Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Counterfactual and Semifactual Explanations for Reinforcement Learning

Jasmina Gajcin

Supervisor: Prof. Ivana Dusparic

August 12, 2025

A dissertation submitted in partial fulfilment
of the requirements for the degree of
PhD (Computer Science)

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this thesis will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement. **Please consult with your supervisor on this last item before agreeing, and delete if you do not consent**

Signed: _____ Date: _____ 22.12.2024._____

Acknowledgements

My deepest gratitude goes to my supervisor, Dr. Ivana Dusparic, for her mentorship throughout the last five years. I truly appreciate her patience, encouragement and empathy during my doctoral studies. This thesis would not be possible without her expertise, research enthusiasm and feedback.

I would also like to thank Dr. Vinny Cahill and Dr. Peter Flach for the lively discussion during my viva and for their insightful comments, which significantly improved the thesis. My gratitude is extended to Dr. Elizabeth Daly and the entire Dublin research team at IBM Ireland for all their guidance and thoughtful discussions during my internships.

Thanks to all of my fellow researchers at DSG, for sharing in the highs and lows of the PhD journey. I also thank my friends whose understanding and support have meant the world to me. My heartfelt gratitude goes to Jovan for always looking at the bright side and keeping me sane, especially during the final months of thesis writing.

Finally, I would like to thank my family for their constant and unwavering support throughout my education journey. They believed in me from the very start, and this thesis would not be possible without them.

Abstract

Reinforcement learning (RL) is a trial-and-error machine learning paradigm where an agent interacts with an environment and receives rewards for its actions. RL aims to learn an optimal policy – a mapping from the observed states to actions maximizing the long-term reward. Deep reinforcement learning (DRL) algorithms use neural networks to learn the policy, increasing their scalability. However, the opacity of neural networks makes the decisions of DRL agents difficult to understand and hinders their applicability to high-risk tasks. Explainable RL (xRL) investigates methods for explaining the behaviour of RL agents.

Counterfactuals and semifactuals are powerful user-friendly explanations. Counterfactuals explain an outcome by contrasting it with the *closest possible world* – a similar scenario resulting in a different outcome. Semifactuals reinforce an outcome by offering an alternative scenario with the same outcome, in the form of a *most distant neighbour*. While extensively researched in supervised learning, there are only a few approaches developing these explanations for RL. Moreover, existing approaches in RL use the same feature-based metrics as used in supervised learning to select, evaluate, and search for these explanations. However, RL differs from supervised learning in its sequential and stochastic execution. Given the differences between the two frameworks, a question arises if the same definitions and methods for generating these explanations can be directly translated from supervised to RL. In this thesis, we aim to develop counterfactual and semifactual explanations for RL, by redefining the concepts of the closest possible world and the most distant neighbour in stochastic and sequential tasks and offering algorithms for counterfactual and semifactual search.

The first contribution of this thesis is an in-depth analysis of the two frameworks which identifies: 1) sequential execution 2) stochastic conditions, and 3) complex explanation types as the three main differences between supervised and RL frameworks, concluding that the same definitions of counterfactual and semifactual explanations cannot be directly translated to RL. We propose a set of five RL-specific counterfactual requirements (validity, reachability, recency, plausibility and certainty of outcome) and six RL-specific semifactual requirements (validity, reachability, recency, plausibility, unexpectedness, and argument strength) that ensure informative explanations in RL. We formally redefine counterfactual and semifactual explanations for stochastic and sequential RL tasks based on the defined requirements.

The second contribution of this thesis is the implementation of RACCER, the first RL-specific algorithm for counterfactual generation. RACCER implements a set of metrics – validity, temporal distance, fidelity and stochastic certainty to evaluate the counterfactual requirements. Additionally, RACCER implements three algorithms to optimize these metrics – RACCER-HTS (using heuristic tree search), RACCER-Advance and RACCER-Rewind (based on evolutionary algorithms).

The third contribution of this thesis is the implementation of SGRL, the first semifactual generation algorithm for RL. SGRL implements five metrics – validity, temporal distance,

fidelity, stochastic uncertainty and exceptionality that can be used to evaluate the RL-specific semifactual requirements. Additionally, SGLR implements two algorithms SGRL-Advance and SGRL-Rewind which use evolutionary algorithms to optimize these metrics and find semifactuals.

We evaluate RACCER and SGRL in five RL environments. Firstly, we evaluate them in standard, increasingly complex benchmark environments: frozen lake, stochastic grid-world, highway driving, and farm environment. We also conduct a case study using CitiBikes, a real-world simulated bike-sharing task. We find that RACCER and SGRL perform better on RL-specific counterfactual and semifactual metrics, producing more diverse and realistic explanations than the baseline approaches. Through a user study, we also find that counterfactual explanations generated by RACCER help users better understand the behaviour of RL agents compared to the counterfactuals generated by a baseline method. In contrast, a user study conducted on semifactual explanations finds that, while users rate highly explanations generated by SGRL, no significant impact has been found on users' perception and understanding of the agent compared to the baseline. A case study in large-scale CitiBikes tasks finds that while RACCER and SGRL approaches generate more realistic explanations, they struggle to scale to environments with large action spaces.

Contents

List of Figures

List of Tables

1 Introduction

Artificial intelligence (AI) solutions have become pervasive in many fields in the last decades [Pouyanfar et al., 2018], partially due to the emergence of neural networks. Neural networks are powerful algorithms that can efficiently process large amounts of unstructured data. With the adoption of neural networks, AI has reached near-human-level performance in many tasks such as image recognition [Krizhevsky et al., 2012], natural language understanding [Otter et al., 2020], and machine translation [Dabre et al., 2020]. However, neural networks rely on a large number of parameters, making their decisions difficult to understand and explain [Arrieta et al., 2020]. Approaches relying on neural networks are often labelled as *black-box* approaches, meaning that their reasoning is hidden and cannot be easily explained. Despite their remarkable performance, AI solutions cannot be straightforwardly deployed to high-risk tasks until their behaviour is fully verified and understood [Lipton, 2017].

Reinforcement learning (RL) [Sutton and Barto, 2018] is a subfield of AI that focuses on developing intelligent agents for sequential decision-making tasks. RL employs a trial-and-error learning approach in which an agent learns how to perform a task from scratch through interactions with its environment. An agent can observe the environment, perform actions that alter its state, and receive rewards from the environment. The goal of RL is to learn an optimal policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ – a mapping between the state space $\mathcal{S}$ and action space $\mathcal{A}$, which maximises the long-term cumulative reward.

Deep reinforcement learning (DRL) algorithms [Arulkumaran et al., 2017] are powerful algorithms that employ neural networks to represent an agent's policy. This allows them to process large amounts of unstructured data and scale to high-dimensional real-life tasks. In recent years, DRL algorithms have shown remarkable success in navigating sequential decision-making problems in games [Mnih et al., 2013, Silver et al., 2017, Vinyals et al., 2019], autonomous driving [Kiran et al., 2021], healthcare [Zhang, 2020, Yu et al., 2021], and robotics [Garaffa et al., 2021]. However, due to their reliance on neural networks, DRL algorithms are inherently black-box and their decisions are difficult to explain. Despite their promising results, the lack of explainability hinders their applicability to real-life high-risk tasks. In the following sections, we discuss the challenges of explaining the decisions of black-box RL systems.

In the remainder of this chapter, we first introduce the motivation behind this thesis in Section 1.1. Section 1.2 defines the research questions, and thesis contributions are detailed in Section 1.3. Section 1.4 describes the evaluation, and Section 1.5 concerns the

dissemination of works emerging from this thesis. Finally, a thesis roadmap is provided in Section 1.6.

1.1 Motivation

In recent years, neural networks have shown remarkable performance in solving complex tasks in supervised and RL. However, their lack of transparency makes it difficult to verify their behaviour and can lead to a lack of trust and engagement by the users of these systems [Saeed and Omlin, 2023]. In recent years, there has been a rise in research focusing not only on improving the performance of these black-box models but also on ensuring their behaviour can be verified and easily understood.

1.1.1 Explainable AI and RL

The field of explainable AI (xAI) develops methods for explaining black-box systems in many AI fields such as supervised learning [Burkart and Huber, 2021], RL [Puiutta and Veith, 2020] and robotics [Anjomshoae et al., 2019]. However, the biggest focus of xAI in recent years has been on explaining the decisions of supervised learning models. In contrast, explainable RL (xRL) is a fairly novel subfield of xAI that has not yet received an equal amount of attention [Puiutta and Veith, 2020].

Understanding the behaviour of RL agents is necessary to verify the correct behaviour before deployment [Puiutta and Veith, 2020], encourage user trust and collaboration [Heuillet et al., 2021], as well as to ensure fairness [Goodman and Flaxman, 2017]. Additionally, explainability is also necessary to correct and prevent "reward hacking" – a phenomenon where an RL agent learns to trick a potentially misspecified reward function and increase its reward without actually accomplishing the task. For example, a vacuum cleaner whose reward increases with the time spent cleaning might learn to eject collected dust [Amodei et al., 2016, Pan et al., 2022]. This lack of transparency is most often highlighted in DRL agents, due to their success in solving complex tasks and reliance on notoriously opaque neural networks. However, it is essential to note that explainability may be necessary for traditional RL algorithms as well. A tabular policy, for example, while free from neural networks, is often obscured by its large size and difficult to understand, especially for non-expert users. In this work, we focus on the problem of explainability as it pertains to the RL framework in general, regardless of the underlying algorithm. However, in the experimental section of the thesis, we evaluate our approaches on DRL algorithms, as these are currently considered state-of-the-art approaches for tackling complex, stochastic environments.

From the perspective of explainability, supervised and RL frameworks bear some similarities. For example, the lack of transparency in both frameworks most often comes from utilising neural networks for their decision-making process. Additionally, both frameworks solve a similar problem of making a prediction (a label in supervised and an action in RL) based on a set of input parameters (input features in supervised and state features

in RL). Thus, given the large body of work on explaining supervised learning, many methods for explaining RL agents have been adapted directly from supervised learning. For example, saliency maps have been adapted from supervised learning [Simonyan et al., 2013] to explain the effect of individual features on the decision of an RL agent [Greydanus et al., 2018, Huber et al., 2019]. However, the RL framework is inherently different from supervised learning due to the role of the RL environment. Firstly, RL agents perform multi-step tasks by interacting with the environment, in contrast with the one-step prediction process of supervised learning. Additionally, to complete the task in the environment, the agent often relies not only on its current state but also its goals and objectives. Finally, the RL environment is often stochastic, resulting in unpredictable events that might affect the agent's behaviour. While a decision in supervised learning can only be influenced by input features and model parameters, In RL it can be influenced by both state features [Simonyan et al., 2013, Greydanus et al., 2018, Puri et al., 2019, Rizzo et al., 2019] and other factors such as past or future states [van der Waa et al., 2018a, Madumal et al., 2020], goals [Beyret et al., 2019] or agent's objectives [Juozapaitis et al., 2019, Huang et al., 2019, Sukkerd et al., 2020]. For this reason, previous research argues that RL agents cannot be fully explained via methods developed for supervised learning and A wider set of more complex, RL-specific explanations is needed to fully capture the behaviour of RL agents [Dazeley et al., 2021c,a].

When developing explanations for AI systems, it is also important to consider to whom the explanation is targeted. Different stakeholders in the AI development process require different explanations [Hoffman et al., 2023]. At the moment, the majority of xAI is developed from the perspective of developers [Miller, 2019] and generates explanations by condensing the neural network into a more interpretable format [Bastani et al., 2017, Lakkaraju et al., 2017], or identifying low-level features responsible for a specific decision [Simonyan et al., 2013, Puri et al., 2019]. Once deployed, however, many AI algorithms will be used by users without in-depth knowledge of AI mechanics [Miller, 2019, Cheng et al., 2019]. These can be expert users, such as medical doctors using an AI-powered diagnostic tool, or non-expert users applying for a loan where an AI system decides the approval. Based on the insights from social sciences Miller [2019] has found that humans prefer explanations that are 1) contrastive, answering why an event P occurred *instead* of Q, 2) selective, focusing on a few causes, and 3) social, presented as a conversation. Non-expert users of AI systems are often more interested in abstract, high-level explanations, which help them better understand and interact with the system [Miller et al., 2017, Miller, 2019].

While there is an ever-growing amount of research on xAI, a staggeringly low number of works engage with end users of AI systems Nauta et al. [2023]. For this reason, in this thesis, we focus on user-friendly explanations that can help end users better understand and interact with the black-box system.

1.1.2 Counterfactual Explanations

One of the most prominent types of user-friendly explanations is counterfactual explanations. Counterfactual explanations explain why a decision y was made given input x by answering the question: *"How could x have been changed for the model to output an alternative decision y'?"* [Wachter et al., 2017]. The counterfactual explanation is then presented as an instance x' as similar as possible to x but producing an alternative outcome y'. For example, a user whose loan application was rejected by an AI system might receive a counterfactual explanation showing a similar application with a higher salary that got approved. Counterfactuals are imagined as belonging to the *closest possible world* where the outcome is changed [Wachter et al., 2017]. Intuitively, an explanation showing that someone with only a slightly higher salary had their loan approved is more informative than one indicating that an application with a much higher salary was successful. Unlike so-called causal explanations, which identify factual causes of a decision, counterfactuals provide richer information as they require simulation of multiple scenarios [Celar and Byrne, 2023]. Research from cognitive science indicates that counterfactuals better elicit creative thinking and enable reasoning such as *modus tollens*, which is difficult based only on factual information [Byrne, 2019].

Counterfactual explanations have the potential to provide user-understandable explanations to all stakeholders in the AI pipeline. However, their potential is most often highlighted for explaining behaviour to expert and non-expert users, as these cohorts are often overlooked in xAI research [Miller et al., 2017]. Counterfactual explanations have been heralded as user-friendly explanations for several reasons [Molnar, 2019, Miller, 2019]. Firstly, counterfactual explanations are contrastive, comparing a factual situation with an alternative situation – a property preferred by human users of AI systems [Miller, 2019]. Additionally, counterfactuals are selective, aiming to show an alternative situation similar to the factual one, indicating a few possible causes for the contested decision. Selective explanations reduce users' cognitive load and have been shown to improve user understanding and engagement with the AI system [Lai et al., 2023]. Finally, counterfactual reasoning is inherent to humans, as research from psychology shows that humans use it to assign blame and understand events [Byrne, 2019]. The benefits of counterfactuals on user understanding have been demonstrated in real-life domains such as medicine [Mertes et al., 2022].

Counterfactual explanations have been extensively explored for explaining supervised learning algorithms [Guidotti et al., 2018]. The main difference between various proposed methods comes from the varying definitions of the closest possible world. Despite differences in exact metrics used, most approaches rely on feature similarity metrics to estimate closeness and metrics based on data density to estimate plausibility.

In RL, counterfactuals have been used to answer the question *"Given that the agent chose action a in state x, how could x be changed for the agent to choose an alternative action a'?"*. The few approaches that generate counterfactuals for RL inherit the definitions of the closest possible world from supervised learning and search for a state similar in

features in which an alternative action is chosen [Olson et al., 2019, Huber et al., 2023]. However, the similarity between the two RL states cannot be fully inferred from their features. Unlike supervised learning, which employs a one-step prediction, RL deals with sequential tasks where actions can have long-term consequences and stochastic processes in the environment can affect execution. For this reason, two RL states can be similar in features, but far away in terms of execution and vice versa. For example, while two positions in a game of chess can differ in only one piece, they require multiple game moves or even be impossible to reach one from the other. In this thesis, we challenge the definition of counterfactuals inherited from supervised learning for explaining RL agents. A part of this thesis examines the similarities and differences between supervised and RL from the perspective of counterfactual explanations and proposes an RL-specific definition of the closest possible world, while another part focuses on implementing algorithms for counterfactual search in RL.

1.1.3 Semifactual Explanations

Counterfactual explanations can answer the *"what if"* question and show how a (potentially small) change in a scenario can lead to a different outcome. Semifactual explanations, on the other hand, answer the *"even if"* question [Aryal and Keane, 2023]. For example, the user whose loan is approved might want to know that *even if* they applied for a higher amount, they would still be successful. They are similar to counterfactuals in that they imagine alternative worlds. However, while counterfactuals search for a change in outcome, semifactuals show examples where the outcome remains the same. Semifactual explanations reinforce the outcome and make it seem immutable, and as such, they have great potential to increase user trust in the system [Kenny and Keane, 2021]. Similar to counterfactuals, they are considered user-friendly explanations, due to their contrastive nature.

The benefits and characteristics of semifactual explanations have long been explored in psychology and philosophy [McCloy and Byrne, 2002]. More recently, semifactuals have entered the field of xAI and have been developed for explaining supervised learning models as *a fortiori* arguments [Nugent et al., 2009, Kenny and Keane, 2021, Artelt and Hammer, 2022, Aryal and Keane, 2023, Kenny and Huang, 2024, Aryal and Keane, 2024]. In this way, semifactuals aim to provide a stronger argument for the outcome compared to the instance being explained. Two main avenues of research have emerged on how to find a strong semifactual in supervised learning. Counterfactually-guided methods imagine semifactuals on the path between the original instance and a counterfactual. On the other hand, counterfactually-free methods rely on feature similarity to find the *most distant neighbour* – an instance with the same outcome as the original one, but as far away from it as possible. In this thesis, we focus on the counterfactually-free methods, which can be widely applied without generating a counterfactual first.

As a fairly novel xAI technique, most of the benefits of semifactuals on user understanding of AI systems are, at the moment, speculative and stem from their similarity to the

user-friendly counterfactuals [Aryal and Keane, 2023]. For example, semifactuals are contrastive and sparse, which is preferred by AI users [Miller, 2019, Lai et al., 2023].

While recently utilised in supervised learning, there are no approaches for generating semifactual explanations in RL. Similar to the counterfactual explanations, the definition of the semifactuals in supervised learning relies on the notion of distance between instances, which does not directly translate into the RL framework, where the similarity in features does not necessarily indicate closeness between states. In this thesis, we aim to enable semifactual explanations for RL tasks. A part of this thesis focuses on defining the concept of the most distant neighbour in the context of stochastic and sequential RL tasks and providing algorithms for semifactual search.

1.2 Research Questions

Understanding the behaviour of RL agents is necessary to ensure trust and apply these powerful algorithms in real-life tasks. Additionally, as AI solutions become pervasive in our everyday lives, research is needed on explanations of black-box systems that are targeted not only at developers but can also be easily understood by users without the knowledge of AI concepts.

As contrastive, user-friendly explanations of counterfactuals and semifactuals have the potential to be powerful methods for explaining RL systems. Finding counterfactuals and semifactuals in RL involves reasoning about distances between states. For example, while two loan applications might differ only in one feature – education level, acquiring a higher level of education might take a long time and require several actions, making these two states far away in terms of execution. However, most of the research on these explanations is targeted at supervised learning and relies on feature-based metrics to estimate the distance between RL states. Unlike supervised learning, RL deals with sequential execution in often stochastic environments. Thus, feature-based metrics cannot fully capture the complex dependencies between RL states. In this thesis, we investigate how the counterfactual concept of the closest possible world and the semifactual concept of the most distant neighbour can be redefined in the context of a sequential and stochastic RL task.

Most current semifactual and counterfactual methods in supervised and RL search for the explanation in the training dataset or rely on it to train a generative model. However, the RL framework does not inherently have a training dataset. Additionally, training data consisting of independent instances is not representative of the underlying temporal connections between states, making it difficult to estimate state distances. For this reason, this thesis aims to provide efficient algorithms for counterfactual and semifactual searches that take into account the stochastic and sequential nature of RL.

To summarise, this thesis sets out to answer the following research questions:

1. **RQ1:** How can the counterfactual concept of the *closest possible world* be defined for stochastic and sequential RL environments?

2. **RQ2:** How can the semifactual concept of the *most distant neighbour* be defined for stochastic and sequential RL environments?
3. **RQ3:** How can we design and implement counterfactual search algorithms in stochastic and sequential RL environments?
4. **RQ4:** How can we design and implement semifactual search algorithms in stochastic and sequential RL environments?

To address the research questions, we take on the following research objectives:

- Design and develop a set of counterfactual requirements describing the closest possible world that a counterfactual explanation in RL should fulfil.
- Design and develop a set of semifactual requirements describing the most distant neighbour that a semifactual explanation in RL should fulfil.
- Design, implement and evaluate counterfactual search algorithms that optimise the RL-specific counterfactual requirements.
- Design, implement and evaluate semifactual search algorithms that optimise the RL-specific semifactual requirements.

When answering the above questions, the thesis makes a number of contributions, which we outline in the following section.

1.3 Thesis Contribution

This thesis investigates the problem of defining, generating, and evaluating user-friendly explanation methods for RL tasks. Specifically, this thesis focuses on counterfactual and semifactual explanations for RL. While counterfactual and semifactual explanations can be used to explain black-box behaviour to any AI stakeholder, they are most often envisioned as a tool for decision support. Decision support tasks are most often targeted at expert users, such as medical doctors using an AI tool, or non-expert users, such as individuals applying for a loan with an automated system. The user-friendly, contrastive, high-level nature of counterfactual and semifactual explanations makes them suitable for this task. Additionally, the absence of xAI tools developed for these stakeholders makes these explanations even more important.

An overarching contribution of this thesis is in mapping out the space of counterfactual and semifactual explanations. Currently, research on these methods has either been limited to supervised learning or directly translated from supervised to RL. For this reason, this thesis analyses supervised and RL and identifies the main differences between them from the perspective of these explainability methods. Additionally, this thesis systematises open questions, research directions, and challenges of developing, applying, and evaluating these explanations in RL.

The first contribution of this thesis is in redefining counterfactual and semifactual explanations from the one-step prediction tasks for which they are currently developed to

sequential and stochastic RL tasks. First, we conduct an in-depth survey of the fields of supervised and RL and identify and categorise differences between the two frameworks from the perspective of counterfactual and semifactual explanations. Furthermore, we define five high-level counterfactual requirements – *validity, reachability, recency, plausibility,* and *certainty of outcome* that describe conditions informative counterfactuals should follow. Similarly, we define six semifactual requirements – *validity, reachability, recency, plausibility, unexpectedness*, and *argument strength* that characterise informative semifactuals in RL. Finally, we define counterfactual and semifactual explanations for RL, taking into account the sequential and stochastic nature of RL tasks, as well as the wide space of decision influences such as the agent's goals, objectives, or outside events.

The second contribution of the thesis is in implementing and evaluating counterfactual search algorithms for RL. To that end, we propose RACCER (Reachable And Certain Counterfactual Explanations for Reinforcement Learning), which is, to the best of our knowledge, the first approach for generating counterfactual explanations for RL tasks which takes into account the sequential and stochastic nature of the RL framework. RACCER proposes four novel RL-specific counterfactual metrics – *validity, temporal distance, fidelity, and stochastic certainty*. These counterfactual metrics correspond to the counterfactual requirements defined above and ensure that counterfactuals are easy to reach and deliver the desired outcome with high probability. We define three optimisation algorithms – RACCER-HTS, RACCER-Rewind, and RACCER-Advance, which search for the counterfactual by optimising the counterfactual metrics. RACCER-HTS is an initial approach, utilising a heuristic tree search. However, due to its inflexible nature and search space limitations, we also implement RACCER-Rewind and RACCER-Advance, which use evolutionary algorithms to evaluate action sequences leading to potential counterfactuals. RACCER-Advance searches for the counterfactuals reachable from the original state, while RACCER-Rewind explores the agent's past interactions to find a counterfactual.

Secondly, we propose SGRL (Semifactual Generation for Reinforcement Learning), the first algorithm for generating semifactual explanations in RL. The main contribution of SGRL is in defining the concept of the most distant neighbour in the sequential and stochastic RL tasks. SGRL implements five metrics that define the most distant neighbour of the state in RL – *validity*, *temporal distance*, *fidelity*, *stochastic uncertainty* and *exceptionality*. These metrics correspond to the semifactual requirements above and ensure that the generated semifactual is far away from the original state and provides strong arguments for the outcome that occurred. SGRL implements two algorithms – SGRL-Rewind and SGRL-Advance, which use an evolutionary algorithm to optimise these metrics and generate diverse semifactuals by exploring the agent's past and future interactions, respectively.

1.4 Evaluation

We evaluate RACCER and SGRL in five environments, increasing in complexity. We use four standard benchmarking environments commonly used to evaluate xRL approaches and one real-life simulated task. All environments are highly stochastic, multi-objective environments where multiple strategies can be employed to successfully finish the task.

1. **Frozen Lake** [Brockman et al., 2016]: where the agent navigates a grid world with frozen squares in which actions can have unintended consequences. We modify the environment to remove the holes and make only certain squares icy.
2. **Stochastic gridworld** [Chevalier-Boisvert et al., 2023]: a variation of the traditional grid world environments. In this version, the agent's goal is to shoot the dragon. To introduce stochasticity in the environment, we include obstacles such as trees and walls which can block the agent's path to the dragon. The agent can destroy obstacles, incurring varying costs, and obstacles can reappear through stochastic processes outside of the agent's control.
3. **Highway Driving** [Leurent, 2018]: where the agent is tasked with driving on a multi-lane highway. The agent can observe the coordinates and speed of neighbouring vehicles and at each step choose one of five discrete actions – change lane to the left, change lane to the right, speed up, slow down, or do nothing (i.e., continue driving without changing the lane or speed). The agent has to optimise multiple criteria such as avoiding collisions, maintaining good speed, and remaining in the rightmost lane.
4. **Farm** [Maillard et al., 2023]: which simulates an agricultural task of growing a tomato plant. At each time step, the agent can apply between 1L and 5L of water or harvest the plant. Positive rewards are obtained by harvesting a ripe plant and every time a plant transitions to a new growth stage. Stochasticity is exhibited in the weather conditions, such as temperature, humidity, and rain.
5. **CitiBikes** [Jiang et al., 2020]: which is modelled after the New York City bike-sharing program. CitiBikes is a real-life simulated task that introduces complexity in the form of a large action and state space, which is not common in standard RL benchmarks. The environment consists of a number of docking stations from which bikes can be borrowed and returned. At each step, the agent can move bikes between stations to satisfy demand. The number of bikes transferred and the shortage of bikes factor into the reward function.

We evaluate each of the approaches against the most closely related state-of-the-art baselines.

We evaluate RACCER against the following baseline:

- **GANterfactual-RL** [Huber et al., 2023]: the only model-agnostic approach for

generating counterfactual explanations for RL. GANterfactual-RL uses generative modelling with a StarGAN architecture [Choi et al., 2018] to generate counterfactual states.

To evaluate SGRL, we adapt a state-of-the-art approach for generating semifactual explanations from supervised learning, as no approaches are available for RL:

- **S-GEN** [Kenny and Huang, 2024]: searches for a semifactual by optimizing three objectives – gain, robustness and diversity. The gain is determined by the distance between two states, taking into account a specified user preference. Robustness is measured in relation to instances in the neighbourhood of the semifactual in the same class. The diversity encourages the generation of semifactuals with diverse features to enable personalisation. To be adapted to RL, the S-GEN approach only requires a dataset of agent interactions.

For each of the approaches, we performed both qualitative and quantitative evaluations.

Quantitative evaluation consists of evaluating approaches and baselines on metrics used to describe suitable explanations. Firstly, we evaluate how explanations perform on RL-specific metrics defined in this thesis. Additionally, we evaluate whether generated explanations are realistic, as well as their diversity. Finally, we evaluate the scalability of the approaches by measuring 1) coverage, as the number of states for which an explanation is successfully generated, and 2) the average time taken to generate a solution.

As explanations are developed for users, qualitative evaluation is performed through user studies. We employ both objective and subjective metrics for evaluating explanations and recourse. Namely, to evaluate RACCER, we use two objective metrics to evaluate the user's ability to 1) predict the behaviour of RL agents and 2) compare RL agents with different strategies. For SGRL, we evaluate only the user's ability to predict the behaviour of RL agents. For SGRL, we do not evaluate the users' ability to compare agents to reduce cognitive load, and as this is difficult for users to deduce using counterfactual explanations in previous work [Huber et al., 2023] and in our own RACCER user study. For both approaches, we utilise subjective metrics and evaluate users' satisfaction with the explanations based on explanation goodness metrics [Hoffman et al., 2018].

1.5 Dissemination

The contributions of this thesis have been presented in different peer-reviewed venues. Below, we list separately the publications describing contributions featured in the thesis and those that presented initial exploratory work that informed the approaches presented in this thesis.

Thesis Publications

The following publications are covered in this thesis:

1. **Gajcin, J.**, & Dusparic, I. (2024). *Redefining Counterfactual Explanations for Reinforcement Learning: Overview, Challenges and Opportunities.* ACM Computing Surveys.

   In this work, we explored how counterfactual explanations differ between supervised and RL and redefined counterfactuals for the RL context. This work identified the main challenges of generating counterfactuals for RL tasks and has served as a roadmap for the remainder of the thesis. Chapter 3 covers most of the content of this paper.

2. **Gajcin, J.**, & Dusparic, I. (2024). *RACCER: Towards Reachable and Certain Counterfactual Explanations for Reinforcement Learning*. In Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS24).

   This work introduces RACCER, an algorithm for generating counterfactual explanations in RL. Section 4.2 is based on this publication.

3. **Gajcin, J.**, Jeromela, J & Dusparic, I. (2024) *Semifactual Explanations for Reinforcement Learning*. In Proceedings of the 12th International Conference on Human-Agent Interaction (HAI24).

   This work proposes SGRL, an approach for generating semifactual explanations for RL, consisting of a set of semifactual metrics and an evolutionary algorithm optimising them. Section 4.3 derives from this publication.

Other Works

Multiple publications resulted from early-stage research on explainability, causality, and human interaction in RL systems. Although not directly covered in this thesis, these works were a first step in identifying challenges of explainable RL and have informed the consequent works that are part of the thesis.

1. **Gajcin, J.**, & Dusparic, I. (2022). *ReCCoVER: Detecting Causal Confusion for Explainable Reinforcement Learning*. In Proceedings of International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems (EXTRAAMAS22)

   This publication explored the issue of causal confusion in RL, where an agent learns to rely on a spurious correlation that does not hold across the entire state space, resulting in a decrease in performance. The work introduced ReCCoVER, an algorithm for detecting and correcting causal confusion.

2. **Gajcin, J.**, Nair, R., Pedapati, T., Marinescu, R., Daly, E., & Dusparic, I. (2021). *Contrastive explanations for comparing preferences of reinforcement learn-*

ing agents. Presented at the Interactive Machine Learning Workshop at AAAI22 (INTERACT22).

In this work, we proposed an xRL approach to generating contrastive explanations for explaining the outcomes of policies with different preferences. This work was a result of an internship at IBM Research and was done in collaboration with IBM supervisors.

3. **Gajcin, J.**, McCarthy, J., Nair, R., Marinescu, R., Daly, E., & Dusparic, I. (2023). *Iterative Reward Shaping using Human Feedback for Correcting Reward Misspecification* In Proceedings of the 49th European Conference on Artificial Intelligence (ECAI2023)

   This work explored interactive reinforcement learning for reward misspecification and proposed ITERS, a human-in-the-loop algorithm that allowed users to detect and correct unwanted behaviour and tailor RL policies to their preferences. This work was a result of another internship at IBM Research and was done in collaboration with IBM supervisors.

4. **Gajcin, J.** *Counterfactual Explanations for Reinforcement Learning Agents*. Proceedings of the Doctoral Consortium at the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS23)

   This extended abstract explored the related work, challenges and open research questions in developing counterfactual explanations for RL systems.

5. He, T., **Gajcin J.** & Dusparic, I. *Causal Counterfactuals for Improving the Robustness of Reinforcement Learning*. Presented at the Autonomous Robots and Multirobot Systems workshop at AAMAS23 (ARMS23).

   This work in the field of causal RL proposed CausalCF, an approach for performing causal interventions and generating counterfactuals to increase the robustness of RL algorithms in robotics tasks.

6. **Gajcin, J.** & Dusparic, I. *ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing RL Policies*. Under review at the 33rd Conference on User Modelling and Personalisation (UMAP25).

   This work explores the user preferences when defining and generating counterfactual recourse in RL to diagnose and correct suboptimal RL policies.

1.6 Roadmap

To summarise, this chapter introduced counterfactual and semifactual explanations as an important and powerful method for enabling trust and interaction with users of AI systems. This thesis explores these explanations in RL tasks, redefines them for the RL context, and provides algorithms for generating counterfactual and semifactual explanations.

The remainder of the thesis is organised as follows. Chapter 2 covers background and related work in the field of explainable RL, with a focus on semifactual and counterfactual explanations. In Chapter 3, we redefine counterfactual and semifactual explanations for RL, the challenges and opportunities for their generation. Chapter 4 covers the implementation of RACCER, the algorithm for generating counterfactual explanations for RL, and SGRL, the algorithm for generating semifactual explanations for RL. Evaluation metrics, scenarios, and results in standard RL benchmarks are presented in Chapter 5. The case study in a real-life, high-dimensional CitiBikes task is presented in Chapter 6. Finally, Chapter 7 provides a conclusion and discusses the limitations and identifies future work directions emerging from this thesis.

2 Background and Related Work

In this chapter, we provide an overview of terminology and related work in the area of explainability in RL, with a focus on counterfactual and semifactual explanations. Section 2.1 covers the preliminaries. Section 2.2 provides the background on explainable AI, while Sections 2.3 and 2.4 focus on explainability of supervised and RL methods, respectively. Section 2.5 covers counterfactual and Section 2.6 semifactual explanations, while explanation evaluation is covered in Section 2.7.

2.1 Reinforcement Learning (RL)

Reinforcement learning (RL) [Sutton and Barto, 2018] is a trial-and-error learning paradigm for sequential tasks. In RL, an agent performs actions in an environment and learns correct behaviour from the environment's reward signals. Formally, RL tasks are modelled as a Markov Decision Process (MDP) $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. State space $\mathcal{S}$ represents a set of possible states the agent can observe, and action space $\mathcal{A}$ is a set of all possible actions. Transition function $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ determines the probabilities of specific transitions between states when taking actions. The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns a real-value reward to each transition. Finally, γ is the discount factor. Intuitively, the discount factor γ determines the trade-off between immediate and delayed rewards in the task.

Agent's behaviour in an environment can be described by a policy $\pi : \mathcal{S} \to \mathcal{A}$, which maps the agent's observed states to its actions. The goal of RL is to learn an *optimal policy* π^* which chooses optimal actions in each state. Agent's behaviour in the environment starting between timesteps t and T can be evaluated according to the reward function and quantified as *a return function* G_t:

$$G_t = \sum_{i=t+1}^{T} \gamma^{i-t-1} \cdot R_i \tag{2.1}$$

In other words, the goal of RL is to find a policy that maximises the return.

One of the first algorithms for finding the optimal policy is Q-learning [Watkins and Dayan, 1992]. Q-learning is based on iterative learning of Q-values. A Q-value $Q_\pi(s, a)$ for policy π describes the expected reward for taking action a in state s and following

policy π thereafter:

$$Q_\pi(s, a) = E_\pi[G_t|s_t = s, a_t = a] \tag{2.2}$$

The goal of the Q-learning algorithm is to approximate the Q-value function of the optimal policy π^* through iterative updates. For each state-action pair, the Q-value is updated according to the update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.3}$$

The parameter α is the learning rate and determines the magnitude of the update. Intuitively, Q-learning uses the immediate reward R_{t+1} and performs a one-step look-ahead to see what the reward is for the following state s_{t+1}. The iterative process continues until convergence. Calculating the optimal Q-value function q^* is equivalent to finding the optimal policy π^* – to perform the optimal action in each state, the agent only needs to choose the action with the highest value for q^*.

Although Q-learning represented a breakthrough in RL research, the algorithm comes with a set of limitations, preventing it from being applied to real-life RL tasks. For one, Q-learning requires Q-value approximations of all state-action pairs to be stored in memory. This is often achieved in tabular form, which becomes infeasible for large state and action spaces present in real-life tasks. Additionally, Q-learning requires updates on all state-action pairs to learn their Q value. In the case of high-dimensional environments, not all state-action pairs might be visited, and the Q-learning algorithm will not be able to estimate the best course of action in these conditions.

2.1.1 Deep Reinforcement Learning (DRL)

Deep reinforcement learning (DRL) algorithms apply concepts from deep learning to RL to overcome the challenges associated with tabular methods such as Q-learning [Li, 2017]. In DRL, a policy is represented using a neural network that maps state features to the agent's actions. By using neural networks, DRL algorithms allow for the processing of large amounts of high-dimensional input, which is common in real-life tasks. Additionally, neural networks allow unstructured input, such as images, greatly increasing the applicability of RL systems.

One of the first DRL approaches, Deep Q Network (DQN), was proposed by Mnih et al. [2013]. DQN is inspired by Q-learning and uses iterative updates to learn to approximate the optimal Q-value function. However, instead of storing Q-values in a tabular form, DQN uses a neural network to approximate their values. DQN takes the agent's state as input and propagates it through the network. The final layer of a DQN estimates Q-values for each possible action. In recent years, DQNs and similar DRL approaches have been successfully applied in many fields such as robotics [Ibarz et al., 2021], healthcare [Yu et al., 2021], and autonomous driving [Kiran et al., 2021].

Despite their success, DRL algorithms lack transparency due to their reliance on neural

Figure 2.1: A summary of goals of XAI depending on the target audience: While the focus of developers is to better understand the abilities of the system and enable successful deployment, experts using the system require explanations to better collaborate with the system. Explanations are necessary for non-expert users to develop trust, ensure system decisions are fair, and give users actionable feedback on how to elicit a different decision from the system.

networks. This makes their decision-making process opaque and difficult to interpret. As a result, these algorithms cannot be safely applied to high-risk real-life RL tasks.

2.2 Explainable AI (xAI)

The emergence of neural networks has led to the development of AI algorithms in many applications in fields like healthcare [Alowais et al., 2023], finance [Li et al., 2023], or recommender engines [Chen et al., 2023]. Neural networks can process large amounts of unstructured data and learn complex patterns to support decision-making. However, these models often rely on optimising millions of parameters, making them difficult to understand. As such, these algorithms are often referred to as black-box, since their decisions are difficult to interpret and understand.

The field of explainable AI (xAI) gathers methods that make AI systems easier to understand [Adadi and Berrada, 2018]. Despite the substantial amount of work in this field in recent years, there is still a lack of precise, unified definitions for what it means for an AI system to be explainable. Adding to the confusion, terms such as interpretability, explainability and transparency are used interchangeably in some works, and distinguished in others. According to Merriam-Webster dictionary, to *interpret* means to "give or provide meaning or to explain and present in understandable terms". Doshi-Velez and Kim [2017]

expand this definition and address the issue that interpretability must provide meaning in terms that are understandable to humans. Guidotti et al. [2018] further define an explanation as an "interface between the human and decision maker". In this work, we adopt the definition from Miller [2019], where an explanation is defined as an answer to a *"why?"* question. This type of explanation focuses on the causes of an event, unlike non-causal explanations, which might be interested in what happened or how an event occurred [Ginet, 2008].

It is important to note that not all AI systems require explainability. Systems that do not interact with humans, or those that operate in low-stakes environments, might not need to be able to explain their every decision. However, most AI systems are developed with end users in mind and targeting real-life high-risk tasks, such as diagnosing diseases, driving autonomous vehicles, or predicting financial markets. For such systems, explainability may be required for various reasons. An overview of explainability requirements for different stakeholders is presented in Figure 2.1.

To ensure that AI systems operate correctly, developers need to understand their decisions. Understanding why a specific decision has been made is necessary to ensure that the model can generalise correctly and behave safely in new scenarios. For example, consider the object recognition model developed in [Beery et al., 2018]. Although the model shows great success in classifying objects during training, there is a flaw within the model's reasoning that can have substantial consequences after deployment. Specifically, the model learns to classify an image as a cow not because of its features, but because of the presence of grass in the photo. As all the images of cows in the training dataset are taken on fields, this error in reasoning will not be exposed before deployment. However, if, after deployment, the model comes across a cow on a beach, its performance will suffer. Similar safeguards have been proven necessary after the 2018 Uber tragic accident, where a car failed to stop for a pedestrian [Knight, 2018]. The object recognition system could not correctly classify the pedestrian as they were pushing a bicycle and thus failed to predict their trajectory. Knowing how a decision has been made by the model is essential in ensuring that correct reasoning is applied and performance is maintained even after deployment.

Many AI systems are envisioned to aid expert professionals, such as medical doctors and loan officers, with their decision-making. However, studies have shown that experts are reluctant to rely on decisions of black-box models if they cannot understand them [Cadario et al., 2021]. To reap the benefits of powerful AI models in high-risk scenarios, their decisions have to be easy to understand and trust.

The end-users of AI systems might not have the expertise needed to understand the low-level inner workings of an AI model. However, they might still interact with AI in their day-to-day, and even more importantly, AI might be making important decisions affecting their lives. For example, AI has previously been developed for selecting suitable candidates during the hiring process [Mujtaba and Mahapatra, 2019] or predicting recidivism rates for incarcerated individuals [Rudin et al., 2020]. From the perspective of

end users, explainability is needed to ensure that users can understand decisions made for them, take actionable recourse to change those decisions and interact with the AI systems.

From the perspective of fairness, understanding how a decision has been made is necessary to avoid discrimination. As AI systems often rely on large amounts of data, their decision-making process can inadvertently inherit historical biases present in that data [Mujtaba and Mahapatra, 2019]. For example, consider the Amazon recruitment system based on AI, which proved to be biased against women applying for engineering roles, as it learned to favour resumes with words more commonly found in men's writing [Dustin, 2018]. Additionally, users of AI systems might have the right to an explanation under a legal framework, such as the GDPR in the EU [Goodman and Flaxman, 2017].

The field of XAI has seen a rapid rise in the number of methods for explaining AI decisions. However, the majority of the methods are targeted at the developers and expert users and provide low-level explanations. On the other hand, non-expert users might be interested in understanding not just why the specific decision was made, but also how to better interact with the AI system and achieve the desired outcome. Previous research in psychology has found that users prefer contrastive-style explanations that answer the question "why not?" [Miller, 2019]. Contrastive-style explanations, such as counterfactual [Guidotti, 2024] and semifactual explanations [Aryal and Keane, 2024], are prominent, user-friendly explanations that can help users better engage with a black-box system. Counterfactual and semifactual explanations explain why an outcome occurred in the specific scenario by contrasting it with an alternative scenario. For example, an AI system deciding to reject a person's loan application could explain its decision counterfactually (e.g. *"If your salary was higher, you would have been approved"*) or semifactually (e.g. *"Even if your request was for a smaller amount, you would have still been rejected"*).

In the remainder of this section, we provide an overview of the most important methods for explaining supervised (Section 2.3) and RL (Section 2.4). Furthermore, we review user-friendly counterfactual (Section 2.5) and semifactual explanations (Section 2.6).

2.3 Explainability in Supervised Learning (IML)

XAI is a broad term that encapsulates explainability methods in various fields within AI, such as supervised learning [Burkart and Huber, 2021], recommender systems [Saeed and Omlin, 2023] and RL [Vouros, 2022]. Still, the majority of XAI research at the moment focuses on explaining a single prediction from a supervised learning algorithm. Methods for interpreting decisions of supervised learning algorithms belong to the field of *interpretable machine learning* (IML). In IML, explanations are focused on uncovering the relationship between the input and the output of a supervised learning model.

IML methods can be categorised along several dimensions, according to their scope,

explanation time, model reliance and target audience. For example, according to their scope, IML methods can be local, explaining only one prediction or global, explaining a black-box model as a whole. Model-specific IML methods are developed for only one type of black-box model, while model-agnostic methods can be applied to any black-box model. Additionally, black-box models can be intrinsically interpretable if their structure is easy to understand (e.g. decision trees or rule-based systems) or post-hoc interpretable if they require explanations on top of their opaque decision-making. Finally, developers, expert users and non-expert users of supervised learning models require different types of explanations (Figure 2.1).

The following sections present an overview of some of the most notable approaches for explaining decisions of ML systems aimed at developers and expert users. A summary of all IML approaches covered in this section according to the taxonomy is presented in Table 2.1.

