness and Fidelity

Additional scoring:
- If the answer is correct, evaluate its fluency (score
     0 to 2).
- Add the score to the final score.

Your output format:
## Explanation: (your explanation)
## Soundness:
## Fidelity:
## Fluency:
## Final Score: Soundness + Fidelity + Fluency
```

A.5. LLM-as-a-judge prompt

```
## Task:
You are evaluating the quality of an explanation for a
     reinforcement learning agent's action.
Two aspects are evaluated: **Soundness** and **Fidelity
     **.

### Soundness:
    - 2: Fully consistent and logically correct.
    - 1: Logically correct but has a misleading or
        unclear statements.
    - 0: Incorrect or contradicts atleast one action
        effect or misinterpret the current observation
        .

### Fidelity:
    - 2: Closely reflects the agent's actual reasoning
        and action effects.
    - 1: Partially reflects the reasoning but with some
        noise or missing attributions.
    - 0: Misrepresents the agent's decision-making or
        states wrong reasons or logically incorrect (
        soundness = 0).
    ---
## Observation:
    {current_observation}
## Action:
    {action}
## Action Effects:
    {action_effect}
## Feature attributions
    {attributions_vector}
## Explanation:
    {explanation}

## Question:
Evaluate this explanation and assign:
    - Soundness (0/1/2):
    - Fidelity (0/1/2):

Remember to keep you reasoning simple and efficient,
     and respect the output format

## Output format:
Please respect this output format as your final output.
      Your output is going to be parsed automatically
     so please respect the following format:
```

```
    {{
        "soundness": ,
        "fidelity": ,
        "explanation": briefly explain why you assigned
            these scores
    }}
```

Appendix B. Implementation details of MCTS

Our implementation of MCTS is inspired from Graham (2025). While the overall search process is described in the pseudo-code in Section 3.1.2, here we elaborate on the key internal mechanisms of the algorithm, including node selection, reward aggregation, and backpropagation.

Node representation and expansion

Each node in the tree corresponds to a candidate explanation generated by the language model. A node stores its textual answer, reward samples, current average quality score Q, visit count, and links to its parent and children. When a node is selected for expansion, the language model generates a refined explanation based on a critique of its parent. This new explanation is encapsulated as a child node.

Reward aggregation

Each node aggregates rewards assigned by the Evaluator model. We compute the node quality score Q as the average between its mean and minimum reward across all evaluations:

$$Q = \frac{\text{mean}(r_1, \dots, r_n) + \min(r_1, \dots, r_n)}{2}$$

This formulation encourages not only average performance but penalizes explanations with low-quality outliers, promoting robustness.

Node selection strategy

At each iteration, the algorithm selects a node for expansion from a pool of non-fully-expanded candidates. A node is considered fully expanded if it has reached the maximum number of children or if any of its children already outperform it in terms of Q.

Selection is guided by the Upper Confidence Bound for Trees formula:

$$\text{UCT}(n) = Q(n) + c \cdot \sqrt{\frac{\log(N)}{n_v + \varepsilon}}$$

where $Q(n)$ is the node's score, N is the number of visits to its parent, n_v is the number of visits to the node, and c is an exploration constant.

We support multiple selection policies:

- **Greedy**: Select the node with the highest UCT value.
- **Importance Sampling**: Sample nodes probabilistically, with weights proportional to their UCT scores.
- **Pairwise Importance Sampling**: Sample pairs of candidates and select the node with the higher UCT in each pair, weighted by their score difference.

Backpropagation

Backpropagation starts when a new explanation is generated and evaluated to update all parent nodes in the tree. However, unlike classical MCTS used for planning, we do not update the Q-values of the parents. In our setup, each node represents a different explanation, and the quality of a child explanation does not mean that the parent explanation is correct. Updating parent Q-values would wrongly make earlier explanations look better than they actually are, which could mislead the search process. Therefore, we only propagate visit counts, making sure that node selection is guided by exploration and not by incorrect reward estimates.

Initialization

The search is initialized with a root node generated under the CoT prompt. The search then proceeds for a fixed number of rollouts, gradually building a tree of increasingly refined explanations.