2.3.1 Feature Importance Methods

Most of the methods for explaining supervised learning models focus on explaining a single decision through the influence of different input features. Feature importance methods are local explanation methods that assign scores to input features based on their contribution to the decision of the AI system.

Partial Dependence Plots (PDP)

Partial dependence plots (PDP) represent a visualisation technique which explains the influence of one or two input features on the output of a model [Friedman, 2001]. PDP plots the partial dependence function $f_S(X_S)$ for a set of features X_S:

$$f_S(X_S) = E_{X_C}[f(X_S, X_C)] = \int f(X_S, X_C) d\mathbb{P}(X_C) \tag{2.4}$$

where f is the black-box model, and X_C a set of variables not being explained. In other words, the partial dependence function estimates the average output of the model when features from X_S are forced to take specific values. The plot shows how individual values of the selected features affect the outcome. A flat PDP indicates that output does not rely on the feature, while more variation in PDP means more significant influence of the feature on the outcome.

Greenwell et al. [2018] propose a way for estimating effects of features in X_S on the output in practice:

$$\bar{f}_S(X_S) = \frac{1}{N} \sum_{i=1}^{N} f(X_S, X_{i,C}) \tag{2.5}$$

where $X_{i,C} = (1, ..., n)$ are values of X_C that occur in the training sample. In other words, the approach averages the output of the model over all other features in the

Table 2.1: Categorisation of IML methods based on the IML taxonomy (Section 2.3). Methods are classified based on their scope, reliance on the black-box model, time of explanation and intended audience.

Explanation type	Explanation subtype	Approach	Scope	Model-reliance	Explanation time	Audience
Feature importance	PDP	Friedman [2001] Greenwell et al. [2018]	Global	Model-agnostic	Post-hoc	Developers/Experts
	ICE	Goldstein et al. [2015]	Local	Model-agnostic	Post-hoc	Developers/Experts
	Permutation feature importance	Breiman [2001] [Pizarroso et al., 2020] Yao et al. [1998] Scardi and Harding Jr [1999]	Global	Model-agnostic	Post-hoc	Developers/Experts
	Shapley values	Štrumbelj and Kononenko [2014]	Local	Model-agnostic	Post-hoc	Developers/Experts
Local surrogates		Ribeiro et al. [2016] Lundberg [2017]	Local	Model-agnostic	Post-hoc	Developers/Experts
Global surrogates	Decision trees	Bastani et al. [2017] Frosst and Hinton [2017]	Global	Model-specific	Post-hoc	Developers/Experts
	Decision sets	Lakkaraju et al. [2017]	Global	Model-specific	Post-hoc	Developers/Experts

Algorithm 1 Calculating partial dependence function

Input: Trained model f, training dataset D, feature of interest x_i, unique values of x_i in dataset $x_i^1, ...x_i^k$

Output: Partial dependence function $f_i(x_i)$

1: **for** $j \in 1, 2, ...k$ **do**
2: $\quad D_i \leftarrow D$
3: $\quad D_i[x_i] \leftarrow x_i^j$
4: $\quad \bar{f} = f(D_i)$
5: $\quad f_i \leftarrow f_i + mean(\bar{f})$
6: **end for**

model for fixed values of features in X_S. High-level approach for estimating the partial dependence function $f_i(x_i)$ for each available value $x_i^1, ...x_i^k$ of the feature $x_i \in X_S$ on the output is shown in Algorithm 1.

Inspired by PDP, Goldstein et al. [2015] proposed ICE (Individual Conditional Expectations) plots for visualising the effect of a feature on the output of an individual data point. Unlike PDP, ICE does not plot the average effect of the feature, but rather presents dependence plots individually for each instance, resulting in one line per data point being plotted. Intuitively, the PDP line can be considered the average of all lines in the ICE plot.

PDP is a straightforward and easy-to-implement approach for visualising the influence of features on the output. Additionally, this method has a causal interpretation – forcibly setting the feature value corresponds to the causal interventions. In this way, PDP estimates causal effects of individual features on the outcome [Zhao and Hastie, 2021]. However, the approach is limited to only showcasing the effects of at most two features at a time. Additionally, PDP assumes independence between features – in other words, it is assumed that forcing change in one feature will not produce a change in others. In situations where two features are correlated, PDP can generate out-of-distribution samples where the two features no longer have correlated values. For example, consider the often correlated features height and weight – forcibly increasing the height variable, without adjusting the weight, would result in an unlikely sample.

Permutation Feature Importance

Permutation feature importance is used to estimate the global effect of a feature on the model accuracy. To determine how reliant the model is on a specific feature, that feature's values are permuted in the dataset, and model predictions are obtained for the modified data. The difference in accuracy between the models trained on the original and permuted data is the permutation feature importance. Since permuting breaks the relationships between a feature and the output, model accuracy will suffer if a feature was instrumental in making predictions. The approach was initially implemented by Breiman [2001] for explaining the behaviour of a random forest model.

The procedure for estimating the importance of feature f_j of a model f using permutation feature importance is described in Algorithm 2. The difference between model performance with original and permuted feature values $\Delta(e_{orig}, e_{perm})$ can be calculated in two main ways: either as a difference $\Delta(e_{orig}, e_{perm}) = e_{perm} - e_{orig}$ or ratio $\Delta(e_{orig}, e_{perm}) = \frac{e_{perm}}{e_{orig}}$ [Molnar, 2019].

Permutation feature importance has also been applied to the problem of *sensitivity analysis*, which attempts to estimate the sensitivity of the neural network's output on the individual features. This approach requires adding a small amount of noise to the values of a specific feature and calculating the difference in performance between the model using the original values and the one relying on the altered feature [Pizarroso et al., 2020]. Yao et al. [1998] used this method to estimate the effect of different variables, such as expert knowledge, marketing information and environment data, on the prediction with neural networks. Similarly, Scardi and Harding Jr [1999] performed a sensitivity analysis of the neural network model for primary production of phytoplankton using noise-based feature permutation.

Permutation feature importance is a simple way for determining how much the model relies on a specific feature. It provides a global interpretation and, when the difference in performance is calculated as a ratio, it is comparable across different models. However, this approach assumes feature independence and suffers from the same issue as the PDP. Namely, in a situation where two features are correlated, randomly permuting one of them, while not changing the other, breaks the correlation between them and can result in unlikely data samples.

Shapley Values

Shapley values originate from game theory and provide a way for assigning payouts to players who collaborate in a game [Shapley, 2016]. In machine learning, Shapley values are used to determine how much each feature contributed to the prediction of a single instance [Molnar, 2019]. Features are considered the *"players"* that participate together in a *"game"* of assigning the label to an instance, and *"gain"* is the prediction.

Shapley value of a feature f_j is calculated as the average marginal contribution of f_j across all *coalitions*. A coalition is any subset of features excluding f_j. To calculate

the Shapley value $\phi(f_j)$ value for feature f_j, based on the black-box model f, for each coalition, the outcome with and without f_j is evaluated, and the difference between the two is taken as the marginal contribution:

$$\phi_j(f_j) = \sum_{S \subseteq 1,\dots,p \setminus j} \frac{|S| \cdot (p - |S| - 1)!}{p!} (\mathrm{val}_x(S \cup j) - \mathrm{val}_x(S)) \tag{2.6}$$

where $\mathrm{val}_x(S)$ is a prediction for feature values in S marginalized over all features not included in S:

$$val_x(S) = \int f(x_1, \dots, x_p) dP_{x \notin S} - E_x(f(X)) \tag{2.7}$$

Shapley values are the only attribution method that satisfies the following four metrics [Molnar, 2019]:

1. *Efficiency* means that the sum of feature contributions equals the difference between the prediction and the average:

$$\sum_{j=1}^{p} \phi_j = f(x) - E_X(f(X)) \tag{2.8}$$

2. *Symmetry* requires that two features j and k get equal contribution if they contribute equally over all coalitions:

$$\text{If } val(S \cup j) = val(S \cup k), \forall S \subseteq 1, \dots p \setminus j, k \Rightarrow \phi_j = \phi_k \tag{2.9}$$

3. *Dummy* property states that if a feature does not change the prediction when added to any coalition, its contribution should be 0:

$$\text{If } val(S \cup j) = val(S), \forall S \subseteq 1, \dots, p \setminus j \Rightarrow \phi_j = 0 \tag{2.10}$$

4. *Additivity* means that for a game with combined payouts $val = val^1 + val^2$, Shapley value can also be obtained as a sum of individual Shapley values:

$$\phi_j = \phi_j^1 + \phi_j^2 \tag{2.11}$$

   This is useful in case of ensemble models (e.g. random forests) where multiple experts influence the final decision. Shapley values for experts can be calculated individually and added together to calculate the total contribution of a feature.

Shapley values require calculations to be carried out for each subset of features. In practice, for a large number of features, it might become infeasible to perform all evaluations directly. Štrumbelj and Kononenko [2014] propose an approximate way to calculate the Shapley value:

$$\phi_j = \frac{1}{M}\sum_{m=1}^{M}(f(x_{+j}^m) - f(x_{-j}^m)) \tag{2.12}$$

Where $f(x_{+j}^m)$ is a prediction for instance x in which a random number of features (excluding feature f_j) has been replaced by values from a randomly selected instance z. Similarly, values of the same random features f_{-j}^m are replaced by values from the same z, but also the value of f_j is copied from z.

Shapley values represent a theoretically sound attribution method with a background in game theory. This is also the only attribution method with efficiency, symmetry, dummy, and additivity properties. However, it requires computations to be performed for each of 2^N coalitions over N features, which can make it infeasible. Additionally, as a permutation-based method, it is prone to relying on out-of-distribution samples.

2.3.2 Local Surrogates

Local surrogate approaches interpret individual decisions of ML systems by generating interpretable surrogate models, which approximate the local behaviour of the black-box model. The most notable approach of this type is LIME (Local Interpretable Model-Agnostic Explanations) [Ribeiro et al., 2016]. To explain the model prediction for a data instance x, LIME starts by perturbing the features in x to get a local dataset P. Intuitively, instances in P should be similar to x and form its neighbourhood. Additionally, a proximity function π_x is defined, and each instance in P is weighted according to its closeness to the original data point x – instances closer to x receive a larger weight. For each instance in P, a label is obtained by querying the black-box model. Finally, an interpretable surrogate model is trained on P to locally mimic the decisions of the black-box model. The obtained surrogate can be further analysed to gain insights into the local behaviour of the decision-making process.

More formally, given a black-box model f, instance x, proximity function π_x, LIME generates a surrogate model G^* which satisfies:

$$G^* = \arg\min_{g \in G}[L(f, g, \pi_x) + \Omega(g)] \tag{2.13}$$

where G is a set of all possible interpretable surrogate models, $L(f, g, \pi_x)$ is a loss function which measures how well the surrogate model approximates the black-box model, and $\Omega(g)$ refers to the model complexity of g. In other words, the generated surrogate model G^* should be as true as possible to the local behaviour of the black-box model, while also not being too complex. Limiting the complexity of a surrogate model is important to ensure that the generated model is simple enough to be interpretable.

Shapley values have also inspired SHAP (SHapley Additive exPlanations) [Lundberg, 2017], an approach at the intersection of linear LIME and Shapley values that serves to determine feature contribution to the output. The linear LIME framework builds a local linear model to approximate the behaviour of the black-box model in the neighbourhood

of an instance. SHAP also attempts to build a local linear model to approximate model behaviour. More formally, the SHAP framework learns an additive feature attribution model g:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j \cdot z_{j'} \tag{2.14}$$

where $z' = 0, 1^M$ is a binary coalition vector, M is the maximum coalition size, and $\phi_0, \ldots, \phi_M$ are Shapley values for features. The high-level logic of SHAP is similar to that of the LIME framework – first, a dataset of local data points is generated, labelled by the black-box model weighted according to selected distance metrics, and then used to fit a local linear model. To generate local data in SHAP, random coalitions are sampled, and for each coalition z' a new data sample is created from the original x using mapping $h_x(z') = z, h_x : 0, 1^M \to \mathbb{R}^p$ in the following way:

- For present features ($z'_i = 1$) value of the feature is copied from the original instance x: $h_x(z'_i) = x_i$
- For absent features ($z'_i = 0$), values are mapped to feature values of a randomly selected instance t: $h_x(z'_i) = t_i$

In LIME, data instances are weighted based on their similarity with the original instance x. For SHAP, however, instances are weighted depending on how many of their features have been randomised – instances where very few or many features have been changed receive large weights. Intuitively, such instances are more informative about the effect of a single feature, because they show a change in prediction for localised changes. Formally, a kernel function is used to calculate weights:

$$\pi_x(z') = \frac{M - 1}{\binom{M}{|z'|}|z'|(M - z')} \tag{2.15}$$

where M is the maximum coalition size and $|z'|$ is the number of present features in coalition z'.

Finally, once the dataset is generated and weights calculated, the linear model is trained by optimising the following loss function:

$$L(f, g, \pi_x) = \sum [f(h_x(z')) - g(z')]^2 \cdot \pi_x(z') \tag{2.16}$$

This corresponds to the loss function used to train the linear LIME model, with $\Omega(g) = 0$.

Local surrogate methods are flexible and often applicable to any data type and black-box model. Additionally, using local approximation, approaches such as otherwise infeasible approaches like SHAP can be adapted to high-dimensional data. However, Alvarez-Melis and Jaakkola [2018] discovered that LIME can generate drastically different explanations

for very similar instances. Authors tested the robustness of LIME by comparing the generated explanations for original and blurred images in the MNIST dataset, and reported that LIME is sensitive to small changes in the input. This can make the approach unstable and unreliable.

2.3.3 Global Surrogates

Global surrogate is an interpretable model which is trained to approximate the behaviour of the black-box model. Explanations can then be obtained by analysing the surrogate model. Most surrogate-based approaches are model-agnostic – they do not depend on the specifics of the black-box model and can be used across different model types. Global surrogate models have to ensure they are approximating the decisions of a black-box model, while also remaining simple enough to be easily understandable. In other words, they need to achieve a balance between two important metrics: *fidelity* and *interpretability*.

Although any interpretable algorithm can be used as a surrogate model, most often the choice falls on decision trees, due to their simplicity. However, unlike some black-box models such as neural networks, decision trees tend to overfit on training data and generalise poorly to unseen data. Methods which use decision trees as surrogate models address this problem in different ways. Bastani et al. [2017] trains a binary decision tree to approximate decisions of a neural network, and proposes an active sampling strategy to avoid overfitting. The tree is constructed greedily, by iteratively splitting the leaves, while the best split is at every step determined based on the data acquired by the active sampling method. Active sampling ensures that data is sampled from the paths that require more information to improve accuracy. On the other hand, Frosst and Hinton [2017] offers an alternative solution to the overfitting problem by training a binary soft decision tree (SDT) as a surrogate for a neural network model. Binary SDT is a binary tree where each inner node is assigned a weight w_i and a bias b_i, and each leaf learns a distribution Q_l. At any inner node, the probability of taking the right path is defined as:

$$p_i(x) = \sigma(xw_i + b_i) \tag{2.17}$$

where x is the input instance, and σ is the sigmoid function. The tree is trained with mini-batch gradient descent, using the dataset of instances labelled by the black-box model. Unlike the greedy training approach, which iteratively splits the tree, SDT size is determined before training, and all parameters are updated simultaneously. A trained SDT can be analysed to extract reasons for making a specific decision. Each node in the SDT represents a specific filter, and by following the decision path down the tree from root to the leaf for a specific instance recovers all filters which contributed to the decision.

Aside from different forms of decision trees, decision sets have also been used as a

Table 2.2: Metrics used for evaluating surrogate rule set $\mathcal{R}$ in Lakkaraju et al. [2017].

Property	Property Explanation	Formula
disagreement($\mathcal{R}$)	Infidelity of the surrogate model	$\sum_{i=1}^{M} \lvert x \lvert x \in D, x \text{ satisfies } q_i \wedge s_i \text{ and } \mathcal{B}(x) \neq c \rvert$
ruleoverlap($\mathcal{R}$)	Overlap in conditions of rules	$\sum_{i=1}^{M} \sum_{j \neq i}^{M} overlap(q_i \wedge s_i, q_j \wedge s_j)$
cover($\mathcal{R}$)	Number of instances coverd by the $\mathcal{R}$	$\lvert x \lvert x \in D, x \text{ satisfies } q_i \wedge s_i \text{ for some } i \in 1 \dots M \rvert$
size($\mathcal{R}$)	Number of rules	$\lvert R \rvert$
maxwidth($\mathcal{R}$)	Maximum number of predicates in a rule	$\max_{e \in \cup(q_i \wedge s_i)} width(e)$
numpred($\mathcal{R}$)	Total number of predicates in $\mathcal{R}$	$\sum_{i=1}^{M} width(q_i) + width(s_i)$
numdsets($\mathcal{R}$)	Number of unique subspace descriptor rules	$\lvert dsets(\mathcal{R}) \rvert$, where $dsets(\mathcal{R}) = \cup_{i=1}^{M} q_i$
featureoverlap($\mathcal{R}$)	Feature overlap in outer and inner rule parts	$\sum_{q \in dset(\mathcal{R})} \sum_{i=1}^{M} featureoverlap(q, s_i)$

surrogate representation. Lakkaraju et al. [2017] learn multiple decision sets which approximate the behaviour of a neural network, each of which is tasked with interpreting a particular region of the feature space. The method uses a two-level if-then decision set which holds multiple rules, each consisting of an outer rule defining the subsection of feature space and an inner rule which defines the decision logic. Formally, each rule is represented as a tuple (q_i, s_i, c), where q_i and s_i are conjunctions of predicates in the form (*feature*, *operator*, *value*) and c_i is the assigned class. q_i refers to the conjunction of predicates in the subspace descriptor of the outer rule, while (s_i, c_i) together represent the inner decision logic. To evaluate the decision set $\mathcal{R}$, $|\mathcal{R}| = M$, concerning interpretability and fidelity, eight metrics are proposed (Table 2.2). The objective function is defined by taking into account all the quantities above, and approximate local search is used to find a decision set which approximates the black-box model well, while remaining simple and interpretable.

Global surrogates represent a flexible method for explaining any black-box model. Various interpretable algorithms can be used for the surrogate, adapting easily to different preferences and requirements. However, these models can struggle to balance between complexity, which is necessary for successful approximation of a black-box model and simplicity, which is needed to ensure interpretability. Additionally, global surrogate models require training another model aside from the original black-box one, which can be time-consuming and expensive.

2.4 Explainable Reinforcement Learning (xRL)

The field of explainable RL (xRL) gathers the methods for interpreting RL agents [Vouros, 2022, Milani et al., 2024]. xRL is a fast-growing field that has seen numerous contributions in the last decade. However, as a fairly new research discipline, there is no unified taxonomy of xRL methods. In this work, we classify the xRL methods according to the taxonomy proposed by Puiutta and Veith [2020], which corresponds to the xAI taxonomy presented in Section 2.3.

Firstly, xRL methods can be divided according to their scope into local and global methods. Global explanation methods aim to distil the black-box policy into a more inter-

Table 2.3: Categorisation of methods for explainable RL based on the xRL taxonomy (Section 2.4). Methods are classified based on their scope, reliance on the black-box model, time of explanation and intended audience.

Method	Subcategory	Approach	Scope	Model-reliance	Explanation time	Audience
Inherently Interpretable	Decision trees	Chapman and Kaelbling [1991] Uther and Veloso [2000] Roth et al. [2019] Paleja et al. [2022]	Global	Model-specific	Intrinsic	Developers/Experts
	Enhanced NNs	Annasamy and Sycara [2019] Mott et al. [2019]	Global	Model-specific	Intrinsic	Developers/Experts
	Custom Models	Custode and Iacca [2022]	Global	Model-specific	Intrinsic	Developers/Experts
Global surrogates	Decision trees	Liu et al. [2018] Coppens et al. [2019] Guo and Wei [2022]	Global	Model-specific	Intrinsic	Developers/Experts
	Rule-based system	Skirzyński et al. [2021]	Global	Model-specific	Intrinsic	Developers/Experts
	Custom models	Verma et al. [2018] Verma [2019]	Global	Model-specific	Intrinsic	Developers/Experts
Summaries		Zahavy et al. [2016] Zrihem et al. [2016] Amir and Amir [2018] Huang et al. [2018] Topin and Veloso [2019] Sequeira and Gervasio [2020] Huber et al. [2021] McCalmon et al. [2022]	Global	Model-agnostic	Post-hoc	All
Feature Importance	Saliency maps	Greydanus et al. [2018] Puri et al. [2019]	Local	Model-agnostic	Post-hoc	Developers/Experts
		Wang et al. [2016]	Local	Model-specific	Post-hoc	Developers/Experts
	Shapley values	Rizzo et al. [2019]	Local	Model-agnostic	Post-hoc	Developers/Experts
		Liessner et al. [2021]	Local/Global	Model-agnostic	Post-hoc	Developers/Experts
Reward explanations		Juozapaitis et al. [2019]	Local	Model-agnostic	Post-hoc	Developers/Experts
		Jenner and Gleave [2022]	Global	Model-agnostic	Intrinsic	Developers/Experts
		Huang et al. [2019]	Global	Model-agnostic	Post-hoc	All
Outcome explanations		Madumal et al. [2020] van der Waa et al. [2018b]	Local	Model-agnostic	Post-hoc	All
		Gajcin et al. [2021]	Global	Model-agnostic	Post-hoc	All

pretable format, such as a rule-based system. The questions about the inner workings of the model can then be simply accessed from the interpretable proxy model. Local explanations, on the other hand, explain why a specific decision has been reached in a specific state. Additionally, xRL methods can be post-hoc or intrinsically interpretable. Intrinsic interpretability means that the model is interpretable as is, often because the reasons for its behaviour can be accessed directly from the model. On the other hand, post-hoc interpretable methods aim to build explanations on top of the black-box model without opening it. The xRL explanation method can be model-specific, if it applies only to specific black-box models, or model-agnostic if it can explain the behaviour of any black-box policy. Additionally, as this thesis focuses on user-friendly explanations, we make a distinction between explanations for different stakeholders of RL systems. The role of explainability also depends on the different stakeholders using the RL system [Bekkemoen, 2024]. In this work, we distinguish between the explainability needs of RL developers, experts working together with RL systems and non-expert end users of these systems.

In the remainder of this section, we cover some of the most notable approaches for explainable RL. In Table 2.3, we provide an overview and taxonomy of all the listed approaches.

2.4.1 Inherently Interpretable Models

While black-box DRL methods have shown remarkable success in learning complex policies, in recent years, substantial research has been developed in learning policies interpretably. Instead of relying solely on neural networks, inherently interpretable methods aim to represent the decision-making process in a human-readable way without sacrificing performance.

Decision trees have been the most popular interpretable structure used to represent RL policy. For example, Chapman and Kaelbling [1991] proposed the G Algorithm, which iteratively splits the state space according to the statistical difference in rewards obtained. However, the size of decision trees required to represent complex policies can often hinder their interpretability. To battle this issue, Uther and Veloso [2000] proposes a Lumberjack algorithm to generate a linked decision forest. Linked decision forests are modelled after decision trees; however, they do not need to repeat the same internal subconcepts in the tree multiple times. Similarly, Roth et al. [2019] aims to build a high-performing decision tree while constraining its size to ensure interoperability. The authors propose to build the tree by only splitting the nodes when the estimated future reward substantially increases. Paleja et al. [2022] proposes Interpretable Continuous Control Tree (ICCT), a tree-based RL approach that can be optimised through gradient descent. Authors evaluate ICCT in autonomous driving tasks and find that it can achieve performance comparable with DRL methods while reducing the number of parameters 300 - 600 times.

While decision trees can be used to generate interpretable policies, they struggle with the trade-off between size and performance and as such as rarely suitable for complex, high-dimensional problems. For this reason, some inherently interpretable methods keep relying on a neural network while enhancing it with additional components that might uncover its decision-making process. Annasamy and Sycara [2019] proposes an interpretable neural network design that can be used to learn the Q function. The network relies on attention, key-value memories, and reconstructive embeddings to provide global explanations of the agent's decisions. The network is modelled after the traditional DQN architecture, with a set of convolutional layers used to extract features from an input image state. Instead of the final layer that maps features to Q values, the authors introduce a key-value store to which the network learns to attend. This can be used to cluster states based on their expected Q-value. The authors show that the network can learn interpretable representations, such as saliency maps that can be used to better understand its behaviour. With appropriate exploration, authors find that this approach can perform comparably with other DRL approaches. However, the authors also note that the learned features extracted by the network are shallow and that the network can easily overfit the training dataset. Mott et al. [2019] proposes a soft attention model for RL, which teaches the agent to focus on the important aspects of the task. The attention mechanism shows which parts of the state the space agent relies on to make a decision. The proposed interpretable model consists of a vision core, which processes

the input image, an attention head that queries the agent, and an LSTM which produces the queries and generates the final decision. The attention head passes a large input tensor through the attention mechanism and uses the answer as the output. The proposed system supports multiple attention heads, allowing the model to focus on different aspects of the input. Authors evaluate the model on Atari tasks and find that it can achieve performance comparable to traditional DRL methods, while being able to correctly attend to key aspects of gameplay, such as enemies and moving objects.

Custode and Iacca [2022] recognise that inherently interpretable methods such as decision trees struggle to process high-dimensional raw data in non-tabular formats. For this reason, they split the RL training process into two parts: 1) extracting high-level features from raw data and 2) learning the optimal policy from high-level features. The authors propose using a vision module with convolutional kernels to represent the first component and a decision tree to represent the second. The pipeline, consisting of the two components, is optimised in an end-to-end fashion using evolutionary algorithms. The approach is evaluated in Atari environments, where it achieved satisfactory performance in a deterministic setting. However, the approach struggles in an environment with stochastic frame-skipping, indicating that it lacks robustness to noisy transitions in the environment.

While there are methods that use interpretable structures, such as decision trees to represent a policy, their performance and interpretability can be limited in complex tasks. On the other hand, some approaches enhance neural networks with additional interpretable components. While this might make it easier to interpret the network's decisions, the underlying decision-making process remains obscured.

2.4.2 Global Surrogates

One of the main approaches to global explanations in RL is distilling already trained black-box policies into intrinsically interpretable models. This often consists of training a black-box DRL model and transforming it into an interpretable AI structure, while retaining performance. The difference between inherently interpretable models and global surrogates is that global surrogates aim to imitate the decision-making process of a black-box model in an interpretable way, while inherently interpretable methods learn an interpretable policy from scratch. Most often, an RL policy is distilled into a tree structure, although several works introduce novel interpretable frameworks for representing RL policies.

Coppens et al. [2019] employs a Soft Decision Tree (SDT) to predict actions of the DRL policy. SDT is a binary tree with predetermined depth, where each node i consists of a perceptron with weight w_i and bias parameter b_i. SDT is trained to predict the optimal action from state features using a dataset of state-action pairs $(s, \pi(s))$ extracted from DRL policy's π interactions with the environment. Explanation for an individual decision is obtained by traversing the trained SDT and observing which spatial features were considered on the decision path. Even though SDT attempts to mimic the DRL policy,

its performance depends on the tree depth and fails to replicate that of the black-box model. Increasing the depth of the tree leads to a larger expected reward in the Mario AI benchmark task [Karakovskiy and Togelius, 2012], but in turn, decreases interpretability since decision paths become longer and more difficult to understand. Guo and Wei [2022] extends this approach by adding a visualisation module that can extract visual explanations from the SDT and evaluate this approach on an aircraft simulator. Similarly, Liu et al. [2018] proposes a regression tree with linear models in leaves, named Linear Model U-Tree (LMUT). In contrast to Coppens et al. [2019], LMUT is not used to predict the agent's next action, but rather represents the black-box policy implicitly, by approximating its Q function. Authors additionally propose an online learning algorithm that combines active interactions with the environment and learning of the Q-function directly from the black-box model. Instead of using state-action pairs as training data, a sequential dataset of interactions $(s_t, a_t, r_t, s_{t+1}, Q(s_t, a_t))$ is gathered from interactions with the environment.

In contrast to tree-based approaches, Verma et al. [2018], Verma [2019] do not use a known structure, but introduce an interpretable RL-specific programming language. Each policy is represented as a program in a high-level, domain-specific language. Verma et al. [2018], Verma [2019] also propose NDPS (Neurally Directed Program Search), a method for searching the program space. NDPS uses a trained black-box policy as an oracle, and attempts to find an interpretable program that approximates this policy best. NDPS consists of iterative updates of the current estimate e of the oracle policy e_N. In each iteration, a neighbourhood of policies structurally similar to the current estimate e is tested, and the one most similar to the oracle is promoted to the current estimate. The iterative process converges after no change to the policy has been made for several iterations. The interpretable policy is evaluated in a car-racing environment and achieves comparable performance to the black-box DRL policy.

Skirzyński et al. [2021] introduce AI-Interpret, an algorithm to distil a black-box policy into decision rules. AI-Interpret aims to identify and present the key aspects of a black-box model in a human-readable way, such as a flowchart. AI-Interpret algorithm clusters demonstrations of a policy and uses Bayesian imitation learning to identify the rules that describe these clusters. Authors conduct an extensive evaluation of AI-Interpret, including a user study, and find that decision rules and flow charts generated by AI-Interpret can help users better understand and create strategies in three sequential decision-making tasks.

Global surrogate models provide an alternative to black-box solutions, but their capabilities are usually limited by the trade-off between interpretability and performance. Moreover, they require additional time and resources for training a separate interpretable policy alongside the original, black-box one.

2.4.3 Summaries

One of the most notable model-agnostic global methods is summary generation. Summaries are usually directed towards non-expert users of the system and aim to showcase specific parts of the agent's behaviour to give the user an idea about the agent's capabilities and limitations. Summaries can also be beneficial to experts and developers as they can emphasise situations where the agent exhibits suboptimal behaviour.

Amir and Amir [2018] proposed HIGHLIGHTS, a method for selecting the most important states to present to the user. A state is considered to be important if making a wrong decision in the state leads to a significant decrease in expected reward. Formally, the importance of a state s is calculated as the difference between the largest and the smallest expected reward attainable from s:

$$\mathcal{I}(s) = \max_{a \in \mathcal{A}} Q(s, a) - \min_{a \in \mathcal{A}} Q(s, a) \tag{2.18}$$

A limited number of transitions relating to states with the highest importance scores are presented to the user in the form of a summary. As the authors also acknowledge, this approach comes with significant drawbacks, such as sensitivity to the number of actions.

Sequeira and Gervasio [2020] addressed some limitations and expanded the HIGHLIGHTS algorithm. The authors propose four heuristics that determine whether a state is interesting enough to be included in the summary:

- *Frequency analysis* detects situations which are frequently or infrequently visited by the agent. Frequency can be calculated for states: $n(s)$, state-action pairs: $n(s, a)$ or transitions: $n(s, a, s')$.
- *Execution certainty* refers to an agent's confidence in its decision in a specific state. Situations of high confidence could indicate the agent's capabilities, while those in which the agent is indecisive could showcase its limitations.
- *Transition value* is a measure of *favorableness* of the situation. Favourable states are those whose state value $V(s)$ is higher than for states in their immediate surroundings, while unfavourable states see the local minimum of $V(s)$.
- *Sequence analysis* uncovers most likely paths from interesting states (extracted according to the previous three conditions) to local maxima states in terms of the state value function.

The authors conducted a user study to evaluate how summaries based on different interestingness elements affect user understanding of an agent's policy. The study finds that presenting users with summaries of suboptimal behaviour where the agent is not confident increases users' understanding of the agent's capabilities.

Huber et al. [2021] extent HIGHLIGHT-DIV algorithm to include local explanations in the form of saliency maps that show the parts of the state agent is focusing on in important

states. Saliency maps are generated using Layer-wise Relevance Propagation (LRP), which assigns utility to each input component. The authors conducted a user study to explore how augmented summaries affect user understanding. The study reported mixed results – while adding local explanations helped users to identify the agent's strategies, it did not significantly affect the overall user understanding of the agent's policy.

Huang et al. [2018] proposes an approach for extracting critical states from the robot's execution to present to the user. The aim of the approach is for the user to build a correct mental model of the robot's behaviour. Critical states can be identified either as states in which the policy strongly prefers a small set of actions over others or those in which acting randomly produces much worse results than following the policy. The authors explore how seeing critical states affects trust in a user study and find that it helps the user calibrate their trust in the robot. However, the authors also note that one limitation of the approach is the increased mental load on the end-user.

Alternative approaches to generating summaries include clustering the state space into similar groups of states and abstracting the decision-making process into an interpretable format such as a decision tree or a graph. Topin and Veloso [2019] introduces Abstract Policy Graphs (APG), which are Markov chains of abstracted states. APG offers insights into which states are treated similarly by the model and provides an interpretable way to see how transitions between different state groups are achieved. Authors also propose APG Gen, an algorithm that builds an APG starting from one abstract state encompassing the entire state space by iteratively splitting abstract states. APG Gen requires a learned value function and a set of demonstrations to build the graph. The states are grouped according to the Feature Importance Ranking Measure (FIRM). Finally, the authors show that APG Gen runs in time increasing quadratically in the number of features and linearly in the number of demonstrations.

Similarly, McCalmon et al. [2022] proposes CAPS, an approach to generating an abstract policy graph with natural language descriptions. CAPS requires user-defined natural language predicates that describe simple behaviours such as "car goes left". CAPS then partitions the state space into an abstract representation by clustering similar states together using the CLTree algorithm. A policy graph where nodes are state clusters is then obtained, and user-defined predicates are used to describe transitions between the nodes. The authors found that CAPS performs comparatively on five benchmark tasks compared to the DRL policies. Additionally, the authors performed a user study and found that CAPS helps users significantly better understand the behaviour of RL agents.

Zahavy et al. [2016] analyses the state representation learned by DQN and uncovers temporal abstractions of state space as well as hierarchical options. The method relies on using t-SNE to visualise the activations in the last hidden layer of the DQN. Additionally, hand-crafted features (such as player position) are recorded and used to colour-code the points on the t-SNE map, depending on their feature value. Before applying t-SNE to activations of the last hidden layer, PCA is used to reduce the dimensionality from 512 to

50 features. By analysing the t-SNE map, the authors extract hierarchical aggregations of state space and identify common attributes of state clusters. Zrihem et al. [2016] upgrade this method by replacing hand-crafted features and manual analysis of the t-SNE with an automated approach. Zrihem et al. [2016] uses the t-SNE map to cluster similar states and uncover the structure of t-SNE. The result of the clustering is a SAMDP (Semi-Agregated Markov Decision Process) – an approximation of the MDP of the process. SAMDP is a directed graph, whose nodes are clusters of t-SNE points with labels C, and arcs are defined by a probability matrix P. The k-means algorithm is used to generate spatial-temporally simple clusters from t-SNE points, based on three proposed measures:

- *Inertia*: $I = \sum_{i=1}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$
- *Entropy*: $e = -\sum_{i=1}^{|C|} |C_i| \sum_{j=1}^{|C|} P_{ij} \log P_{ij}$
- *Intensity factor*: $F = \sum_{j \in C} \frac{P_{jj}}{\sum_{j \in C} P_{ji}}$

Generated SAMDP is laid over the t-SNE map, with each node of SAMDP positioned over the average location of its corresponding cluster.

Summary methods provide a way of condensing an agent's behaviour and locating situations where the agent acts in an unexpected or otherwise interesting way. At the moment, however, these methods rely on heuristics to determine whether a state is important enough to be included in the summary. Additionally, while useful for visual tasks, summarising might not be an appropriate technique for explaining tasks with other types of input, such as personal assistants.

2.4.4 Feature Importance Methods

Most local explainability methods aim to explain a decision by identifying which state features contributed to it. This field of research corresponds to feature importance methods developed for supervised learning, which explore how input features are responsible for the prediction. Many of the methods have been initially developed for supervised learning and later adapted to RL tasks.

Saliency maps have originally been used to interpret decisions of neural networks for supervised learning tasks [Simonyan et al., 2013], but have also been applied as an out-of-the-box solution to the problem of explainable RL. Saliency maps explain why a specific action was chosen by the agent by highlighting specific parts of the image that the agent focused on while making the decision. There are two approaches to generating saliency maps: *perturbation-based* and *gradient-based*. Perturbation-based methods distress parts of the image and compare the model's decision before and after alteration, to assess the importance of a feature. This can mean either removing, blurring or in another way altering the part of the image. Intuitively, an agent's decision changes significantly after removing a feature, and then it is considered important. Gradient-based methods, however, rely on calculating the Jacobian of the neural network with respect to the input to see how much specific parts of the image impacted the decision.

Gradient-based methods require access to the model's internal parameters and structure, while perturbation methods use the model as an "oracle" and only require access to the model's output.

Greydanus et al. [2018] used perturbation-based saliency maps to explain agents' actions in Atari games. To determine their contribution to a specific action, a Gaussian blur was added to the image features. The approach was used to expose how agents learn interesting strategies, such as tunnelling in *Breakout* environment. Tunnelling refers to a situation where the agent aims the ball through the brick wall to create a tunnel and then propels the ball to the top of the screen, allowing it to bounce between the ceiling and the brick wall, collecting dense rewards. Saliency maps uncovered that while creating the tunnel agent attends to the top of a brick wall, where its future rewards lie. Additionally, Greydanus et al. [2018] applies saliency maps to uncover overfitting, in a situation where the agent learns to rely on a feature which is correlated with the optimal action but does not cause it. An overfitting scenario is created by adding hint pixels, representing most likely expert action, to the raw Atari state. Although the overfitting agent shows good performance, saliency maps uncover that it bases its decisions on the added hint feature. Finally, Greydanus et al. [2018] also showcased the potential benefits of saliency maps for debugging. After observing the suboptimal behaviour of the agent Pacman environment, the authors apply saliency maps to uncover that the agent was not attending to the ghosts, resulting in an inability to avoid them. Greydanus et al. [2018] recognise that capturing this kind of faulty behaviour can inspire developers to re-examine the reward function that the hat agent is following. Similarly, Puri et al. [2019] proposed SARFA (Specific and Relevant Feature Attribution), a method for generating perturbation-based saliency maps and applied it to explain the moves of a chess agent. In this work, perturbation is performed by removing a piece from the chessboard. Action is further explained by comparing the difference in the expected reward of taking the action in the original and perturbed state. SARFA addresses two limitations of previous perturbation-based methods – *specificity* and *relevance* and uses their harmonic mean to calculate feature saliency. Specificity means that the saliency of a feature should only focus on the action being explained. Perturbation of a feature should only affect the saliency of that feature if it significantly affects the action being explained, while not impacting other actions. Relevance addresses the need for saliency of a feature to only include those perturbations whose effects are relevant to the action being explained. In other words, perturbations should have a significant effect on the action in question, without impacting other actions, for the feature to be considered important.

Gradient-based saliency maps have not yet found broad applications in the field of RL. Few works utilise saliency maps to demonstrate and evaluate the behaviour of newly proposed methods. Wang et al. [2016] introduces Duelling Networks, which separately estimate the state-value function and advantage of each action, by implementing two different streams in the neural network structure. Duelling networks can distinguish between states in which making a good decision is crucial and those where there is no significant difference between actions. Saliency maps are used to demonstrate and

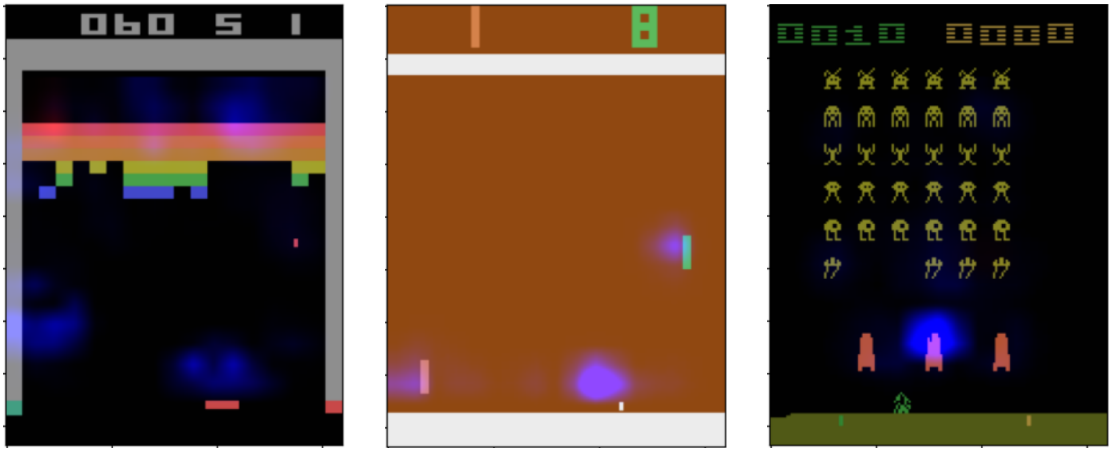


Figure 2.2: Saliency maps for explaining decision of Atari agents for *Breakout*, *Pong* and *Space Invaders* environments, obtained with algorithm provided in Greydanus et al. [2018]. When deciding the pictured states, the agent focuses on the highlighted areas in the image.

validate the behaviour of duelling networks in a driving environment, showing that in critical situations, advantage stream attends to nearby cars, while if the road is clear, it does not attend to any features, as its decision is not important.

While they can highlight parts of the input responsible for a decision, research has found that saliency maps can be too finely grained for non-expert users to be able to recognise important factors. Huber et al. [2019] aims to make saliency maps more focused and reduce the user's cognitive load by selecting only the most important contributions as a part of explanations. Authors use Layer-wise Relevance Propagation (LRP) to identify which neurons in the convolutional layers contributed the most to a decision. Authors propose an argmax approach with selects only the most important neurons in each layer and creates sparse and focused saliency maps.

Similar to saliency maps, Shapley values have also been translated from supervised learning to RL to assign importance to state features. Rizzo et al. [2019] uses Shapley values to explain the connection between features and an agent's decisions in a traffic scenario. Authors use the KernelSHAP algorithm [Lundberg, 2017] to estimate the effect of traffic flow on an agent's decisions in a complex roundabout scenario. Similarly, Liessner et al. [2021] proposes RL-SHAP, a visual representation of feature influence on an agent's actions. RL-SHAP can explain in each timestep which features contributed negatively and positively to a decision. Additionally, RL-SHAP can also generate a global explanation showing how different features contributed to actions along the episode. The authors evaluate RL-SHAP in an autonomous driving scenario.

Feature importance methods offer a way of understanding a decision based on the state features, often in a visual way. They are easily adaptable to RL and thus can take advantage of the large body of work in feature importance methods in IML. However, an RL agent's decisions are often not motivated only by its current state but can depend on its objectives, goals or outside events.

2.4.5 Reward Function Explanations

The reward function is an essential part of the RL process, and it guides the learning process. A misspecified reward function can lead to policy failure, missed objectives or reward hacking, where the agent learns to game the reward without achieving the intended goal. For this reason, understanding how a reward function affects learning is necessary for developers to debug and verify behaviour. Additionally, non-expert and expert users of RL systems might need to understand reward functions to understand whether they align with their preferences. This issue is especially prevalent in multi-objective scenarios, where an agent needs to balance different, often conflicting goals.

Jenner and Gleave [2022] utilises the fact that many reward functions are equivalent and encourage the same behaviour to enable reward function explanations. For example, scalar multiplication of the reward function by a positive factor does not change the agent's behaviour. The authors propose a reward preprocessing algorithm approach which simplifies a reward function to make it more interpretable while maintaining the same agent behaviour. To explain the reward function R, the authors search for the equivalent reward function R' that is equal to the potential shaping. According to the potential shaping [Ng et al., 1999], two rewards R and R' are equivalent if there exists a function $\Phi : \mathcal{S} \to \mathbb{R}$ so that:

$$R'(s, a, s') = R(s, a, s') + \gamma \cdot \Phi(s') - \Phi(s) \tag{2.19}$$

Authors then search for the potential shaping function Φ while also ensuring the interpretability of the reward function R' by optimising two metrics: 1) sparsity and 2) smoothness. By optimising sparsity, users can focus only on a few states with non-zero rewards to understand them. Smoothness aims to minimise fluctuations between adjacent transitions so that the user can observe the reward as a trend in time.

Sukkerd et al. [2020] propose a method for generating contrasting explanations that show how the trade-off between different objectives contributed to the choice of the action. An explanation is presented in terms of quality attributes (QA), which represent specific objectives (e.g. execution time, energy consumption...). To provide contrasting explanations, the authors propose comparing the policy with other policies on the Pareto front. Specifically, subset Pareto-optimal policies are chosen, such that each one π_i improves upon the original policy in one of the QAs QA_i. Specifically, the "i-improving policy" π_i is the one that reaches value θ_i for quality attribute QA_i, which has a magnitude δ improvement of the same QA in the original policy. Policy π_i is learned by using a mixed integer linear program (MILP) to optimise an MDP with a reward function that prioritises QA_i, with the constraint that the expected value of the quality attribute QA_i for the solution is less than or equal to θ_i. After comparing the original policy with a subset of Pareto-optimal policies, natural language contrasting explanations are generated in the form:

I could improve on QA_i by following policy π_i. However, this would worsen

the QA_j by the amount qa_j. I chose action because improvement in QA_i is not worth the deterioration of QA_j.

Similarly, Juozapaitis et al. [2019] converts the single-objective problem into a multi-objective one by decomposing the reward into semantically meaningful components. Furthermore, they explain the difference between the two actions in one state in terms of reward components. Formally, authors split the reward into a reward vector $\vec{R} : S \times A \rightarrow \mathbb{R}^{|C|}$ where C represents a set of different reward types. Similarly, the Q-value can also be split into individual components Q_c^π, where each component corresponds to one reward type. Authors then propose a novel learning algorithm *drQ*, inspired by Q-learning, which supports individual rewards and updates separately each Q_c^π. The difference between two actions a_1 and a_2 in state s is then explained by analysing the difference in Q-values given by reward difference explanation (RDX): $\Delta(s, a_1, a_2) = \vec{Q}(s, a_1) - \vec{Q}(s, a_2)$. However, depending on the number of components, RDX can be large in dimension and difficult to interpret. To ensure a more compact representation, authors propose to extract a minimal sufficient explanation of MSX in the form of a tuple (MSX^+, MSX^-) where MSX^+ represents a minimal set of critical positive and MSX^- negative reasons for explaining the preference for a_1 over a_2.

Huang et al. [2019] aims to reduce the time users need to create an accurate mental model of an agent's objectives by curating a dataset of the most informative observations that can be presented to the user. The most informative scenarios are those in which the agent making a trade-off between objectives is the most noticeable. Authors use the algorithmic teaching approach to model how users formulate their mental models. The human user is modelled with a prior $P(\theta)$, and the goal is to find the sequence of environments $E_1, \ldots, E_k$ such that the person observing them maximises the probability of building the correct mental model θ^*. The authors propose an exact inference approach modelled after inverse RL and multiple approximate inference approaches, which are necessary to account for uncertainty in human inference.

The reward function is one of the most important components of the RL framework, and understanding how it affects behaviour is essential to both developers and users of RL systems. However, there is scarce research in this area. Additionally, current approaches often rely on evaluation in simple environments and do not include user studies. Understanding the best way in which these explanations can be presented to the users is an open research question.

2.4.6 Outcome Explanations

Unlike supervised learning, which deals with one-step predictions, RL specialises in sequential decision-making. For this reason, an agent's decisions can often be motivated by some delayed outcome or a goal it is pursuing. Various explainability methods have been developed to explain an agent's decisions through the lens of an expected outcome.

To compare two policies globally, Gajcin et al. [2021] explores their outcomes to generate contrastive explanations. The two policies are assumed to be trained to perform the

task adequately and have the same abilities, but differ in their preferences. The authors propose a global approach for generating contrastive explanations for comparing the preferences of two policies, which consists of three steps. Firstly, a dataset of disagreement states, in which two policies choose different actions, is collected by unrolling policies in the environment. Secondly, the dataset is filtered so that only disagreement data, which is a consequence of the agent's different preferences, and not abilities, is obtained. To do this, only those states are kept in which both policies are confident in their choice of action, both policies similarly evaluate the state, in terms of a state value, and after separately executing policies in the environment for a set number of steps, both policies arrive at similarly good states. Intuitively, situations where both policies see and execute the same potential in the environment, but strongly disagree on their preferred way to do so, are collected, indicating that their disagreement is a consequence of their opposing preferences. Finally, filtered preference-based disagreement data is used to compare outcomes of following policies from a state of disagreement to uncover in what kind of states, in terms of state features, policies prefer to end up.

van der Waa et al. [2018b] generates contrastive explanations by comparing the expected outcomes of two actions in a specific state. Using a hand-crafted interpretable set of features C, states are transformed into a user-understandable form with mapping $k : S \rightarrow C$. Similarly, hand-crafted outputs are used to model the consequences of action in a state: $t : C \times A \rightarrow Pr(O)$. Outputs O describe the agent's position in understandable terms – in a grid-world environment, for example, the agent can be at the goal, in a trap or near a forest. Authors propose a method for understanding why the agent chose one or more consecutive actions of the fact policy π_t and did not follow the alternative foil π_f. Foil policy is extracted directly from the user's question by estimating the action-value function Q_f of the foil policy π_f as:

$$Q_f(s, a) = Q_t(s, a) + Q_l(s, a) \tag{2.20}$$

Where Q_l is the user's preference obtained from the question. Formally, if the user preferred action b over a in state s, $Q_l(s, b)$ is chosen such that $Q_f(s, b) > Q_t(s, a)$. Policies π_t and π_f are both unrolled in the environment, starting in state s and their visited paths $\gamma(s_t, \pi) = (s_0, a_0), \ldots, (s_n, a_n) | T, \pi$ are recorded. To obtain a contrastive explanation, their paths are first transformed into their interpretable counterparts, using mappings t and k:

$$Path(s_t, \pi) = (c_0, o_0), \ldots, (c_n, o_n) \tag{2.21}$$

where $c_i = k(s_i), o_i = t_i(s_i, a_i) \forall (s_i, a_i) \in S, A$. To generate contrastive explanations two paths $Path(s_t, \pi_t)$ and $Path(s_t, \pi_f)$ can be compared, to uncover reasons for following π_t over π_f. Authors propose taking a set difference $Path(s_t, \pi_t) \setminus Path(s_t, \pi_f)$ to obtain outcomes unique to π_t, or calculating symmetrical difference $Path(s_t, \pi_t) \Delta Path(s_t, \pi_f)$, which would return differences of both policies.

Similarly, Madumal et al. [2020] answers the question *Why did the agent take action a instead of b in state s?* by comparing the outcomes of two potential actions. Instead of unrolling the alternative policies in the environment, however, authors use a causal model named *action-influence model*. Action-influence model is an extension of the structural causal model (SCM), adapted to the specifics of RL. SCM is a directed acyclic graph (DAG) where nodes represent variables, and arcs indicate causal relationships. Each arc is also assigned a structural equation which defines the nature and strength of the causal effect. Action-influence model extends standard SCM by assigning an action to each arc, representing a causal effect after the action is performed. It also allows for a special type of node called the reward node to represent the effect of actions and features on the reward. Nodes in the action-influence model are state features. Authors use a hand-crafted DAG and learn structural equations from transition data using linear regression. To generate contrasting explanations, causal chains of actions a and b are compared. To reduce explanation complexity, the authors propose that only minimal chains are used to represent the causal effects of an action. For a complete chain, a minimal chain consists of only the first and last arcs, including their source and destination nodes, ignoring the intermediate results. The final contrastive explanation is obtained by comparing the differences between minimal causal chains of actions a and b.

Explanations that include outcomes as explanation components can explain longer chains of events that are characteristic of RL. However, as a novel field of research, there are few methods dealing with this issue. Current methods that include agents' outcomes in explanations often require hand-crafted components describing the rules of the environment, which can be time-consuming or even infeasible to obtain, especially in high-dimensional tasks.

2.5 Counterfactual Explanations

At the moment, a substantial amount of research is focused on developing explanations for experts familiar with AI systems. However, from the perspective of non-expert users, such explanations might be too detailed and difficult to understand. End-users are more likely to be interested in more abstract explanations of the system [Dazeley et al., 2021c]. For example, consider the 2017 incident when an Uber self-driving vehicle accidentally killed a pedestrian after failing to avoid them while crossing the road [Knight, 2018]. In this situation, developers and experts might be interested in uncovering which parameters or input features contributed to the fatal failure, to be able to repair it. On the other hand, a non-expert user is more likely to be interested in high-level questions, such as *"Did the car not recognise the person in front of it?"* or *"Did the car believe it had the right of way?"* [Dazeley et al., 2021c]. Although the research in developing low-level explanations has gained momentum, user-friendly explanation methods lag. However, developing user-friendly explanations is necessary to build trust and encourage cooperation between non-expert users and black-box systems.

In this section, we explore one of the most notable examples of user-friendly explanations

Table 2.4: Some important terms in counterfactual explanations research.

Term	Symbol	Meaning
Factual (original) instance	x	Instance being explained
Original class	y	Class that x is assigned by the black-box model f; $y = f(x)$
Counterfactual class	y'	Desired class; $y' \neq y$
Counterfactual instance	x'	Instance similar to x, but classified as the counterfactual class y'
Counterfactual perturbation	δ	Perturbation that needs to be added to x to obtain x'; $x' = x + \delta$

– counterfactual explanations. Counterfactuals are local explanations which attempt to discover the causes of an event by answering the question: *"Given that input x has been classified as y, what is the smallest change to x that would cause it to be classified as an alternative class y′?"*. The explanation is usually in the form of a *counterfactual instance* – an instance x' as similar as possible to the original x, but classified by the system into an alternative class y'. The alternative class y', to which the counterfactual belongs, can either be specified or omitted, in which case the counterfactual can be an instance with any label other than y. Current methods either directly search for a counterfactual instance x' similar to x, or a small perturbation δ which can be added to x to achieve a desired outcome, in which case $x' = x + \delta$. A summary of the terminology used throughout this thesis is given in Table 2.4.

Counterfactuals address questions that users interacting with an AI system might be interested in, for example: *"Since my application was rejected, what are some examples of successful loan applications similar to mine?"* or *"How can I change my application for it to be accepted in the future?"*. This gives the user actionable advice, which they can use to improve their application. Counterfactuals are also contrastive since they compare real and imagined worlds, and they are selective, focusing only on a small number of features that need to be changed for a different outcome. All of this makes counterfactuals user-friendly explanations.

Additionally, counterfactuals correspond to the third tier of Pearl's Ladder of Causation [Pearl and Mackenzie, 2018] (Table 2.5). The first rung of the hierarchy corresponds to associative reasoning and the question *"If I observe X, what is the probability of Y occurring?"*. This rung also aligns with statistical reasoning, as the probability in question can be estimated from data. The second rung requires interventions to answer the question *"How would Y change if I changed X?"*. Finally, the third rung addresses the question *"Was it X that caused Y?"* Answering this question requires imagining alternative words where X did not happen and estimating whether Y would occur in such circumstances. Humans rely on generating counterfactuals in their everyday lives. By imagining alternative worlds, humans learn about cause-and-effect relationships between events. For example, by considering the counterfactual claim *"Had the car driven slower, it would have avoided the accident"*, the relationship between the car speed and the accident can be deduced. Additionally, humans use counterfactuals to assign blame or fault for a negative event [Byrne, 2019].

Counterfactual explanations are often imagined as belonging to the closest possible world

– an instance as similar as possible to the original one, where an alternative outcome is obtained. While most works agree on this definition of counterfactuals, they differ significantly in how to measure and search for the closest possible world. Here we select several counterfactual requirements which are often optimised by counterfactual generation approaches [Verma et al., 2020]:

1. *Validity*: counterfactual must be assigned by the model to a different class than that of the original instance. If a specific counterfactual class y' is provided, then validity is satisfied if the counterfactual is classified as y'.
2. *Proximity*: counterfactual x' should be as similar as possible to the original instance x.
3. *Actionability*: counterfactual explanation should provide users with meaningful insights into how they can change their features to achieve the desired outcome. This means that suggesting that a user should change their sensitive features (e.g. race) or immutable features (e.g. age, country of origin) is not helpful and may be offensive.
4. *Sparsity*: ideally, counterfactuals should require only a few features to be changed. This corresponds to the notion of a selective explanation, which is easier to understand [Molnar, 2019].
5. *Data manifold closeness*: Counterfactual instances rely on modifying existing data points, which can lead to the generation of out-of-distribution samples. To be feasible for users, counterfactuals need to be realistic.
6. *Causality*: while modifying the original sample, counterfactuals should abide by the causal relationship between features. For illustration, consider an example by Mahajan et al. [2019] of a counterfactual which suggests that a user should increase their education to a Master's, without adjusting their age. Even though such a counterfactual instance would be actionable and fall within the data manifold (as there probably are people of the same age as the user, but with a master's education), it would not be feasible.
7. *Recourse*: ideally, the user should be offered a set of discrete actions that can transform the original instance into a counterfactual. This property is closely related to actionability and causality – the offered sequence of actions should not change the immutable features, and should consider causal relationships between the variables, to understand how changing one feature can influence the others.

While most counterfactual search methods agree on these counterfactual requirements, they often resort to different metrics to evaluate them and different algorithms to optimise them. In the next section, we provide an overview of methods for generating counterfactuals in supervised learning.

Table 2.5: Pearl's Ladder of Causation [Pearl and Mackenzie, 2018]

Rung	Activity	Question	Example
Association	Observing	What is the probability of Y if I observed X?	What does a symptom tell us about the disease?
Intervention	Intervening	What will Y be if I change X?	If I take aspirin, will my headache be cured?
Counterfactual	Imagining	Was it X that caused Y?	Was it aspirin that cured my headache?

Table 2.6: An overview of counterfactual generation approaches for supervised learning covered in Section 2.5.1 according to the taxonomy proposed by [Guidotti, 2024].

Approach	Type	Search algorithm
Wachter et al. [2017]	Loss optimization	Gradient descent
Looveren and Klaise [2021]		FISTA [Beck and Teboulle, 2009]
DICE [Mothilal et al., 2020]		Gradient descent
DECE [Cheng et al., 2020]		Gradient descent
REVISE [Joshi et al., 2019]		Gradient descent
Mahajan et al. [2019]		Generative modelling
ReLACE [Chen et al., 2021]		RL (P-DQN [Xiong et al., 2018])
Samoilescu et al. [2021]		RL (DDPG [Lillicrap et al., 2015])
Dandl et al. [2020]	Heuristic Search	NSGA-II
Growing spheres [Laugel et al., 2017]		Growing Spheres
GECO [Schleich et al., 2021]		Genetic algorithm
FACE [Poyiadzi et al., 2020]	Instance-based	Graph search
NICE [Brughmans et al., 2024]		Iterative updates
LORE [Guidotti et al., 2019]	Decision tree-based	Tree search
FoilTree [van der Waa et al., 2018a]		Tree search

2.5.1 Counterfactual Explanations in Supervised Learning

In this section, we review some of the most notable methods for generating counterfactuals in IML. We classify the algorithms based on the counterfactual search strategy as proposed by Guidotti [2024] and recognise four categories – loss optimisation, heuristic search, instance-based and decision tree-based. Loss optimisation is the most common approach, where a loss function consisting of some or all counterfactual metrics is optimised over the training dataset to find the most suitable counterfactual. In heuristic search, an iterative search which at each step minimises a cost function is performed. Instance-based approaches search for the counterfactuals based on their relation with their surrounding instances. Finally, decision tree-based approaches distil the black-box model into a decision tree, a search for a counterfactual that is close to the original one in the tree. An overview of counterfactual search methods classified according to this taxonomy is provided in Table 2.6.

Loss Optimization

One of the most common approaches for generating counterfactuals consists of defining a loss function, usually with multiple objectives corresponding to different counterfactual metrics, and performing a search over the training dataset to find the instance that minimises the loss.

Wachter et al. [2017] In one of the first approaches for generating counterfactu-

als, Wachter et al. [2017] proposed minimizing a loss function over instances x' in the dataset:

$$L(x, x', y, y') = \lambda(f(x') - y)^2 + d(x, x') \tag{2.22}$$

where f is the black-box model, x the original instance, y label of the original instance and y' counterfactual outcome. The first term in the loss function ensures that the counterfactual is classified as close as possible to the desired class, while the second minimises the distance between the original and counterfactual instance. The distance $d(\cdot)$ is defined as a Manhattan distance, normalised by the inverse median absolute deviation:

$$d(x, x') = \sum_{j=1}^{p} \frac{|x_j - x'_j|}{MAD_i} \tag{2.23}$$

where p is the number of features. Parameter λ determines the balance between two objective quantities – for higher values of λ, counterfactuals with outcomes close to y' are preferred, while lower λ opt for counterfactuals more similar to x. Finally, gradient descent is used to minimise the loss function over the training dataset. This approach focuses only on two criteria, validity and proximity, and can result in counterfactuals which are unrealistic or require a large number of features to be changed.

Looveren and Klaise [2021] To ensure that generated counterfactuals fall within the data manifold Looveren and Klaise [2021], define prototypes of classes and direct the search towards one of these classes. This method does not require the counterfactual class to be provided, and by default generates counterfactuals of any different class. Formally, the approach requires an autoencoder ENC(x), trained to project instances onto an E-dimensional latent space and a representative sample of the dataset, labelled by the black-box model. For each class, samples belonging to that class are encoded by ENC(x) and sorted by increasing distance from the original data point x. The prototype for class i is then defined as an average over K top-most encodings of instances with label i:

$$\text{proto}_i = \frac{1}{K} ENC(x_k^i) \tag{2.24}$$

To find the counterfactual perturbation δ, the approach first detects the closest prototype of a different class:

$$j = \arg\min_{i \neq y} ||ENC(x) - \text{proto}_i||_2^2 \tag{2.25}$$

And uses it to define the prototype loss:

$$L_{proto} = ||ENC(x + \delta) - \text{proto}_j||_2^2 \tag{2.26}$$

The prototype loss pushes the search in the direction of the nearest prototype of a suitable class, thus speeding up the optimisation. The final objective function is defined as:

$$L = c \cdot L_{pred} + L_{dist} + L_{proto} \tag{2.27}$$

where $L_{pred} = \max([f(x_o+\delta)]_y - max_{i \neq y}[f(x_0+\delta)]_i, -\kappa)$ and $L_{dist}(\delta) = \beta \cdot ||\delta||_1 + ||\delta||_2^2$. L_{pred} represents the validity constraint, while L_{dist} measures the distance between the counterfactual and original instance. To optimise the objective function, a fast iterative shrinkage-thresholding algorithm (FISTA) [Beck and Teboulle, 2009] is applied. To ensure actionability, the approach also allows users to define immutable features.

DICE Extending Wachter et al. [2017], DICE [Mothilal et al., 2020] focuses on generating diverse counterfactuals, to offer the user several possible ways to change their input. Diversity objective is defined for a set of counterfactuals $C = c_1, \cdots, c_k$ as:

$$dpp\_diversity(C) = det(K) \tag{2.28}$$

where $K_{ij} = \frac{1}{1+dist(c_i,c_j)}$. A loss function is defined over a set of counterfactuals and includes proximity, validity and diversity objectives:

$$\begin{aligned} L(C, x, y') = \arg \min_{c_1,\cdots,c_k} & \frac{1}{k} \sum_{i=1}^{k} dist(f(c_i), y') \\ & + \frac{\lambda_1}{k} \sum_{i=1}^{k} dist(c_i, x) \\ & - \lambda_2 \cdot dpp\_diversity(C) \end{aligned} \tag{2.29}$$

where y' is the desired counterfactual class. Parameters λ_1 and λ_2 are used for balancing between the different objectives. The obtained loss function is optimised over the dataset using gradient descent. To ensure actionability, the user can provide constraints on immutable features. Additionally, adjustments are made to generated counterfactuals to ensure their sparsity and each continuous feature is restored back to its value in x in a greedy approach until $f(c)$ changes. Although generating a diverse set of counterfactuals offers the user more options to change their input, DICE does not ensure that generated counterfactuals fall within the data manifold and can thus result in unrealistic explanations.

DECE Extending DICE, Cheng et al. [2020] propose DECE, an interactive visual interface for counterfactual generation. DECE implements the same counterfactual search as DICE and optimises validity, proximity and diversity metrics. However, DECE enables counterfactuals to be generated not only for single instances but also for subgroups of the population. To reduce cognitive load, DECE introduces r-counterfactuals, counterfactuals restricted to only one feature change.

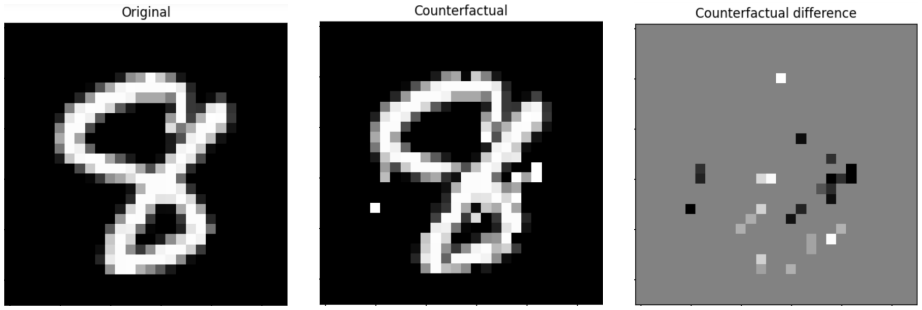


Figure 2.3: Counterfactual generation using growing spheres [Laugel et al., 2017] for MNIST dataset. **Left**: original instance classified as 8 **Middle**: the closest counterfactual instance classified as 9. **Right**: pixel difference between the original and counterfactual instances.

REVISE To ensure counterfactual instance falls within the data manifold, Joshi et al. [2019] propose REVISE, an approach which performs loss function optimisation only over likely instances:

$$x' = \arg \min_{x^* | p(x^*) > \gamma} dist(x^*, x) \tag{2.30}$$

where γ is a threshold regulating how likely an instance has to be to be considered. A generative model is trained to sample instances from the data distribution. Generative models usually assume that some underlying latent representation z is mapped to x by some deterministic function $G_\theta(z)$ parametrised by θ. Decoder of a variational autoencoder (VAE) is used to represent the generative model. The cost function can then be modified to include the generative model:

$$x' = \arg \min_{z \sim G_\theta(z)} \min_{\lambda} (l(f(G_\theta(z)), y') + \lambda \cdot c(x, G_\theta(z)) \tag{2.31}$$

The first component refers to validity constrain, while the second one measures the distance between the counterfactual and the original instance. The loss function is optimised in the latent space using gradient steps until the prediction constraint is satisfied. REVISE also addresses the problem of generating counterfactuals for causal models, where the goal is to predict the outcome after intervening on a treatment variable. Due to confounding factors, the effect of the treatment on the outcome often cannot be straightforwardly calculated. Assuming that -1 and 1 are possible outcomes, and x is the original instance with a negative outcome ($y = -1$), REVISE proposes optimising the following loss function in causal models:

$$x' = \arg \min_{z \sim \mathcal{G}_\theta(z)} \min_{\lambda} l(logy_{do(t)}, 1) + \lambda dist(x, \mathcal{G}_\theta(z)) \tag{2.32}$$

This way, the output of the model is estimated under different interventions on the treatment variable.

Mahajan et al. [2019] To ensure that counterfactuals obey the causal constraints

between features, Mahajan et al. [2019] propose a causal proximity measure that can be included in any loss-based counterfactual method. The approach assumes access to the structural causal model (SCM) of the features. SCM is a directed graph, where each node corresponds to a feature, and arcs indicate causal relationships. Each arc is assigned a structural equation, which describes how the source node influences the sink. The authors start by dividing features into two groups, U and V. Features in V have at least one parent in SCM, while features from U have none. For features in U, a standard Euclidean proximity measure is used. On the other hand, for each feature $v \in V$, where $v_{p_1}, \cdots, v_{p_k}$ are parent features of v in SCM, and $v = f(v_{p_1}, \cdots, v_{p_k}) + \epsilon$, where ϵ is independent random noise, causal proximity between features x_v and counterfactual x'_v is defined as:

$$distCausal(x_v, x'_v) = dist_v(x'_v, f(x'_{v_{p1}}, \cdots, x'_{v_{pk}})) \tag{2.33}$$

Causal proximity does not depend on the original instance x and instead ensures that features in the counterfactual instance follow causal constraints. The complete loss function is defined as:

$$\mathcal{L}(x, x') = \sum_{u \in U} dist(x_u, x'_u) + \sum_{v \in V} distCausal(x_v, x'_v) \tag{2.34}$$

Using the standard proximity measure on features without parents, and causal proximity on the others, ensures that the generated counterfactual is similar to the original instance while obeying the causal relationships between the features.

ReLACE ReLACE [Chen et al., 2021] reimagines the problem of finding counterfactuals as a search for a sequence of actions that lead from x to x'. The authors suggest that this problem can be tackled using RL algorithms. Specifically, an MDP is defined, and a policy which learns the optimal sequence of actions between x and x' is obtained using traditional RL methods. In each time step instance, x is modified using the available set of actions, and the reward is assigned to the agent corresponding to how the current state satisfies counterfactual metrics of proximity and validity. An MDP state at time step t is the current modified instance x_t. An agent can choose actions which directly alter feature values in the state. The optimisation is terminated when a state which satisfies the validity constraint is obtained. To ensure sparsity, the number of time steps and the number of changed features are limited. The reward in each time step is defined as:

$$R(x_t, a_t) = \begin{cases} 1 - \lambda(dist(x, x_t) - dist(x, x_{t-1})), & \text{if } f(x_t) \neq f(x) \\ -\lambda(dist(x, x_t) - dist(x, x_{t-1})), & \text{otherwise} \end{cases} \tag{2.35}$$

To learn a policy which generates counterfactual instances P-DQN approach [Xiong et al., 2018] is used. The authors also offer two versions of their algorithm. ReLACE-GLOBAL

assumes that one RL agent is trained to generate counterfactuals for each data instance. ReLACE-LOCAL, however, generates local counterfactuals for just one instance to encapsulate its peculiarities. ReLACE-LOCAL requires a synthetic dataset to be generated by sampling instances from the neighbourhood of x. Additionally, for faster learning, the policy learned using ReLACE-GLOBAL can be used to initialise the local algorithm. This approach is model-agnostic, and RELACE can quickly generate counterfactuals for each instance simultaneously. Additionally, it ensures that counterfactuals are sparse and close to the original instance. However, this approach does not address the problem of data manifold closeness of generated instances. Specifically, since each action forcibly changes feature values, causality constraints that exist between features may be violated in counterfactual instances.

Samoilescu et al. [2021] Similarly, Samoilescu et al. [2021] proposes using DRL algorithms to efficiently generate batches of counterfactuals in one forward pass. In contrast to Chen et al. [2021], the authors search for the counterfactual perturbation in the latent space instead of the high-dimensional state space. The encoder is trained to transform instances into their latent representation. To learn an RL policy which converts instances to their counterfactual counterparts, the DDPG [Lillicrap et al., 2015] algorithm is used. The actor network serves as a counterfactual generator, meaning that instead of outputting an action to alter the instance, the actor outputs the counterfactual itself. During training, instance x is encoded into its latent representation $z = ENC(x)$ and passed to the actor-network μ. The actor then outputs a counterfactual instance z' in the latent space, and the decoder is used to transform it back to the state space and obtain a counterfactual instance x'. To ensure actionable counterfactuals, users can define constraint vectors c, which are used during post-processing to clip feature values. The actor is trained on two loss functions:

$$\begin{aligned} \mathcal{L}_{sparsity} &= \frac{1}{|B|} \sum_{B} |\mathcal{L}_1(x, x') + \mathcal{L}_0(x, x')| \\ \mathcal{L}_{consist} &= \frac{1}{|B|} \sum_{B} (ENC(pp(x', c)) - z')^2 \end{aligned} \tag{2.36}$$

Where B is the training batch and pp is the post-processing procedure for adjusting feature values according to the constraint vector c. $\mathcal{L}_{sparsity}$ ensures sparsity by minimising the number of feature changes, while the aim of $\mathcal{L}_{consist}$ is to guide the search towards in-distribution instances. The proposed algorithm is model-agnostic and can efficiently generate batches of counterfactuals. However, the authors do not explore whether the generated counterfactuals conform to the causal rules of the environment.

Loss optimisation provides a flexible approach to counterfactual search where a variety of different metrics can be combined and even weighed according to preferences. However, loss-based approaches conflate all counterfactual metrics into one and require handcrafted weights of their importance. Additionally, virtually all approaches use different metrics to describe the same counterfactual requirements, which makes benchmarking difficult.

Heuristic Search

Heuristic search utilises an iterative approach where, in each step, a counterfactual x' is updated to minimise a predefined cost function.

Dandl et al. [2020] To address the issues of sparsity and data manifold closeness, Dandl et al. [2020] optimises four objectives. The first objective relates to the notion of validity and ensures that the counterfactual belongs to the counterfactual class:

$$o_1(f(x'), y) = \begin{cases} 0 & \text{if } f(x') = y' \\ \inf|f(x') - y'| & \text{otherwise} \end{cases} \tag{2.37}$$

The second objective addresses proximity and measures the distance between the original and counterfactual example:

$$o_2(x, x') = \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x'_j) \tag{2.38}$$

where $\delta_G(\cdot)$ is Gower distance defined as:

$$\delta_G(x_j, x'_j) = \begin{cases} \frac{1}{R_j}|x_j - x'_j| & \text{if } x_j \text{ is numerical} \\ \mathbb{I}_{x_j \neq x'_j} & \text{if } x_j \text{ is categorical} \end{cases} \tag{2.39}$$

and R_j is the observed range of feature j.

The third objective ensures sparsity by measuring the number of features that had to be changed to generate the counterfactual:

$$o_3(x, x') = \sum_{j=1}^{p} \mathbb{I}_{x_j \neq x'_j} \tag{2.40}$$

The last objective addresses the problem of generating realistic data points and ensures that the obtained counterfactual belongs to the data manifold by minimising the distance between the counterfactual and the nearest observed instance:

$$o_4(x', X_{obs}) = \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x'_j) \tag{2.41}$$

Instead of optimising the collapsed, weighted loss function, the authors employ a multi-objective optimisation strategy in order to generate counterfactuals which allow exploration of different trade-offs between objectives. To obtain the counterfactual instance, the Nondominated Sorting Genetic Algorithm (NSGA-II) [Deb et al., 2002] was used, which selects suitable instances through repeated random recombination, mutation and ranking of the obtained solutions. Each instance is represented as a vector of features,

which can be recombined and mutated, and candidates are ranked according to the four metrics. The approach generates a Pareto set of counterfactuals, which represent different trade-offs between the objectives. To ensure actionability, values of numerical features are clipped to fit within a predefined range, and users can additionally define immutable features. However, since the approach directly recombines and mutates features of candidate counterfactuals, it is possible that generated counterfactuals do not conform to causal relationships between features.

Growing Spheres Another similar heuristic approach is proposed by Laugel et al. [2017] and uses growing spheres to locate the counterfactual instance.

The loss function is defined as:

$$x' = \arg\min_{x' \in \mathcal{X}} dist(x, x') \quad | \quad f(x) \neq f(x') \tag{2.42}$$

Where $dist(x, x')$ represents a combined objective, ensuring proximity to the original instance and sparsity:

$$dist(x, x') = ||x - x'||_2 + \gamma \cdot ||x - x'||_0 \tag{2.43}$$

The second part of the objective measures how many features are different between x and x', representing sparsity objective and γ is a balancing parameter between proximity and sparsity objectives. Growing spheres algorithm utilizes a two-step approach and optimizes the two objectives sequentially, instead of simultaneously. The algorithm does not require access to the training data but rather relies on generating instances. To optimize the proximity objective, the algorithm iteratively generates instances in the vicinity of x and examines them, until the validity constraint is met. After obtaining the closest counterfactual x', its feature values are iteratively reset to their values in the original instance x, starting with the features with the smallest difference in values between x and c. This step ensures that the generated counterfactual is sparse. Growing spheres rely on generating new instances, which can result in unrealistic counterfactuals. The approach also does not address the issue of actionability.

GECO [Schleich et al., 2021] recognize that counterfactual explanations need to be generated in real time in order to help users better understand the decisions of AI systems. GECO is an interactive counterfactual explainer that generates plausible and actionable counterfactuals in real-time. To restrict the state space, authors introduce PLAF, plausibility-feasibility constraint language and restrict the search space according to the defined constraints. PLAF encodes constraints in the following format:

$$\text{IF} \quad \Phi_1 \quad \text{AND} \quad \Phi_2 \quad \text{AND} \quad \cdots \quad \text{THEN} \quad \Phi_0 \tag{2.44}$$

Each Φ_i represents an atomic predicate that take the form $E_1 \quad OP \quad E_2$, where E_1 and E_2 are features and OP is an operator such that $OP \in =, \neq, \leq, \geq, <, >$. A genetic algorithm is utilized to search for a sparse counterfactual that satisfies the plausibility

and feasibility constraints. Additionally, GECO optimises sparsity and first explores those counterfactuals that change only one feature, before moving on to other counterfactuals. By employing this early-end search strategy, GECO can generate counterfactuals in less time without exploring the whole potential space of counterfactual instances.

Heuristic search algorithms aim to speed up the counterfactual search as they enable early stopping while iteratively updating the currently best solution. On the other hand, a drawback of this approach is that not the whole search space is explored and a potentially better counterfactual might be omitted in this process.

Instance-based

Instance-based counterfactual methods search for the counterfactual by investigating its relationship with other instances.

FACE Poyiadzi et al. [2020] focus on the problem of ensuring counterfactual instance is likely and propose a graph-based approach FACE. Each instance in the dataset is presented as a node in a directed graph, and the presence of an edge between two nodes indicates that there is a connection between them. Additionally, each edge is weighted according to the density of instances between the two nodes. Users can also implicitly define immutable features, by removing transitions from a node to its neighbour if the transition includes a change of the immutable feature. Authors offer 3 different ways in which a graph can be constructed: KDE, k-NN and ϵ-graph. The shortest path algorithm is then used to find an instance closest to x with different predictions. Since all nodes represent data instances, generated counterfactuals will actually belong to the training dataset. Additionally, since the graph is weighted according to data density, the authors ensure that there is a dense path between the original and counterfactual instances.

NICE Brughmans et al. [2024] propose a nearest unlike neighbour approach, NICE, that guarantees to find a counterfactual for any factual instance. Given a factual instance x, NICE first finds the closest instance x_n that is classified into a counterfactual class. NICE then extracts the list of non-overlapping features – the difference in features between x and x_n. Counterfactual x' is then generated by changing the features of x iteratively according to the features in x_n. In each iteration i, potential counterfactuals are generated by changing exactly one feature in the current best counterfactual x_c^{i-1} to the feature's values in x_n. The best intermediate counterfactual is chosen according to a cost function. NICE implements three cost functions based on three counterfactual metrics – sparsity, proximity and plausibility. In this way, diverse explanations can be generated depending on the choice of the cost function. The iterative process terminates once an intermediate counterfactual x_c is valid.

By focusing the counterfactual search around the original instance, these methods aim to create plausible explanations without time-consuming searching over the entire space of possible counterfactuals.

Decision tree-based

Most of the counterfactual search methods recognize that counterfactuals should be similar to the original instance and optimize metrics to ensure this. However, most of the methods rely only on feature-based metrics such as proximity or sparsity to ensure similarity. Decision tree-based methods, however, search for counterfactuals that are similar to the original instance in terms of the reasoning that led to their classification. These methods aim to distil the black-box model into a decision tree and search for the counterfactual by making minimal split changes to the path leading from the root to the leaf containing the original instance.

LORE Guidotti et al. [2019] propose LORE (Local Rule-based Explainer). Given a black-box model M LORE generates a neighbourhood Z around the original instance x such that for some $z \in Z, M(z) = M(x)$ and for some $z \in Z, M(z) \neq M(x)$. A generative algorithm is used to generate Z and optimizes the following fitness functions:

$$\begin{aligned} f^x_{=}(z) &= \mathsf{I}_{M(z)=M(x)} + (1 - d(z, x)) - \mathsf{I}_{x=z} \\ f^x_{\neq}(z) &= \mathsf{I}_{M(z)\neq M(x)} + (1 - d(z, x)) - \mathsf{I}_{x=z} \end{aligned} \tag{2.45}$$

where d is a distance function and I is an identity function such that $\mathsf{I}_{true} == 1$ and $\mathsf{I}_{false} == 0$. A decision tree is then trained on Z. LORE extracts the *factual rule* as the path leading from the root of the tree to the leaf containing x. Additionally, LORE generates *counterfactual rules* – paths leading to nodes where the decision tree prediction is different to $M(x)$. To ensure counterfactuals are similar to the original instance, LORE measures minimality as the number of splits in the counterfactual rule which are not satisfied by the original instance. The explanation is presented as the tuple $< r, \Phi >$ where r is the factual rule and Φ is a set of counterfactual rules.

FoilTree Similar to LORE, van der Waa et al. [2018a] propose FoilTree, an approach to generating counterfactuals using decision trees. The FoilTree approach also starts by generating a neighbourhood around the factual instance x. However, unlike LORE, FoilTree weighs generated samples based on their similarity to the original instance, similar to LIME [Ribeiro et al., 2016]. A decision tree is then trained in a one-vs-all fashion to distinguish between the counterfactual (or foil) class and all the other classes. The fact leaf is identified as the leaf in which the original instance x is classified. The foil leaf is then selected as the leaf classified in the counterfactual class with the smallest number of nodes between it and the fact leaf. The explanation is then presented as the difference between the two paths leading to the fact and foil leaves.

Decision-tree-based methods provide an alternative to counterfactual search which focuses not only on the difference in features but also the differences in reasoning leading to the factual and counterfactual instance. One drawback of these approaches is, however, that they require the training of a surrogate model.

2.5.2 Counterfactual Explanations in RL

Although there is substantial work on counterfactual explanations in supervised learning, they have not been thoroughly explored for RL agents. Similar to IML, counterfactuals in xRL are local explanations that explain the choice of action a in a state s by finding a counterfactual state s' as similar as possible to s, in which the agent chose an alternative action a'. In supervised learning, a search is performed over the training set to find the counterfactual instance. RL problems, however, do not have a notion of a training set, making it necessary to find alternative ways of obtaining counterfactuals.

In the first counterfactual-based xRL approach, Olson et al. [2019] propose generating counterfactuals using a generative deep learning architecture. Authors set out to explain decisions of Atari RL agents and search for counterfactuals in the latent space of the policy network. The approach assumes that policy is represented with a deep neural network and starts by dividing the network into two parts. The first part A contains all but the last hidden layer of the network and serves for mapping input state s into latent representation z: $A(s) = z$. The second part π consists only of the last fully connected layer followed by a softmax and is used to provide action probabilities based on the latent state. Policy network can be imagined as a composition of the network parts $\pi(A(s))$. To generate counterfactual states, authors propose a deep generative model consisting of encoder (E), discriminator(D), generator (G) and a Wasserstein encoder (E_w, D_w). Encoder E and generator G work together as an encoder-decoder pair, with encoder mapping input state into low-dimensional latent representation, and generator performing reconstruction from encoding to original state. Discriminator D learns to predict the probability distribution over actions $\pi(z)$ given a latent state representation $z = E(s)$. Counterfactual search is performed in the latent space, which may have holes, resulting in unrealistic instance. For this reason, authors include a Wasserstein encoder (E_w, D_w) to map latent z to an even lower-dimensional representation z_w in order to obtain a more compact format and ensure more likely counterfactuals. The generative architecture is trained on a set of agent's transitions in the environment $\mathcal{X} = (s_1, a_1), \dots, (s_n, a_n)$. Finally, to generate counterfactual state s' of state s, authors first locate the latent representation z_w^* of s' by solving:

$$\begin{aligned} &\text{minimize} && ||E_w(A(s)) - z_w^*||_2^2 \\ &\text{subject to} && \arg\max_{a \in A} \pi(D_w(z_w^*, a)) = a' \end{aligned} \tag{2.46}$$

Where a' is the counterfactual action. The objective can be simplified to:

$$z_w^* = \arg\min_{z_w} ||z_w - E_w(A(s))||_2^2 + \log(1 - \pi(D_w(z_w), a')) \tag{2.47}$$

The process for finding the counterfactual instance then consists of encoding the original state into Wasserstein encoding $z_w = E_w(A(s))$ and optimising Equation 2.47 to find

z_w^*. Latent counterfactual instance z_w^* can then be decoded to obtain $\pi(D_w(z_w^*))$, which is finally passed to the generator to obtain the counterfactual instance s'.

While Olson et al. [2019] proposes an approach for generating realistic counterfactual states, their approach requires access to internal model parameters. In contrast, Huber et al. [2023] propose GANterfactual-RL a generative model-agnostic approach that uses domain transfer architecture to generate counterfactual states. Domains are determined by actions and contain states in which a specific action would be chosen by the agent. Understanding what is required for the action to change requires translating the state from its original to an alternative domain. The algorithm is based on the StarGAN architecture [Choi et al., 2018] and includes training a discriminator D and generator G. The generator receives as input a state and a target domain and produces a translated state. The role of the discriminator is to distinguish between real and fake images. The discriminator takes as input a state and produces two outputs – D_{cls} and D_{src}. The first output D_{cls} classifies the state to its action domain. The second output D_{src} outputs the probability of the state being real, instead of generated by the generator.

The generator and discriminator are trained on states extracted from the agent's policy. The StarGAN architecture is trained using multiple loss components. The first component is the adversarial loss, which ensures highly realistic states are generated. Authors use a Wasserstein objective with a gradient penalty to implement this loss:

$$\mathcal{L}_{adv} = \mathbb{E}_s[D_{src}(s)] - \mathbb{E}_{s,a'}[D_{src}(G(s,a'))] - \lambda_{gp}\mathbb{E}_{\hat{s}}[(||\nabla_{\hat{s}} D_{src}(\hat{s})||_2 - 1)^2] \tag{2.48}$$

The second loss component uses the discriminator's classification output D_{cls} which learns to approximate which action the agent will take in a state. This loss ensures that the counterfactual action is taken in the generated state. The classification loss depends on whether the network is fed original input from the training data or generated input:

$$\begin{aligned} \mathcal{L}_{cls}^{a} &= \mathbb{E}_{s,a}[-logD_{cls}(a|s)] && \text{if } s \text{ is a real state} \\ \mathcal{L}_{cls}^{a'} &= \mathbb{E}_{s,a'}[-logD_{cls}(a'|G(s,a'))] && \text{if } s \text{ is a generated state} \end{aligned} \tag{2.49}$$

Finally, reconstruction loss is used to ensure that the generated states are similar to the original states. Reconstruction loss penalises unnecessary feature changes:

$$\mathcal{L}_{rec} = \mathbb{E}_{s,a,a'}[||s - G(G(s,a'),a)||_1] \tag{2.50}$$

The discriminator and generator can then be simultaneously trained by optimising the following loss functions:

$$\begin{aligned}\mathcal{L}_D &= -\mathcal{L}_{adv} + \lambda_{cls} * \mathcal{L}^{a}_{cls} \\ \mathcal{L}_G &= \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}^{a'}_{cls} + \lambda_{rec}\mathcal{L}_{rec}\end{aligned} \tag{2.51}$$

Authors evaluate counterfactuals generated by GANterfactual-RL in a user study and found that they significantly improve user understanding of RL agents in Atari games, compared to those generated by Olson et al. [2019].

Counterfactuals represent a human-friendly approach to answering questions about the model's decision. They can help users of AI systems get actionable advice on how to change the outcome in the future. However, since current methods rely on generating models, they risk generating counterfactuals that might be unlikely or unattainable.

2.6 Semifactual Explanations

Semifactual explanations are another type of explanation method that has emerged alongside counterfactuals [Aryal and Keane, 2023]. Semifactuals are similar to counterfactuals in the sense that they, too, imagine alternative worlds; however, semifactuals search for those in which the outcome did not change. For example, when explaining why a loan application was rejected, a semifactual might state that *"Even if your salary was twice as high, the application would still have been rejected."*

Previous research in psychology has found that the semifactual *"even if"* and counterfactual *"what if"* statements are perceived differently by humans. For instance, McCloy and Byrne [2002] conducted a study in which they aimed to explain to participants the event of Philip's allergic reaction to ice cream using counterfactual and semifactual explanations. A semifactual stated that *"Even if Philip had chosen the banana split, he would still have had an allergic reaction"* and the counterfactual *"If only Philip had not chosen the ice-cream sundae, he wouldn't have had an allergic reaction"*. While counterfactuals led users to perceive the antecedent (the choice of dessert in this case) as more causally related to the outcome, semifactuals resulted in users judging a weaker causal connection. In other words, semifactuals convinced the users that the outcome (allergic reaction) would have occurred either way and as such is not causally connected to the choice of dessert. Additionally, McCloy and Byrne [2002] found that counterfactuals lead users to focus more on the alternative antecedents that undo the outcome. On the other hand, semifactuals result in a higher focus on antecedents that would change the outcome.