Final selection

After all rollouts are complete, the answer associated with the node of highest Q score across the tree is returned as the final explanation.

Appendix C. SFT configuration

To improve the explanation performance of smaller language models, we applied supervised fine-tuning using Low-Rank Adaptation (LoRA) with the `Unsloth` library and Hugging Face's tooling. The LLaMA 3.1 8B model was fine-tuned using a LoRA rank of 32 and an alpha value of 16, with a batch size of 8, a learning rate of 2e−4, and trained for 10 epochs using mixed-precision training (fp16 or bf16). For the larger LLaMA 3.1 70B model, we used a reduced LoRA rank of 8 to accommodate its size, and loaded it in 4-bit quantized format. This model was trained for 8 epochs with a batch size of 4 and the same learning rate.

Additionally, we fine-tuned OpenAI's GPT-4o using their API with 6 training epochs, a batch size of 1, and a learning rate multiplier of 2. Due to the model's proprietary architecture, only the high-level training parameters could be configured.

In all cases, we used a dataset of 500 explanation examples, split into 400 for training and 100 for validation. Each input consisted of a state description and a feature attribution vector, with the corresponding explanation as the output. The models were trained using next-token prediction.

Appendix D. Computational cost analysis

Unit definition

To provide a hardware- and vendor-agnostic comparison of computational efficiency, we express the runtime of each method in units of a single LLM forward pass, denoted as x. Here, x represents the median wall-clock time required for one prompt–completion inference call under the typical token budget used in our experiments.

Table D.7
Training time for LoRa supervised fine-tuning across different LLMs.

Model	Training hardware	Training time
LLaMA 3.1 (8B)	1 × A100 40 GB	~45 min
LLaMA 3.1 (70B)	1 × A100 40 GB	~3.5 h
GPT-4o	API fine-tuning endpoint	~1 h

Per-state computational cost

CoT: CoT prompting requires a single LLM call per state:

$$T_{\text{CoT}} = x.$$

This serves as the reference cost for comparison.

MCTS: Each rollout in the MCTS framework invokes four LLM calls: a generator, a critic, an evaluator, and a refiner (which reuses the generator). With B rollouts, the total per-state computational cost is therefore:

$$T_{\text{MCTS}} = 4Bx,$$

and the number of calls per state is $4B$.

SFT: For SFT, we report the actual training wall-clock time in Table D.7, which depends on the dataset size, batch size, and number of epochs parameters reported in Appendix C. Inference after fine-tuning requires only a single LLM call per state:

$$T_{\text{SFT-inf}} = x.$$

Discussion

This formulation provides a direct comparison of computational trade-offs between methods. MCTS improves explanation quality at a cost proportional to the number of rollouts ($4Bx$), whereas SFT restores single-call efficiency after an initial fine-tuning phase. Reporting runtime in units of x abstracts away provider-dependent latency and allows for transparent comparison across models and settings.

Appendix E. Agreement & correlation metrics

Let $y_i \in \{0, 1, 2\}$ be the expert score for explanation i (Soundness or Fidelity), and $\hat{y}_i$ the judge score after majority vote across the three LLM judges (ties resolved by the maximum). We report:

Per-explanation metrics. **Accuracy** is the exact-match rate: Acc $= \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[\,\hat{y}_i = y_i\,]$. **Weighted Cohen's κ** treats the scale as ordinal with K=3 categories. Let O_{ij} be the observed frequency in cell (i, j) and E_{ij} the expected frequency from marginals. With quadratic weights we have:

$$w_{ij} = \frac{(i-j)^2}{(K-1)^2}, \qquad \kappa_w = 1 - \frac{\sum_{i,j} w_{ij}\, O_{ij}}{\sum_{i,j} w_{ij}\, E_{ij}}.$$

(We use quadratic weights by default; linear weights $w_{ij} = \frac{|i-j|}{K-1}$ give similar trends.)