As they engage in a different psychological process, semifactuals also have additional benefits compared to counterfactuals [Aryal and Keane, 2023]. Firstly, semifactuals focus on confirming the outcome even in the alternative scenarios. They make the outcome seem more correct and immutable, making them more convincing. Secondly, semifactuals are selective explanations, aiming to change only one feature. This makes them easier to process and reduces the cognitive load on users. Finally, semifactuals

seem to elicit less negative emotions in users when it comes to understanding negative outcomes such as loan rejection. While counterfactuals might potentially be perceived as blaming the users for the outcome, semifactuals reinstate the outcome and reconfirm that nothing could have been done to prevent it.

After analysing semifactuals, Aryal and Keane [2023] introduces the requirements for ensuring that these explanations are useful to users:

1. *Validity*: Same as in counterfactuals, validity refers to the alternative scenario achieving the counterfactual outcome. In the case of semifactuals, the outcome should not change from the factual one.
2. *Sparsity*: semifactuals aim to change only one feature.
3. *Plausibility/Actionability*: the feature change should be realistic, and the semifactual should fall within the data manifold.
4. Convincing: Semifactuals can sometimes present unexpected conclusions. However, to maintain trust in the system, they should be convincing.
5. *Causality*: semifactuals should change the user's preconceived causal model of the environment, by showing them new, more convincing information.
6. *Fairness*: from the perspective of fairness, semifactuals should not be misleading. Aryal and Keane [2023] proposes three conditions that semifactuals should satisfy in this regard:
   (a) semifactuals should not change proxy variables.
   (b) Semifactuals should be robust to adversarial attacks.
   (c) semifactuals should follow the ceteris paribus (i.e. all other things being equal), meaning that the outcome should not depend on minor interactions between variables.

While the above are high-level requirements for semifactuals in supervised learning, current semifactual generation approaches take into account a subset of them. Additionally, semifactual methods differ in metrics for evaluating these requirements. In the following section, we provide an overview of current state-of-the-art methods for semifactual generation in supervised learning.

2.6.1 Semifactual Explanations in Supervised Learning

Previous work by Aryal and Keane [2024] divides the approaches for generating semifactuals into two categories: counterfactually-guided and counterfactually-free. Counterfactually-guided explanations require a counterfactual to be generated and search for the semifactual in relation to the counterfactual, while counterfactually-free methods do not have this requirement. An overview of the methods reviewed in this section, with their classification based on this taxonomy, is given in Table 2.7.

Table 2.7: An overview of counterfactual generation approaches for supervised learning covered in Section 2.6 according to their requirements and search strategy.

Semifactual Method	Type	Search Strategy
Nugent et al. [2009]	Counterfactually-free	Local surrogate approximation
DSER [Artelt and Hammer, 2022]		Iterative search
MDN [Aryal and Keane, 2023]		Instance scoring
S-GEN [Kenny and Huang, 2024]		Evolutionary algorithm
KLEOR [Cummins and Bridge, 2006]	Counterfactually-guided	Instance scoring
PIECE [Kenny and Keane, 2021]		Generative network
CVC [Zhao et al., 2022]		Variational encoder

Counterfactually-guided Methods

Counterfactually-guided approaches require a counterfactual to be generated to guide the search for the semifactual. Intuitively, semifactuals are often imagined between the original instance and the decision boundary. Given that counterfactuals are often the closest instances that cross the decision boundary, it makes sense to search for semifactuals by following the path that leads to a counterfactual but stopping before the class changes.

Cummins and Bridge [2006] propose KLEOR, a selection of counterfactually-guided methods for generating example-based explanations. KLEOR consists of three methods: Sim-Miss, Global-Sim and Attr-Sim. All of the approaches require access to a counterfactual defined as the nearest miss (NM) – instance most similar to the original but of a different class. Sim-Miss is the simplest of the three methods and given the original instance x searches for a semifactual x' that is 1) classified in the same class as x and 2) close to NM. However, Sim-Miss is only applicable in cases where the decision boundary separates the feature space into convex regions. Global-Sim addresses this issue and searches not only for instance close to NM, but one that is between NM and the original instance. Specifically, Global-Sim adds constraints:

$$Sim(x, x') \leq Sim(x, NM) \tag{2.52}$$

where $Sim(x, x')$ is a similarity measure between instances. Since Global-Sim relies on a composite similarity between all features, a few larger features can outweigh the others. For this reason, authors introduce the third method Attr-Sim, which uses an attribute-based similarity instead of the one defined in Eq. 2.52:

$$Sim_a(x_a, x'_a) > Sim_a(x_a, NM_a) \tag{2.53}$$

where x_a are features of the instance x.

PIECE algorithm [Kenny and Keane, 2021] generates plausible counterfactual and semifactual explanations for image-based tasks. The algorithm is model-specific and developed for convolutional neural networks. PIECE models the distribution of latent features

and finds exceptional features for a counterfactual class. A counterfactual is then found by resetting the exceptional features to their expected values for the counterfactual class. A semifactual generation follows the same process except it terminates just before the decision boundary is reached. A generative network is used to visualize explanations. For a neuron N_i in a convolutional layer, θ_i is the probability of that neuron being activated for the counterfactual class y'. An exceptional feature is then defined as:

$$\begin{aligned} N_i = 0 \quad \text{if} \quad p(1-\theta_i) < \alpha \\ N_i > 0 \quad \text{if} \quad p(\theta_i) < \alpha \end{aligned} \tag{2.54}$$

Where α is the threshold value. Intuitively, an exceptional feature is the one represented by a neuron that activates despite its probability of activating for the counterfactual class being low and vice versa. Exceptional features are ordered from lowest to highest probability and changed to expected values of the counterfactual class. The semifactual is generated as the final instance before crossing the decision boundary.

CVC [Zhao et al., 2022] extends PIECE to enable its use in one-shot learning scenarios, where only a small amount of data is available. CVC uses a class-to-class variational encoder (C2C-VAE), which learns an embedding of space representing feature differences. C2C-VAE consists of an encoder f and decoder f'. To train the encoder-decoder pair, C2C-VAE samples pairs of instances x_s and x_t belonging to different classes s and t. CVC algorithms consist of two steps. In the first step, a guiding instance of a counterfactual class is retrieved. To this end, CVC starts by sampling K vectors in the embedding space and decoding them back into instances. The guiding instance is chosen as the one that minimises the mean squared distance to the original instance in pixel space. In the second stage, CVC interpolates between the original and guide instances:

$$x' = f'((1-\lambda)\cdot f(s) + \lambda \cdot f(t)), \quad \lambda \in [0,1] \tag{2.55}$$

Where t is the guiding instance. A semifactual is obtained by stopping the interpolation before a class is changed.

Counterfactually-guided methods imagine semifactuals as an extension of the counterfactual and thus can be generated by expanding counterfactual search approaches. However, as they search for semifactuals in relation to a counterfactual, the quality of the semifactual is limited by the quality of the guiding counterfactual explanation.

Counterfactually-free Methods

Counterfactually free methods often rely on some sort of distance metric to choose the furthest instance classified in the same class as the query. Nugent et al. [2009] provide one of the first approaches for generating semifactual explanations. The motivation for the approach comes from case-based reasoning. The semifactual explanation is generated in

order to reaffirm a classification by showing a similar example of the same class. Authors see semifactuals as *a fortriori* arguments – arguments that lay between the query and the decision boundary and strengthen the case. For example, when explaining why a pay increase was given to an employee with five years of experience, a semifactual should show a similar employee with less experience who also received the increase, rather than one with more experience. The authors use a local surrogate approach to approximate the decision boundary around the query instance. Firstly, a set of instances is generated in the neighbourhood of the query instance, and a logistic regression model is used to classify them. The semifactual is then chosen as the instance with the lowest probability of being assigned the same class as the query instances:

$$x' = \arg\min_{c \in N} LR(x) \tag{2.56}$$

where N is the neighbourhood of instance x, and $LR(x)$ is the logistic regression probability. Intuitively, this method finds a similar example that is close to an approximated decision boundary, strengthening the classification.

Due to low certainty, machine learning models can sometimes refrain from providing a prediction on a specific instance. DSER (Diverse Semifactual Explanations of Reject) [Artelt and Hammer, 2022] has been developed to explain the model's decision to reject an instance by comparing it with a similar instance that the model is sure about rejecting. DSER is very similar to counterfactual loss-based methods, and aims to find a good semifactual by optimising a composite loss function:

$$x' = \arg\min[l_{feasable}(x) + l_{sparse}(x) + l_{similar}(x) + l_{diverse}(x)] \tag{2.57}$$

The loss function combines four different desired metrics – feasibility, sparsity, proximity and diversity. Feasibility ensures that the semifactual instance is rejected and that the model is more certain about the semifactual than the query instance:

$$l_{feasible}(x) = C_f \max(r(x') - \theta, 0) + C_{sf} \max(r(x) - r(x'), 0) \tag{2.58}$$

where $r(x)$ estimates model's uncertainty when classifying x. The first part of the feasibility loss ensures that the model rejects the semifactual because its certainty falls below the threshold θ. The second part of the equation ensures that the semifactual is convincing by being more certain than the query instance. Parameters C_f and C_{sf} are used to balance between semifactual being convincing and valid. The second loss component $l_{sparse}(x)$ ensures that as few feature changes as possible occur when generating the semifactual:

$$l_{sparse}(x) = C_{sparse} \cdot \max(\sum_{f \in F} [(x_f - x'_f) \neq 0] - \mu, 0) \tag{2.59}$$

where F is the set of features. The parameter μ determines the maximum number of

feature changes allowed. The third component $l_{similar}$ promotes instances as dissimilar to the original query as possible. Specifically, the Euclidean distance metric is used:

$$l_{similar}(x) = -C_{similar} \cdot ||x - x'||_2 \tag{2.60}$$

Finally, $l_{diverse}$ ensures that multiple, diverse semifactuals are offered. Diversity loss ensures that generated semifactuals change different features:

$$l_{diverse}(x) = C_{diverse} \cdot \sum_{f \in F_{used}} ((x_f - x'_f) \neq 0) \tag{2.61}$$

Where F_{used} represents the set of features already changed in previously generated semifactuals. Diversity loss ensures that these features are not changed again.

Similarly, Aryal and Keane [2023] proposes MDN (Most Distant Neighbour), a naive benchmark that searches for the most distant instance of the same class as the query instance. Instead of looking for a dissimilar instance in general, MDN searches for a semifactual which significantly differs from the original instance in one key feature. Given a feature f MDN first splits the dataset into two sets – HighSet(f) contains instances whose value of f is higher than in the original instance x and LowSet(f) where the value of f is lower. The sets are then separately ordered according to the scoring function:

$$sfs(x', X, f) = \frac{\mathbb{I}(x' = x)}{|F|} + \frac{diff(x_f, x'_f)}{\max(diff(x_f, x'_f))} \tag{2.62}$$

where F is the feature set. The first part of the SF score ensures sparsity by penalising feature changes. The second part ensures high diversity in the feature between semifactual and original instances. For each feature, an instance with the highest *sfs* score is chosen. The final semifactual is chosen as the one with the highest *sfs* score over all features:

$$x' = \arg\max_{x \in X} sfs(x) \tag{2.63}$$

Kenny and Huang [2024] states that semifactual explanations should be used to explain positive outcomes, unlike counterfactuals, which target negative decisions. Authors propose S-GEN, a method for generating semifactuals that can help users make better decisions even when they receive a positive outcome. For example, a semifactual could inform the farmer that had they used half the fertiliser amount, the yield would have remained the same. To generate semifactuals, authors start by creating a structural causal model (SCM) around the query instance. Neighbouring instances represent nodes, while actions that can transform one state into another are actions. In this way, authors can define a structural semifactual which takes into account the causal structure of the

task. Achieving the semifactual means taking an action $a \in a_1, ..., a_m$ in the SCM. S-GEN optimises the objective function:

$$x' = \max_{a_1,...,a_m} \frac{1}{m} \sum f(P()G()) + \gamma R(q'_1. ... , q'_m) \quad s.t. \quad \forall i, j : q'_i = S(q, a_i), H_j(q_i) \leq 0 \tag{2.64}$$

Where. The objective consists of three main components – gain G, diversity R and robustness H. Gain is similar to the cost used in counterfactual explanations, and measures how much the user can gain by following the semifactual:

$$G(x, a) = P \circ diff(x, x') \tag{2.65}$$

Function P is an oracle function that maps the distance between two states into a gain. Intuitively, gain depends on how distant two states are, as well as a predefined notion of what a favourable change is to the specific user. Diversity is defined over all semifactuals and ensures they are as distant as possible from each other:

$$R(q_1, ... , q_m) = \frac{2}{m(m-1)} \sum_{i=1}^{m} \sum_{j>i}^{m} L_p \circ diff(q_i, q_j) \tag{2.66}$$

Robustness refers to the semifactual's ability to withstand distributional shift. S-GEN considers the semifactual to be robust if instances in its neighbourhood are classified in the same class as well:

$$H(x, a) = \min_{\theta \in \mathcal{B}(x,a)} h(\theta) - \mu \tag{2.67}$$

where h is the black-box model, $\mathcal{B}(x, a)$ is the neighbourhood of x achieved by making a actionable feature change a in the SCM. The parameter μ indicates the sensitivity to a wrong prediction. S-GEN uses an evolutionary algorithm to optimise the loss function. The authors also conducted a user study where they showed that users prefer semifactuals over counterfactuals when explaining positive outcomes.

2.7 Evaluating Explanations

Despite the large amount of work in XAI on generating explanations for black-box systems, there are still no unified metrics for evaluating explanations. This makes it difficult to benchmark and compare different methods across tasks. One of the reasons for the lack of universally accepted metrics is that explanations are inherently subjective – what is a satisfactory explanation for one person might not be for another [Zhou et al., 2021]. Additionally, different users might not be interested in the same information, and different situations might require different granularities of explanations. For example, for a

person wondering why the autonomous vehicle took a left turn, only the most important cause will be enough, and presenting them with additional reasons could be overwhelming. On the other hand, an investigator of a car accident caused by an autonomous vehicle is likely to require a full explanation.

According to Doshi-Velez and Kim [2017], evaluation approaches can be divided into three tiers. An overview of this classification is provided in Table 2.8.

1. **Application-grounded (AG) evaluation** requires experiments to be conducted with real humans executing real tasks. This way, explanations are evaluated based on how much they assist the person using the system (e.g. assisting a doctor with a diagnosis). A good baseline for this evaluation is to observe how well *human-generated* explanations assist real humans in performing the real task.
2. **Human-grounded (HG) evaluation** require real humans executing a simplified task. Since the task is simplified, both expert and non-expert users can participate in the experiment. This evaluation is used when we are more interested in general metrics of explanations (e.g. which are easier to comprehend) than their usefulness in real tasks. For example, the human-grounded experiment could involve presenting two explanations to the user and asking them to choose the one they think is of higher quality.
3. **Functionally-grounded (FG) evaluation** assumes that explanations are evaluated without human experiments, using a proxy for explanation quality. The common proxies are cost, explanation length or sparsity. For example, for counterfactual explanations, there is a set of metrics that can serve as proxies for evaluation, such as proximity, validity or sparsity (Section 2.5).

Functionally-grounded evaluation does not require human experiments, making it more cost-efficient and easier to execute. However, to ensure that the explanation is suitable for a task, human studies are often necessary. While evaluating explanations using proxies is straightforward, measuring the quality of explanations in human studies is more complex. Mohseni et al. [2021] proposes four different metrics that can be explored in human studies:

1. **Mental model** represents the way the user understands the behaviour of the black-box system. Testing mental models can uncover how explanations affect users' understanding of how the system works. For example, Sequeira and Gervasio [2020] evaluates their summary generation method by exploring whether it helps users understand in which situations the agent performs well and in which it might need more practice. In another work, Olson et al. [2019] compares how confident users feel that they understand an agent's behaviour before and after seeing counterfactual explanations.
2. **Usefulness and satisfaction** refer to how helpful the user finds the explanations. For example Ribeiro et al. [2016] evaluate the usefulness of their algorithm LIME for choosing the best model by measuring the user's ability to select a better model

Table 2.8: Classification of evaluation approaches in XAI based on their requirements for users and the task presented to the users.

Approach	Evaluation Subjects	Evaluation Task
Application-grounded	Real humans	Real task
Human-grounded	Real humans	Simplified task
Functionally-grounded	No humans	Proxy task

between the two presented based on explanations generated by LIME.

3. **User trust** is an important component in enabling the integration of AI systems and encouraging collaboration between the user and the system. Experts need to trust the system to rely on its decisions. User trust is also related to user understanding of the system – if users can predict and understand the behaviour of the system, they are more likely to have trust in it. Greydanus et al. [2018] uses the user's ability to distinguish between optimal and overfitting agents in Atari games as a proxy of their trust. Similarly, Madumal et al. [2020] hypothesises that enabling users to understand an agent's behaviour will lead to an increase in trust. Authors evaluate their contrasting explanations in the StarCraft 2 task by allowing users to ask questions and receive explanations. Afterwards, the user's understanding is evaluated by asking the user to predict the next action in a particular state and record the user's trust before and after receiving explanations.

4. **Human-AI performance** refers to the ability of explanations to improve a user's performance in a task. This corresponds to application-grounded evaluation, whose goal is to evaluate whether providing explanations to experts using black-box systems increases their performance. For example, Puri et al. [2019] performs a human study to measure how their explanation method, SARFA, helps users to solve chess puzzles. Users are asked to solve puzzles with and without a saliency map, which highlights pieces that are most useful in choosing an optimal move.

The overview of evaluation techniques used in XAI approaches is presented in Table 2.9. Explanation evaluation remains a challenging task, due to the abstract and highly subjective metrics such as trust or user satisfaction. Additionally, most of the works rely on functionally grounded evaluation. Few works that conduct user studies often use human-grounded evaluation, with only a very small number of works performing application-grounded evaluation with real humans executing real tasks. This hinders AI research as it makes benchmarking difficult.

Table 2.9: Evaluation of explanations generated by state-of-the-art IML and xRL methods presented in Section 2.3, 2.4, 2.5.1 and 2.6. For simplicity, only approaches which performed evaluation are listed.

Approach	Field	Explanation Type	Evaluation Type	Evaluation Metric	Evaluation Subjects
Pizarroso et al. [2020]	IML	Feature importance	FG	Computational cost	-
Štrumbelj and Kononenko [2014]	IML	Feature importance	HG	Mental model	Non-experts/Developers
Ribeiro et al. [2016]	IML	Local surrogate	HG	User trust + Usefulness	Non-experts
Lundberg [2017]	IML	Feature importance	HG + FG	Mental model	Non-experts
Bastani et al. [2017]	IML	Global surrogate	HG + FG	Mental model	Experts/Developers
Frosst and Hinton [2017]	IML	Global surrogate	FG	Fidelity	-
Lakkaraju et al. [2017]	IML	Global surrogate	HG + FG	Mental model	Non-experts
Roth et al. [2019]	xRL	Inherently interpretable	FG	Performance + Simplicity	-
Paleja et al. [2022]	xRL	Inherently interpretable	FG	Performance + Simplicity	-
Annasamy and Sycara [2019]	xRL	Inherently interpretable	FG	Performance	-
Mott et al. [2019]	xRL	Inherently interpretable	FG	Generalization + Robustness	-
Custode and Iacca [2022]	xRL	Inherently interpretable	FG	Performance	-
Coppens et al. [2019]	xRL	Global surrogate	FG	Fidelity	-
Liu et al. [2018]	xRL	Global surrogate	FG	Fidelity	-
Guo and Wei [2022]	xRL	Global surrogate	FG	Performance + Fidelity	-
Skirzyński et al. [2021]	xRL	Global surrogate	AG	Human-AI Performance	Non-experts
Verma et al. [2018]	xRL	Global surrogate	FG	Fidelity	-
Zrihem et al. [2016]	xRL	Summaries	FG	Performance + Simplicity	-
Amir and Amir [2018]	xRL	Summary	HG	Mental model + Usefulness	Non-experts
Huang et al. [2018]	xRL	Summaries	HG	Mental model + Trust	Non-experts
Topin and Veloso [2019]	xRL	Summaries	FG	Performance + Simplicity	-
Sequeira and Gervasio [2020]	xRL	Summary	HG	Mental model	Non-experts
Huber et al. [2021]	xRL	Summaries	HG	Mental model	Non-experts
McCalmon et al. [2022]	xRL	Summaries	FG + HG	Mental model + Fidelity	Non-experts
Greydanus et al. [2018]	xRL	Feature importance	HG	User trust	Non-experts
Puri et al. [2019]	xRL	Feature importance	AG	Human-AI performance	Experts
Wang et al. [2016]	xRL	Feature importance	FG	Performance	-
Juozapaitis et al. [2019]	xRL	Reward explanations	FG	Fidelity	-
Jenner and Gleave [2022]	xRL	Reward explanations	FG	Simplicity	-
Huang et al. [2019]	xRL	Reward explanations	HG	Mental model	Non-experts
Madumal et al. [2020]	xRL	Contrasting	HG	Mental model + Trust	Non-experts
van der Waa et al. [2018b]	xRL	Outcome explanations	HG	Satisfaction	Non-experts
Wachter et al. [2017]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
Looveren and Klaise [2021]	IML	Counterfactual explanations	FG	Proximity + Sparsity + In-distributionness + Cost	-
Mothilal et al. [2020]	IML	Counterfactual explanations	FG	Proximity + Validity + Diversity	-
Cheng et al. [2020]	IML	Counterfactual explanations	AG	Usefulness	Experts
Joshi et al. [2019]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
Mahajan et al. [2019]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
Chen et al. [2021]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
Samoilescu et al. [2021]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
Dandl et al. [2020]	IML	Counterfactuals	FG	Counterfactual metrics	-
Laugel et al. [2017]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
Schleich et al. [2021]	IML	Counterfactual explanations	FG	Counterfactual metrics + Computational cost	-
Poyiadzi et al. [2020]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
Brughmans et al. [2024]	IML	Counterfactual explanations	FG	Counterfactual metrics + Computational cost	-
Guidotti et al. [2019]	IML	Counterfactual explanations	FG	Counterfactual metrics	-
van der Waa et al. [2018a]	IML	Counterfactual explanations	FG	Simplicity + Fidelity	-
Olson et al. [2019]	xRL	Counterfactual explanations	HG	Mental model	Non-experts
Huber et al. [2023]	xRL	Counterfactual explanations	HG	Mental models + Satisfaction	Non-experts
Nugent et al. [2009]	IML	Semifactual explanations	HG	Mental model + Usefulness	Non-experts
Artelt and Hammer [2022]	IML	Semifactual explanations	FG	Semifactual metrics	-
Aryal and Keane [2023]	IML	Semifactual explanations	FG	Semifactual metrics	-
Kenny and Huang [2024]	IML	Semifactual explanations	HG	Usefulness	Non-experts
Cummins and Bridge [2006]	IML	Semifactual explanations	HG	Satisfaction	Non-experts
Kenny and Keane [2021]	IML	Semifactual explanations	FG	Semifactual metrics	-
Zhao et al. [2022]	IML	Semifactual explanations	FG	Semifactual metrics	-

3 Redefining Counterfactual and Semifactual Explanations for RL

The problem of generating counterfactual explanations has been explored in depth in supervised learning [Laugel et al., 2017, Wachter et al., 2017, Guidotti et al., 2018, Dandl et al., 2020, Poyiadzi et al., 2020, Mothilal et al., 2020]. In RL, however, a substantially smaller number of works investigate this problem [Olson et al., 2019, Huber et al., 2023]. Similarly, while semifactuals have been explored in psychology and philosophy for decades [McCloy and Byrne, 2002], they only recently emerged alongside counterfactuals for xAI [Aryal and Keane, 2023]. In supervised learning, several works consider these explanations a powerful complement to counterfactual explanations [Kenny and Keane, 2021, Aryal and Keane, 2023, Kenny and Huang, 2024]. However, despite counterfactuals finding applications in xRL, semifactuals have not yet been used to explain the behaviour of RL agents.

In this chapter, we aim to lay the theoretical groundwork for defining and generating counterfactual and semifactual explanations for RL tasks. As these explanations have so far been primarily defined for and applied in supervised learning tasks, we first analyse the differences between supervised learning and RL from the perspective of counterfactual and semifactual explanations (Section 3.1). We use the insights from this analysis to redefine counterfactual explanations for RL in Section 3.5 and semifactual explanations for RL in Section 3.6.

3.1 Supervised vs. Reinforcement Learning

While semifactuals and counterfactuals are different explanation types, they share many similarities. Both explanation types search for an alternative world in which some feature of the world is changed, and which achieves the desired outcome. The key idea behind generating counterfactuals is finding an instance belonging to the *closest possible world* [Wachter et al., 2017] – an instance similar to the original one, but eliciting a different outcome. Conversely, semifactuals are often defined as the *most distant neighbour* – an instance with the same outcome far away from the original instance being explained [Aryal and Keane, 2023]. For this reason, both semifactuals and counterfactuals require precise definitions of concepts such as 1) desired outcome, 2) distance between instances, and 3) plausibility of an instance. In the current work in supervised and RL, these

concepts are often defined as a function of instance features and do not take into account RL's sequential or stochastic nature. In this section, we investigate how the sequential and stochastic conditions of RL tasks affect these and other concepts of importance for counterfactual and semifactual explanations and how they can be defined for complex RL tasks. An illustrative example showcasing the effect of stochasticity and sequential execution on counterfactual and semifactual explanations is shown in Figure 3.1. For simplicity, the remainder of this chapter refers to semifactual and counterfactual explanations under the joined name of *Xfactuals*.

3.1.1 Sequential Execution

One of the main differences between supervised and RL is in the type of tasks they solve. While supervised learning is focused on one-step prediction tasks, RL is developed for sequential problems. Consequently, in supervised learning, all instances are considered independent. In RL, on the other hand, there is an underlying temporal connection between states – the agent can transition between states using actions. Temporal distances and the underlying connections between states have to be taken into account to estimate how distant two states are from one another, i.e. how many RL actions are needed to transition between them, or whether transitioning is even at all possible.

Explaining vs. Correcting Behaviour

In supervised learning, the purpose of Xfactual explanations is twofold: 1) explaining how the black-box model reached the decision, and 2) offering users actionable advice on how they can change the input to achieve the desired output from the model. In other words, explaining the model and correcting the input can be done at the same time. This is because supervised learning models are one-step prediction models that make decisions based on the user's features but have no influence over those features. The model's decision-making process is separate from the user's actions, and the user's recourse is limited to changing the input features.

In RL, due to its sequential nature, the roles of the black-box model and user are no longer independent. If the user were to change the input state, this would also change the RL environment and thus affect the decision-making process of the black-box policy. For this reason, Xfactuals cannot be used in RL to suggest to the user how to change the input features. For example, consider an autonomous driving RL agent making a mistake and ending in a crash. To prevent the crash, a recourse could be issued advising the user that the crash would have been avoided if the speed had been lower. While this type of explanation provides an insight into the rules of the environment (e.g., high speed could lead to crashes), it does not offer any information about the behaviour of this particular agent. Namely, it gives no insight into why the agent chose to maintain dangerously high speed – was it because it misjudged its distance from the next car, or did it assume the car in front was moving fast enough to accommodate the increase in speed?

The process of providing counterfactual explanations to users and counterfactual recourse is separate in RL. This is because counterfactual recourse can, by definition, only be achieved by using actions that are different from the ones the agent would choose, and for that reason are not representative of the agent's behaviour and cannot be used to explain it. While counterfactual recourse can offer a correction to the policy that would elicit a desired outcome, it does not explain that policy. In this work, we focus on using Xfactuals in order to explain the behaviour of RL agents, and counterfactual recourse falls out of the scope of this thesis.

State Distance

In supervised learning, two instances are considered close if their features are similar. This can mean they only differ in a few features (measured by sparsity) or that their features are similar according to Euclidean distance or a similar metric (often referred to as proximity). Additionally, user-defined metrics can be used to deem certain features as more costly to change, making any instances that differ in that specific feature far away from the original instance. As instances are independent in supervised learning, relying solely on features to estimate the similarity between instances is reasonable.

In RL, however, states are connected through temporal dependencies via state transitions, which are not necessarily compatible with feature changes. This means that two states with almost identical state feature values might be distant from each other in terms of execution. For example, consider a farmer asking an AI system why it has predicted a low yield for their tomato plant. A feature-based closest-world counterfactual might explain that had the farmer planted beans instead of tomatoes, then the yield would have been higher. Similarly, a feature-based most distant neighbour semifactual could state that even if the farmer planted sunflowers, the yield would not have changed. This explanation differs only in one feature from the original scenario — the choice of plant. However, in terms of execution, both of these explanations are far away from the original scenario as they diverge from the original world in the early steps of the episode when the choice of plant is made. An alternative explanation, indicating that if more water had been applied yesterday, the yield would have been higher, is more informative of the agent's reasoning. Therefore, to define state closeness metrics in RL, we have to take into account the temporal dependencies between states.

Plausibility

In supervised learning, one aspect of ensuring an instance is plausible is verifying that it can be reached from the original instance using realistic feature changes. For example, user-defined actionability metrics discourage counterfactuals that cannot be obtained from the original instance or require changing immutable features, such as country of birth or race.

In RL, however, there is an underlying structure that determines whether a state can be reached from another state. As states are temporally connected, the transition from one

state to another is possible only if there exists a sequence of actions transitioning between them. Xfactuals that are not temporally connected to the original instance through RL actions can represent states that would never occur in the agent's experience (e.g. a chess game position without kings). As such, they are not representative of the agent's behaviour and can be misleading.

Direction of Change

In supervised learning, we deal with one-step prediction tasks. This means that there is no difference between Xfactuals that explain an event through changes in the future and those that change the past. Namely, in supervised learning statements such as "*The outcome would have been the same if it rained yesterday*" and "*The outcome will not change even if it rains tomorrow*" are often used interchangeably.

For counterfactuals, previous research in psychology distinguishes between two ways of answering a *what if* question – *prefactual*, which explains an outcome by hypothesising about the future and *counterfactual*, which offers alternative pasts to explain an outcome [Byrne and Egan, 2004, Ferrante et al., 2013, Dai et al., 2022]. For example, for a user wondering why their loan application has been rejected, a counterfactual explanation could state: *Had you had a higher income, your loan would have been approved*, while the prefactual explanation would advise: *If you obtain a higher income in the future, your next loan application will be approved* [Dai et al., 2022]. Previous research in psychology has found that the two explanation types include different psychological processes [Byrne and Egan, 2004, Ferrante et al., 2013]. While an analogous classification is possible for semifactuals, at the moment, there is no research investigating their effect on human cognition.

Although supervised learning does not distinguish between the two types of statements, RL is designed for sequential tasks, where this distinction makes more sense. For example, a choice to attack an opponent by an RL agent playing chess could be explained by previous actions that lead to a well-prepared attack, as well as by future hopes of obtaining an advantageous position. However, current methods for generating Xfactuals in RL [Olson et al., 2019, Huber et al., 2023] do not distinguish between Xfactuals that exist in the agent's past and its future. As they have been shown to produce different cognitive responses in a counterfactual setting, this distinction has to be investigated in RL.

3.1.2 Stochastic Conditions

Achieving Xfactuals requires changing the features of the original state. Most methods for generating Xfactuals in supervised learning assume that features are independent and that changes to these features are deterministic. In many RL tasks, however, the actions that change state features can exhibit stochastic behaviour. Specifically, due to stochastic events outside of the agent's control, taking the same action in a state multiple times does not always lead to the same state. Additionally, the agent's policy

might not be deterministic, and the agent's actions might be chosen randomly based on a probability distribution This is also a characteristic of real-life problems, where decision models can change through time, and performing an action might not have a deterministic outcome. For example, the threshold for a loan can change over time, and a medical intervention might not always be successful. Similarly, a poker-playing agent might want to adopt a non-deterministic policy to prevent predictable behaviour. Without considering stochasticity in RL, we cannot define closeness between two states or the likelihood of obtaining a desired outcome.

State Distance

In supervised learning, Xfactuals are often considered to be obtained from the original instance using actions with deterministic outcomes. Few works acknowledge that actions transforming the original into a counterfactual instance can have non-deterministic outcomes [Delaney et al., 2021]. However, they only present the uncertainty associated with counterfactuals as additional information to the user, but do not use it as an important factor when choosing a counterfactual.

In RL, however, the stochasticity is often interwoven in the environment. For example, an agent's actions can have stochastic consequences, impacting the closeness between states. Two states that can be reached in one action might still be far away if the outcome of that action is highly uncertain. For example, consider a scenario where an automated loan approval system explains its decision to deny a loan to a user. An explanation could state that had they had more disposable income, the loan would have been approved. One possible action that transitions to the counterfactual state in that case is playing the lottery, which could increase their disposable income. However, while it is possible that taking this action only once will lead to the desired state, it is more likely that this action will have to be taken repeatedly. For that reason, stochasticity in the environment has to be considered when defining closeness between RL states.

Plausibility

In supervised learning, the plausibility of an instance is often evaluated by its position within the data manifold. Intuitively, if an Xfactual instance has features that are common in the dataset and there is a populated path between the original and counterfactual instance, it is considered to belong to a possible world [Poyiadzi et al., 2020]. For example, some methods aim to estimate how likely the counterfactual instance is given the training dataset by estimating data density around the counterfactual instance or on the path between the original and counterfactual instance [Poyiadzi et al., 2020]. Similarly, Kenny and Keane [2021] measured the distance from the semifactual to the original instance in the latent space to estimate its plausibility in the dataset. As there are no connections between states in supervised learning, relying on feature-based metrics is the only way to estimate the plausibility of the instance.

In RL, the likelihood of transitioning from one state to another depends on the agent's

policy. Xfactuals that are not likely under the agent's policy can be misleading. For example, consider again a farmer asking an agricultural AI tool why its yield prediction is not higher. Suppose this AI tool is extremely environmentally conscious and refrains from using pesticides. Without considering whether counterfactuals are possible under the agent's policy, an answer could state that had more pesticides been applied, the yield projection would increase. However, this counterfactual is not representative of the agent's behaviour, as the action of applying pesticides would not be taken by the AI. On the other hand, a counterfactual that considers the agent's policy might state that yield projection would be higher if more water is applied (given that applying water is more likely under the agent's policy). In order to estimate whether a state belongs to a possible world, not only features but also state transitions under the agent's policy have to be considered in RL.

Certainty of Outcome

In supervised learning, Xfactuals that deliver the desired outcome under stochastic conditions have been discussed under the name of robust or consistent counterfactuals [Mishra et al., 2021]. Research has been focusing on ensuring that Xfactuals are robust to small changes in input features [Dominguez-Olmedo et al., 2022] as well as changes to the model [Black et al., 2021, Upadhyay et al., 2021]. However, most of these methods rely on ensuring that the Xfactuals are found in the data manifold as a proxy for robustness.

In RL, however, actions can have stochastic outcomes, and stochastic events outside of the agent's control can affect the environment. For example, consider an AI medical tool erroneously treating a patient with the wrong treatment T, after which the patient's condition worsens. Suppose an alternative treatment T' was available, which in 99% of cases leads to moderate recovery and in 1% of cases to full recovery. A what-if question can then be answered by showing either the state of a moderately or fully recovered patient after administering T'. However, the first is more likely and representative of the environment, while the second is highly unlikely to be counterfactual, which can be misleading about the true effects of T'. For this reason, stochasticity has to be considered when defining the certainty of outcome for Xfactuals in RL.

Actionable Explanations

Actionability is another requirement of Xfactuals that is often considered essential to ensure that they are useful and not misleading the users. Actionability is often defined as the user's ability to enact changes to the original instance to turn it into the Xfactual. Non-actionable explanations could be those that change immutable features, such as race or birth country, or those that propose out-of-range changes, such as increasing salary by an extreme amount. Actionability has often been considered alongside or instead of plausibility.

Current methods for generating Xfactuals in supervised and RL inherently assume the

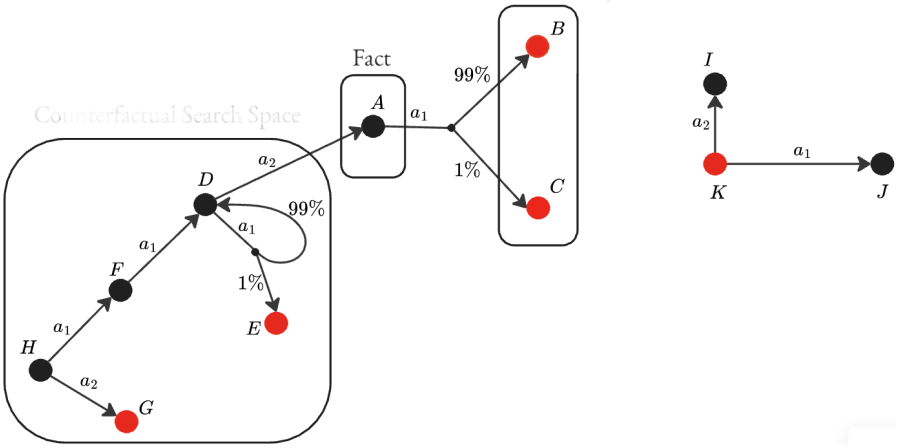


Figure 3.1: The effect of the stochastic and sequential nature of RL on Xfactual generation. To explain the factual state A with outcome y, we consider the red states B, C, E, G, and K as potential Xfactual explanations that achieve the desired outcome y'. States B and C correspond to forward semi/counterfactual states as they can be reached from the fact state. In contrast, states E and G are backwards semi/counterfactual states that are a product of different behaviour or stochastic conditions in the agent's past. 1) G is further away in terms of RL actions from A compared to B and C. 2) K is not temporally connected to A 3) Even though E can be reached in one action from D this action is uncertain as such E is on average more distant from A than G 4) Despite being equally distant from A in terms of actions, B is more likely than C.

user has some control over the input instance. In RL, however, changes to a state in RL can generally occur from two sources – the agent's actions or stochastic processes in the environment. While the first one results in actionable changes, the second one is non-actionable. For example, while an RL agent can choose to increase the amount of fertiliser in a farming task, it cannot control whether it rains. Thus, potential Xfactual explanations might differ from the factual state in the amount of fertiliser added, in the weather "observed", or in both the weather and fertiliser.

Previous work has considered actionability to be an important requirement of Xfactuals [Poyiadzi et al., 2020, Kenny and Keane, 2021] and preferred Xfactuals that utilise only actionable changes. However, it is difficult to directly translate the notion of actionability to RL, as users do not have agency over the agent's state. The changes to the environment can come from two sources – agents' actions or the stochastic processes in the environment. In this thesis, we consider Xfactuals based on both of these changes as essential for explaining the agent's behaviour. While the former can help the users understand how an agent's actions affect outcomes, the latter could explain how stochastic processes in the environment affect the outcome. In this thesis, we allow both types of changes and see them as equal when looking for Xfactuals. However, further research is needed into how these different types of Xfactuals affect user understanding of RL policies.

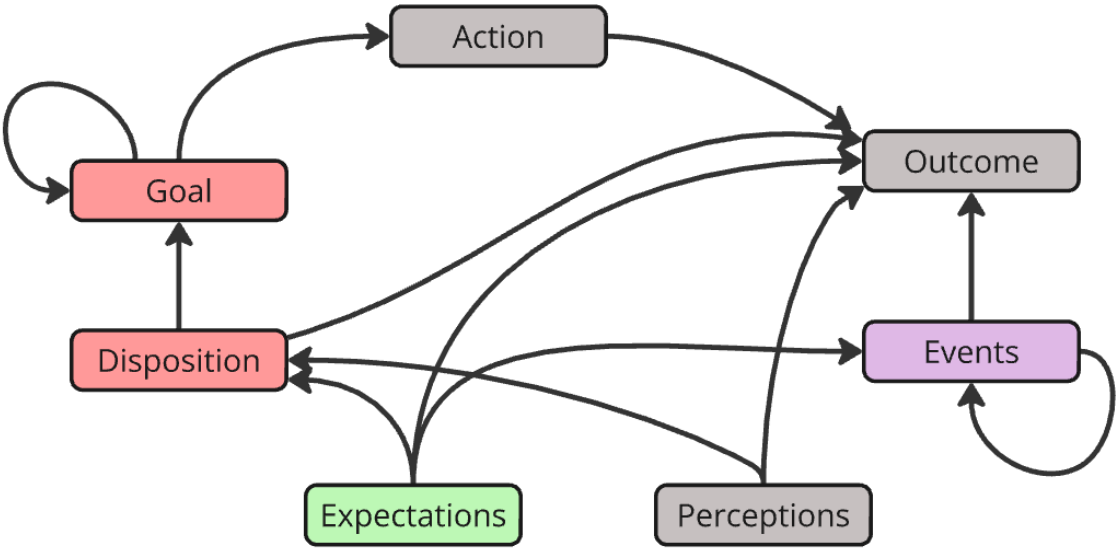


Figure 3.2: Causal xRL Framework as presented by Dazeley et al. [2021b]. RL outcomes can be explained through a large number of concepts, such as perceptions (corresponding to the agent's observations), the agent's goals or dispositions (corresponding to the objective function), and outside events or expectations.

3.1.3 Explanation Components

In supervised learning, Xfactual explanations rely only on input features to explain an outcome. As decisions in supervised learning are affected only by input features, this definition is reasonable.

However, in RL, multiple different factors can affect an agent's behaviour. According to the causal attribution model (CAM) presented by Dazeley et al. [2021a], an agent's decisions can be directly or indirectly affected by five factors – *perception* (corresponding to state features), *goals*, *disposition* (corresponding to objectives), *events*, and *expectations*. The visualisation of the CAM is presented in Figure 3.2. While Xfactual explanations could be used to understand the interplay between different components, their main purpose is in helping users understand the agent's behaviour. For this reason, in this work, we disentangle the explanation components and observe their effect on the agent's behaviour through the lens of Xfactual explanations. In this section, we explore how these different factors can be used to explain outcomes with Xfactuals.

State Features

Current Xfactuals explanations use an agent's perception as a variable and only consider state features as the possible causes of an outcome. The Xfactual question can then be phrased as

Goals

An agent's current goal can also influence an outcome. Xfactual explanations can then be used to address the question *"Given that outcome y was obtained in state x while following goal G, for what alternative goal G' would the desired outcome (y for semi-*

factuals and y′ for counterfactuals) be obtained?". This question is especially useful in multi-goal or hierarchical RL tasks [Al-Emran, 2015, Pateria et al., 2021], where the agent has a set of goals that need to be fulfilled and can switch between them. In such environments, knowing which goal an agent is pursuing in a state is necessary to fully understand its behaviour. For example, consider an autonomous taxi vehicle. The task of the ride-sharing agent can be split into two subgoals – picking up passengers and delivering them to their destinations. A passenger picked up in such a taxi might be confused if the agent takes a longer route to their destination. The passenger might lose trust in the agent and even suspect the agent is lost or malfunctioning. However, if an agent is allowed to explain its behaviour by employing a counterfactual statement: *"I am following a goal of picking up a different passenger. Had I followed a goal of delivering the current passenger, I would have taken the shorter route to the destination."*, the user can be reassured that the agent is indeed following the best course of action.

Objectives

In multi-objective environments, an agent's preference for a specific objective can guide its behaviour. This corresponds to the effect of the agent's disposition on its decisions as described in Dazeley et al. [2021a]. The Xfactual question can then be posed as *"Given that outcome y was obtained in state x while the agent was following objective O, for which alternative objective O′ would the desired outcome (y for semifactuals and y′ for counterfactuals) be obtained?"*. An agent's internal preference for one objective over another is likely to influence its actions, and this information is necessary to fully understand the agent's behaviour.

Many real-life tasks fit this description. For example, the behaviour of an autonomous driving agent is at each time step guided by different objectives such as speed, safety, fuel consumption, user comfort, and so on. Different agents can have different preferences in regard to these objectives. If an agent does not slow down in a critical situation, and as a consequence ends up colliding with the vehicle in front of it, the counterfactual explanation could identify that, had the agent preferred safety over speed in the critical state, it would have chosen to slow down and the accident would have been avoided. This type of explanation can also be useful for developers when designing multi-objective reward functions for complex tasks. Deciding on the correct weights for different objectives is a notoriously challenging task and requires substantial time and effort [Liu et al., 2014, Amodei et al., 2016, Pan et al., 2022, Gajcin et al., 2023]. Understanding how different preferences of objectives influence an agent's decision, especially in critical states, can be useful for adjusting the appropriate weights.

Events

Due to the stochasticity in the environment, Unexpected outside events can alter an agent's behaviour and, as such, should also be featured in Xfactual explanations. The counterfactual question can then be defined as *"Given the unexpected event E occurred and outcome y obtained in state x, for which event E′ would the desired outcome*

(y for semifactuals and y′ for counterfactuals) be obtained?". For example, consider an autonomous driving agent that had to change its trajectory due to an unexpected blockage on the road. The user might be confused as to why the agent decided on the longer path, and counterfactual explanations could restore their trust in the system by asserting that *"Had the shorter path not been blocked, I would have taken it."*. Such explanations might be useful in the field of safe RL, which argues that, to be safely deployed, agents must be able to avoid danger and correctly respond to unexpected events [Garcıa and Fernández, 2015].

Expectations

In the taxonomy introduced by Dazeley et al. [2021b], expectations refer to a set of social and cultural constraints that are imposed on the agent and can affect its decisions. These can be anything from social conventions and laws to ethical rules. Xfactual explanations could focus on how changing expectations affects an agent's behaviour by posing the question *"Given the expectation E_X occurred and outcome y was obtained in state x, for which expectation E'_X would the desired outcome (y for semifactuals and y′ for counterfactuals) be obtained?"*.

Including expectations in the explanations can help frame an agent's behaviour in the broader social and cultural context. Users who are not familiar with the expectations the agent is implementing may find it strange if the agent diverges from its primary goals due to a social norm. Explanations that identify social expectations as the cause of unexpected behaviour could help users understand and regain trust in the model. For example, a robot might need to explain that it chose to wait and hold the door for someone instead of rushing through to complete the task as fast as possible because it was following social expectations.

While decisions of a supervised learning model depend only on the input features, the RL agent's behaviour is often motivated by a wider range of causes, such as goals, objectives, and expectations. For that reason, the definition of Xfactual explanations can be extended to include all factors influencing an agent's decisions. In this thesis, we focus only on explanations that use state features to explain the outcome and discuss the broad set of other potential explanation components as future work (Section 7.3).

3.1.4 Explanation Outcomes

In supervised learning, Xfactual explanations are used to explain only one type of outcome – the model's predictions, as this is the only output of a black-box model in supervised learning.

In RL tasks, however, multiple possible scores could describe an agent's behaviour in a specific state. For example, an agent's decision in a specific state is not only represented by its action choice in that state. Each agent's decision often cannot be understood on its own but constitutes a part of a larger plan or policy. Relying only on a current action might not be enough to capture the desired change in behaviour, especially in complex,

continuous environments. In game environments, such as chess or StarCraft, the user might be more interested in knowing why an agent chose one specific plan or sequence of steps instead of another. Similarly, in an autonomous driving task with continuous control, the user is unlikely to ask the agent why it changed the steering angle by a small amount, but might be interested to know why it chose a specific route over another. Interpreting a single action in a complex or continuous environment might be too low-level for non-expert users, and better suited for experts or developers. Non-expert users, however, tend to be more comfortable with high-level explanations, corresponding to plans or policies. Xfactual explanations in RL have to be extended to include more complex behaviour that is inherent to RL agents to be user-friendly.

Furthermore, Xfactual explanations can be used to explain more complex outcomes than those directly relying on the agent's action choice. For example, one straightforward use for Xfactual explanations could be to explain the reward achieved in a specific timestep. Similarly, Xfactuals can be used to explain a specific positive or negative outcome, which can be described as a function of state features. For example, Xfactuals could be utilised to explain a car crash in an autonomous car, which can be deduced from the positions of the car and other vehicles. Additionally, the user might be inclined to define their own negative or positive outcomes that depend on their preferences. For example, a safety-oriented user might want the autonomous vehicle to explain every time it exceeds a user-defined speed limit. To account for diverse user needs, Xfactual explanations in RL need to allow different definitions of scores that extend beyond the agent's current action choice.

Despite their similarities, supervised and RL frameworks are inherently different in the types of tasks they solve, and as such, require different explanation methods. An overview of differences between the two frameworks from the perspective of Xfactuals is given in Table 3.6. In the remainder of this chapter, we redefine counterfactual (Section 3.5.3) and semifactual explanations for RL (Section 3.6).

3.2 Counterfactual Explanations – Redefined

In the previous section, we identified the main differences between supervised and RL from the perspective of counterfactual explanations. In this section, we lay the theoretical foundations for defining counterfactual explanations for stochastic and sequential RL tasks. Specifically, in this section we 1) redefine the concepts of counterfactual variable and outcome for RL (Section 3.5.1) 2) propose a set of counterfactual requirements in RL (Section 3.5.2) and 3) provide an RL-specific definition of counterfactual explanations for RL (Section 3.5.3). This section is in large part based on the publication Gajcin and Dusparic [2024] (publication 1 as listed in Section 1.5).

Table 3.1: Overview of the differences between supervised and RL from the perspective of Xfactual explanations.

Framework	Supervised Learning	Reinforcement Learning	
Framework requirement	One-step prediction	Sequential Execution	Stochastic Conditions
State Distance	Feature similarity metrics (e.g. proximity, sparsity)	Temporal distance metrics (e.g. number of actions)	Transition probability metrics (e.g. average number of actions)
Outcome Certainty	Data density metrics (e.g. data manifold closeness, robustness)	–	Stochastic metrics (e.g. outcome probability)
Plausibility	Data density metrics (e.g. data manifold closeness, robustness)	Connection constraints (e.g. RL path existence)	Outcome likelihood metrics (e.g. average success rate)
Counterfactual Variables	Input Features	State features Goals Objectives Outside Events Expectations	
Counterfactual Outcome	Model prediction	Action choice Plan choice Policy choice Inferred outcomes (e.g. low reward) User-defined outcomes	

3.2.1 Counterfactual Variables and Outcomes

To redefine counterfactual explanations, we build on the current definition used by methods in both supervised and reinforcement learning. Wachter et al. [2017] define counterfactuals as:

> Score p was returned because variables V had values $(v_1, v_2, \dots)$ associated with them. If V instead had values $(v_1', v_2', \dots)$ and all other variables had remained constant, score p' would have been returned.

Score in this context corresponds to the notion of explanation outcome as discussed before. In RL, current counterfactual explanations focus on interpreting a single decision in a specific state [Olson et al., 2019, Huber et al., 2023]. Variables are simply state features, and the score is an agent's choice of action in the state. This corresponds to the definition of counterfactuals used in supervised learning, where input features are replaced by state features and a model's prediction of a class label is with an action prediction. However, as discussed in Sections 3.1.3 and 3.4.1, RL offers a broader set of possible explanation components and outcomes that can be used for defining counterfactuals.

To redefine counterfactual explanations for RL, we start by redefining the concept of the counterfactual variable for RL.

Definition 3.2.1 (Counterfactual Variable in RL)**.** Given the original state x and outcome y being explained, the counterfactual variables can be either state features, the agent's goals, the agent's objectives, outside events, or expectations.

The goal of counterfactuals is to explain the outcome of an RL policy. While current work focuses on explaining only the agent's current decision, Section 3.4.1 identifies other plausible outcomes, such as the agent's plans or user-defined outcomes that can be explained. We propose a broad definition of outcome in this work, referring to any function of the state.

Definition 3.2.2 (Counterfactual Outcome in RL)**.** Given a state x the counterfactual outcome O is any function $\mathcal{F} : \mathcal{S} \to \mathbb{R}$ of the state x:

$$O(x) \coloneqq \mathcal{F}(x) \tag{3.1}$$

In this way, counterfactuals can be used to explain the agent's choice of action or plan, a reward function achieved in the state, or another user-defined outcome.

Having defined the variables and outcomes in the RL setting, we can form counterfactuals by changing variables in order to achieve the desired outcome. However, according to the definition, essentially any RL state achieving a desired outcome can be considered counterfactual. This can include uninformative or even misleading counterfactuals. For that reason, in the next section, we define the requirements for counterfactual explanations in RL that can be used to evaluate and search for informative counterfactuals.

3.2.2 Counterfactual Explanation Requirements

There are often a large number of alternative states that achieve the desired outcome. To choose the most useful one, the concept of the closest possible world is often used. However, we cannot reuse the same requirements describing the closest possible world from current work, as they do not account for the stochastic and sequential nature of RL tasks (Section 3.1). For this reason, in this section, we define five requirements that describe the closest possible world counterfactual in sequential and stochastic RL tasks:

Definition 3.2.3 (Counterfactual Requirements for RL)**.** Given a state x_n where outcome $y = O(x_n)$ is being explained, a sequence of previous k state-action pairs $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ leading to x_n, and a counterfactual outcome y', a counterfactual x' should satisfy the following requirements:

1. **Validity:** The desired outcome y' should be achieved in the counterfactual state x'. Depending on the definition of outcome, this could mean the agent choosing a target action in x', the agent achieving a higher value of the reward function, or obtaining another user-defined metric in x'.

2. **Reachability:** The counterfactual x' should be connected to the original instance through the underlying temporal connection of the RL environment using RL actions. Specifically, x' should be reachable from the original state x_n or from another state from the agent's past. In this way, a counterfactual answers the what-if question by exploring alternative past and future worlds.

3. **Recency**: Counterfactuals that are reachable in a few steps should be preferred over more distant ones.

4. **Plausibility**: Counterfactual x' should be representative of the agent's policy π that is being explained. That means it should be reachable not by just any policy, but by π.

5. **Certainty of outcome**: The counterfactual should be reachable with high certainty even in stochastic environments. This means that, given two potential counterfactuals, the one that can be reached via a more certain path should be preferred.

These requirements are inspired partly by their counterparts in supervised learning, adapted to stochastic and sequential RL tasks and partly by the counterfactual explanation research in psychology. Validity defined above, for example, corresponds to the notion of validity used in supervised learning approaches.

Reachability corresponds in part to multiple counterfactual requirements used in supervised learning, such as proximity (to ensure closeness of instances), data manifold closeness (ensuring a path between the original and counterfactual), and actionability (preventing changing of immutable features). Counterfactuals that are not reachable can be misleading, as they might represent unrealistic changes to the original instance.

Similarly, research in human reasoning uncovered that humans reason about changing the most recent events to explain outcomes [Byrne, 2019]. We apply this insight to define recency. For example, a medical AI tool explaining its decision to prescribe treatment should not tell the user that if conditions 10 years ago were different, they would have chosen a different treatment if there exists a more recent counterfactual. This requirement is similar to the proximity and sparsity requirements defined for supervised learning, as they attempt to ensure "closeness" between the counterfactual and original instance.

Plausibility, as defined above, corresponds to similar metrics such as plausibility or data manifold closeness in supervised learning, which ensure the counterfactual is representative of the dataset. Counterfactuals that are not likely under the agent's policy will not be representative of its behaviour. For example, a counterfactual showing that patients would have improved if they received a highly risky treatment is not representative of a cautious AI medical tool's behaviour and cannot be used to explain it.

In RL, the stochastic nature of the environment can make a counterfactual instance invalid during the time it takes to reach it from the original state. While validity ensures that the desired outcome is achieved in the counterfactual state, it does not measure how likely it is to reach the counterfactual state under stochastic conditions. For example, consider an AI medical tool explaining why the patient has not been discharged after unsuccessfully applying treatment with a 10% chance of full recovery and 50% chance of moderate recovery. An uncertain counterfactual would show the patient fully recovered and state that if the treatment worked, they would be discharged. A more certain counterfactual would offer the state in which the patient is moderately recovered and ready for discharge. The uncertain counterfactual might be misleading about the efficacy of the treatment. Certainty of outcome is similar to validity in supervised learning (ensuring the counterfactual outcome) or plausibility (ensuring a highly dense path between the original and counterfactual instance) in supervised learning.

Table 3.2: Comparison between the counterfactual requirements in supervised and RL. Causality is implicitly satisfied through reachability, as executing RL actions maintains the causal relationships in the environment.

Supervised Learning		Reinforcement Learning	
Counterfactual requirement	**Promotes counterfactuals that:**	**Counterfactual requirement**	**Promotes counterfactuals that:**
Validity	Achieve the desired outcome	Validity	Achieve the desired outcome
Proximity	Have similar features to x	Recency	Can be reached in few steps
Sparsity	Change few features		
Actionability	Do not change immutable features	Reachability	Can be reached using RL actions
Plausibility	Likely under the dataset	Plausibility	Likely under the agent's policy
Robustness	Maintain outcome under small changes	Certainty of Outcome	Deliver the desired outcome with high probability
Causality	Preserve causal feature dependencies	Reachability (implicitly)	Can be reached using RL actions

While these requirements have been redefined to fit the stochastic and sequential RL tasks, they still share the overall goals of close plausible counterfactuals that deliver the desired outcome and are compatible with similar requirements used in supervised learning. However, there are other requirements, more prominently used in supervised learning, that are either obsolete or do not translate to RL tasks. For completeness, we list them here:

1. **Actionability**: As discussed in Section 3.1, actionability is one of the common requirements of counterfactuals in supervised learning. In RL, however, both counterfactuals that can be reached by RL actions and those that are a consequence of a change in stochastic conditions in the environment can be useful to explain the effect of stochastic events on an agent's behaviour. For example, a non-actionable counterfactual in an agricultural scenario could explain that had the temperature been higher the agent would advise less watering. While neither the RL agent nor the user has control over the temperature, this explanation can help the user understand the effect of temperature on the agent's actions and reassure them that the agent is not wasteful and knows when to save water.

2. **Causality**: In supervised learning, the underlying causal model is often unknown, and changing features risks making changes that do not correspond to the causal rules for the environment. For example, increasing the education level in a loan applicant's profile without adjusting their age would break the causal relationship between the two. However, this requirement is obsolete in RL if a requirement such as reachability is considered. As RL actions implicitly encode the rules of the environment, counterfactuals reachable by those actions will still follow causal rules.

3. **Feature-based Similarity Metrics (Proximity/Sparsity)**: While they are used in supervised learning to estimate the similarity between instances, relying on these requirements can be misleading in RL. For example, despite the small difference in state features, two states can be far apart from each other in terms of execution and vice versa.

In this section, we defined five counterfactual requirements which can be used to evaluate counterfactual explanations in RL. Some of the requirements are translated from supervised learning, while some are RL-specific and capture the stochastic and sequential

nature of RL tasks. Table 3.7 provides a summary of The counterfactual requirements of RL and their counterparts in supervised learning.

3.2.3 Counterfactual Explanations for RL

After having defined the concepts of counterfactual variable, counterfactual outcome, and the requirements for counterfactual explanations in RL, we now formalise a definition of counterfactual explanations for the RL task.

Definition 3.2.4 (Counterfactual explanations for RL)**.** Suppose a state x_n where an outcome $y = O(x_n)$ is being explained, and let the counterfactual outcome be y'. Suppose a sequence of previous k state-action pairs $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ leading to x_n is available. Given a counterfactual variable V with value v in state x_n we define two types of counterfactual explanations:

- **Backward counterfactual** x' is the *most recent plausible state* in which variable v takes an alternative value v' that delivers the counterfactual outcome y' with high certainty, such that there exists a trajectory P starting in x_{n-k} and leading to x'

$$x' = P(x_{n-k}) \tag{3.2}$$

- **Forward counterfactual** x' is the *most recent plausible state* in which variable v takes an alternative value v' that delivers the counterfactual outcome y' with high certainty such that there exists a trajectory P starting in x_n and leading to x'

$$x' = P(x_n) \tag{3.3}$$

The definition of counterfactuals relies on finding an alternative world where the counterfactual variable holds a different value. The counterfactual variable can be a state feature, the agent's goal, an objective or expectation, or an outside event. Depending on the choice of counterfactual variable, five different counterfactual explanations can be formed for any state. We summarise these in Table 3.8. In this thesis, we focus on explanations that use features as variables and agents' actions as the outcome and discuss other counterfactual explanation types as future work in Section 7.3.

We note that forward counterfactuals are sometimes referred to as prefactuals in psychology research, while backwards counterfactuals are referred to as simply counterfactuals. To prevent confusion, we rename these explanations as backward and forward counterfactuals. We make a formal distinction between backward and forward counterfactuals, as previous research in psychology found that they evoke different cognitive processes in humans when used to explain decisions of supervised learning models [Dai et al., 2022].

We also note here that we purposefully keep the definition of a counterfactual explanation in RL abstract, without explicitly formalising the concepts such as recency and plausibility at this stage. This is in line with high-level definitions provided for coun-

Table 3.3: Different formulations of forward and backwards counterfactual explanations for different choices of counterfactual variables.

Counterfactual Variable	Backward Counterfactual Explanation	Forward Counterfactual Explanation
Feature	Had the state features taken different values (in the past), outcome y' would be achieved instead of y	If the state features were to take different values (in the future), outcome y' would be achieved instead of y
Goal	Had the agent followed a different goal (in the past), outcome y' would be achieved instead of y	If the agent were to follow a different goal (in the future), outcome y' would be achieved instead of y
Objective	Had the agent preferred a different objective (in the past), outcome y' would be achieved instead of y	If the agent were to prefer a different objective (in the future), outcome y' would be achieved instead of y
Events	Had the agent encountered a different event (in the past), outcome y' would be achieved instead of y	If the agent were to encounter a different event (in the future), outcome y' would be achieved instead of y
Expectations	Had the agent faced a different expectation (in the past), outcome y' would be achieved instead of y	If the agent were to face a different expectation (in the future), outcome y' would be achieved instead of y

terfactuals in supervised learning [Wachter et al., 2017]. In Section 4.2.2, we provide concrete metrics for evaluating these requirements for a specific experimental use case. However, we acknowledge that explanation quality is inherently subjective, and different use cases might require different metrics to evaluate these requirements. For example, when explaining a non-deterministic policy, a plausible counterfactual might be reached with high probability. For a deterministic policy, however, plausibility might be measured by how "good" actions leading to the counterfactual are according to the policy (e.g. by using Q-values). We expect this definition to provide a guideline for searching for counterfactual explanations, with low-level implementation of the requirements subject to use case and user preference.

3.3 Semifactual Explanations – Redefined

In supervised learning, semifactual explanations are often defined as the most distant neighbour of the original instance. Distance between instances is defined using feature-based similarity metrics (e.g. proximity and sparsity), while plausibility is evaluated using data density metrics. However, as shown in Section 3.1, due to the stochastic and sequential nature of RL tasks, the definitions of distance and plausibility do not translate straightforwardly from supervised to RL tasks.

In this section, we redefine semifactual explanations for RL tasks. Firstly, we start by redefining the concepts of semifactual variables and outcomes for RL (Section 3.6.1). Next, in Section 3.6.2, we propose a set of requirements for semifactuals to satisfy in stochastic and sequential tasks, and finally, in Section 3.6.3, we propose a new definition for semifactual explanations in RL.

3.3.1 Semifactual Variables and Outcomes

To redefine semifactual explanation we build on their current definition in xAI. However, there is no formal definition of semifactuals in current work, and they are often defined informally as a special case of counterfactual explanations that do not change the outcome [Aryal and Keane, 2023]. For this reason, we utilise the same definition used for counterfactual explanations [Wachter et al., 2017], but ensure that the outcome remains the same instead of being changed:

> Score p was returned because variables V had values $(v_1, v_2, \dots)$ associated with them. Even if V instead had values $(v_1', v_2', \dots)$ and all other variables had remained constant, the same score p would have been returned.

Score in this context corresponds to the notion of explanation outcome as discussed before. In supervised learning, variables are simply input features, and the outcome is the prediction label. In RL, however, a broader range of variables and outcomes can be defined to explain the complex behaviour of RL agents. Firstly, we define the semifactual variable in RL as:

Definition 3.3.1 (Semifactual Variable in RL)**.** Given an original state x being explained, a semifactual x' can be obtained by changing state variable values, the agent's goals and objectives, outside events, or expectations.

Similar to counterfactual explanations (Section 3.5.3), the goal of the semifactuals is to explain an outcome. While in supervised learning semifactuals have been used to explain only model output, in RL, multiple possible outcomes can be explained (Section 3.4.1). For this reason, we implement a broad definition (same as for counterfactual outcome):

Definition 3.3.2 (Semifactual Outcome for RL)**.** Given a state x the semifactual outcome is any function $\mathcal{F} : \mathcal{S} \to \mathbb{R}$ of the state x:

$$O(s) \coloneqq \mathcal{F}(s) \tag{3.4}$$

In this way, semifactual explanations can be used to explain the agent's actions and plans, but also more complex outcomes such as the agent's rewards or user-defined outcomes.

Having defined the semifactual variables and outcomes in RL, we can generate semifactuals by changing variables while maintaining the outcome. However, this would lead to a large space of potential semifactual explanations, as each RL state with different variables but the same outcome would be considered. For this reason, the next section introduces a set of semifactual requirements which can be used to evaluate and search for informative semifactuals.

3.3.2 Semifactual Explanation Requirements

In order to define a good semifactual, we also need to define the most distant neighbour in RL. To that end, we define a set of six semifactual requirements that take into account the stochastic and sequential nature of RL tasks, and describe a most distant neighbour semifactual:

Definition 3.3.3 (Semifactual Requirements for RL)**.** Given a state x_n where outcome $y = O(x_n)$ is being explained, a set of previous k state-action pairs $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ leading to x_n a semifactual x' should satisfy the following objectives:

1. **Validity:** The semifactual state x' should deliver the semifactual outcome y.
2. **Reachability:** The semifactual x' should be connected to the original instance through the underlying temporal connections of the RL environment. Specifically, x' should be reachable from the original state x_n or from another state from the agent's past.
3. **Recency**: The semifactual x' that are reachable in a few steps should be preferred over more distant ones.
4. **Plausibility:** The semifactual x' should be representative of the agent's policy π. That means it should be reachable not by just any policy, but by π.
5. **Unexpectedness:** The semifactual x' should be a consequence of an unexpected event.
6. **Argument Strength:** To be informative, a semifactual x' explanation must offer a stronger argument for the outcome than the original state x.

The defined requirements are inspired in part by their counterparts for supervised learning, adapted to sequential and stochastic RL tasks, and in part by semifactual research in psychology. Validity as defined above, for example, corresponds to the one used for counterfactuals (Section 3.5.2) but maintains the outcome. It also corresponds to the notion of validity for semifactuals in supervised learning [Aryal and Keane, 2023].

Reachability is the same as the definition of reachability for counterfactual explanations in Section 3.5.2. Additionally, reachability is similar to the metrics of actionability or data manifold closeness, which ensures the semifactual is represented in the dataset. Semifactuals that are not reachable by using RL actions might represent states that are not representative of the RL environment.

Recency corresponds to the requirement of the same name defined for the counterfactuals in Section 3.5.2. Additionally, this requirement is similar to sparsity and proximity, which are sometimes used in supervised learning, to ensure closeness between the original and semifactual instance. Intuitively, telling a farmer that increased watering of the fields 10 years ago would not have increased the yield is not as reassuring as indicating that increased watering last month would also not have changed the outcome.

This definition of plausibility corresponds to the plausibility as defined for counterfactual explanations (Section 3.5.2). Additionally, it is similar to the data manifold closeness and plausibility metrics used in supervised learning. A semifactual that is not representative of the agent's policy can be misleading. For example, consider a cautious AI tool explaining why it chose to treat a patient with a high-cost, highly efficient treatment T. A semifactual could state that even if a more risky treatment T' was applied earlier, the agent would choose T at this stage. This semifactual is not representative of the agent's policy, as it shows a state in which the cautious agent never would have ended up. While this explanation gives us an insight into the connection between different treatments and the disease, it does not explain why the agent applied treatment T. An

alternative semifactual states that even if the disease progressed more quickly, the agent would still choose treatment T gives us more information about the decisions and is more representative of the agent's policy, as it could be reached by the agent's policy under different stochastic conditions.

The unexpectedness requirement stems from the work of Aryal and Keane [2023], who argue that semifactuals should be surprising in supervised learning. For example, knowing that the yield would have been the same even if the field suffered a drought might reassure the farmer in their decisions and reinforce their trust in the system. Thus, informative semifactuals ought to be exceptional.

The requirement for the strength of the argument is inspired by the works presenting semifactuals as *a fortiori* arguments [Nugent et al., 2009]. For example, an AI medical tool prescribing the most aggressive possible treatment can explain its decision by stating that even if the disease were twice as difficult, it would have made the same choice. This explanation, however, does not offer any new information about the agent's decision-making process, as the original decision is already extreme. An alternative semifactual stating that even if the disease was less dangerous it would still prescribe the same treatment offers a stronger argument for understanding the agent as favouring harsher treatments.

The defined requirements correspond in part to semifactual requirements developed for supervised learning but adapted for stochastic and sequential tasks. However, there are requirements often used in supervised learning that do not directly translate to RL settings. For completeness, we list these here:

1. **Actionability**: As discussed in section 3.1, actionability does not directly translate from supervised into RL. While we require semifactuals to be reachable from the original instance, we do not make a distinction between changes that occur via the agent's actions or changes in stochastic conditions, as both of these have a role in explaining RL agents.

2. **Causality**: Similar to counterfactuals, semifactuals are not supposed to violate the causal constraints of the task. While in supervised learning, the causal model is often difficult to obtain, in RL, it is ingrained in the environment. As long as the semifactual search relies on RL actions, causal constraints will not be violated. For this reason, we do not include causality as a desired requirement of semifactuals for RL.

3. **Feature-based similarity metrics**: Previous works in supervised learning argue that semifactuals should be allowed to change only one feature in the original state [Kenny and Keane, 2021, Aryal and Keane, 2023]. However, following this constraint in RL would automatically reject semifactuals that are close in execution to the original instance but have changes in multiple features, some of them not necessarily important for the decision choice.

In this section, we introduced a set of requirements for evaluating semifactuals in stochas-

Table 3.4: Comparison of semifactual requirements for supervised and RL..

Supervised Learning		Reinforcement Learning	
Semifactual requirement	**Promotes semifactual that:**	**Semifactual requirement**	**Promotes semifactuals that:**
Validity	Achieve the desired outcome	Validity	Achieve the desired outcome
Proximity	Have similar features to x	Recency	Can be reached in few steps
Sparsity	Change few features		
Actionability	Do not change immutable features	Reachability	Can be reached using RL actions
Plausibility	Likely under the dataset	Plausibility	Likely under the agent's policy
Surprising	Surprising/Shocking	Unexpectedness	Follow unexpected events
Convincing	Provide strong arguments for the decision	Argument strength	Provide strong arguments for the decision
Causality	Preserve causal feature dependencies	Reachability (implicitly)	Can be reached using RL actions

tic and sequential tasks. While some of these requirements are adapted from similar ones in supervised learning, some represent novel RL-specific requirements. In Table 3.9, we summarise the main similarities and differences between counterfactual requirements in supervised and RL. In the next section, we use the defined concepts of semifactual variables and outcomes and the semifactual requirements in RL to define semifactual explanations for RL.

3.3.3 Semifactual Explanations for RL

In previous sections, we formalised the concepts of semifactual variables and outcomes, as well as proposed a set of requirements that describe the most distant neighbour semifactuals in RL. With that in mind, we define semifactual explanations for RL tasks:

Definition 3.3.4 (Semifactual explanations for RL)**.** Suppose a state x_n where an outcome $y = O(x_n)$ is being explained. Let $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ be a sequence of previous k state-action pairs leading to x_n. Given a semifactual variable V taking value v in x_n we define two types of semifactual explanations:

- **Backward semifactual** x' is the *most recent, unexpected, plausible, argument-strong state* in which variable v takes an alternative value v' that still delivers the outcome y, such that there exists a sequence of state-action pairs P starting in x_{n-k} and leading to x'

$$x' = P(x_{n-k}) \tag{3.5}$$

- **Forward semifactual** x' is the *most recent, unexpected, plausible, argument-strong state* in which variable v takes an alternative value v' that still delivers the outcome y, such that there exists a sequence of state-action pairs P starting in x_n and leading to x'

$$x' = P(x_n) \tag{3.6}$$

Intuitively, we define semifactuals as alternative states that differ in some semifactual variable V from the original state. Variable V can be either one of the five semifactual variables defined in Section 3.1.3 – state features, goals, objectives, expectations, or events. Depending on the variable, different semifactual explanations can be formed. For the summary of different semifactuals depending on the choice of semifactual variable, see Table 3.10.

Table 3.5: Different formulations of forward and backward semifactual explanations for different choices of semifactual variables.

Semifactual Variable	Backward Semifactual Explanation	Forward Semifactual Explanation
Feature	Even if the state features taken different values (in the past), outcome y would still be achieved.	Even if the state features take different values (in the future), outcome y will still be achieved.
Goal	Even if the agent followed a different goal (in the past), outcome y would still be achieved.	Even if the agent follows a different goal (in the future), outcome y will still be achieved.
Objective	Even if the agent preferred a different objective (in the past), outcome y would still be achieved.	Even if the agent prefers a different objective (in the future), outcome y will still be achieved.
Events	Even if the agent encountered a different event (in the past), outcome y would still be achieved.	Even if the agent encounters a different event (in the future), outcome y will still be achieved.
Expectations	Even if the agent faced a different expectation (in the past), outcome y' would still be achieved.	Even if the agent faces a different expectation (in the future), outcome y will still be achieved.

The above definition is similar to that in supervised learning, as both define semifactuals as the most distant neighbour. However, the distance includes RL-specific requirements of unexpectedness, plausibility, and argument strength, which are based on the stochastic and sequential nature of RL tasks. Additionally, our definition distinguishes between forward and backward semifactuals, which search for the semifactual in the agent's future and past experiences, respectively.

3.4 Chapter Summary and Contributions

This section focuses on two research questions posited in Section 1.2. Firstly, our goal was to answer RQ1, which deals with redefining counterfactual explanations for RL

> How can the counterfactual concept of the *closest possible world* be defined in stochastic and sequential environments?

Secondly, we tackled RQ2, which aims to redefine semifactual explanations for RL:

> How can the semifactual concept of the *most distant neighbour* be defined in stochastic and sequential environments?

To answer these questions, we investigated how counterfactual and semifactual explanations can be defined in RL tasks. Firstly, we analysed the differences between supervised and RL from the perspective of counterfactual and semifactual explanations (Section 3.1). We identified 1) sequential execution, 2) stochastic conditions and 3) the broader set of explanation components in RL as the main factors that require the redefinition of counterfactuals and semifactuals for RL. In Section 3.5.3, we redefined the concepts of counterfactual variable and outcome for RL tasks. Additionally, we provided high-level guidelines for finding the closest possible world in stochastic and sequential environments by defining five RL-specific counterfactual requirements. Section 3.6 provides the same redefinition for semifactual explanations by first redefining the semifactual variable and outcome, and providing six RL-specific semifactual requirements to serve as guidelines for choosing the most distant neighbour. *"Given that outcome y was obtained in state x with state feature values $f_1, ..., f_k$, how can the features be changed for a desired outcome (y for semifactuals and y' for counterfactuals) to be obtained?". This type of explanation is most suitable for single-goal, single-objective, deterministic tasks, where*

each action depends only on the current state the agent is perceiving. In such tasks, it can be assumed that the user knows the ultimate goal of the agent's behaviour and the agent does not need to balance between different objectives. In such environments, an agent's actions have deterministic consequences, and unexpected events cannot influence the agent's behaviour. This corresponds to simple environments, such as deterministic games, where the agent's goal is to win by optimising a single objective function. In more complex, real-world environments, however, relying only on the current state to make a decision is unlikely, and additional factors might need to be taken into account.

Goals

An agent's current goal can also influence an outcome. Xfactual explanations can then be used to address the question "Given that outcome y was obtained in state x while following goal G, for what alternative goal G' would the desired outcome (y for semi-factuals and y' for counterfactuals) be obtained?". This question is especially useful in multi-goal or hierarchical RL tasks [Al-Emran, 2015, Pateria et al., 2021], where the agent has a set of goals that need to be fulfilled and can switch between them. In such environments, knowing which goal an agent is pursuing in a state is necessary to fully understand its behaviour. For example, consider an autonomous taxi vehicle. The task of the ride-sharing agent can be split into two subgoals – picking up passengers and delivering them to their destinations. A passenger picked up in such a taxi might be confused if the agent takes a longer route to their destination. The passenger might lose trust in the agent and even suspect the agent is lost or malfunctioning. However, if an agent is allowed to explain its behaviour by employing a counterfactual statement: "I am following a goal of picking up a different passenger. Had I followed a goal of delivering the current passenger, I would have taken the shorter route to the destination.", the user can be reassured that the agent is indeed following the best course of action.

Objectives

In multi-objective environments, an agent's preference for a specific objective can guide its behaviour. This corresponds to the effect of the agent's disposition on its decisions as described in Dazeley et al. [2021a]. The Xfactual question can then be posed as "Given that outcome y was obtained in state x while the agent was following objective O, for which alternative objective O' would the desired outcome (y for semifactuals and y' for counterfactuals) be obtained?". An agent's internal preference for one objective over another is likely to influence its actions, and this information is necessary to fully understand the agent's behaviour.

Many real-life tasks fit this description. For example, the behaviour of an autonomous driving agent is at each time step guided by different objectives such as speed, safety, fuel consumption, user comfort, and so on. Different agents can have different preferences in regard to these objectives. If an agent does not slow down in a critical situation, and as a consequence ends up colliding with the vehicle in front of it, the counterfactual explanation could identify that, had the agent preferred safety over speed in the critical state,

it would have chosen to slow down and the accident would have been avoided. This type of explanation can also be useful for developers when designing multi-objective reward functions for complex tasks. Deciding on the correct weights for different objectives is a notoriously challenging task and requires substantial time and effort [Liu et al., 2014, Amodei et al., 2016, Pan et al., 2022, Gajcin et al., 2023]. Understanding how different preferences of objectives influence an agent's decision, especially in critical states, can be useful for adjusting the appropriate weights.

Events

Due to the stochasticity in the environment, Unexpected outside events can alter an agent's behaviour and, as such, should also be featured in Xfactual explanations. The counterfactual question can then be defined as "Given the unexpected event E occurred and outcome y obtained in state x, for which event E′ would the desired outcome (y for semifactuals and y′ for counterfactuals) be obtained?". For example, consider an autonomous driving agent that had to change its trajectory due to an unexpected blockage on the road. The user might be confused as to why the agent decided on the longer path, and counterfactual explanations could restore their trust in the system by asserting that "Had the shorter path not been blocked, I would have taken it.". Such explanations might be useful in the field of safe RL, which argues that, to be safely deployed, agents must be able to avoid danger and correctly respond to unexpected events [Garcıa and Fernández, 2015].

Expectations

In the taxonomy introduced by Dazeley et al. [2021b], expectations refer to a set of social and cultural constraints that are imposed on the agent and can affect its decisions. These can be anything from social conventions and laws to ethical rules. Xfactual explanations could focus on how changing expectations affects an agent's behaviour by posing the question "Given the expectation E_X occurred and outcome y was obtained in state x, for which expectation E'_X would the desired outcome (y for semifactuals and y′ for counterfactuals) be obtained?".

Including expectations in the explanations can help frame an agent's behaviour in the broader social and cultural context. Users who are not familiar with the expectations the agent is implementing may find it strange if the agent diverges from its primary goals due to a social norm. Explanations that identify social expectations as the cause of unexpected behaviour could help users understand and regain trust in the model. For example, a robot might need to explain that it chose to wait and hold the door for someone instead of rushing through to complete the task as fast as possible because it was following social expectations.

While decisions of a supervised learning model depend only on the input features, the RL agent's behaviour is often motivated by a wider range of causes, such as goals, objectives, and expectations. For that reason, the definition of Xfactual explanations

can be extended to include all factors influencing an agent's decisions. In this thesis, we focus only on explanations that use state features to explain the outcome and discuss the broad set of other potential explanation components as future work (Section 7.3).

3.4.1 Explanation Outcomes

In supervised learning, Xfactual explanations are used to explain only one type of outcome – the model's predictions, as this is the only output of a black-box model in supervised learning.

In RL tasks, however, multiple possible scores could describe an agent's behaviour in a specific state. For example, an agent's decision in a specific state is not only represented by its action choice in that state. Each agent's decision often cannot be understood on its own but constitutes a part of a larger plan or policy. Relying only on a current action might not be enough to capture the desired change in behaviour, especially in complex, continuous environments. In game environments, such as chess or StarCraft, the user might be more interested in knowing why an agent chose one specific plan or sequence of steps instead of another. Similarly, in an autonomous driving task with continuous control, the user is unlikely to ask the agent why it changed the steering angle by a small amount, but might be interested to know why it chose a specific route over another. Interpreting a single action in a complex or continuous environment might be too low-level for non-expert users, and better suited for experts or developers. Non-expert users, however, tend to be more comfortable with high-level explanations, corresponding to plans or policies. Xfactual explanations in RL have to be extended to include more complex behaviour that is inherent to RL agents to be user-friendly.

Furthermore, Xfactual explanations can be used to explain more complex outcomes than those directly relying on the agent's action choice. For example, one straightforward use for Xfactual explanations could be to explain the reward achieved in a specific timestep. Similarly, Xfactuals can be used to explain a specific positive or negative outcome, which can be described as a function of state features. For example, Xfactuals could be utilised to explain a car crash in an autonomous car, which can be deduced from the positions of the car and other vehicles. Additionally, the user might be inclined to define their own negative or positive outcomes that depend on their preferences. For example, a safety-oriented user might want the autonomous vehicle to explain every time it exceeds a user-defined speed limit. To account for diverse user needs, Xfactual explanations in RL need to allow different definitions of scores that extend beyond the agent's current action choice.

Despite their similarities, supervised and RL frameworks are inherently different in the types of tasks they solve, and as such, require different explanation methods. An overview of differences between the two frameworks from the perspective of Xfactuals is given in Table 3.6. In the remainder of this chapter, we redefine counterfactual (Section 3.5.3) and semifactual explanations for RL (Section 3.6).

Table 3.6: Overview of the differences between supervised and RL from the perspective of Xfactual explanations.

Framework	Supervised Learning	Reinforcement Learning	
Framework requirement	One-step prediction	Sequential Execution	Stochastic Conditions
State Distance	Feature similarity metrics (e.g. proximity, sparsity)	Temporal distance metrics (e.g. number of actions)	Transition probability metrics (e.g. average number of actions)
Outcome Certainty	Data density metrics (e.g. data manifold closeness, robustness)	–	Stochastic metrics (e.g. outcome probability)
Plausibility	Data density metrics (e.g. data manifold closeness, robustness)	Connection constraints (e.g. RL path existence)	Outcome likelihood metrics (e.g. average success rate)
Counterfactual Variables	Input Features	State features Goals Objectives Outside Events Expectations	
Counterfactual Outcome	Model prediction	Action choice Plan choice Policy choice Inferred outcomes (e.g. low reward) User-defined outcomes	

3.5 Counterfactual Explanations – Redefined

In the previous section, we identified the main differences between supervised and RL from the perspective of counterfactual explanations. In this section, we lay the theoretical foundations for defining counterfactual explanations for stochastic and sequential RL tasks. Specifically, in this section we 1) redefine the concepts of counterfactual variable and outcome for RL (Section 3.5.1) 2) propose a set of counterfactual requirements in RL (Section 3.5.2) and 3) provide an RL-specific definition of counterfactual explanations for RL (Section 3.5.3). This section is in large part based on the publication Gajcin and Dusparic [2024] (publication 1 as listed in Section 1.5).

3.5.1 Counterfactual Variables and Outcomes

To redefine counterfactual explanations, we build on the current definition used by methods in both supervised and reinforcement learning. Wachter et al. [2017] define counterfactuals as:

> *Score p was returned because variables V had values* $(v_1, v_2, \dots)$ *associated with them. If V instead had values* $(v_1', v_2', \dots)$ *and all other variables had remained constant, score* p' *would have been returned.*

Score in this context corresponds to the notion of explanation outcome as discussed before. In RL, current counterfactual explanations focus on interpreting a single decision in a specific state [Olson et al., 2019, Huber et al., 2023]. Variables are simply state features, and the score is an agent's choice of action in the state. This corresponds to the definition of counterfactuals used in supervised learning, where input features are replaced by state features and a model's prediction of a class label is with an action prediction. However, as discussed in Sections 3.1.3 and 3.4.1, RL offers a broader set of possible explanation components and outcomes that can be used for defining counterfactuals.

To redefine counterfactual explanations for RL, we start by redefining the concept of the counterfactual variable for RL.

Definition 3.5.1 (Counterfactual Variable in RL)**.** Given the original state x and outcome y being explained, the counterfactual variables can be either state features, the agent's goals, the agent's objectives, outside events, or expectations.

The goal of counterfactuals is to explain the outcome of an RL policy. While current work focuses on explaining only the agent's current decision, Section 3.4.1 identifies other plausible outcomes, such as the agent's plans or user-defined outcomes that can be explained. We propose a broad definition of outcome in this work, referring to any function of the state.

Definition 3.5.2 (Counterfactual Outcome in RL)**.** Given a state x the counterfactual outcome O is any function $\mathcal{F} : \mathcal{S} \to \mathbb{R}$ of the state x:

$$O(x) \coloneqq \mathcal{F}(x) \tag{3.7}$$

In this way, counterfactuals can be used to explain the agent's choice of action or plan, a reward function achieved in the state, or another user-defined outcome.

Having defined the variables and outcomes in the RL setting, we can form counterfactuals by changing variables in order to achieve the desired outcome. However, according to the definition, essentially any RL state achieving a desired outcome can be considered counterfactual. This can include uninformative or even misleading counterfactuals. For that reason, in the next section, we define the requirements for counterfactual explanations in RL that can be used to evaluate and search for informative counterfactuals.

3.5.2 Counterfactual Explanation Requirements

There are often a large number of alternative states that achieve the desired outcome. To choose the most useful one, the concept of the closest possible world is often used. However, we cannot reuse the same requirements describing the closest possible world from current work, as they do not account for the stochastic and sequential nature of RL tasks (Section 3.1). For this reason, in this section, we define five requirements that describe the closest possible world counterfactual in sequential and stochastic RL tasks:

Definition 3.5.3 (Counterfactual Requirements for RL)**.** Given a state x_n where outcome $y = O(x_n)$ is being explained, a sequence of previous k state-action pairs $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ leading to x_n, and a counterfactual outcome y', a counterfactual x' should satisfy the following requirements:

1. **Validity:** The desired outcome y' should be achieved in the counterfactual state x'. Depending on the definition of outcome, this could mean the agent choosing a target action in x', the agent achieving a higher value of the reward function, or obtaining another user-defined metric in x'.

2. **Reachability:** The counterfactual x' should be connected to the original instance through the underlying temporal connection of the RL environment using RL actions. Specifically, x' should be reachable from the original state x_n or from another state from the agent's past. In this way, a counterfactual answers the what-if question by exploring alternative past and future worlds.

3. **Recency**: Counterfactuals that are reachable in a few steps should be preferred over more distant ones.

4. **Plausibility**: Counterfactual x' should be representative of the agent's policy π that is being explained. That means it should be reachable not by just any policy, but by π.

5. **Certainty of outcome**: The counterfactual should be reachable with high certainty even in stochastic environments. This means that, given two potential counterfactuals, the one that can be reached via a more certain path should be preferred.

These requirements are inspired partly by their counterparts in supervised learning, adapted to stochastic and sequential RL tasks and partly by the counterfactual explanation research in psychology. Validity defined above, for example, corresponds to the notion of validity used in supervised learning approaches.