Per-model metrics. For each model m, we compute the mean expert score $\bar{y}_m$ and mean judge score $\bar{\hat{y}}_m$ across its explanations. We then report:

$$\text{MAE} = \frac{1}{M} \sum_{m=1}^{M} \left| \bar{\hat{y}}_m - \bar{y}_m \right|.$$

Pearson's r between $(\bar{y}_m)$ and $(\bar{\hat{y}}_m)$:

$$r = \frac{\sum_m (\bar{y}_m - \bar{y}_\cdot)(\bar{\hat{y}}_m - \bar{\hat{y}}_\cdot)}{\sqrt{\sum_m (\bar{y}_m - \bar{y}_\cdot)^2}\sqrt{\sum_m (\bar{\hat{y}}_m - \bar{\hat{y}}_\cdot)^2}}.$$

Where $\bar{y}_\cdot = \frac{1}{M} \sum_{m=1}^{M} \bar{y}_m$ and $\bar{\hat{y}}_\cdot = \frac{1}{M} \sum_{m=1}^{M} \bar{\hat{y}}_m$ denote the across-model means of the expert and judge scores, respectively.

Spearman's ρ (rank correlation). Let $R_m = \operatorname{rank}(\bar{y}_m)$ and $S_m = \operatorname{rank}(\bar{\hat{y}}_m)$, where $\operatorname{rank}(\cdot)$ assigns *fractional (average) ranks* in $\{1, \ldots, M\}$ with 1 = smallest value. Then

$$\rho = \frac{\sum_{m=1}^{M} (R_m - \bar{R})\,(S_m - \bar{S})}{\sqrt{\sum_{m=1}^{M} (R_m - \bar{R})^2}\,\sqrt{\sum_{m=1}^{M} (S_m - \bar{S})^2}}.$$

Where $\bar{R} = \frac{1}{M} \sum_{m=1}^{M} R_m = \frac{M+1}{2}$ and $\bar{S} = \frac{1}{M} \sum_{m=1}^{M} S_m = \frac{M+1}{2}$.

Data availability

Data will be made available on request.

References

Abu-Rasheed, H., Weber, C., & Fathi, M. (2024). Knowledge graphs as context sources for LLM-based explanations of learning recommendations. In *2024 IEEE global engineering education conference* (pp. 1–5). IEEE.

Ahmad, W., Shahzad, A. R., Amin, M. A., Bangyal, W. H., Alahmadi, T. J., & Khan, S. H. (2025). Machine learning driven dashboard for chronic myeloid leukemia prediction using protein sequences. *PLoS One*, *20*, Article e0321761.

Ali, A., Xia, Y., Zia, M. F., Bangyal, W. H., & Iqbal, M. (2025). Trustworthy load forecasting with generative AI: A dual-attention convlstm and VAE-based approach. *IEEE Transactions on Consumer Electronics*.

Ameur, M., Brik, B., & Ksentini, A. (2024). Leveraging LLMs to explain DRL decisions for transparent 6G network slicing. In *2024 IEEE 10th international conference on network softwarization* (pp. 204–212). IEEE.

Ashraf, A., Qingjie, Z., Bangyal, W. H. K., & Iqbal, M. (2023). Analysis of brain imaging data for the detection of early age autism spectrum disorder using transfer learning approaches for internet of things. *IEEE Transactions on Consumer Electronics*, *70*, 4478–4489.

Ashwood, Z., Jha, A., & Pillow, J. W. (2022). Dynamic inverse reinforcement learning for characterizing animal behavior. *Advances in Neural Information Processing Systems*, *35*, 29663–29676.

Atrey, A., Clary, K., & Jensen, D. (2020). Exploratory not explanatory: Counterfactual analysis of saliency maps for deep RL. In *International conference on learning representations*.

Attai, K., Ekpenyong, M., Amannah, C., Asuquo, D., Ajuga, P., Obot, O., Johnson, E., John, A., Maduka, O., Akwaowo, C., & Uzoka, F.-M. (2024). Enhancing the interpretability of malaria and typhoid diagnosis with explainable AI and large language models. *Tropical Medicine and Infectious Disease*, *9*, 216.

Badshah, S., & Sajjad, H. (2024). Reference-guided verdict: Llms-as-judges in automatic evaluation of free-form text. arXiv preprint arXiv:2408.09235.

Bastani, O., Pu, Y., & Solar-Lezama, A. (2018). Verifiable reinforcement learning via policy extraction. In *Proceedings of the 32nd international conference on neural information processing systems* (pp. 2499–2509). Red Hook, NY, USA: Curran Associates Inc..

Beechey, D., Smith, T. M. S., & Şimşek, Ö. (2023). Explaining reinforcement learning with Shapley values. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of machine learning research: Vol. 202*, *Proceedings of the 40th international conference on machine learning* (pp. 2003–2014). PMLR.

Boggess, K., Kraus, S., & Feng, L. (2022). Toward policy explanations for multi-agent reinforcement learning. arXiv preprint arXiv:2204.12568.

de Souza Loureiro, A., Valverde-Rebaza, J., Noguez, J., Escarcega, D., & Marcacini, R. (2025). Advancing multi-step mathematical reasoning in large language models through multi-layered self-reflection with auto-prompting. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 205–223). Springer.

Evans, B. D., Jordaan, H. W., & Engelbrecht, H. A. (2023). Comparing deep reinforcement learning architectures for autonomous racing. *Machine Learning with Applications*, *14*, Article 100496.

Feit, F., Metzger, A., & Pohl, K. (2022). Explaining online reinforcement learning decisions of self-adaptive systems. In *2022 IEEE international conference on autonomic computing and self-organizing systems* (pp. 51–60). IEEE.

Feldhus, N., Wang, Q., Anikina, T., Chopra, S., Oguz, C., & Möller, S. (2023). InterroLang: Exploring NLP models and datasets through dialogue-based explanations. arXiv preprint arXiv:2310.05592.

Fukuchi, Y., Osawa, M., Yamakawa, H., & Imai, M. (2017). Autonomous self-explanation of behavior for interactive reinforcement learning agents. In *Proceedings of the 5th international conference on human agent interaction* (pp. 97–101). ACM.

Graham, B. (2025). mcts-llm: Monte Carlo tree search for LLMs (GitHub repository). https://github.com/BrendanGraham14/mcts-llm. (Accessed 01 September 2025).

Greydanus, S., Koul, A., Dodge, J., & Fern, A. (2018). Visualizing and understanding atari agents. In *International conference on machine learning* (pp. 1792–1801). PMLR.

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., Wang, Y., & Guo, J. (2024). A survey on LLM-as-a-judge. ArXiv, abs/2411.15594.

Gupta, K. D., Sadman, N., Sadmanee, A., Sarker, M. K., & George, R. (2023). Behavioral recommendation engine driven by only non-identifiable user data. *Machine Learning with Applications*, *11*, Article 100442.

Hu, X., Liu, A., & Dai, Y. (2024). Combining ChatGPT and knowledge graph for explainable machine learning-driven design: A case study. *Journal of Engineering Design*, 1–23.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *ICLR*. OpenReview.net.

Huber, T., Schiller, D., & André, E. (2019). Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In *KI 2019: Advances in artificial intelligence: 42nd German conference on AI* (pp. 188–202). Springer.

Itaya, H., Hirakawa, T., Yamashita, T., Fujiyoshi, H., & Sugiura, K. (2021). Visual explanation using attention mechanism in actor-critic-based deep reinforcement learning. In *2021 international joint conference on neural networks* (pp. 1–10). IEEE.

Jiang, Y., Gu, S. S., Murphy, K. P., & Finn, C. (2019). Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, *32*.

Jiang, Z., & Luo, S. (2019). Neural logic reinforcement learning. In *International conference on machine learning* (pp. 3110–3119). PMLR.

Juozapaitis, Z., Koul, A., Fern, A., Erwig, M., & Doshi-Velez, F. (2019). Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI workshop on explainable artificial intelligence*.

Kamrani, A. S., Dini, A., Dagdougui, H., & Sheshyekani, K. (2025). Multi-agent deep reinforcement learning with online and fair optimal dispatch of EV aggregators. *Machine Learning with Applications*, *19*, Article 100620.