Reachability corresponds in part to multiple counterfactual requirements used in supervised learning, such as proximity (to ensure closeness of instances), data manifold closeness (ensuring a path between the original and counterfactual), and actionability (preventing changing of immutable features). Counterfactuals that are not reachable can be misleading, as they might represent unrealistic changes to the original instance.

Similarly, research in human reasoning uncovered that humans reason about changing the most recent events to explain outcomes [Byrne, 2019]. We apply this insight to define recency. For example, a medical AI tool explaining its decision to prescribe treatment should not tell the user that if conditions 10 years ago were different, they would have chosen a different treatment if there exists a more recent counterfactual. This requirement is similar to the proximity and sparsity requirements defined for supervised learning, as they attempt to ensure "closeness" between the counterfactual and original instance.

Plausibility, as defined above, corresponds to similar metrics such as plausibility or data manifold closeness in supervised learning, which ensure the counterfactual is representative of the dataset. Counterfactuals that are not likely under the agent's policy will not be representative of its behaviour. For example, a counterfactual showing that patients would have improved if they received a highly risky treatment is not representative of a cautious AI medical tool's behaviour and cannot be used to explain it.

In RL, the stochastic nature of the environment can make a counterfactual instance invalid during the time it takes to reach it from the original state. While validity ensures that the desired outcome is achieved in the counterfactual state, it does not measure how

likely it is to reach the counterfactual state under stochastic conditions. For example, consider an AI medical tool explaining why the patient has not been discharged after unsuccessfully applying treatment with a 10% *chance of full recovery and* 50% *chance of moderate recovery. An uncertain counterfactual would show the patient fully recovered and state that if the treatment worked, they would be discharged. A more certain counterfactual would offer the state in which the patient is moderately recovered and ready for discharge. The uncertain counterfactual might be misleading about the efficacy of the treatment. Certainty of outcome is similar to validity in supervised learning (ensuring the counterfactual outcome) or plausibility (ensuring a highly dense path between the original and counterfactual instance) in supervised learning.*

While these requirements have been redefined to fit the stochastic and sequential RL tasks, they still share the overall goals of close plausible counterfactuals that deliver the desired outcome and are compatible with similar requirements used in supervised learning. However, there are other requirements, more prominently used in supervised learning, that are either obsolete or do not translate to RL tasks. For completeness, we list them here:

1. ***Actionability**: As discussed in Section 3.1, actionability is one of the common requirements of counterfactuals in supervised learning. In RL, however, both counterfactuals that can be reached by RL actions and those that are a consequence of a change in stochastic conditions in the environment can be useful to explain the effect of stochastic events on an agent's behaviour. For example, a non-actionable counterfactual in an agricultural scenario could explain that had the temperature been higher the agent would advise less watering. While neither the RL agent nor the user has control over the temperature, this explanation can help the user understand the effect of temperature on the agent's actions and reassure them that the agent is not wasteful and knows when to save water.*

2. ***Causality**: In supervised learning, the underlying causal model is often unknown, and changing features risks making changes that do not correspond to the causal rules for the environment. For example, increasing the education level in a loan applicant's profile without adjusting their age would break the causal relationship between the two. However, this requirement is obsolete in RL if a requirement such as reachability is considered. As RL actions implicitly encode the rules of the environment, counterfactuals reachable by those actions will still follow causal rules.*

3. ***Feature-based Similarity Metrics (Proximity/Sparsity)**: While they are used in supervised learning to estimate the similarity between instances, relying on these requirements can be misleading in RL. For example, despite the small difference in state features, two states can be far apart from each other in terms of execution and vice versa.*

In this section, we defined five counterfactual requirements which can be used to evaluate counterfactual explanations in RL. Some of the requirements are translated from

Table 3.7: Comparison between the counterfactual requirements in supervised and RL. Causality is implicitly satisfied through reachability, as executing RL actions maintains the causal relationships in the environment.

Supervised Learning		Reinforcement Learning	
Counterfactual requirement	**Promotes counterfactuals that:**	**Counterfactual requirement**	**Promotes counterfactuals that:**
Validity	Achieve the desired outcome	Validity	Achieve the desired outcome
Proximity	Have similar features to x	Recency	Can be reached in few steps
Sparsity	Change few features		
Actionability	Do not change immutable features	Reachability	Can be reached using RL actions
Plausibility	Likely under the dataset	Plausibility	Likely under the agent's policy
Robustness	Maintain outcome under small changes	Certainty of Outcome	Deliver the desired outcome with high probability
Causality	Preserve causal feature dependencies	Reachability (implicitly)	Can be reached using RL actions

supervised learning, while some are RL-specific and capture the stochastic and sequential nature of RL tasks. Table 3.7 provides a summary of The counterfactual requirements of RL and their counterparts in supervised learning.

3.5.3 Counterfactual Explanations for RL

After having defined the concepts of counterfactual variable, counterfactual outcome, and the requirements for counterfactual explanations in RL, we now formalise a definition of counterfactual explanations for the RL task.

Definition 3.5.4 (Counterfactual explanations for RL)**.** Suppose a state x_n where an outcome $y = O(x_n)$ is being explained, and let the counterfactual outcome be y'. Suppose a sequence of previous k state-action pairs $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ leading to x_n is available. Given a counterfactual variable V with value v in state x_n we define two types of counterfactual explanations:

- **Backward counterfactual** x' is the *most recent plausible state* in which variable v takes an alternative value v' that delivers the counterfactual outcome y' with high certainty, such that there exists a trajectory P starting in x_{n-k} and leading to x'

$$x' = P(x_{n-k}) \tag{3.8}$$

- **Forward counterfactual** x' is the *most recent plausible state* in which variable v takes an alternative value v' that delivers the counterfactual outcome y' with high certainty such that there exists a trajectory P starting in x_n and leading to x'

$$x' = P(x_n) \tag{3.9}$$

The definition of counterfactuals relies on finding an alternative world where the counterfactual variable holds a different value. The counterfactual variable can be a state feature, the agent's goal, an objective or expectation, or an outside event. Depending on the choice of counterfactual variable, five different counterfactual explanations can be formed for any state. We summarise these in Table 3.8. In this thesis, we focus on explanations that use features as variables and agents' actions as the outcome and discuss other counterfactual explanation types as future work in Section 7.3.

Table 3.8: Different formulations of forward and backwards counterfactual explanations for different choices of counterfactual variables.

Counterfactual Variable	Backward Counterfactual Explanation	Forward Counterfactual Explanation
Feature	Had the state features taken different values (in the past), outcome y' would be achieved instead of y	If the state features were to take different values (in the future), outcome y' would be achieved instead of y
Goal	Had the agent followed a different goal (in the past), outcome y' would be achieved instead of y	If the agent were to follow a different goal (in the future), outcome y' would be achieved instead of y
Objective	Had the agent preferred a different objective (in the past), outcome y' would be achieved instead of y	If the agent were to prefer a different objective (in the future), outcome y' would be achieved instead of y
Events	Had the agent encountered a different event (in the past), outcome y' would be achieved instead of y	If the agent were to encounter a different event (in the future), outcome y' would be achieved instead of y
Expectations	Had the agent faced a different expectation (in the past), outcome y' would be achieved instead of y	If the agent were to face a different expectation (in the future), outcome y' would be achieved instead of y

We note that forward counterfactuals are sometimes referred to as prefactuals in psychology research, while backwards counterfactuals are referred to as simply counterfactuals. To prevent confusion, we rename these explanations as backward and forward counterfactuals. We make a formal distinction between backward and forward counterfactuals, as previous research in psychology found that they evoke different cognitive processes in humans when used to explain decisions of supervised learning models [Dai et al., 2022].

We also note here that we purposefully keep the definition of a counterfactual explanation in RL abstract, without explicitly formalising the concepts such as recency and plausibility at this stage. This is in line with high-level definitions provided for counterfactuals in supervised learning [Wachter et al., 2017]. In Section 4.2.2, we provide concrete metrics for evaluating these requirements for a specific experimental use case. However, we acknowledge that explanation quality is inherently subjective, and different use cases might require different metrics to evaluate these requirements. For example, when explaining a non-deterministic policy, a plausible counterfactual might be reached with high probability. For a deterministic policy, however, plausibility might be measured by how "good" actions leading to the counterfactual are according to the policy (e.g. by using Q-values). We expect this definition to provide a guideline for searching for counterfactual explanations, with low-level implementation of the requirements subject to use case and user preference.

3.6 Semifactual Explanations – Redefined

In supervised learning, semifactual explanations are often defined as the most distant neighbour of the original instance. Distance between instances is defined using feature-based similarity metrics (e.g. proximity and sparsity), while plausibility is evaluated using data density metrics. However, as shown in Section 3.1, due to the stochastic and sequential nature of RL tasks, the definitions of distance and plausibility do not translate straightforwardly from supervised to RL tasks.

In this section, we redefine semifactual explanations for RL tasks. Firstly, we start by redefining the concepts of semifactual variables and outcomes for RL (Section 3.6.1). Next, in Section 3.6.2, we propose a set of requirements for semifactuals to satisfy in

stochastic and sequential tasks, and finally, in Section 3.6.3, we propose a new definition for semifactual explanations in RL.

3.6.1 Semifactual Variables and Outcomes

To redefine semifactual explanation we build on their current definition in xAI. However, there is no formal definition of semifactuals in current work, and they are often defined informally as a special case of counterfactual explanations that do not change the outcome [Aryal and Keane, 2023]. For this reason, we utilise the same definition used for counterfactual explanations [Wachter et al., 2017], but ensure that the outcome remains the same instead of being changed:

> *Score p was returned because variables V had values* $(v_1, v_2, \dots)$ *associated with them. Even if V instead had values* $(v_1', v_2', \dots)$ *and all other variables had remained constant, the same score p would have been returned.*

Score in this context corresponds to the notion of explanation outcome as discussed before. In supervised learning, variables are simply input features, and the outcome is the prediction label. In RL, however, a broader range of variables and outcomes can be defined to explain the complex behaviour of RL agents. Firstly, we define the semifactual variable in RL as:

Definition 3.6.1 (Semifactual Variable in RL)**.** Given an original state x being explained, a semifactual x' can be obtained by changing state variable values, the agent's goals and objectives, outside events, or expectations.

Similar to counterfactual explanations (Section 3.5.3), the goal of the semifactuals is to explain an outcome. While in supervised learning semifactuals have been used to explain only model output, in RL, multiple possible outcomes can be explained (Section 3.4.1). For this reason, we implement a broad definition (same as for counterfactual outcome):

Definition 3.6.2 (Semifactual Outcome for RL)**.** Given a state x the semifactual outcome is any function $\mathcal{F} : \mathcal{S} \to \mathbb{R}$ of the state x:

$$O(s) \coloneqq \mathcal{F}(s) \tag{3.10}$$

In this way, semifactual explanations can be used to explain the agent's actions and plans, but also more complex outcomes such as the agent's rewards or user-defined outcomes.

Having defined the semifactual variables and outcomes in RL, we can generate semifactuals by changing variables while maintaining the outcome. However, this would lead to a large space of potential semifactual explanations, as each RL state with different variables but the same outcome would be considered. For this reason, the next section introduces a set of semifactual requirements which can be used to evaluate and search for informative semifactuals.

3.6.2 Semifactual Explanation Requirements

In order to define a good semifactual, we also need to define the most distant neighbour in RL. To that end, we define a set of six semifactual requirements that take into account the stochastic and sequential nature of RL tasks, and describe a most distant neighbour semifactual:

Definition 3.6.3 (Semifactual Requirements for RL)**.** Given a state x_n where outcome $y = O(x_n)$ is being explained, a set of previous k state-action pairs $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ leading to x_n a semifactual x' should satisfy the following objectives:

1. **Validity:** The semifactual state x' should deliver the semifactual outcome y.
2. **Reachability:** The semifactual x' should be connected to the original instance through the underlying temporal connections of the RL environment. Specifically, x' should be reachable from the original state x_n or from another state from the agent's past.
3. **Recency**: The semifactual x' that are reachable in a few steps should be preferred over more distant ones.
4. **Plausibility:** The semifactual x' should be representative of the agent's policy π. That means it should be reachable not by just any policy, but by π.
5. **Unexpectedness:** The semifactual x' should be a consequence of an unexpected event.
6. **Argument Strength:** To be informative, a semifactual x' explanation must offer a stronger argument for the outcome than the original state x.

The defined requirements are inspired in part by their counterparts for supervised learning, adapted to sequential and stochastic RL tasks, and in part by semifactual research in psychology. Validity as defined above, for example, corresponds to the one used for counterfactuals (Section 3.5.2) but maintains the outcome. It also corresponds to the notion of validity for semifactuals in supervised learning [Aryal and Keane, 2023].

Reachability is the same as the definition of reachability for counterfactual explanations in Section 3.5.2. Additionally, reachability is similar to the metrics of actionability or data manifold closeness, which ensures the semifactual is represented in the dataset. Semifactuals that are not reachable by using RL actions might represent states that are not representative of the RL environment.

Recency corresponds to the requirement of the same name defined for the counterfactuals in Section 3.5.2. Additionally, this requirement is similar to sparsity and proximity, which are sometimes used in supervised learning, to ensure closeness between the original and semifactual instance. Intuitively, telling a farmer that increased watering of the fields 10 *years ago would not have increased the yield is not as reassuring as indicating that increased watering last month would also not have changed the outcome.*

This definition of plausibility corresponds to the plausibility as defined for counterfactual

explanations (Section 3.5.2). Additionally, it is similar to the data manifold closeness and plausibility metrics used in supervised learning. A semifactual that is not representative of the agent's policy can be misleading. For example, consider a cautious AI tool explaining why it chose to treat a patient with a high-cost, highly efficient treatment T. A semifactual could state that even if a more risky treatment T' was applied earlier, the agent would choose T at this stage. This semifactual is not representative of the agent's policy, as it shows a state in which the cautious agent never would have ended up. While this explanation gives us an insight into the connection between different treatments and the disease, it does not explain why the agent applied treatment T. An alternative semifactual states that even if the disease progressed more quickly, the agent would still choose treatment T gives us more information about the decisions and is more representative of the agent's policy, as it could be reached by the agent's policy under different stochastic conditions.

The unexpectedness requirement stems from the work of Aryal and Keane [2023], who argue that semifactuals should be surprising in supervised learning. For example, knowing that the yield would have been the same even if the field suffered a drought might reassure the farmer in their decisions and reinforce their trust in the system. Thus, informative semifactuals ought to be exceptional.

The requirement for the strength of the argument is inspired by the works presenting semifactuals as a fortiori arguments [Nugent et al., 2009]. For example, an AI medical tool prescribing the most aggressive possible treatment can explain its decision by stating that even if the disease were twice as difficult, it would have made the same choice. This explanation, however, does not offer any new information about the agent's decision-making process, as the original decision is already extreme. An alternative semifactual stating that even if the disease was less dangerous it would still prescribe the same treatment offers a stronger argument for understanding the agent as favouring harsher treatments.

The defined requirements correspond in part to semifactual requirements developed for supervised learning but adapted for stochastic and sequential tasks. However, there are requirements often used in supervised learning that do not directly translate to RL settings. For completeness, we list these here:

1. ***Actionability**: As discussed in section 3.1, actionability does not directly translate from supervised into RL. While we require semifactuals to be reachable from the original instance, we do not make a distinction between changes that occur via the agent's actions or changes in stochastic conditions, as both of these have a role in explaining RL agents.*
2. ***Causality**: Similar to counterfactuals, semifactuals are not supposed to violate the causal constraints of the task. While in supervised learning, the causal model is often difficult to obtain, in RL, it is ingrained in the environment. As long as the semifactual search relies on RL actions, causal constraints will not be violated. For this reason, we do not include causality as a desired requirement of semifactuals*

Table 3.9: Comparison of semifactual requirements for supervised and RL..

Supervised Learning		Reinforcement Learning	
Semifactual requirement	**Promotes semifactual that:**	**Semifactual requirement**	**Promotes semifactuals that:**
Validity	Achieve the desired outcome	Validity	Achieve the desired outcome
Proximity	Have similar features to x	Recency	Can be reached in few steps
Sparsity	Change few features		
Actionability	Do not change immutable features	Reachability	Can be reached using RL actions
Plausibility	Likely under the dataset	Plausibility	Likely under the agent's policy
Surprising	Surprising/Shocking	Unexpectedness	Follow unexpected events
Convincing	Provide strong arguments for the decision	Argument strength	Provide strong arguments for the decision
Causality	Preserve causal feature dependencies	Reachability (implicitly)	Can be reached using RL actions

for RL.

3. ***Feature-based similarity metrics**: Previous works in supervised learning argue that semifactuals should be allowed to change only one feature in the original state [Kenny and Keane, 2021, Aryal and Keane, 2023]. However, following this constraint in RL would automatically reject semifactuals that are close in execution to the original instance but have changes in multiple features, some of them not necessarily important for the decision choice.*

In this section, we introduced a set of requirements for evaluating semifactuals in stochastic and sequential tasks. While some of these requirements are adapted from similar ones in supervised learning, some represent novel RL-specific requirements. In Table 3.9, we summarise the main similarities and differences between counterfactual requirements in supervised and RL. In the next section, we use the defined concepts of semifactual variables and outcomes and the semifactual requirements in RL to define semifactual explanations for RL.

3.6.3 Semifactual Explanations for RL

In previous sections, we formalised the concepts of semifactual variables and outcomes, as well as proposed a set of requirements that describe the most distant neighbour semifactuals in RL. With that in mind, we define semifactual explanations for RL tasks:

Definition 3.6.4 (Semifactual explanations for RL)**.** Suppose a state x_n where an outcome $y = O(x_n)$ is being explained. Let $(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})$ be a sequence of previous k state-action pairs leading to x_n. Given a semifactual variable V taking value v in x_n we define two types of semifactual explanations:

- **Backward semifactual** x' is the *most recent, unexpected, plausible, argument-strong state* in which variable v takes an alternative value v' that still delivers the outcome y, such that there exists a sequence of state-action pairs P starting in x_{n-k} and leading to x'

$$x' = P(x_{n-k}) \tag{3.11}$$

- **Forward semifactual** x' is the *most recent, unexpected, plausible, argument-strong state* in which variable v takes an alternative value v' that still delivers the outcome y, such that there exists a sequence of state-action pairs P starting in x_n

Table 3.10: Different formulations of forward and backward semifactual explanations for different choices of semifactual variables.

Semifactual Variable	Backward Semifactual Explanation	Forward Semifactual Explanation
Feature	Even if the state features taken different values (in the past), outcome y would still be achieved.	Even if the state features take different values (in the future), outcome y will still be achieved.
Goal	Even if the agent followed a different goal (in the past), outcome y would still be achieved.	Even if the agent follows a different goal (in the future), outcome y will still be achieved.
Objective	Even if the agent preferred a different objective (in the past), outcome y would still be achieved.	Even if the agent prefers a different objective (in the future), outcome y will still be achieved.
Events	Even if the agent encountered a different event (in the past), outcome y would still be achieved.	Even if the agent encounters a different event (in the future), outcome y will still be achieved.
Expectations	Even if the agent faced a different expectation (in the past), outcome y' would still be achieved.	Even if the agent faces a different expectation (in the future), outcome y will still be achieved.

and leading to x'

$$x' = P(x_n) \tag{3.12}$$

Intuitively, we define semifactuals as alternative states that differ in some semifactual variable V from the original state. Variable V can be either one of the five semifactual variables defined in Section 3.1.3 – state features, goals, objectives, expectations, or events. Depending on the variable, different semifactual explanations can be formed. For the summary of different semifactuals depending on the choice of semifactual variable, see Table 3.10.

The above definition is similar to that in supervised learning, as both define semifactuals as the most distant neighbour. However, the distance includes RL-specific requirements of unexpectedness, plausibility, and argument strength, which are based on the stochastic and sequential nature of RL tasks. Additionally, our definition distinguishes between forward and backward semifactuals, which search for the semifactual in the agent's future and past experiences, respectively.

3.7 Chapter Summary and Contributions

This section focuses on two research questions posited in Section 1.2. Firstly, our goal was to answer RQ1, which deals with redefining counterfactual explanations for RL

> *How can the counterfactual concept of the closest possible world be defined in stochastic and sequential environments?*

Secondly, we tackled RQ2, which aims to redefine semifactual explanations for RL:

> *How can the semifactual concept of the most distant neighbour be defined in stochastic and sequential environments?*

To answer these questions, we investigated how counterfactual and semifactual explanations can be defined in RL tasks. Firstly, we analysed the differences between supervised and RL from the perspective of counterfactual and semifactual explanations (Section 3.1). We identified 1) sequential execution, 2) stochastic conditions and 3) the broader set of explanation components in RL as the main factors that require the redefinition of counterfactuals and semifactuals for RL. In Section 3.5.3, we redefined the concepts of

counterfactual variable and outcome for RL tasks. Additionally, we provided high-level guidelines for finding the closest possible world in stochastic and sequential environments by defining five RL-specific counterfactual requirements. Section 3.6 provides the same redefinition for semifactual explanations by first redefining the semifactual variable and outcome, and providing six RL-specific semifactual requirements to serve as guidelines for choosing the most distant neighbour. We also offered a systematic taxonomy of counterfactual (Table 3.8) and semifactual explanations (Table 3.10) that can be used to guide future research in this field.

In the following chapter, we build on the above definitions and requirements and propose algorithms for generating counterfactual and semifactual explanations in stochastic and sequential environments. In Section 4.2, we implement counterfactual metrics to evaluate RL-specific counterfactual requirements defined in this chapter and propose three novel algorithms optimising these metrics and generating counterfactual explanations in RL. Similarly, in Section 4.3 we propose semifactual metrics for evaluating semifactual requirements and two novel algorithms for optimising them and generating semifactual explanations for RL tasks.

4 Generating Counterfactual and Semifactual Explanations in RL

In the previous chapter, we redefined counterfactual and semifactual explanations for RL tasks. While the previous chapter lays the theoretical groundwork for defining and evaluating Xfactuals, it does not offer a practical approach for evaluating Xfactual requirements or for searching for these explanations. In this chapter, we propose algorithms for generating Xfactual explanations for RL tasks. Firstly, in Section 4.1 we introduce the necessary prerequisites. Further, in Section 4.2, we describe RACCER (Reachable and Certain Counterfactual Explanations for Reinforcement Learning), the first approach for generating counterfactual explanations which accounts for the stochastic and sequential nature of RL tasks. In Section 4.3 we describe the SGRL (Semifactual Generation for Reinforecement Learning), the first approach for generating semifactual explanations in RL.

4.1 Preliminaries

In the previous chapter, we defined multiple possible counterfactual and semifactual formulations based on different choices of variables and outcomes (Table 3.8 and Table 3.10). In this thesis, we focus on a specific type of Xfactuals – those that change the state features to explain the action choice. In other words, we focus on Xfactuals that use state features as variables and the agent's action choice as the outcome. This type of Xfactual explanation is the only one explored both in supervised [Aryal and Keane, 2023, Guidotti, 2024] and RL [Olson et al., 2019, Huber et al., 2023] (because it can be naturally extended from supervised learning). As such it represents a suitable test bed for initial solutions for developing RL-specific Xfactual explanations. In this section, we define the necessary prerequisites for developing algorithms to search for this type of Xfactuals.

Counterfactual and semifactual explanations with other types of variables and outcomes as defined in Sections 3.5.3 and 3.6.3 are considered out of scope for this work. We discuss these alternative Xfactual explanations as future work (Section 7.3) and offer future research directions to integrate these explanation types in RL.

In this thesis, we focus on explaining the agent's action choice. To that end, we define the action choice outcome:

Definition 4.1.1 (Action Choice Outcome). Given agent's policy π, state space $\mathcal{S}$ and action space $\mathcal{A}$, we define *action choice outcome* of a state x as a function $O_A(x) : S \rightarrow \mathcal{A}$:

$$O_A(x) = \pi(x) \tag{4.1}$$

Stochasticity is one of the main characteristics of real-life RL tasks. As such, it features in both counterfactual and semifactual requirements defined in Sections 3.5.2 and 3.6.2. For this reason, we need precise definitions of stochastic processes in the environment in order to evaluate these requirements. To reason formally about stochastic processes, we start by defining stochastic context:

Definition 4.1.2 (Stochastic Context). Suppose $W^1, \dots, W^I$ are stochastic processes in the environment outside of the agent's control. Stochastic context $\mathcal{W}$ is the collection of all such processes:

$$\mathcal{W} = \bigcup W^i \tag{4.2}$$

We then define a stochastic configuration $\mathcal{W}_T$ as the stochastic context along a particular trajectory. A trajectory $T = (x_1, a_1), \dots, (x_n, a_n)$ is a sequence of state-action pairs.

Definition 4.1.3 (Stochastic Configuration). Given a trajectory $T = \{(x_1, a_1), \dots, (x_N, a_N)\}$ consisting of N state-action pairs, a *stochastic configuration* $\mathcal{W}_T$ is the instantiation of the stochastic context $\mathcal{W}$ along the trajectory T:

$$\mathcal{W}_T = \bigcup W^i_T \tag{4.3}$$

To illustrate these definitions, let us reconsider the example of an AI agricultural tool explaining its yield prediction. Factors such as daily precipitation or atmospheric temperature are among factors that cannot be influenced by the farmer and are thus elements of the stochastic context, $\mathcal{W}$. An example of a trajectory T is the past days leading to the day on which the outcome (the harvest yield) is determined. The amounts of rain and measured temperatures during these days would then be in the stochastic configuration $\mathcal{W}_T$. Determining the actual values in $\mathcal{W}_T$ allows looking into the past states in T and identifying factors that can be changed, thus enabling the generation of informative Xfactuals.

4.2 RACCER: Reachable and Certain Counterfactual Explanations for RL

In Chapter 3, we redefined counterfactual explanations for RL tasks and proposed a set of counterfactual requirements that counterfactual explanations should optimise. In this section, we describe RACCER (Reachable and Certain Counterfactual Explanations for Reinforcement Learning), our approach for generating counterfactual explanations, which optimises the RL-specific counterfactual requirements as defined in Section 3.5.2.

In the remainder of this section, we first introduce the problem statement (Section 4.2.1). To address the problem statement, we first developed and implemented a set of four counterfactual metrics that can be used to evaluate and search for counterfactuals in RL (Section 4.2.2). These metrics evaluate RL-specific counterfactual requirements that describe the closest possible world counterfactual in RL (Section 3.5.2). To search for the counterfactual explanations, we present a family of three algorithms that optimise the defined metrics. Firstly, we present RACCER-HTS, which uses a heuristic tree search to search for forward counterfactuals (Section 4.2.3). In Section 4.2.3 we present more efficient counterfactual search algorithms RACCER-Rewind and RACCER-Advance based on an evolutionary algorithm that can generate a diverse set of backwards and forward counterfactuals, respectively.

4.2.1 Problem Statement

In this thesis, we focus on counterfactual explanations, which explain an action by presenting alternative state features where an alternative action would have been chosen. Formally, given an action choice outcome O_A, RACCER addresses the following problem statement (which corresponds to the RQ3 as defined in Section 1.2):

Definition 4.2.1 (Problem Statement)**.** Consider state x where $O_A(x) = a$ and whose state features $F_1, \dots, F_M$ take values $f_1, \dots, f_M$. Given the counterfactual action a', what alternative values $f'_1, \dots, f'_M$ should state features of a counterfactual explanation x' take to achieve the counterfactual outcome $O_A(x') = a'$, and how can we search for such a counterfactual state x'?

To generate a counterfactual explanation x', RACCER requires 1) oracle access to the agent's policy π, 2) an approximation of the agent's policy in the form of a probability of an action given a state ($\mathbb{P}(a|s)$) and 3) access to the RL environment. RACCER can be applied to explain RL policies learned by both value-based and policy-based algorithms. For value-based algorithms, a Q-value function can be used to approximate $\mathbb{P}(a|s)$, while policy-based methods directly learn $\mathbb{P}(a|s)$. However, due to the third requirement, RACCER is not directly applicable to offline scenarios where full access to the environment is not available. We discuss these limitations and propose several solutions for future work (Section 7.2).

RACCER does not search for the counterfactual state x' directly in the state space, but rather indirectly by investigating and evaluating action sequences leading to counterfactual explanations. This way of counterfactual search is necessary according to the counterfactual requirements, most of which (specifically plausibility, reachability, recency and certainty of outcome) reference action sequences and require both the counterfactual and the action sequence leading to it to verify them.

Table 4.1: Counterfactual requirements and metrics utilised in RACCER and their counterparts in supervised learning (SL). Reachability is satisfied implicitly, as RACCER searches for the counterfactual by exploring the agent's execution, ensuring the original and counterfactual states are connected through the temporal connections in the environment.

Counterfactual Requirement (RL)	Counterfactual Requirement (SL)	Counterfactual Metric (RACCER)
Validity	Validity	Validity
Reachability	Actionability/Data manifold closeness	Satisfied by the counterfactual search algorithm
Recency	Proximity/Sparsity	Temporal Distance
Plausibility	Actionability/Data manifold closeness	Fidelity
Certainty of outcome	Robustness	Stochastic certainty

4.2.2 Counterfactual Metrics

In Section 3.5.2, we proposed five counterfactual requirements – validity, reachability, recency, plausibility, and certainty of outcome, which describe the closest possible world counterfactual in sequential and stochastic RL tasks. While these requirements offer a high-level road map for developing and evaluating counterfactual explanations in RL, efficient metrics that evaluate these requirements are needed to generate counterfactuals in practice.

In this section, we propose four counterfactual metrics that can be used to evaluate the counterfactual requirements defined in Section 3.5.2. These metrics can be used to search for and evaluate counterfactual explanations. However, we define some of these metrics not as a function of the counterfactual state directly, but rather of the sequence of actions connecting the original and the counterfactual state. This is because, according to the counterfactual requirement of reachability, counterfactual and original states have to be connected via RL actions. Additionally, some counterfactual requirements, specifically plausibility and certainty of outcome, depend on the action sequence leading to the counterfactual state. For this reason, RACCER searches for counterfactuals by evaluating sequences of actions leading to potential counterfactual states. When a counterfactual is found by executing the sequence of actions A, it is assigned the counterfactual metric values of A. In this way, counterfactuals can be directly compared and evaluated. The summary of counterfactual metrics and their corresponding counterfactual requirements is given in Table 4.1.

Validity

Validity is a constraint present in virtually every counterfactual approach and refers to the counterfactual state achieving the counterfactual outcome. Similarly, we identify validity as an important counterfactual requirement in Section 3.5.2. Here we define a metric of the same name, which can be used to evaluate whether a counterfactual outcome is achieved in the state. Specifically, in this work, we use the action outcome as defined in Section 4.1.

To evaluate validity, we use the same metric previously established in supervised and RL [Wachter et al., 2017, Olson et al., 2019, Huber et al., 2023]. Formally, given the state

x being explained, agent's policy π and the counterfactual action a', we define validity V of a counterfactual state x' as:

$$V(x', a') = (O_A(x') = a') = (\pi(x') = a') \tag{4.4}$$

Validity does not depend on the sequence of actions leading to the counterfactual, but rather just on the counterfactual state and action. Validity defined in this way is suitable for explaining discrete actions. In continuous action spaces, however, it is unlikely that a counterfactual could achieve the exact continuous outcome. Validity in continuous action spaces could be changed to ensure that the counterfactual belongs to a counterfactual range, rather than one specific action. In this work, we generate counterfactuals only for tasks with discrete action spaces, as this is currently the only scenario considered in supervised and RL, and we leave continuous actions for future work. We discuss the challenges of generating counterfactuals for explaining continuous outcomes in Section 7.2.

Temporal Distance

We defined recency as the requirement for the counterfactual state to be connected to the original state in as few RL actions as possible (Section 3.5.2). To evaluate recency, we propose measuring temporal distance. Consider a factual state x_n and a trajectory $\{(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})\}$ leading to x_n. For an action sequence $A = \{a'_1, \dots, a'_k\}$, where A is leading from x_{n-k} to x' in case of backward counterfactuals and from x_n to x' in case of forward counterfactuals, we define temporal distance TD as:

$$TD(x_n, A) = \begin{cases} |A| & \text{, for forward counterfactuals} \\ |a'_j, \dots, a'_k|,\ j = min\{i | a_{n-k+i} \neq a'_i\} & \text{, for backward counterfactuals} \end{cases} \tag{4.5}$$

Temporal distance is calculated in different ways depending on whether we are searching for forward or backward counterfactuals. For forward counterfactuals, we measure the temporal distance simply as the number of RL actions needed to navigate from x_n to x'. For backward counterfactuals, however, we allow the counterfactual to be reachable from any one of the previous k visited states $x_{n-k}, \dots x_{n-1}$. As such, we measure only the number of actions leading to the counterfactual x' starting in x_{n-k} that diverge from the original actions.

By minimising temporal distance, we ensure counterfactuals can be reached in as few actions as possible, either from the current state x_n (for forward counterfactuals), or from a state in its past x_{n-k} (for backward counterfactuals). Temporal distance is one of the counterfactual metrics that is defined for the sequence of actions leading to the counterfactual x' rather than the counterfactual itself.

It is important to note that, although our goal is to minimise temporal distance, we

do not want it to be zero – that is, we want there to exist a sequence of actions A of non-zero length connecting the original and counterfactual state. A temporal distance of zero would indicate that the counterfactual state is the same as the original state, which would violate the validity requirement.

Fidelity

In Section 3.5.2, we defined plausibility as an important counterfactual requirement, calling for counterfactuals to be connected to the original instance through a sequence of actions likely under the agent's policy. For a factual state x_n and sequence of state-action pairs $\{(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})\}$ leading to x_n, consider a counterfactual state x' and a sequence of actions $A = \{a'_1, \dots, a'_k\}$ connecting x_n and x' in case of forward counterfactuals or x_{n-k} and x' in case of the backward counterfactuals. To estimate the plausibility of A, we propose measuring how likely a sequence of actions A is under the agent's policy π. However, due to possible stochasticity in the environment, executing the action sequence A could generate different trajectories. To account for the stochasticity, let $\mathcal{P}$ be a set of all possible trajectories T obtained by executing actions A starting in x_n (for forward counterfactuals) or x_{n-k} (for backward counterfactuals) over all possible stochastic configurations $\mathcal{W}_T$:

$$\begin{aligned} \mathcal{P} &= \{(x^*, A)|_{\mathcal{W}_T} \quad \forall \mathcal{W}_T \in \mathcal{W}\} \\ \textit{where} \quad x^* &= \begin{cases} x_n & \textit{for forward counterfactuals} \\ x_{n-k} & \textit{for backward counterfactuals} \end{cases} \end{aligned} \tag{4.6}$$

To evaluate the likelihood of A under the policy π, we propose measuring fidelity. We define fidelity F as the average probability of the policy π choosing the action a_j in state x_j for each $(x_j, a_j) \in T$ along all trajectories $T \in \mathcal{P}$:

$$F(x_n, A) = \frac{\sum_{T \in \mathcal{P}} \sum_{x_j, a_j \in T} \mathbb{P}[\pi(x_j) = a_j]}{|\mathcal{P}|} \tag{4.7}$$

To evaluate the probability of taking an action a in the state x, we require access to the agent's approximation of the distribution $\mathbb{P}(a|x)$. For RL policies learned using a policy-based algorithm, this probability distribution is available, as it is learned during the policy learning process. To explain policies learned using the value-based algorithms, we use the agent's Q-value function:

$$\begin{aligned} \mathbb{P}[\pi(x) = a] =& \text{softmax}(Q(x, \mathcal{A}))[a] \\ =& \frac{\exp\left(Q(x, a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(Q(x, a')\right)} \end{aligned} \tag{4.8}$$

where $Q(x, a)$ is the Q-value of taking action a in state x, and $\mathcal{A}$ is the action space of the task.

By optimising fidelity, we ensure that generated counterfactuals are representative of the agent's behaviour. The same definition of fidelity is used in both the forward and backward search. The only difference is the starting state of the action sequence A – for backward search fidelity is evaluated over trajectories $\mathcal{P}$ connecting x_{n-k} and x', and in forward search it is evaluated over trajectories $\mathcal{P}$ connecting x_n to x'.

Stochastic Certainty

The certainty of outcome is defined in Section 3.5.2 as an important requirement ensuring that the counterfactuals are reachable with high certainty even in stochastic environments. To ensure that users are presented with counterfactuals that are likely to produce the counterfactual output, we propose optimising stochastic certainty. For a factual state x_n and sequence of state-action pairs $\{(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})\}$ leading to x_n, consider a sequence of actions $A = \{a'_1, \dots, a'_k\}$ leading to a counterfactual x' from x_n (for forward counterfactuals) or x_{n-k} (for backward counterfactuals). To estimate the stochastic certainty of A, we measure the probability that a state obtained after executing the actions A eliciting the counterfactual action a' under all possible stochastic configurations $\mathcal{W}_\mathcal{T}$:

$$
\begin{aligned}
&S_t(x_n, A, a') = \mathbb{P}\left[\pi(x') = a' \mid x' = \textit{execute}(x^*, A, \mathcal{W}_T),\ \mathcal{W}_T \in \mathcal{W}\right] \\
&\textit{where} \quad x^* = \begin{cases} x_n & \textit{for forward counterfactuals} \\ x_{n-k} & \textit{for backward counterfactuals} \end{cases}
\end{aligned} \tag{4.9}
$$

where $\textit{execute}(x^*, A, \mathcal{W}_T)$ returns the state reached by executing action sequence A in state x^* under the stochastic configuration $\mathcal{W}_T$. By maximising stochastic certainty, we ensure the counterfactual x' can be reached with high certainty even in stochastic environments. The same definition of stochastic certainty is used for evaluating both the forward and backward counterfactuals, and the only difference is the starting state.

4.2.3 Counterfactual Search

Counterfactual metrics defined in Section 4.2.2 provide a blueprint for finding a counterfactual explanation in RL. One of the main characteristics of counterfactuals in RL is that they have to be reachable from the original instance by RL actions (either by exploring the future or the past). For that reason, to find a suitable counterfactual, we need to explore the space around the original instance using RL actions. We limit the search to k actions. We refer to this parameter as horizon and note that the horizon depends on the task and outcome that is being explained. For example, to explain a car crash, we might focus only on a small number of previous steps, while to explain a treatment decision of a medical AI tool, we might need to look back months or even

years.

Given a state x_n and a sequence of state-action pairs $\{(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})\}$ leading to x_n we search for a counterfactual state x' either by exploring agent's past or future interactions. In other words, our goal is to find x' by generating a trajectory $T = \{(x_i, a_i)\}|_{i \in 1,\dots,k}$ which leads from x_n to x' for forward and from x_{n-k} to x' in backward counterfactuals.

A brute-force approach to searching the space around an RL state would involve unrolling all possible action sequences A of length k starting in x_n for forward and x_{n-k} for backward counterfactuals. Either a breadth-first or a depth-first exhaustive search could be used. The process would continue until a suitable counterfactual is found or the entire state space reachable by k actions from the starting state is explored. Naturally, such an approach is extremely computationally inefficient. The time complexity is additionally exacerbated by the stochasticity in the environment, as taking an action can lead to not just one, but multiple new states, depending on the stochastic processes in the environment.

For that reason, in this section, we explore heuristic approaches for searching the space around the original instance. Firstly, we implement a heuristic tree search algorithm, RACCER-HTS, that generates forward counterfactuals (Section 4.2.3). However, RACCER-HTS exhibited a number of drawbacks, such as poor scalability to complex environments, the ability to generate only forward counterfactuals and a limitation to only one counterfactual per factual query. For this reason, in Section 4.2.3 we implement two counterfactual search algorithms, RACCER-Advance and RACCER-Rewind, which generate diverse forward and backward counterfactuals by utilising a multi-objective search using an evolutionary algorithm.

Heuristic Tree Search

The goal of counterfactual search is to investigate the state space around the original instance. As the first approach, we propose a heuristic tree search algorithm, RACCER-HTS, that searches for forward counterfactuals that can be reached from the original state via RL actions. RACCER-HTS relies on the counterfactual metrics defined in Section 4.2.2 as a heuristic to decide which trajectories in the environment to explore. Specifically, RACCER-HTS aims to find a counterfactual state x' that minimises a weighted loss function of counterfactual metrics:

$$\begin{gathered} x' = \arg\min_{A}(L(x_n, A, a')) \quad s.t. \quad V(x', a') = 1 \quad where \\ L(x_n, A, a') = \alpha TD(x_n, A) + \beta(1 - F(x_n, A)) + \gamma(1 - S_t(x_n, A, a')) \end{gathered} \tag{4.10}$$

where α, β and γ are parameters determining the importance of different counterfactual metrics. By minimising L, RACCER-HTS aims to find counterfactuals that can be reached in a few actions, are representative of the agent's policy and are achieved with

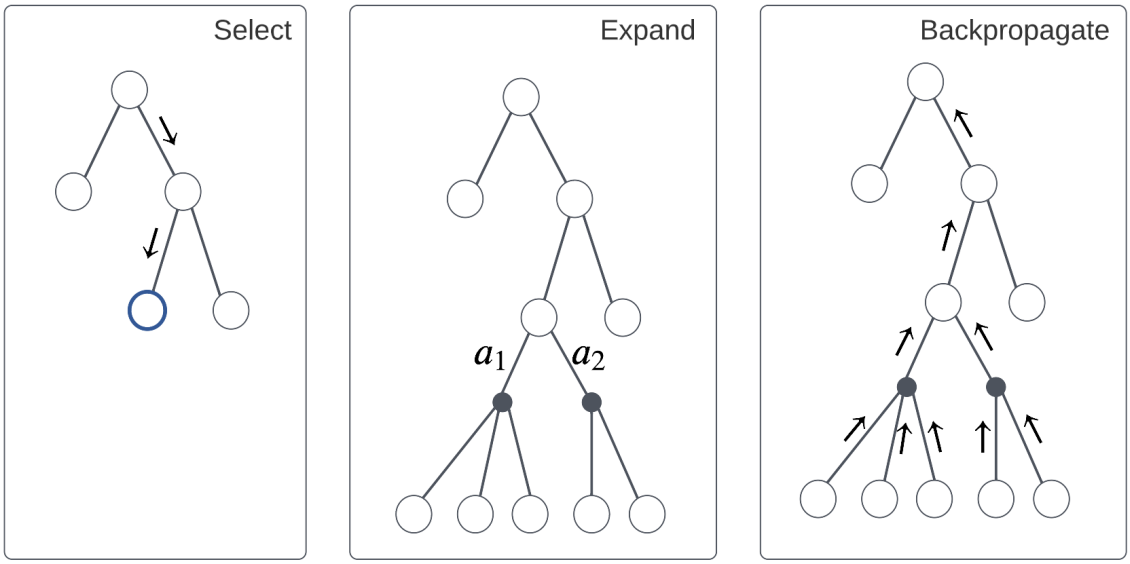


Figure 4.1: Heuristic tree search: In each iteration, a node is selected by navigating the tree from the root to a leaf by choosing actions according to the UCT formula. The node is expanded by performing all possible actions and appending all obtained states as children of the node. Finally, newly generated nodes are evaluated and their values are propagated back to the root to update the values of parent nodes. The white nodes represent states, while black nodes are determination nodes that serve to instantiate all possible child states of a node in a stochastic environment.

high certainty even in highly stochastic environments. Notably, validity is considered a constraint in this problem,m and only counterfactuals that satisfy validity are considered as explanations.

RACCER-HTS iteratively builds a tree to represent the agent's execution – each node corresponds to a state, and each edge to one action. Children of a node are obtained by taking a specific action in that node. To account for the stochasticity in the environment, we apply determinization to the expanding process by adding hidden determinization nodes each time an action is performed. The children of determinization nodes are sampled from the possible states that result from performing a specific action. The iterative tree-building process is visualised in Figure 4.1.

Each node n is also associated with a value $val(n)$, and each edge is assigned a value $Q(n, a)$. These values are based on the loss function L and are used to determine which node should be expanded in the next iteration. To calculate $val(n)$, we compute the value of $L(x_n, A, a')$, where A is the sequence of actions that navigates from root x_n to node n in the tree. $Q(n, a)$ is calculated for each node n and action a as the average of values val of the children nodes obtained when performing a in n.