Kim, H., Chen, H., Li, C., & Lee, J. M. (2025). TalkToAgent: A human-centric explanation of reinforcement learning agents with large language models. ArXiv, abs/2509.04809.

Kroeger, N., Ley, D., Krishna, S., Agarwal, C., & Lakkaraju, H. (2023). In-context explainers: Harnessing LLMs for explaining black box models.

Kwon, M., Xie, S. M., Bullard, K., & Sadigh, D. (2023). Reward design with language models. arXiv preprint arXiv:2303.00001.

Lee, Y., Kim, J., Kim, J., Cho, H., Kang, J., Kang, P., & Kim, N. (2024). CheckEval: A reliable LLM-as-a-judge framework for evaluating text generation using checklists. arXiv preprint arXiv:2403.18771.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lu, W., Zhao, X., Spisak, J., Lee, J. H., & Wermter, S. (2025). Mental modelling of reinforcement learning agents by language models. *Transactions on Machine Learning Research*.

Lyu, D., Yang, F., Liu, B., & Gustafson, S. (2019). SDRL: Interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 33*, (pp. 2970–2977).

Martens, D., Hinns, J., Dams, C., Vergouwen, M., & Evgeniou, T. (2025). Tell me a story! narrative-driven xai with large language models. *Decision Support Systems*, Article 114402.

Masadome, S., & Harada, T. (2025). Reward design using large language models for natural language explanation of reinforcement learning agent actions. *IEEJ Transactions on Electrical and Electronic Engineering*.

Mavrepis, P., Makridis, G., Fatouros, G., Koukos, V., Separdani, M. M., & Kyriazis, D. (2024). XAI for all: Can large language models simplify explainable AI?. arXiv preprint arXiv:2401.13110.

Mekrache, A., Mekki, M., Ksentini, A., Brik, B., & Verikoukis, C. (2024). On combining XAI and LLMs for trustworthy zero-touch network and service management in 6G. *IEEE Communications Magazine*.

Metzger, A., Bartel, J., & Laufer, J. (2023). An AI chatbot for explaining deep reinforcement learning decisions of service-oriented systems. In *International conference on service-oriented computing* (pp. 323–338). Springer.

Milani, S., Topin, N., Veloso, M., & Fang, F. (2024). Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, *56*, 1–36.

Milani, S., Zhang, Z., Topin, N., Shi, Z. R., Kamhoua, C., Papalexakis, E. E., & Fang, F. (2022). Maviper: Learning decision tree policies for interpretable multi-agent reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 251–266). Springer.

Mirchandani, S., Karamcheti, S., & Sadigh, D. (2021). Ella: Exploration through learned language abstraction. *Advances in Neural Information Processing Systems*, *34*, 29529–29540.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., & Jimenez Rezende, D. (2019). Towards interpretable reinforcement learning using attention augmented agents. *Advances in Neural Information Processing Systems*, *32*.

Nangue Tasse, G., James, S., & Rosman, B. (2020). A boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, *33*, 9497–9507.

Olson, M. L., Khanna, R., Neal, L., Li, F., & Wong, W.-K. (2021). Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artificial Intelligence*, *295*, Article 103455.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., teusz Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N. A., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., & Zhang, L. M. (2019). Solving Rubik's cube with a robot hand. ArXiv, abs/1910.07113.

Pandya, R., Zhao, M., Liu, C., Simmons, R., & Admoni, H. (2024). Multi-agent strategy explanations for human-robot collaboration. In *2024 IEEE international conference on robotics and automation* (pp. 17351–17357). IEEE.

Payani, A., & Fekri, F. (2020). Incorporating relational background knowledge into reinforcement learning via differentiable inductive logic programming. arXiv preprint arXiv:2003.10386.

Peng, X., Riedl, M., & Ammanabrolu, P. (2022). Inherently explainable reinforcement learning in natural language. *Advances in Neural Information Processing Systems*, *35*, 16178–16190.

Qing, Y., Liu, S., Song, J., Wang, H., & Song, M. (2022). A survey on explainable reinforcement learning: Concepts, algorithms, challenges. arXiv preprint arXiv:2211.06665.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*, 1–8.

Rezazadeh, F., Chergui, H., & Mangues-Bafalluy, J. (2023). Explanation-guided deep reinforcement learning for trustworthy 6G RAN slicing. In *2023 IEEE international conference on communications workshops* (pp. 1026–1031). IEEE.

Rodriguez, I. D. J., Killian, T. W., Son, S., & Gombolay, M. C. (2020). Optimization methods for interpretable differentiable decision trees in reinforcement learning. In *Proc. of the 23rd int. conf. on artificial intelligence and statistics*.

Ruiz-Rodríguez, M. L., Kubler, S., Robert, J., Voisin, A., & Le Traon, Y. (2025). Evolutionary multi-objective multi-agent deep reinforcement learning for sustainable maintenance scheduling. *Engineering Applications of Artificial Intelligence*, *156*, Article 111126.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Shu, T., Xiong, C., & Socher, R. (2017). Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. arXiv preprint arXiv:1712.07294.

Slack, D., Krishna, S., Lakkaraju, H., & Singh, S. (2023). Explaining machine learning models with interactive natural language conversations using TalkToModel. In *Nature machine intelligence*: *Vol. 5*, (pp. 873–883).

Spitzer, P., Celis, S., Martin, D., Kühl, N., & Satzger, G. (2024). Looking through the deep glasses: How large language models enhance explainability of deep learning models. In *Proceedings of mensch und computer 2024* (pp. 566–570).

Stein, G. J. (2021). Generating high-quality explanations for navigation in partially-revealed environments. In A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems*.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning* (pp. 3319–3328). PMLR.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

Tan, H., Yan, H., & Yang, Y. (2025). LLM-guided reinforcement learning: Addressing training bottlenecks through policy modulation. ArXiv, abs/2505.20671.

Topin, N., Milani, S., Fang, F., & Veloso, M. (2021). Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 35*, (pp. 9923–9931).

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LlaMA: Open and efficient foundation language models. arXiv:2302.13971.

Towers, M., Du, Y., Freeman, C., & Norman, T. J. (2024). Explaining an agent's future beliefs through temporally decomposing future reward estimators. arXiv preprint arXiv:2408.08230.

Wang, J., Liang, Y., Meng, F., Sun, Z., Shi, H., Li, Z., Xu, J., Qu, J., & Zhou, J. (2023). Is chatGPT a good NLG evaluator? A preliminary study. In Y. Dong, W. Xiao, L. Wang, F. Liu, & G. Carenini (Eds.), *Proceedings of the 4th new frontiers in summarization workshop* (pp. 1–11). Singapore: Association for Computational Linguistics.

Yildirim, M., Dagda, B., Asodia, V., & Fallah, S. (2025). Highwayllm: Decision-making and navigation in highway driving with rl-informed language model. *Robotics and Autonomous Systems*, Article 105114.

Zhang, X., Guo, Y., Stepputtis, S., Sycara, K., & Campbell, J. (2023). Explaining agent behavior with large language models. arXiv preprint arXiv:2309.10346.

Zhang, D., Huang, X., Zhou, D., Li, Y., & Ouyang, W. (2024). Accessing gpt-4 level mathematical olympiad solutions via Monte Carlo tree self-refine with llama-3 8b. arXiv preprint arXiv:2406.07394.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ... Wen, J.-R. (2025). A survey of large language models. arXiv:2303.18223.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Proceedings of the 37th international conference on neural information processing systems*. Red Hook, NY, USA: Curran Associates Inc..

Ziaei, F., & Ranjbar, M. (2023). A reinforcement learning algorithm for scheduling parallel processors with identical speedup functions. *Machine Learning with Applications*, *13*, Article 100485.

Zytek, A., Pido, S., Alnegheimish, S., Berti-Equille, L., & Veeramachaneni, K. (2024). Explingo: Explaining ai predictions using large language models. In *2024 IEEE international conference on big data* (pp. 1197–1208). IEEE.

Zytek, A., Pidò, S., & Veeramachaneni, K. (2024). Llms for xai: Future directions for explaining explanations. arXiv preprint arXiv:2405.06064.