To estimate $L(x_n, A, a')$, we need to calculate the values of individual counterfactual metrics of temporal distance, fidelity, and stochastic certainty for nodes in the tree. We calculate the temporal distance of node n as the length of the trajectory (in terms of edges) between the root node and n. To evaluate fidelity and stochastic certainty for a node n, we start by identifying the action sequence A leading from the root node to n. Then, we run N_{sim} simulations by executing actions A starting in x_n and use the

Algorithm 2 Counterfactual heuristic tree search used by RACCER-HTS

1: **Input**: state x, counterfactual action a'
2: **Parameters**: number of iterations N_{iter}, number of simulations N_{sim}
3: **Output**: counterfactual state x'
4: $t = \{x\}$ {Initializing search tree}
5: $i = 0$
6: **while** $i < N_{iter}$ **do**
7: n = select(t) {Select state n to be expanded}
8: S_e = expand(n) {Expand n by performing available actions}
9: **for all** $s \in S_e$ **do**
10: $val(s) = L(x, A, a')$ {Evaluate states in S_e according to L}
11: $t+ = s$
12: **end for**
13: backpropagate() {Propagate values back to the root}
14: $i+ = 1$
15: **end while**
16: $p = []$ {Initializing set of potential counterfactuals}
17: **for all** $s \in t$ **do**
18: **if** $valid(s)$ **then**
19: $p+ = s$ {Filter valid counterfactuals}
20: **end if**
21: **end for**
22: $x' = \arg\min_{cf \in p} val(cf)$ {Select the best counterfactual}

obtained trajectories to approximate a set of all possible stochastic configurations $\mathcal{W}$. Fidelity is then calculated using the agent's Q-value function according to Equation 4.7, and stochastic certainty follows Equation 4.9. Additionally, we normalise the values for temporal distance so that they fall within the $[0, 1]$ *range, while fidelity and stochastic certainty values naturally belong to that range. We can then evaluate a node in a tree by combining and weighting the three counterfactual metrics to obtain $L(x_n, A, a')$ as shown in Equation 4.10.*

The overview of the RACCER-HTS is given in Algorithm 2. At the start of the search, a tree is constructed with only the root node corresponding to the state x_n that is being explained (Algorithm 2, line 4). At each step of the algorithm, a node in the tree is chosen (Algorithm 2, line 7) and the tree is expanded with the node's children (Algorithm 2, line 8). All actions are expanded simultaneously in the node. The resulting child nodes are then evaluated against L (Algorithm 2, lines 9 -12), and the results are propagated back to the tree root to update the value of nodes and edges (Algorithm 2, line 13). To decide which node is expanded in each iteration, we navigate the tree from the root, at each node n taking the action decided by the Upper Confidence Bound applied for Trees (UCT) formula [Kocsis and Szepesvári, 2006]:

$$a^* = \arg\max_{a \in A} \left\{ Q(a, n) + C\sqrt{\frac{\ln(N(n))}{N(s, a)}} \right\} \tag{4.11}$$

where C is the exploration constant, $N(n)$ number of times n was visited and $N(n, a)$

number of times a was chosen in n. UCT balances between following the trajectories of high value and exploring under-represented trajectories through the exploration constant C. The process is repeated until a predetermined maximum number of iterations N_{iter} is reached.

Once the tree is fully grown, all nodes are first filtered according to the validity constraint (Equation 4.4) to filter out the states that do not satisfy this constraint (Algorithm 2, lines 17 - 20). The remaining nodes are potential counterfactual explanations. Then, all nodes are evaluated against L. The state corresponding to the node in the tree with the minimum value for L is presented to the user as the best counterfactual (Algorithm 2, line 22).

RACCER-HTS explores the space around the original instance in a heuristic way, promoting sequences of actions with better counterfactual metric scores. It has the benefit of quickly generating counterfactuals if they are connected to the original state in only a few actions, as it iteratively grows the search tree. However, RACCER-HTS is limited to only looking for forward counterfactuals. Additionally, this approach uses a conflated loss function consisting of a weighted average of all counterfactual metrics. In this way, only one counterfactual can be generated per query, and the weights have to be manually adjusted according to different preferences. To address these limitations of RACCER-HTS, we implement a multi-objective search using evolutionary algorithms in the next section.

Multi-objective Search (NSGA-II)

When implementing and evaluating RACCER-HTS, we came across three main drawbacks of the approach: 1) inefficiency, 2) limited search space, and 3) a weighted loss function, which requires manual weight adjustment and limits the algorithm to produce only one counterfactual per factual query. Firstly, RACCER-HTS involves searching through a tree of an agent's execution tree. In environments with large state spaces and high stochasticity, the search space becomes prohibitively large. Secondly, RACCER-HTS is equipped to only search for forward counterfactuals that can be reached from the original state; it cannot be utilised to find backward counterfactuals that would have occurred if the conditions of the agent's past experiences were different. Finally, RACCER-HTS uses a weighted loss function, which introduces two issues. Firstly, it is not clear how the weights for this loss function should be chosen, as different users and domains might have differing preferences about counterfactual metrics. Secondly, using a weighted loss function limits RACCER-HTS to finding just one counterfactual. However, ideally, a set of diverse counterfactuals should be generated to apply to users of different preferences [Mothilal et al., 2020, Laugel et al., 2023].

To overcome the limitations of the RACCER-HTS identified above, we implement two novel counterfactual search algorithms based on evolutionary algorithms – RACCER-Rewind and RACCER-Advance. While RACCER-Advance searches for forward counterfactual states that can be reached from the original state in k actions, RACCER-Rewind

searches for backward counterfactuals that arise from the agent choosing different actions or experiencing different stochastic conditions in the previous k actions.

Prior work often utilised evolutionary algorithms for counterfactual search [Dandl et al., 2020, Sharma et al., 2019, Guidotti et al., 2019], in part due to their suitability to discrete search spaces [Verma et al., 2024] and flexibility, as they make no assumptions about the underlying search space. Unlike tree search algorithms, evolutionary algorithms enable non-sequential search of the agent's execution space, allowing us to explore promising parts of the search space without evaluating all states on the way.

RACCER-Rewind and RACCER-Advance utilise an evolutionary algorithm, NSGA-II, to generate and evaluate action sequences that might lead to the optimal counterfactual state. NSGA-II (Nondominated Sorting Genetic Algorithm II) [Deb et al., 2002] is an evolutionary algorithm that uses mutation, crossover, and combination to modify a set of solutions. Only the best solutions according to a fitness function are selected and propagated to future generations. NSGA-II provides a multi-objective optimisation and can simultaneously optimise different conflicting objectives. In this way, NSGA-II can generate a set of diverse solutions optimising different objectives. For both RACCER-Rewind and RACCER-Advance, we use NSGA-II with three metrics – temporal distance, fidelity, and stochastic certainty – as optimisation objectives and the validity metric as a constraint.

An overview of the NSGA-II algorithm as used by RACCER-Advance and RACCER-Rewind is given in Algorithm 3. Through G generations, NSGA-II maintains a population $P_g, g \in 1, \dots, G$ of N solutions and uses selection, combination, and mutation to identify sequences of actions leading to the most promising counterfactual states. For a parent population P_g, in each iteration $g \in 1, \dots, G$, a child population Q_g is generated through selection, combination and mutation processes (Algorithm 3, line 9). Child and parent populations are joined together $R_g = P_g + Q_g$ (Algorithm 3, line 10).

Each action sequence $A_g^i \in R_g$ is then evaluated according to the three counterfactual metrics – temporal distance, fidelity, and stochastic certainty. Temporal distance is calculated according to Equation 4.5, the number of actions in A_g^i. To approximate the set of all stochastic configurations needed to evaluate fidelity and stochastic certainty, we run N_{sim} simulations by executing A_g^i in x_n (for RACCER-Advance) or x_{n-k} (for RACCER-Rewind) in the same way as for RACCER-HTS (Section 4.2.3) (Algorithm 3, Lines 12-16). Fidelity is then evaluated according to Equation 4.7, and stochastic certainty using Equation 4.9. Additionally, the value for the temporal distance metric is normalised to the [0,1] range, while fidelity and stochastic certainty metrics already belong to this range. Additionally, while running the N_{sim} simulations, we also collect all states reached by this rollout as a set CF of potential counterfactual states (Algorithm 3, line 14). Only states that satisfy validity are recorded at this step. Each counterfactual state is assigned the values of the counterfactual metrics evaluated for the sequence of actions leading to it. The evaluations of individual counterfactual metrics assigned to each action sequence are used by the NSGA-II to further guide the selection process.

Algorithm 3 The main loop of the NSGA-II [Deb et al., 2002] adapted for the RACCER-Advance and RACCER-Rewind algorithms

1: **Parameters:** Horizon k; Population size N; Number of generations G; Number of simulations N_{sim}
2: **Input:** Counterfactual action a'; Start state x {x_n for forward; x_{n-k} for backward CFs)}
3: **Constraints:** $\mathcal{C} = V(x')$
4: **Objectives:** $\mathcal{O} = \{TD(x, A), 1 - S_t(x, A, a'), 1 - F(x, A)\}$
5: **Output:** Set of counterfactual explanations CF
6: $g = 0$; P_g =init_population() {Initialize population}
7: CF = {}
8: **while** $g < G$ **do**
9: Q_g = generate_child_population(P_g) {Using selection, combination, and mutation}
10: $R_g = P_g + Q_g$ {Combine the parent and child populations}
11: $j = 0$
12: **while** $j < N_{sim}$ **do**
13: CF_g = unroll_and_validate(s, P_g, V) {Collect valid states by unrolling P_i}
14: $CF = CF \cup CF_g$
15: $j+ = 1$
16: **end while**
17: $(F_0, F_1, \ldots, F_t)$ = nondominated_sorting($R_g, \mathcal{C}, \mathcal{O}$) {Get non-dominated solutions}
18: $l = 0$; $P_{g+1} = \{\}$
19: **while** $|P_{g+1}| + |F_l| < N$ **do**
20: $P_{g+1} = P_{g+1} \cup F_l$ {Add solutions ordered descendingly into P_{g+1}}
21: **end while**
22: **if** $|P_{g+1}| < N$ **then**
23: C_{dist} = crowding_distance(F_l, C_{dist}) {Calculate the distance for the latest n.d. set}
24: F'_l = sort_descending(F_l, C_{dist}) {Sort based on the crowding distance}
25: **end if**
26: **for all** $f \in F'_l$ **do**
27: $P_{g+1} = P_{g+1} \cup f$ {Add the best solution from the latest front into the population}
28: **if** $|P_{g+1}| \geq N$ **then**
29: **break**
30: **end if**
31: **end for**
32: $g+ = 1$
33: **end while**
34: CF = select_pareto_front($CF, \mathcal{O}$) {Select Pareto front of cfs according to objectives}

Each state in CF is assigned the same values as the action sequence A^i_g leading to it. A non-dominated sorting algorithm is then used to split the population R_g into fronts $F_0, \ldots, F_t$, where F_0 contains the best solutions (Algorithm 3, line 17). A new parent population P_{g+1} is then generated by adding solutions from the best fronts to P_{g+1} until population size N is reached (Algorithm 3, lines 19-21). If F_{l-1} is the last full front that could be inserted into P_{g+1}, then the crowding distance is calculated for instances in the following front F_l (Algorithm 3, line 23). F_l is sorted decreasingly according to the

crowding function (Algorithm 3, line 24), and the best solutions are appended to P_{g+1} until the population size N is reached (Algorithm 3, lines 26-31). In this way, NSGA-II ensures that, in the case of solutions with similar fitness function values, those from the less populated part of the solution space are preferred. This leads to diversification of the solution set. After G generations, a Pareto front of all potential counterfactual states is selected from CF, and these are presented as the results (Algorithm 3, line 34).

Formally, to explain the agent's choice of action a over the counterfactual action a′ in a state x_n, RACCER-Advance searches for a forward counterfactual x′ which satisfies:

$$\underset{x'=A(x_n)}{\arg\min}\left[TD(x_n, A), 1 - F(x_n, A), 1 - S_t(x_n, A, a')\right] \quad s.t. \quad V(x', a') = 1$$

Conversely, RACCER-Rewind searches for backward counterfactuals in the agent's past. Formally, RACCER-Rewind optimises the following when explaining a choice of action a over a counterfactual action a′ in state x_n:

$$\underset{x'=A(x_{n-k})}{\arg\min}\left[TD(x_n, A), 1 - F(x_n, A), 1 - St(x_n, A, a'),\right] \quad s.t. \quad V(x', a') = 1$$

By utilising multi-objective search using NSGA-II, RACCER overcomes the three drawbacks of the HTS algorithm. Firstly, the evolutionary algorithm does not require building and navigating a tree structure. We show empirically that this increases the scalability of the approach in Section 5.2. Additionally, RACCER-Rewind and RACCER-Advance can search for backwards and forward counterfactuals, respectively. Finally, RACCER-Advance and RACCER-Rewind enable multi-objective optimisation and can generate a set of diverse counterfactual solutions without requiring hyperparameter optimisation for the objective weights. We show empirically in Section 5.2.4 that RACCER-Rewind and RACCER-Advance generate sets of multiple diverse counterfactual explanations.

We note here that while NSGA-II enables multi-objective optimisation with diverse solutions, it also comes with several drawbacks. Firstly, NSGA-II relies on a series of parameters such as population size or the horizon k, which have to be carefully tuned for the specific use case. Additionally, the computational complexity due to fitness function approximation can make the algorithm computationally expensive. In section 7.3, we discuss possible additions to the algorithm that could reduce the search time.

An overview of the characteristics of RACCER-HTS, RACCER-Advance, and RACCER-Rewind is shown in Table 4.2. Full evaluation of RACCER-HTS, RACCER-Rewind, and RACCER-Advance is presented in Section 5.2.

Table 4.2: An overview of RACCER-HTS, RACCER-Rewind, and RACCER-Advance algorithms for generating counterfactual explanations in RL tasks and their characteristics.

Algorithm	RACCER-HTS	RACCER-Rewind	RACCER-Advance
Backward search	✗	✓	✗
Forward search	✓	✗	✓
Diverse explanations	✗	✓	✓
Optimization	Single objective	Multi-objective	Multi-objective
Search algorithm	Heuristic tree search	NSGA-II	NSGA-II

4.3 SGRL: Semifactual Generation for RL

In Chapter 3, we redefined semifactual explanations for RL tasks and proposed a set of semifactual requirements describing the most distant neighbour in stochastic and sequential RL tasks. While Section 3.6 provides a theoretical blueprint for defining and evaluating semifactual explanations in RL, it does not offer a practical approach for generating these explanations. In this chapter, we implement SGRL (Semifactual Generation for Reinforcement Learning) – the first approach for generating semifactual explanations for RL.

In the remainder of this section, we describe the implementation of SGRL. In Section 4.3.1, we describe the problem statement. To address the problem statement, we first design and implement a set of five semifactual metrics that can be used to evaluate and search for semifactuals in RL (Section 4.3.2). These metrics evaluate the RL-specific semifactual requirements as defined in Section 3.5.2. In Section 4.3.3, we implement two search algorithms, SGRL-Advance and SGRL-Rewin, that generate semifactuals by optimising the semifactual metrics.

4.3.1 Problem Statement

In this work, we focus on semifactuals that use state features to explain an agent's action choices. Specifically, given a factual state x, where the agent chooses action a, our goal is to answer the question "Why did the agent not choose an alternative action a'?". We refer to the alternative action a' as the semifactual action in the remainder of this chapter. To answer this question in the "even if" format, our goal is to generate a semifactual state x' in which the agent still does not choose the action a', thus reaffirming its decision in the factual state.

Formally, given an action choice outcome O_A, SGRL addresses the following problem statement (which corresponds to RQ4 as defined in Section 1.2):

Definition 4.3.1 (Problem Statement)**.** Consider state x where $O_A(x) = a$ and whose state features $F_1, \dots, F_M$ take values $f_1, \dots, f_M$, and a semifactual action a'. What alternative values $f'_1, \dots, f'_M$ should state features of a semifactual explanation x' take such

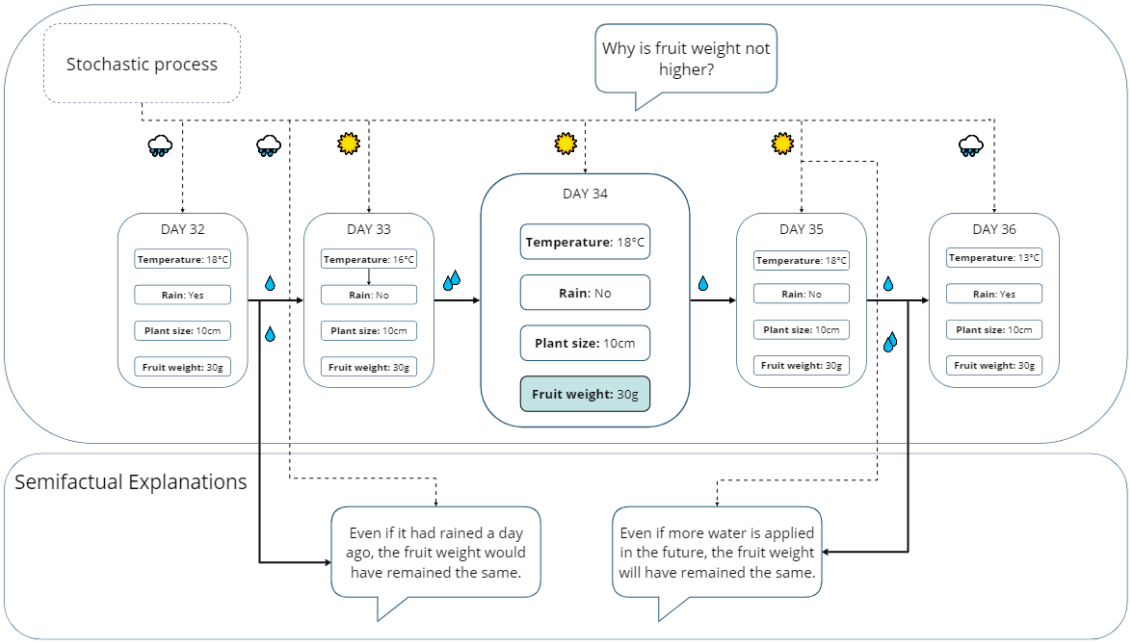


Figure 4.2: Forward and backward semifactuals: In an agricultural task, semifactuals can be used to explain why fruit weight at a specific time was not higher. The backward semifactual looks into the past and reports that more rain would not have changed the fruit weight. Conversely, the forward semifactual looks into the future and states that more watering in the future is not going to change the outcome. Dashed lines represent the effect of stochastic processes on the environment's state. In this case, a backward semifactual relies on a non-actionable change of weather, while the forward one explains the outcome through an actionable change.

that the semifactual action is still not chosen in x' ($O_A(x') \neq a'$), and how can we search for such a counterfactual state x'?

To generate a semifactual explanation x', SGRL requires 1) oracle access to the agent's policy π, 2) an approximation of the agent's policy in the form of a probability of an action given a state ($\mathbb{P}(a|s)$) and 3) access to the RL environment. These requirements are the same as defined for the RACCER algorithm (Section 4.2.1). Similar to RACCER, SGRL can be applied to explain RL policies learned by both value-based and policy-based algorithms. For value-based algorithms, a Q-value function can be used to approximate $\mathbb{P}(a|s)$, while policy-based methods directly learn $\mathbb{P}(a|s)$. SGRL is not directly applicable to offline scenarios where full access to the environment is not available. We discuss these limitations and propose several solutions in Section 7.2.

SGRL does not search for the semifactual state x' directly in the state space, but rather indirectly by investigating and evaluating action sequences leading to semifactual explanations. This way of semifactual search is necessary according to the semifactual requirements, most of which (specifically plausibility, recency, reachability and unexpectedness) reference action sequences and require both the semifactual and the action sequence leading to it to verify them.

Additionally, we note here that an alternative formulation is possible for semifactuals. Namely, given the factual state x where $\pi(x) = a$, a semifactual could be searched for as a state x' where $\pi(x') = a$. Such a semifactual reaffirms the choice of action a in x.

Table 4.3: Semifactual requirements and metrics utilised in SGRL and their counterparts in supervised learning (SL).

Semifactual Requirements (RL)	Semifactual Requirements (SL)	Semifactual Metric (SGRL)
Validity	Validity	Validity
Reachability	Actionability/Data manifold closeness	Satisfied by definition
Recency	Sparsity	Temporal distance
Plausibility	Actionability/Data manifold closeness	Fidelity
Argument strength	Robustness	Stochastic uncertainty
Unexpectedness	–	Exceptionality

In this way, however, the semifactual can only answer the "why" question, but not "why not?". For this reason, we utilise the former formulation for semifactuals and use them not to reaffirm a choice of particular action, but to reaffirm the choice of not executing an action of interest.

4.3.2 Semifactual metrics for RL

In Section 3.6.2 we defined six RL-specific semifactual requirements – validity, reachability, recency, plausibility, unexpectedness and argument strength. These requirements provide a blueprint for evaluating and selecting useful semifactuals in stochastic and sequential environments. However, we require efficient metrics for evaluating these metrics in real-life tasks to generate semifactuals for RL. In this section, we introduce metrics that can be used to evaluate these requirements for semifactual explanations. Same as for counterfactual metrics defined in Section 4.2.2, some semifactual metrics defined in this section are defined with respect to the action sequence leading to the semifactual, rather than as a function of the semifactual itself. This is because some semifactual requirements, such as reachability or plausibility, require a sequence of actions leading to the semifactual to be evaluated. An overview of these metrics and their corresponding semifactual requirements is shown in Table 4.3.

Validity

In Section 3.6.2, we defined validity as an important requirement for semifactual explanations, ensuring that the semifactual explanations reinforce the outcome. Similarly, validity is an essential requirement for all semifactual generation approaches in supervised learning [Aryal and Keane, 2023]. Here we define a metric of the same name, which can be used to evaluate if the semifactual outcome is achieved in a state. Specifically, we use the action outcome as defined in Section 4.1. Formally, given a state x where the outcome $O_A(x) = a$, and a semifactual action $a' \neq a$, we define the validity of a semifactual x' as:

$$V(x') = (O_A(x') \neq a') = (\pi(x') \neq a') \tag{4.12}$$

This metric of validity of semifactuals is analogous to the validity of counterfactuals

implemented for RACCER (Section 4.2.2). Similarly to RACCER, this definition of validity is suitable for explaining discrete actions. In continuous action spaces, however, it could be difficult to find a semifactual which reinforces an exact continuous action, and validity might need to be redefined to refer to a range rather than one action. In this work, we generate semifactuals only for tasks with discrete action spaces, as this is the only scenario considered by the current state-of-the-art semifactual generation approaches in supervised learning. We leave the topic of explaining continuous actions using semifactuals for future work and discuss the challenges of this task in Section 7.2.

Temporal Distance

In Section 3.6.2, we defined recency as the requirement for the semifactual state to be reached in as few actions as possible from the original instance in terms of RL actions. To evaluate recency in semifactuals, we propose measuring temporal distance. Consider a factual state x_n and a trajectory $\{(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})\}$ leading to x_n. For an action sequence $A = \{a'_1, \dots, a'_k\}$, where A is leading from x_{n-k} to x' in case of backward semifactuals and from x_n to x' in case of forward semifactuals, we define temporal distance TD as:

$$TD(x_n, A) = \begin{cases} |A| & \text{, for forward counterfactuals} \\ |a'_j, \dots, a'_k|,\ j = min\{i | a_{n-k+i} \neq a'_i\} & \text{, for backward counterfactuals} \end{cases} \tag{4.13}$$

Temporal distance defined in this way is equivalent to the counterfactual metric of the same name defined for RACCER in Section 4.2.2. By minimising temporal distance, we ensure counterfactuals can be reached in as few actions as possible, either from the current state x_n (for forward semifactuals), or from a state in its past x_{n-k} (for backward semifactuals).

Similar to the temporal distance defined for counterfactuals, it is important to note that we do not want the temporal distance to be zero – that is, we want there to exist a sequence of actions A of non-zero length connecting the original and semifactual state. A temporal distance of zero would indicate that the semifactual state is the same as the factual one, and as such would be uninformative.

Fidelity

Plausibility is a semifactual requirement that ensures the semifactual is representative of the agent's policy (Section 3.6.2). To estimate the plausibility of an instance, we rely on the fidelity, as defined for counterfactuals in Section 4.2.2. For a fact state x_n and a trajectory $\{(x_{n-k}, a_k), \dots, (x_{n-1}, a_{n-1})\}$ leading to x_n, consider a semifactual state x' and a sequence of actions $A = \{a'_1, \dots, a'_k\}$ connecting x and x' (for forward semifactuals) or x_{n-k} (for backward semifactuals). Due to possible stochasticity in the environment,

taking the same actions might have different outcomes. Let $\mathcal{P}$ be a set of all possible trajectories T obtained by executing actions A starting in x_n (for forward semifactuals) and x_{n-k} (for backward search) over all possible stochastic configurations $\mathcal{W}_T$:

$$\begin{aligned} \mathcal{P} &= \{(x^*, A)|_{W_T} \quad \forall W_T \in \mathcal{W}\} \\ \textit{where} \quad x^* &= \begin{cases} x_n & \textit{for forward search} \\ x_{n-k} & \textit{for backward search} \end{cases} \end{aligned} \tag{4.14}$$

Fidelity (as defined for RACCER in Section 4.2.2) is the probability of the agent's policy π choosing the actions from A starting in x_n (for forward semifactuals) or x_{n-k} (for backward semifactuals) along all stochastic configurations $\mathcal{W}_T$:

$$F(x_n, A) = \frac{\sum_{T \in \mathcal{P}} \sum_{x_j, a_j \in T} \mathbb{P}[\pi(x_j) = a_j]}{|\mathcal{P}|} \tag{4.15}$$

To evaluate the probability of taking an action a in the state x, we require access to the agent's approximation of the distribution $\mathbb{P}(a|x)$. For RL policies learned using a policy-based algorithm, this probability distribution is available, as it is learned during the policy learning process. To explain policies learned using the value-based algorithms, we use the agent's Q-value function:

$$\begin{aligned} \mathbb{P}[\pi(x) = a] =& \text{softmax}(Q(x, \mathcal{A}))[a] \\ =& \frac{\exp\left(Q(x, a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(Q(x, a')\right)} \end{aligned} \tag{4.16}$$

where $Q(x, a)$ is the Q-value of taking action a in state x, and $\mathcal{A}$ is the action space of the task.

By optimising fidelity, we ensure that generated semifactuals are representative of the agent's behaviour. The same definition for fidelity is used in the search for both the forward and backward semifactuals. The only difference is the starting state of the action sequence A – for backward search fidelity is evaluated over trajectories $\mathcal{P}$ connecting x_{n-k} and x', and in forward search it is evaluated over trajectories $\mathcal{P}$ connecting x_n to x'.

Stochastic Uncertainty

To be informative, a semifactual explanation must offer a stronger argument than the state in question (Section 3.6.2). To choose semifactuals that offer a strong argument, we propose prioritising semifactuals that are located in the part of the search space where validity is unlikely to be satisfied. To estimate the argument strength of a sequence of actions A leading to a semifactual x', we measure the probability of achieving validity after executing the sequence of actions A leading to x' from x_n (for forward counterfactu-

als) or x_{n-k} (for backward counterfactuals). Due to the stochasticity in the environment, executing the sequence of actions A will not always generate the same trajectory. To account for stochasticity, we propose measuring stochastic uncertainty, which estimates the probability of a valid outcome after executing action sequence A under all possible stochastic configurations $\mathcal{W}_T$. Given the factual state x_n, a sequence of actions A and the semifactual action a' we define stochastic uncertainty as:

$$S_t(x_n, A, a') = \mathbb{P}\left[\pi(x') == a' \mid x' = execute(x^*, A, \mathcal{W}_T),\ \mathcal{W}_T \in \mathcal{W}\right]$$
$$where \quad x^* = \begin{cases} x_n & for\ forward\ counterfactuals \\ x_{n-k} & for\ backward\ counterfactuals \end{cases} \tag{4.17}$$

where $execute(x^*, A, \mathcal{W}_T)$ calculates a state reached by executing action sequence A in state x^* under the stochastic configuration $\mathcal{W}_T$. By maximising stochastic uncertainty, we ensure that a semifactual is found in the part of the state space where achieving validity is unlikely, thus showing a strong argument for the decision. The same definition of stochastic uncertainty is used for evaluating both the forward and backward counterfactuals, and the only difference is the starting state.

We call this metric stochastic uncertainty, in line with the stochastic certainty property defined for counterfactual evaluation by RACCER (Section 4.2.2). The difference between the two is that the stochastic certainty complements the counterfactual validity property, as both aim to ensure the counterfactual action a' is chosen in a state. On the other hand, the stochastic uncertainty property conflicts with the semifactual validity property, as the validity ensures the semifactual action is not chosen in the semifactual state, and stochastic uncertainty aims to maximise the probability of choosing the semifactual action.

Exceptionality

We defined the unexpectedness of semifactual explanations in RL as a requirement in Section 3.6.2, ensuring that the semifactuals are surprising. To evaluate the unexpectedness of semifactuals, we propose measuring exceptionality, which prioritises explanations that can be reached by following trajectories where the unexpected happens. We define the exceptionality of a semifactual x' reached by a trajectory $T = (x_n, a'_n), \ldots, (x_{n+k}, a'_{n+k})$ (for forward semifactuals) or $T = (x_{n-k}, a'_{n-k}), \ldots, (x_{n-1}, a'_{n-k})$ (for backward semifactuals) as:

$$E(x_n, T) = \frac{1}{|T|} \sum_{(x_i, a_i) \in T} \mathbb{P}(x_{i+1} | x_i, a_i) \tag{4.18}$$

To estimate the probability of state transition $\mathbb{P}(x_{i+1}|x_i, a_i)$, we learn an approximation of the state transition probability function using Monte Carlo simulations. Specifically, we

estimate $\mathbb{P}(x_{i+1}|x_i, a_i)$ as the share of times the agent transitions to x_{i+1} after executing a_i in state x_i:

$$\mathbb{P}(x_{i+1}|x_i, a_i) = \frac{N(x_{i+1})}{N(x_i, a_i)} \tag{4.19}$$

It should be noted that here we have a likelihood (sum of probabilities), rather than a cumulative probability, for computational reasons. Intuitively, exceptionality prioritises trajectories in the environment where unexpected state transitions happen.

4.3.3 Semifactual Search

Semifactual metrics defined in the previous section can be used to evaluate and compare semifactual explanations. One of the main characteristics of semifactuals in RL reachability – the semifactual state should be connected to the original instance by RL actions (either by exploring the future or the past). For that reason, to find a suitable semifactual, we need to explore the space around the original instance using RL actions. We limit the search to k actions. Similarly to RACCER (Section 4.2), we refer to this parameter as horizon and note that the horizon depends on the task and outcome that is being explained.

Given a state x_n and a trajectory $\{(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})\}$ leading to x_n we search for a semifactual state x' utilizing backward or forward search. In other words, our goals is to find x' by generating a trajectory $T = \{(x_i, a_i)\}|_{i \in 1,\dots,k}$ which leads from x_n to x' for forward semifactuals and from x_{n-k} to x' for backward semifactuals.

Similar to the argument presented for counterfactual search in Section 4.2.3, an exhaustive search that explores each possible action sequence would be too time-consuming even in small environments. For that reason, in this section, we explore a heuristic approach for searching the space around the original instance. After successfully applying a multi-objective search for counterfactuals in Section 4.2.3, we utilise the same approach for semi-factual explanations. Specifically, we use NSGA-II to evaluate and propagate promising sequences of actions according to the semifactual metrics. In the next section, we describe the implementations of SGRL-Rewind and SGRL-Advance, the first semifactual generation approaches, which are fueled by the NSGA-II algorithm.

Multi-objective Search (NSGA-II)

To find a semifactual that optimises the five semifactual metrics, we define two search algorithms, SGRL-Rewind and SGRL-Advance. Both algorithms are counterfactual-free, i.e. they do not require searching for counterfactuals first. SGRL-Rewind and SGRL-Advance search for the semifactual state by exploring the neighbourhood around the original instance. While SGRL-Advance searches for forward semifactual states that can be reached from the original state in k actions, SGRL-Rewind searches for backward semifactuals that arise from the agent choosing different actions or observing different stochastic conditions in the previous k actions.

SGRL-Rewind and SGRL-Advance utilise an evolutionary algorithm, NSGA-II, to generate and evaluate potential action sequences which might lead to the optimal semifactual state. NSGA-II (Nondominated Sorting Genetic Algorithm II) [Deb et al., 2002] is an evolutionary algorithm that uses mutation, crossover, and combination to modify a set of solutions. Only the best solutions according to a fitness function are selected and propagated to future generations. NSGA-II provides a multi-objective optimisation and can simultaneously optimise different conflicting objectives. We use the same implementation of NSGA-II for SGRL as for the RACCER-Advance and RACCER-Rewind algorithms (Section 4.2).

An overview of NSGA-II for SGRL can be found in Algorithm 3. The algorithm is the same as the one used by RACCER (Algorithm 3), except it uses semifactual metrics as objectives and constraints, instead of counterfactual ones. For both SGRL-Rewind and SGRL-Advance, we use NSGA-II with four semifactual metrics – temporal distance, fidelity, stochastic uncertainty and exceptionality – as optimisation objectives and the semifactual validity metric as a constraint. Semifactual metrics of temporal distance, fidelity and stochastic uncertainty are already defined for sequences of actions. However, exceptionality is defined as a function of the trajectory leading to the semifactual (Equation 4.18). For this reason, we evaluate the exceptionality of the action sequence A as the average exceptionality of all trajectories that result in valid semifactuals by executing A from x_n (for forward semifactuals), or x_{n-k} (for backward semifactuals.)

Formally, to explain state x_n and a semifactual action a', SGRL-Advance aims to find a state x' that optimises the following:

$$\underset{x'=A(x_n)}{\arg\min} \Big[TD(x_n, A), 1 - S_t(x_n, A, a), \\ 1 - F(x_n, A), 1 - E(x_n, A) \Big] \text{ and } V(x', a') = 1 \tag{4.20}$$

Conversely, SGRL-Rewind searches for backward counterfactuals. Given a sequence of k previous state-action pairs $[(x_{n-k}, a_{n-k}), \dots, (x_{n-1}, a_{n-1})]$ leading to the factual state x_n, SGRL-Rewind explores semifactual states that could be reached if the agent had chosen an alternative sequence of actions starting in x_{n-k}. Formally, NSGA-Rewind optimises the following when explaining state x_n:

$$\underset{x'=A(x_{n-k})}{\arg\min} \Big[TD(x_n, A), 1 - S_t(x_n, A, a), \\ F(x_n, A), 1 - E(x_n, A) \Big] \quad \text{such that} \quad V(x') = 1 \tag{4.21}$$

Table 4.4 summarises the characteristics of the SGRL-Advance and SGRL-Rewind algorithms. Full evaluation in benchmark environments of SGRL algorithms is given in

Table 4.4: An overview of SGRL-Rewind and SGRL-Advance algorithms for generating semifactual explanations in RL tasks and their metrics.

Algorithm	SGRL-Rewind	SGRL-Advance
Backward search	✓	✗
Forward search	✗	✓
Diverse explanations	✓	✓
Optimization	Multi-objective	Multi-objective
Search algorithm	NSGA-II	NSGA-II

Chapter 5, while Chapter 6 details the evaluation on a real-life simulated task.

4.4 Chapter Summary and Contributions

In this section, we investigated the remaining two research questions defined in Section 1.2. Firstly, we sought to answer RQ3:

> *How can we design and implement counterfactual search algorithms in stochastic and sequential RL environments?*

Secondly, we investigate RQ4:

> *How can we design and implement semifactual search algorithms in stochastic and sequential RL environments?*

To answer RQ3, we introduced RACCER, the first RL-specific counterfactual generation method in Section 4.2. Firstly, we introduced a set of counterfactual metrics that enable us to evaluate RL-specific counterfactual requirements introduced in Section 3.5.2. We also offered the initial algorithm RACCER-HTS based on the heuristic tree search that optimises the above metrics and searches for the optimal counterfactual (Section 4.2.3). Additionally, we propose two algorithms, RACCER-Advance and RACCER-Rewind, based on an evolutionary algorithm which provides a more efficient search with a larger state space compared to the initial algorithm (Section 4.2.3).

To investigate RQ4, we implemented SGRL, the first semifactual generation method for RL. Firstly, we defined five semifactual metrics – validity, temporal distance, plausibility, stochastic uncertainty and exceptionality, which can be used to evaluate semifactual requirements defined in Section 3.6.2. Furthermore, we implemented two algorithms SGRL-Advance and SGRL-Rewind, based on evolutionary algorithms which optimise the above metrics and generate semifactual explanations for RL (Section 4.3.3).

To summarise, the contributions of this section are as follows:

1. *We define and implement a set of four counterfactual metrics that can be used to evaluate RL-specific counterfactual requirements.*

2. *We provide three algorithms, RACCER-HTS, RACCER-Rewind, and RACCER-Advance, which optimise the above metrics and can be used to search for counterfactual explanations in RL.*
3. *We define and implement a set of five semifactual metrics that can be used to evaluate RL-specific semifactual requirements.*
4. *We provide two algorithms, SGRL-Advance and SGRL-Rewind, that optimise the defined semifactual metrics and can be used to search for semifactual explanations in RL.*

In the following section, we provide an extensive evaluation of counterfactual and semifactual explanations generated using RACCER and SGRL approaches and compare them to state-of-the-art baselines.

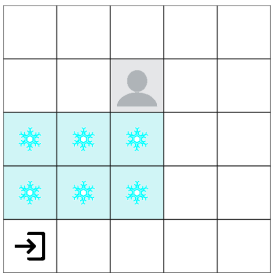
5 Evaluation

In this chapter, we present the evaluation of RACCER and SGRL algorithms for counterfactual and semifactual generation in RL. We start by describing evaluation environments in Section 5.1. Further sections describe evaluation hypotheses, metrics, baselines and evaluation results for counterfactual (Section 5.2) and semifactual explanation generation methods (Section 5.3).

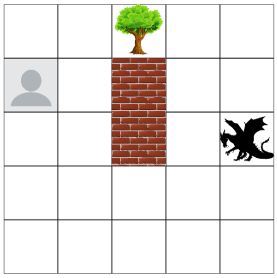
5.1 Evaluation Environments

We evaluate RACCER and SGRL in four standard benchmarking RL tasks – frozen lake, stochastic gridworld, highway driving and farm environment. We start by evaluating RACCER and SGRL on toy environments, frozen lake and stochastic gridworld with small, discrete state and action spaces. Such toy environments are commonly used for initial demonstration and evaluation of xRL methods [Eberhardinger et al., 2023, Liu et al., 2023, Saulières et al., 2023]. Additionally, similar toy environments are often chosen as use cases for user studies, as they allow for fast explanation generation and can be easily understood by users without prior knowledge that might influence their behaviour in the study [Amir and Amir, 2018, Huber et al., 2023, Deshmukh et al., 2023]. Furthermore, we evaluate RACCER and SGRL on more complex real-world highway driving and farm tasks, with discrete and continuous state features. With larger action and state spaces, and more complex behaviours, these environments present a suitable use case for evaluating the quality and scalability of explanations to more complex scenarios and have been used for this purpose by previous xRL works [Samadi et al., 2024, Yildirim et al., 2024, Gautron et al., 2022]. Additionally, all environments are stochastic and feature multiple conflicting objectives. Snapshots of the environments are presented in Figure 5.1.

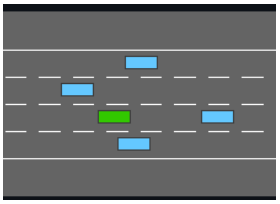
In the following sections, we provide in-depth descriptions of the benchmarking environments. Additionally, for each environment, we define a set of constraints that realistic states in this environment must satisfy. A summary of the characteristics of the environments is shown in Table 5.1. For each environment, we include in Table 5.1 a measure of stochastic complexity as the average time (in seconds) of running 10 RL actions in this environment. As RACCER and SGRL approaches rely on running simulations, we need this approximation to evaluate the scalability of these approaches to environments with increasingly complex stochastic processes. Some examples of counterfactual and semifactual explanations generated for these environments are available in Appendix



(a) Frozen lake



(b) Stochastic gridworld



(c) Highway driving

(d) Farm

Figure 5.1: Snapshots of the stochastic gridworld, frozen lake, highway driving and farm environments.

A2.

5.1.1 Frozen Lake

The frozen lake is a well-known stochastic environment in which the agent is tasked with reaching the goal while navigating a grid where some squares are covered in ice [Brockman et al., 2016]. In this thesis, we utilise a 5x5 frozen lake grid and fix the frozen states to specific squares for simplicity. An example of the environment is shown in Figure 5.1a. At each timestep, the agent can observe the entire gridworld, including the positions of frozen squares and the goal square. At each time step agent can either choose to move (by performing actions LEFT, RIGHT, UP, DOWN) or exit the grid by choosing action EXIT. The EXIT action results in the termination of the episode if the agent is at the goal, and otherwise, it does not have an effect on the environment. The episode ends when the agent successfully exits the episode or when the maximum number of timesteps is reached.

A negative base penalty of -1 is associated with each action. A positive reward of +10 is awarded after the agent successfully exits the environment. Stochasticity in the environment comes from the frozen states. Taking an action in these states leads the agent to the intended square in 50% of cases. Otherwise, the agent remains in its position. Positions of icy states are fixed between episodes. All states which contain an agent are considered realistic in this environment.

5.1.2 Stochastic Gridworld

Stochastic gridworld is a simple 5×5 *grid world, based on the commonly-used gridworld task [Chevalier-Boisvert et al., 2023]. An example of the environment is shown in Figure 5.1b. We alter the original environment to include trees, walls and a dragon, to increase complexity and introduce stochasticity. At each step, the agent can observe the positions of all the objects in the gridworld. At each step, the agent can move one step in any direction (UP, DOWN, LEFT, or RIGHT) or perform SHOOT and CHOP actions.*

The goal of the task is to kill the dragon. To successfully shoot the dragon, the agent has to be in the same file or row as the dragon, and the space between them has to be empty. In that situation, the agent can successfully perform the SHOOT action and win the game. However, trees and walls can block the agent's path to the dragon. An agent can chop down a tree or a wall by performing a CHOP action when located directly next to it. The episode ends when the dragon is shot or when the maximum number of time steps is reached.

The negative base penalty of -1 is applied to each action. A positive reward of +10 is awarded if the dragon is successfully killed. Stochasticity in the environment comes from the tree and wall objects, which can regrow after destruction. At each time step, there is a 25% *probability that a tree will grow and a* 25% *probability that a wall will be built at a specific square along the middle file. Trees can grow in place of walls after they are destroyed, and vice versa. To judge if a state is realistic under the rules of this environment, we check two constraints:*

1. *Agent and monster are always present in the environment.*
2. *No two objects (agent, monster, trees or walls) can populate the same square in the gridworld at the same time.*

Any state that fails any of these constraints is considered unrealistic under the rules of the environment.

5.1.3 Highway Driving

In the highway driving environment [Leurent, 2018], the agent is tasked with driving on a multi-lane highway. An example of the environment is given in Figure 5.1c. The highway consists of three lanes and is populated by other vehicles. The agent can observe the coordinates and speed of itself and four of its closest neighbouring vehicles. At each step agent chooses one of the five discrete actions – LANE LEFT, LANE RIGHT (to change lanes), SLOWER, FASTER (to change speed), or IDLE (i.e., continue driving without changing the lane or speed). The episode terminates when the agent crashes into another vehicle.

Highway driving is a multi-objective environment in which an agent needs to balance different competing objectives. Specifically, positive rewards are awarded for maintaining good speed in order to achieve good progress on the road. Negative rewards, however, are a result of vehicle crashes. Taking action a in state s elicits the following reward:

$$R(s,a) = 0.4 \cdot \frac{\mathcal{V} - \mathcal{V}_{min}}{\mathcal{V}_{max} - \mathcal{V}_{min}} - collision \tag{5.1}$$

where $\mathcal{V}$ *is the ego vehicle's velocity,* $\mathcal{V}_{max}$ *is the maximum and* $\mathcal{V}_{min}$ *minimum velocity. Stochasticity in this environment comes from the behaviour of other vehicles on the road which operate independently of the agent. All states whose features are within the range* $[0, 1]$ *are considered realistic in this environment.*

Table 5.1: Summary of the characteristics of the evaluation environments: frozen lake, stochastic gridworld, highway driving and farm. Stochastic complexity is calculated as the average time for a fixed-length simulation in the environment.

Task	State space		Action space		Stochastic complexity (s)
	Num. of features	Feature type	Num. of actions	Action type	
Frozen lake	8	Discrete	5	Discrete	$1.30 \cdot 10^{-4}$
Stochastic gridworld	7	Discrete	6	Discrete	$8.00 \cdot 10^{-4}$
Highway	30	Mixed	5	Discrete	$5.95 \cdot 10^{-1}$
Farm	10	Mixed	7	Discrete	$1.67 \cdot 10^{-2}$

5.1.4 Farm

Farm-gym [Maillard et al., 2023] is a collection of environments that simulate diverse agricultural tasks, adapted for RL. In this thesis, we focus on a farm environment that involves growing a tomato plant. An example of the environment is given in Figure 5.1d. The goal of the task is to successfully grow a plant and obtain a large harvest. The agent can observe plant progress indicators such as its growth stage or fruit weight. Additionally, the agent observes outside conditions such as weather, rain and temperature. In total, there are 10 *state features observed at each time step. Additionally, at each timestep agent chooses between* 7 *discrete actions – to harvest the plant or to apply between 0L and 5L of water.*

When harvesting a ripe plant, a positive reward equal to the weight of the harvested fruit is issued. Additionally, each time a plant transitions to a new growth stage, a positive reward of +1 is awarded. Stochasticity is exhibited in the weather conditions, such as temperature, humidity, and rain. To evaluate if a state is realistic in this environment, we check the following constraints:

1. *The maximum day temperature feature must be higher than the minimum day temperature feature.*
2. *The average day temperature feature value must be between the minimum and the maximum day temperature feature values.*

Any state that does not comply with any of the above constraints is considered unrealistic under the rules of this environment.

5.2 Evaluating Counterfactual Explanations

In this section, we evaluate RACCER-HTS, RACCER-Rewind, and RACCER-Advance approaches for generating counterfactual explanations for RL tasks introduced in Chapter 4. Firstly, in Section 5.2.1 we describe baseline approaches. Section 5.2.2 outlines the research hypotheses evaluated in this thesis. Details of the evaluation setup are described in Section 5.2.3. Finally, results are presented in Section 5.2.4.

5.2.1 Baselines

At the moment, there are two approaches for generating counterfactual explanations for RL tasks – CSE [Olson et al., 2019] and GANterfactual-RL [Huber et al., 2023]. However, CSE is a model-specific approach and requires access to the black-box model's internal parameters to generate counterfactual explanations. GANterfactual-RL is, however, a model-agnostic approach and can be used to explain any model. Additionally, Huber et al. [2023] found that CSE can produce misleading counterfactuals of extremely low validity and has shown GANterfactual-RL to perform better both in quantitative evaluation of counterfactual properties as well as in a user study compared to CSE. For these reasons, we use GANterfactual-RL as a baseline to evaluate RACCER.

GANterfactual-RL

GANterfactual-RL [Huber et al., 2023] is the only model-agnostic approach for generating counterfactual explanations for RL tasks. GANterfactual-RL algorithm is based on the domain translation problem using a StarGAN architecture [Choi et al., 2018]. Generating counterfactuals using GANterfactual-RL involves training two neural networks - a discriminator D and a generator G. The generator G is trained to translate the input state x into x', conditioned on the target domain c. The role of the discriminator is to distinguish between real states and fake ones produced by G. The GANterfactual-RL approach requires a dataset of states split into domains between which states can be translated. Each domain corresponds to an RL action and contains states in which the agent chooses that action.

5.2.2 Evaluation Hypotheses and Metrics

Our goal is to evaluate RACCER-HTS, RACCER-Advance, RACCER-Rewind and the baseline approach GANterfactual-RL on both quantitative and qualitative metrics. The following section outlines the precise evaluation metrics.

Quantitative Evaluation Hypotheses

We posit four quantitative evaluation hypotheses to evaluate counterfactual explanations generated by RACCER-HTS, RACCER-Advance, RACCER-Rewind and the baseline approach GANterfactual-RL. A summary of the quantitative evaluation hypotheses with their corresponding evaluation metrics is given in Table 5.2.

Firstly, our goal is to evaluate whether RACCER performs better on RL-specific metrics defined in Section 4.2.2. To that end, we compare RACCER-HTS, RACCER-Advance and RACCER-Rewind with GANterfactual-RL on the RL-specific metrics of validity, temporal distance, fidelity and stochastic certainty.

Secondly, our goal is to evaluate whether counterfactual explanations generated by RACCER and GANterfactual-RL are realistic under the rules of the environment. As GANterfactual-RL uses generative modelling to generate counterfactuals, it does not

explicitly ensure that the explanations are realistic under the rules of the environment. RACCER, on the other hand, searches for the counterfactuals in the agent's execution, ensuring they belong to the state space of the RL task. To evaluate how realistic counterfactual explanations are, we evaluate explanations generated by RACCER approaches RACCER-HTS, RACCER-Advance and RACCER-Rewind and those generated by the baseline GANterfactual-RL based on task-specific constraints described in Section 5.1.

Thirdly, we evaluate the diversity of explanations generated by RACCER approaches and GANterfactual-RL. Diversity has been recognised as an important property of counterfactual explanations [Mothilal et al., 2020, Laugel et al., 2023]. Different users have different preferences and require different explanations. Additionally, a single counterfactual might be too restrictive, as it only shows how the decision is affected by only a small subset of potentially important factors [Wachter et al., 2017]. Offering diverse explanations could help users understand how the decision is affected by different factors. For example, consider an AI agricultural tool explaining why it applied a large amount of water to the field. While one counterfactual might state that less water would be needed if the weather were more rainy, another might signal that less water would be applied today had the temperature been lower. To evaluate diversity, we measure 1) the number of generated explanations per factual state and 2) the pair-wise feature diversity of generated counterfactual explanations. For a factual state x and a set X' of generated counterfactual explanations, we measure pair-wise diversity as:

$$D(x, X') = \frac{1}{|X'|} \sum_{i \in \{1,\dots,|X'|\}} \sum_{\substack{j \in \{i+1,\dots,|X'|\} \\ j \neq i}} ||x'_i - x'_j||_2 \tag{5.2}$$

Finally, our goal is to compare RACCER approaches based on evolutionary algorithms, RACCER-Advance and RACCER-Rewind, to RACCER-HTS, which utilises heuristic tree search. While these approaches optimise the same properties, we hypothesise that evolutionary algorithms will produce counterfactuals of similar counterfactual property values but in less time compared to RACCER-HTS. To evaluate the scalability of these approaches, we measure 1) coverage, defined as the number of factual queries for which a counterfactual is successfully generated and 2) average generation time for these approaches.

In summary, we evaluate approaches based on RACCER on four quantitative hypotheses:

- ***H1.1 [Counterfactual Metrics]:** Counterfactual explanations generated by RACCER-HTS, RACCER-Advance and RACCER-Rewind will perform better on RL-specific metrics of validity, temporal distance, fidelity and stochastic certainty compared to the baseline.*
- ***H1.2 [Realistic Explanations]:** RACCER approaches RACCER-HTS, RACCER-Advance and RACCER-Rewind will produce counterfactual explanations that are*

Table 5.2: A summary of quantitative evaluation goals and metrics for evaluating counterfactual explanation generation methods.

Hypothesis	Evaluation Goal	Evaluation Metrics
H1.1	Generating closest possible world counterfactuals	Validity Temporal Distance Fidelity Stochastic Certainty
H1.2	Generating realistic counterfactuals	Task-specific constraints
H1.3	Generating diverse sets of counterfactuals	Number of Explanations Feature Diversity
H1.4	Scalable counterfactual generation	Coverage Generation Time

realistic under the rules of the RL environment more often compared to the baseline.

- ***H1.3 [Diversity]:*** *RACCER approaches RACCER-Advance and RACCER-Rewind will produce a more diverse set of counterfactual explanations compared to the baseline.*
- ***H1.4 [Scalability]:*** *RACCER approaches RACCER-Advance and RACCER-Rewind will produce counterfactual explanations in less time compared to the baselines and the RACCER-HTS approach.*

Qualitative Evaluation

Counterfactual explanations are ultimately intended to assist humans in real-life tasks, and evaluating them in this context is necessary to ensure their usefulness. To that end, we perform a qualitative evaluation of RACCER in a user study. Specifically, we compare counterfactual explanations generated by RACCER-HTS to those generated by the baseline approach GANterfactual-RL. The user study has three research hypotheses:

- ***H2.1 [Understanding Agent behaviour]:*** *RACCER-HTS will produce counterfactuals that can help users better understand and predict the behaviour of RL agents compared to the baseline.*
- ***H2.2 [Comparing Agents]:*** *RACCER-HTS will produce counterfactuals that help users better choose between agents with different preferences compared to the baseline.*
- ***H2.3 [User Satisfaction]:*** *Counterfactual explanations generated by RACCER-HTS will be perceived as more satisfactory by users compared to explanations generated by the baseline.*

To evaluate how well users understand agents, we ask users to predict agents' actions and report accuracy. To evaluate users' ability to compare different agents, we ask them to choose between agents given a specific preference. Finally, to evaluate user

Table 5.3: A summary of qualitative evaluation goals and metrics for evaluating counterfactual explanations.

Hypothesis	Evaluation Goal	Evaluation Metrics
H2.1	Understanding agent behaviour	Prediction accuracy
H2.2	Comparing agents with different preferences	Choice Accuracy
H2.3	User satisfaction	Useful Detailed Complete Satisfying Actionable Reliable Trustworthy

satisfaction, we use explanation goodness metrics [Hoffman et al., 2018], which provide a 1-5 Likert scale for measuring explanations' usefulness, detailedness, completeness, user satisfaction, actionability and reliability. Additionally, we measure the user's perceived trustworthiness of the generated explanations.

An overview of qualitative evaluation goals and metrics is available in Table 5.3, and the user study design is described in Section 5.2.3.

5.2.3 Experiment Design

In this section, we explain the design of experiments conducted to evaluate counterfactual explanations generated by RACCER and GANterfactual-RL algorithms. The following sections describe the training details for RL policies being explained, the process of generating factual queries and the user study design.

Training RL Policies

Our goal is to evaluate RACCER and GANterfactual-RL algorithms when explaining RL policies. For this purpose, for each task, we train an RL policy π until convergence. To train RL policies, we employ a DQN [Mnih et al., 2013] algorithm in all tasks. The training parameters and performance results for these policies are available in Table 5.4.

Additionally, in the stochastic gridworld environment, we train another policy π', which is trained on a different reward function than π. Specifically, while destroying trees and walls is equally costly for π, π' is trained on a reward function with a stronger -5 penalty for destroying walls, compared to a -1 penalty for destroying trees. We train π' for the same number of timesteps as π. Policies π and π' both converge to different behaviours, corresponding to the preferences encoded in their reward functions. Other than the reward function, policies π and π' are trained with the same parameters, available in Table 5.4. We use these two policies to investigate the qualitative hypothesis H2.2, which addresses the problem of the users' ability to distinguish between two good policies with different preferences.

Table 5.4: Training parameters and performance of RL policies in a frozen lake, stochastic gridworld, highway driving and farm environments.

Task	Training Parameters				Training timesteps	Average reward
	Architecture	Learning rate	Exploration Fraction	Gamma		
Frozen lake	[256, 256]	10^{-4}	0.80	0.90	$3 \cdot 10^5$	-3.76 ± 32.49
Stochastic gridworld	[256, 256]	10^{-4}	0.80	0.90	10^5	7.83 ± 1.54
Highway	[256, 256]	$5 \cdot 10^{-4}$	0.80	0.90	$2 \cdot 10^4$	24.26 ± 0.14
Farm	[256, 256]	$5 \cdot 10^{-4}$	0.80	0.90	10^5	30.82 ± 2.42

Generating Factual States

For each RL policy being explained, we focus on counterfactual questions about why an action was chosen in a state. Specifically, given that in state x the RL policy π chose action a, we generate counterfactuals to answer the question "Why not action a'?" for each alternative possible action a'. To select factual states for a policy π, we unroll it in the environment and, for each action a_i, we record a set of states $S_F(a_i)$ where π chooses action a_i. As we aim to search for counterfactuals in the agent's past, we only record states which have been preceded by a state-action path of at least length k, where k is the horizon parameter in RACCER and SGRL algorithms. Finally, 100 *states are selected at random from each $S_F(a_i)$ as factual states.*

Finally, for each target action a_i and factual state $x \in S_F(a_i)$ we utilise RACCER-HTS, RACCER-Advance, RACCER-Rewind and GANterfactual-RL approaches to generate counterfactual explanations. In all environments, we set the horizon parameter for RACCER approaches to $k = 5$. The horizon parameter is task-specific and should be chosen to be large enough that performing k actions can lead to a significant change in the agent's state, but small enough to enable a feasible search of the space of action sequences of length k. In this thesis, we find that in all environments, five actions are enough to substantially change the state (e.g. a plant can grow into a different stage in the farm environment, and an agent can change lanes and velocity in the highway driving). The parameters used by these methods are summarised in Table 5.5. Examples of counterfactual explanations generated by these approaches can be found in Appendix A2.

User Study Design

To evaluate hypotheses H2.1, H2.2, and H2.3, we conducted a user study to compare the counterfactual explanations produced by GANterfactual-RL and RACCER-HTS. We conducted the study in the stochastic gridworld environment, as it has simple rules and requires no prior knowledge from users. In this way, we ensure their answers are based on the presented explanations, rather than influenced by the prior experience with the task (as might be the case with, for example, driving). A detailed description of the user study is available in the Appendix A1.

We sourced 153 participants through the Prolific platform (https://www.prolific.com/) from English-speaking countries (UK, Ireland, Canada, USA, Australia, and New Zealand) and split them into two groups. The first group received counterfactuals generated by

Table 5.5: Training parameters for the GANterfactual-RL, RACCER-HTS, RACCER-Advance, and RACCER-Rewind approaches for generating counterfactual explanations for stochastic gridworld, frozen lake, highway driving and farm environments.

Algorithm	Parameter	Frozen Lake	Stochastic Gridworld	Farm	Highway Driving
GANterfactual-RL	Dataset size	$5 \cdot 10^5$	$5 \cdot 10^5$	$5 \cdot 10^5$	$5 \cdot 10^5$
	Batch size	16	512	128	512
	Generator architecture	[128, 128]	[128, 128]	[128, 128]	[128, 128]
	Discriminator architecture	[256,256]	[256,256]	[256,256]	[256,256]
	Training timesteps	$3 \cdot 10^3$	$3 \cdot 10^3$	$3 \cdot 10^3$	$3 \cdot 10^3$
RACCER-HTS	Number of iterations	300	300	300	300
	Number of simulations	10	10	10	10
	Loss parameter α	1	1	1	1
	Loss parameter β	1	1	1	1
	Loss parameter γ	1	1	1	1
	Horizon	5	5	5	5
RACCER-Advance	Number of generations	25	25	25	25
	Population size	24	24	24	24
	Number of simulations	10	10	10	10
	Horizon	5	5	5	5
RACCER-Rewind	Number of generations	25	25	25	25
	Population size	24	24	24	24
	Number of simulations	10	10	10	10
	Horizon	5	5	5	5

GANterfactual-RL, and the second counterfactuals produced by RACCER-HTS. After filtering participants for those who had passed attention checks, 58 participants remained in the first group and 63 in the second group. Participants were remunerated according to the Prolific payment policy.

The study design follows that used to evaluate the GANterfactual-RL algorithm in Huber et al. [2023]. Before the study, users were shown general information about the task and the study. Users were also shown a definition and examples of counterfactual explanations and asked to answer test questions to ensure a full understanding of the task. The template for the study can be found at: https://qrxhyre44mt.typeform.com/to/cpeLrWbZ. The study consisted of 3 parts: evaluating user understanding of the agent's behaviour, evaluating user understanding of the agent's preferences, and evaluating user satisfaction.

Firstly, in the training phase, the users were shown the behaviour of two agents A and B, with different policies, described in more detail in Section 5.2.3. For each agent, users go through two stages: the training and testing stages. In the training stage, users are shown a game state, and the action agent chooses that state. Then, users are presented with counterfactual states, describing in which situations the agent would choose an alternative action. For each agent, the user sees 10 training states. In the testing phase, users are presented with 10 *states and asked to predict an action the agent would take, without being given the explanations. Examples of the states and explanations presented to users are available in Appendix A1.*

The factual states in the training and testing phase are selected by the HIGHLIGHTS-DIV algorithm [Amir and Amir, 2018], inspired by the setup from Huber et al. [2023]. This way, users are presented with the most informative states of the agent's game-

play. We modify the HIGHLIGHTS-DIV algorithm to include states with a diverse range of Q-values, since using the original HIGHLIGHTS-DIV algorithm results in a mostly homogeneous set of states in which the agent should perform the SHOOT action. We generate the 20 most informative states according to HIGHLIGHTS-DIV and randomly split them into training and testing sets. To be able to show counterfactuals to the users, they need to be realistic. For that reason, we additionally filter the states obtained by the HIGHLIGHTS-DIV algorithm to ensure they are realistic. We present two counterfactual states for each factual one to reduce the cognitive load required by the experiment. We show counterfactuals for actions CHOP and SHOOT, as these actions represent the most interesting gameplay.

Secondly, after completing the training and testing phase for both agents, users were asked to choose a more suitable one according to a specific preference. Specifically, users are asked which agent they would choose if they wanted to keep the cost minimal, regardless of the time it takes to finish the task. Conversely, they were also asked which agent they would choose to finish the task quickly, regardless of the cost.

At the end of the study, users were asked to rank the explanations based on the explanation goodness metrics [Hoffman et al., 2018] on a $1-5$ *Likert scale (*1 *- strong disagreement,* 5 *- strong agreement). Users reported whether explanations were useful, satisfying, complete, detailed, actionable, trustworthy, and reliable.*

5.2.4 Evaluation Results and Analysis

In this section, we present the evaluation results for counterfactual generation methods presented in this work and state-of-the-art baseline approaches. The following sections showcase the results of evaluating the four quantitative and three qualitative evaluation hypotheses defined in Section 5.2.3.

We find that running RACCER-HTS is computationally infeasible in farm and highway environments, as generating only one counterfactual in these tasks could take as long as one hour. As we evaluate approaches on 500 *factual states, generating counterfactuals using RACCER-HTS would have been impossible. However, we include RACCER-HTS results on smaller frozen lake and stochastic gridworld environments.*

Counterfactual Metrics

To evaluate H1.1 (Section 5.2.2), we evaluate RL-specific metrics validity, temporal distance, fidelity and stochastic certainty for generated counterfactual explanations. The evaluation results for all tasks are presented in Table 5.6.

Counterfactual metrics, temporal distance, fidelity and stochastic certainty are defined with respect to the action sequence leading to the counterfactual explanation. For this reason, we can naturally calculate them for counterfactuals generated by RACCER-HTS, RACCER-Advance and RACCER-Rewind, as these algorithms search for counterfactuals by executing RL actions in the environment. However, the baseline approach

Table 5.6: Evaluation results for RL-specific counterfactual metrics when explaining optimal policies in the frozen lake, stochastic gridworld, highway and farm task for GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance approaches.

Task	Algorithm	Counterfactual Metrics			
		Validity (= 1)	Temporal Distance (↓)	Fidelity (↑)	Stochastic Certainty (↑)
Frozen lake	GANterfactual-RL	**1.0**	0.97	0.01	0.03
	RACCER-HTS	**1.0**	**0.38**	**0.16**	0.50
	RACCER-Rewind	**1.0**	0.45	**0.16**	**0.62**
	RACCER-Advance	**1.0**	0.62	0.07	0.59
Stochastic gridworld	GANterfactual-RL	**1.0**	1.0	0.0	0.0
	RACCER-HTS	**1.0**	**0.38**	0.14	**0.58**
	RACCER-Rewind	**1.0**	0.48	0.05	0.51
	RACCER-Advance	**1.0**	0.42	**0.16**	0.55
Highway	RACCER-Rewind	**1.0**	**0.22**	**0.32**	0.80
	RACCER-Advance	**1.0**	0.30	0.21	**0.81**
Farm	RACCER-Rewind	**1.0**	**0.40**	**0.09**	0.25
	RACCER-Advance	**1.0**	0.63	0.04	**0.28**

GANterfactual-RL uses generative modelling to generate counterfactuals and relies in no way on the notion of RL actions. For this reason, we cannot directly estimate the counterfactual metrics for this algorithm. Given the original state x_n and the counterfactual x' generated by GANterfactual-RL, we run a genetic algorithm to explore different action sequences that could connect x_n to x' or x_{n-k} to x'. If such an action sequence is found, the counterfactual metrics are evaluated with respect to it, and the values are assigned to the counterfactual x'. In frozen lake and stochastic gridworld environments with discrete state spaces, we find the appropriate action sequence connecting the original and counterfactual state in less than 5% of the cases. We include the obtained results in the Table 5.6. However, in environments with larger state spaces and continuous state features, such as farm and highway driving, it becomes computationally prohibitive to search for action sequences connecting two states. Additionally, given the low reachability achieved in smaller environments, it is unlikely that counterfactuals would be easier to reach in even larger state spaces with continuous state features. For this reason, it is infeasible to compute action sequences for counterfactuals generated by GANterfactual-RL to evaluate counterfactual metrics.

In all environments, all generated explanations are valid (with validity = 1.0). This is because all approaches considered in this work see validity as a constraint and only generate valid counterfactual explanations.

RACCER-HTS achieves the best results on the temporal distance metric in frozen lake (TD = 0.38) and stochastic gridworld environments (TD = 0.38). This means that a counterfactual generated by RACCER-HTS is reachable on average in just under two RL actions from the original state. As RACCER-HTS grows, the search tree iteratively starting in the original state, it is not surprising that it can find counterfactuals that are closer to the original state. In the highway and farm environment, the best results on temporal distance are achieved by the RACCER-Rewind approach. In particular, in the highway environment, RACCER-Rewind achieves a temporal distance of 0.22, meaning counterfactuals are reachable in just one action.

On the fidelity metric, RACCER-HTS and RACCER-Rewind approaches show better results in the frozen lake task ($F = 0.16$ for both), while RACCER-Advance is the best in the stochastic gridworld ($F = 0.16$) and RACCER-Rewind in the farm task ($F = 0.09$). However, all RACCER approaches show similar results for fidelity, with no approach achieving fidelity higher than 0.32 (where the best fidelity score would be 1, and the worst 0). This result is not surprising – while fidelity ensures that the counterfactual is likely under the agent's policy, reaching a counterfactual state often requires taking actions different to those the agent deems the best.

In all environments, all RACCER approaches achieve similar results in the stochastic certainty metric. In the frozen lake task, RACCER-Rewind produces the most certain counterfactuals ($S_t = 0.62$), while in the stochastic gridworld, it is RACCER-HTS ($S_t = 0.58$) and in the farm environment, RACCER-Advance ($S_t = 0.28$). In the farm environment, the counterfactuals are the most uncertain, with RACCER-Advance achieving $S_t = 0.28$ and RACCER-Rewind $S_t = 0.25$ indicating high stochasticity of the environment.

We find that counterfactuals generated by GANterfactual-RL are reachable from the original state in only around 5% of cases, resulting in low performance on RL-specific counterfactual metrics, which rely on the underlying connection between the counterfactual and original states. Additionally, we find that RACCER-HTS can produce counterfactuals reachable in fewer RL actions, as a consequence of its algorithm iteratively increasing the number of actions it is considering. Similarly, RACCER-HTS is comparable to RACCER-Rewind and RACCER-Advance in the frozen lake and stochastic gridworld on fidelity and stochastic certainty. ***In conclusion, RACCER-HTS, RACCER-Rewind, and RACCER-Advance perform similarly to each other, and outperform the baseline GANterfactual-RL approach on RL-specific counterfactual metrics, confirming hypothesis H1.1.***

Generating Realistic Explanations

The evaluate hypothesis H1.2 as defined in Section 5.2.2, we measure the percentage of realistic counterfactual explanations generated by GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance approaches. Whether a counterfactual explanation is realistic depends on task-specific constraints described for each environment in Section 5.1. The results of this evaluation are presented in Table 5.7.

RACCER-HTS, RACCER-Advance and RACCER-Rewind produce only realistic instances in all environments. RACCER-HTS, RACCER-Advance and RACCER-Rewind approaches can only produce realistic states by design. This is because they traverse the environment in counterfactual search and choose counterfactual explanations among existing states. On the other hand, GANterfactual-RL uses generative modelling and can produce counterfactuals which cannot exist within the environment.

In the frozen lake environment, GANterfactual-RL produces only realistic explanations for the frozen lake and highway driving tasks. In the stochastic gridworld, however, only 32.17% explanations generated by GANterfactual-RL are realistic. Similarly, in the farm

Table 5.7: Evaluation results on generating realistic instances, diversity and scalability metrics when generating counterfactual explanations for explaining a policy π using GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance algorithms.

Task	Algorithm	Realistic expl. (%) (↑)	Diversity		Scalability	
			Num. of expl. (↑)	Feature diversity (↑)	Coverage (↑)	Gen. time (s) (↓)
Frozen lake	GANterfactual-RL	**100**	1	0	69.20	**0.0003**
	RACCER-HTS	**100**	1	0	80.20	26.88
	RACCER-Rewind	**100**	8.28	**3.28**	71.4	3.31
	RACCER-Advance	**100**	**9.89**	2.39	**81.4**	3.19
Stochastic gridworld	GANterfactual-RL	32.17	1	0	18.17	**0.003**
	RACCER-HTS	**100**	1	0	**84.67**	44.32
	RACCER-Rewind	**100**	3.07	**5.28**	74.00	6.84
	RACCER-Advance	**100**	**3.24**	1.47	79.83	6.29
Highway	GANterfactual-RL	**100**	1	0	65.60	**0.0004**
	RACCER-Rewind	**100**	2.33	0.57	**100**	452.10
	RACCER-Advance	**100**	**2.55**	**0.64**	**100**	496.70
Farm	GANterfactual-RL	50.26	1	0	25.71	**0.0003**
	RACCER-Rewind	**100**	1.66	**2.95**	**35.29**	83.56
	RACCER-Advance	**100**	**1.75**	2.67	32.29	98.81

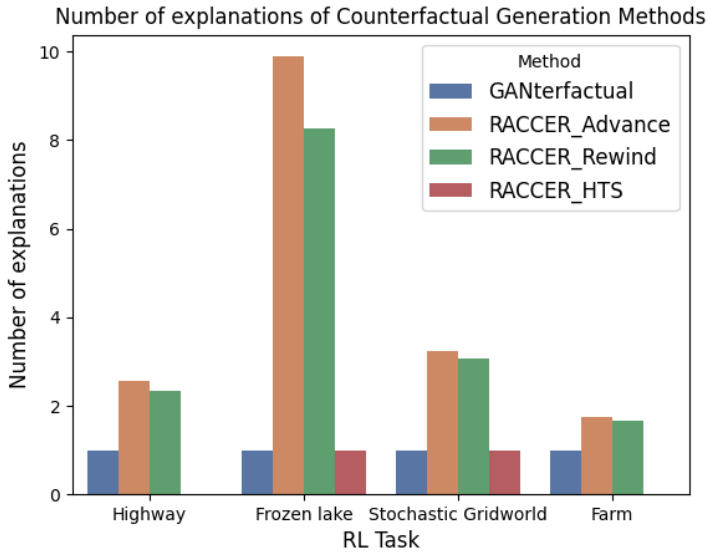


Figure 5.2: Number of counterfactual explanations generated by GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance algorithms in the frozen lake, stochastic gridworld, highway driving and farm environments.

environment, 50.26% *of explanations generated are realistic.*

In conclusion, we find that in all four environments, RL-based methods RACCER-HTS, RACCER-Advance and RACCER-Rewind produce only realistic counterfactuals, while GANterfactual-RL does not always produce realistic explanations, confirming hypothesis H1.2.

Generating Diverse Explanations

To evaluate hypothesis H1.3 defined in Section 5.2.2, we evaluate diversity for counterfactual explanations generated by GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance. The results in all tasks are presented in Section 5.7.

GANterfactual-RL and RACCER-HTS generate only one counterfactual explanation by design. As they do not generate a set of explanations, we cannot evaluate their feature diversity and assign them 0 *for this metric.*

In all environments, RACCER-Advance generates the highest number of explanations, ranging from 1.75 *explanations in the farm environment to* 9.89 *explanations per factual state in the frozen lake. RACCER-Rewind generates a similar number of explanations, with* 1.66 *in the farm and* 8.28 *in the frozen lake environment. In the stochastic gridworld, RACCER-Advance generates* 3.24 *and RACCER-Rewind* 3.07 *explanations on average. Similarly, in a highway environment, RACCER-Advance generates* 2.55 *and RACCER-Rewind* 2.33 *explanations on average per factual state. A visualisation of the number of generated counterfactuals per counterfactual search algorithm and environment is given in Figure 5.2.*

However, despite the higher number of explanations generated, RACCER-Rewind shows higher feature diversity than RACCER-Advance in frozen lake, stochastic gridworld and farm environments. The biggest difference is visible in the stochastic gridworld environment, where RACCER-Rewind achieves feature diversity at 5.28 *and RACCER-Advance at* 1.47*. In the stochastic gridworld, RACCER-Advance achieves* 2.39*, compared to RACCER-Rewind at* 3.28*. In the highway environment, RACCER-Advance shows feature diversity of* 0.64*, compared to* 0.57 *achieved by RACCER-Rewind.*

In conclusion, we find that in all environments, RACCER-Advance and RACCER-Rewind generate a larger number of explanations with higher diversity compared to the baseline GANterfactual-RL approach, confirming hypothesis H1.3.

Scalability

To evaluate hypothesis H1.4 defined in Section 5.2.2, we evaluate how the tree-search-based RACCER-HTS approach scales to higher complexity environments compared to RACCER-Advance and RACCER-Rewind based on evolutionary algorithms. For completeness, we report and discuss scalability evaluation results for the GANterfactual-RL approach in this section as well, even though this approach is not explicitly a part of H1.4. The results of this evaluation are presented in Table 5.7.

In all environments, RACCER-HTS achieves similar coverage to the RACCER-Rewind and RACCER-Advance approaches. In the frozen lake, stochastic gridworld, and highway tasks, RACCER-HTS, RACCER-Advance, and RACCER-Rewind achieve coverage at over 70%*. The lowest coverage is achieved in the farm task, where all approaches report coverage lower than* 40%*. RACCER-Rewind generates the most counterfactuals in the farm task at* 35.29%*, followed by RACCER-Advance at* 32.39 *and GANterfactual-RL at* 25.71%*. We suspect this drop in coverage in the farm task is due to the higher number of actions in this environment, resulting in a smaller search space of potential counterfactuals.*

All three RACCER approaches also outperform GANterfactual-RL on coverage in all environments. The contrast is particularly noticeable in the stochastic gridworld, where GANterfactual-RL produces counterfactuals for less than 20% *of factual queries, while all other approaches report coverage above* 70%*. Similarly, in the highway environ-*

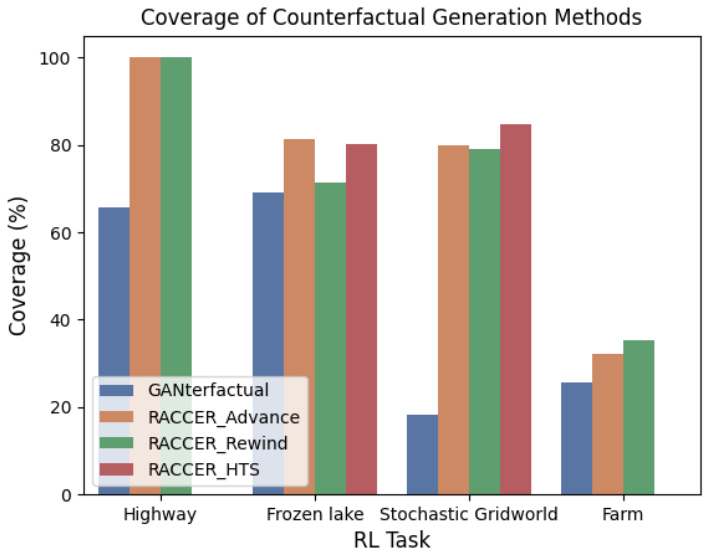


Figure 5.3: Coverage of the GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance counterfactual generation algorithms in the frozen lake, stochastic gridworld, farm and highway environments.

ment, RACCER-Advance and RACCER-Rewind generate counterfactuals for each state, while GANterfacutal-RL achieves coverage of only 65.60%. *Coverage of counterfactual generation algorithms is visualised in Figure 5.3.*

GANterfactual-RL reports the lowest time to generate an explanation at well below one second. However, this might be misleading, as GANterfactual requires a longer pretraining, which can take several hours depending on the environment. RACCER-HTS, RACCER-Rewind and RACCER-Advance, on the other hand, require no pretraining but take longer to generate an explanation. However, in each environment, RACCER-Rewind and RACCER-Advance produce counterfactuals faster than RACCER-HTS. In the stochastic gridworld, RACCER-Advance and RACCER-Rewind require around 3*s compared to around* 26*s for RACCER-HTS per factual state. Similarly, in the stochastic gridworld, RACCER-Rewind and RACCER-Advance generate a counterfactual of around* 6*s, while RACCER-HTS requires over* 40*s on average. In the highway environment, RACCER-Rewind and RACCER-Advance take less than two minutes per factual state, while in the highway task, they require the most time to generate the counterfactual at around eight minutes. In contrast, RACCER-HTS takes around one hour, a substantially longer time to generate a counterfactual in farm and highway tasks, making it infeasible to evaluate in these environments.*

We find that the generation time of RACCER-HTS, RACCER-Advance and RACCER-Rewind depends primarily on the time needed to run simulations in that environment. For this reason, less time is needed in gridworld and frozen lake environments, where stochastic processes are simple to calculate and simulate. More time is needed in the farm and highway environments as their stochastic processes are more complex and more time-consuming to calculate (e.g. simulating the weather conditions or the behaviour of other vehicles on the road).

To summarise, although RACCER-HTS achieves similar coverage as RACCER-Advance and RACCER-Rewind, it requires around eight times longer on average to generate a counterfactual explanation, confirming H1.4.

User Study

In this section, we present the results of the user study conducted to evaluate the qualitative hypotheses H2.1, H2.2 and H2.3 defined in Section 5.2.2.

Firstly, we aim to evaluate H2.1, which concerns the user's ability to understand the agent's behaviour after seeing counterfactual explanations. In keeping with previous studies on counterfactual explanations [Dai et al., 2022, Huber et al., 2023], we use the prediction accuracy of the agent's actions in the testing phase as a metric for measuring user understanding of the agent's behaviour. Users who have seen counterfactuals generated by RACCER-HTS have shown 76.19% *accuracy in predicting agents' actions. In contrast, users who have been presented with counterfactuals generated using the GANterfactual-RL approach have achieved an accuracy of* 70.94%*. After conducting a non-parametric one-tailed Mann-Whitney U test, we find a significant difference between the prediction accuracy of the two approaches (*$p = 0.0185$*). This* ***indicates that RL-specific counterfactuals help users better understand and predict the behaviour of RL agents, and confirms H2.1.***

Secondly, we evaluate H2.2, which addresses users' ability to distinguish between agents with different preferences after seeing counterfactual explanations. Users presented with RACCER-HTS explanations choose the correct agent in 53.17%*, while users who have seen GANterfactual-RL explanations made a correct choice in* 58.62% *of cases. We condu,ct a non-parametric one-tailed Mann-Whitney U test and found no significant difference between the two approaches (*$p = 0.6509$*). This indicates that* ***contrary to H2.2, RACCER-HTS is not better at helping users distinguish between agents with different policies compared to GANterfactual-RL.*** *This result is in line with previous research on counterfactuals in supervised learning [Huber et al., 2023] which also found that, while counterfactuals can help users better understand and predict agent's actions, they do not lead to an improvement in the ability to compare agents. One reason for this might be that counterfactuals as local explanations are more suitable for understanding and predicting a single action, rather than interpreting an agent's global preferences. In Section 7.3 we discuss possible integration of counterfactuals with other xAI methods to enable agent comparison.*

Thirdly, we evaluate H2.3 by measuring users' satisfaction on a 1-5 Likert scale of explanation goodness metrics [Hoffman et al., 2018]. The results of this part of the study are presented in Figure 5.4. After conducting a non-parametric one-tailed Mann-Whitney U test, we find that users perceive explanations generated by ***RACCER-HTS to be significantly more useful for understanding the agent ($p = 0.0057$), more detailed ($p = 0.0190$) and complete ($p = 0.0095$) compared to those generated by the GANterfactual-RL approach.*** *However, there is no significant difference be-*

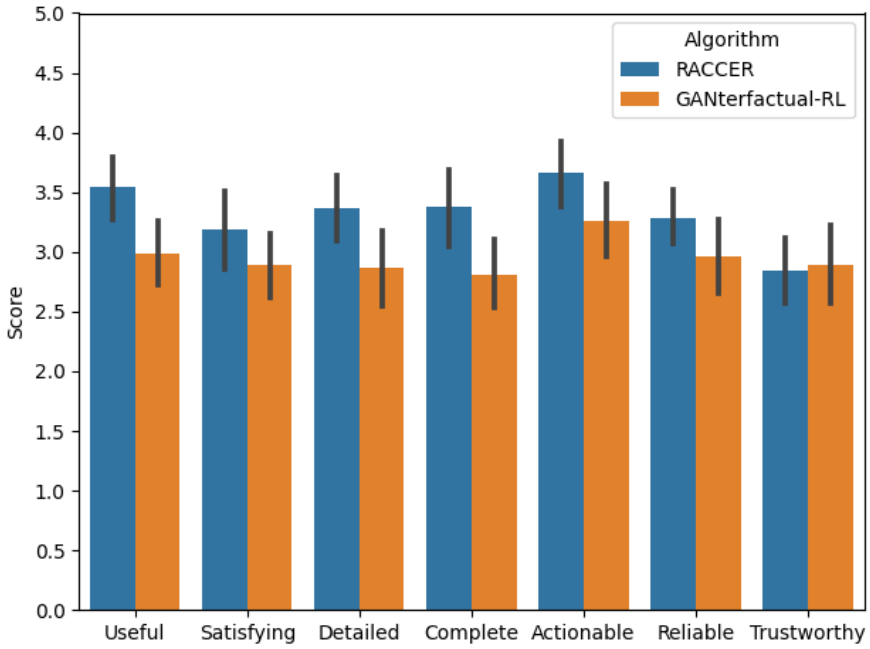


Figure 5.4: User satisfaction score on explanation goodness metrics [Hoffman et al., 2018] for counterfactual explanations generated by RACCER-HTS and GANterfactual-RL approaches in a stochastic gridworld environment.

tween the approaches in the perceived trustworthiness ($p = 0.7901$), reliability ($p = 0.1446$), and actionability ($p = 0.0729$) of explanations, ***resulting in H2.3 being only partially confirmed by our experiments.***

Additional Results: Explaining Suboptimal Policies

In this section, we include some initial, smaller-scale results which have emerged from the initial investigation of these algorithms. Specifically, we investigated how RACCER algorithms and GANterfactual-RL baseline could be used to explain not only fully converged but also suboptimal policies.

Most often, counterfactual explanations have been applied to explain the behaviour of fully trained agents. However, understanding suboptimal agents is necessary for verification and debugging. For example, Olson et al. [2019] has used counterfactuals to help users recognise agents that relied on artificially inserted pixels correlated with an action choice. These agents do not base decisions on actual game elements, and their performance suffers when the spurious correlation is broken.

GANterfactual-RL trains supervised learning models to translate states between domains. We hypothesise that this approach, although suitable for explaining fully-trained agents, cannot be applied to suboptimal ones. This is because domains for training the generator and discriminator models in GANterfactual-RL are defined based on which actions the RL agent would make in a state. However, for a suboptimal agent, some randomness in the decision-making process is likely. This introduces randomness into domains, resulting in domains that are difficult to separate, making supervised learning of discriminator and generator models challenging. To evaluate this hypothesis, we use all metrics from the

Table 5.8: Evaluation results for RL-specific counterfactual metrics when explaining suboptimal policies in a frozen lake, stochastic gridworld tasks for GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance approaches.

Task	Algorithm	Counterfactual Metrics			
		Validity (= 1.0)	Temporal Distance (↓)	Fidelity (↑)	Stochastic Certainty (↑)
Frozen lake	GANterfactual-RL	**1.0**	0.97	0.02	0.01
	RACCER-HTS	**1.0**	**0.41**	**0.10**	0.55
	RACCER-Rewind	**1.0**	0.44	**0.10**	**0.63**
	RACCER-Advance	**1.0**	0.58	0.04	0.58
Stochastic gridworld	GANterfactual-RL	**1.0**	1.0	0.0	0.0
	RACCER-HTS	**1.0**	**0.39**	0.11	**0.60**
	RACCER-Rewind	**1.0**	0.43	**0.15**	0.43
	RACCER-Advance	**1.0**	0.54	0.06	0.54

above three hypotheses used to evaluate the quality of counterfactual explanations (i.e. RL-specific counterfactual metrics, realistic constraints and diversity metrics).

To perform an initial investigation on this topic, we train a suboptimal policy $\pi_{suboptim}$ in a frozen lake and stochastic gridworld tasks. This policy is trained on the same reward function and same training parameters as policy π described in Table 5.4. However, policy $\pi_{suboptim}$ is trained for only one-tenth of the training steps used to train π. Factual states for this evaluation are gathered in the same way as for explaining the converged policy π as described in Section 5.2.3.

We start by investigating the ability of approaches to explain suboptimal policies on smaller tasks frozen lake and stochastic gridworld. For each factual state, we generate explanations using GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance algorithms. We report the results on counterfactual metrics in Table 5.8, and the results on the percentage of realistic instances, diversity and scalability are presented in Table 5.9.

In both environments, all approaches show similar results on RL-specific metrics when explaining optimal (Table 5.6) and suboptimal policies (Table 5.8). Similarly, we find that RACCER-HTS produces the closest counterfactual for suboptimal policies, with a temporal distance of 0.41 in frozen lake (compared to 0.38 for optimal policy in Table 5.6), and 0.39 in gridworld (compared to 0.38 for optimal policy in Table 5.8).

The biggest difference between explaining optimal and suboptimal policies is exhibited in the coverage metric and the ability to generate realistic explanations. When it comes to coverage, RACCER-HTS, RACCER-Rewind and RACCER-Advance achieve similar coverage as for optimal policies in a frozen lake and stochastic gridworld environments. However, the coverage of GANteractual-RL suffers in all environments, dropping from 69.20% *for optimal to* 20% *for suboptimal policy in the frozen lake task. On the other hand, GANterfactual-RL shows increased coverage in the stochastic gridworld task when explaining a suboptimal policy with counterfactuals generated for* 22.30% *of factual states, compared to* 18.17% *when explaining an optimal policy. When it comes to the ability to generate realistic instances, the percentage of realistic explanations generated by GANterfactual-RL increases to* 54% *In the stochastic gridworld environment, compared to* 32% *reported for optimal policies.*

Table 5.9: Evaluation results on generating realistic instances, diversity and scalability metrics when generating counterfactual explanations for suboptimal policies on the frozen lake and stochastic gridworld tasks using GANterfactual-RL, RACCER-HTS, RACCER-Rewind and RACCER-Advance algorithms.

Task	Algorithm	Realistic expl. (%) (↑)	Diversity		Scalability	
			Num. of expl. (↑)	Feature diversity (↑)	Coverage (↑)	Gen. time (s) (↓)
Frozen lake	GANterfactual-RL	**100**	1	0	20.00	**0.0003**
	RACCER-HTS	**100**	1	0	**88.40**	26.80
	RACCER-Rewind	**100**	5.80	3.28	83.80	2.95
	RACCER-Advance	**100**	**7.34**	**8.28**	88.20	3.08
Stochastic gridworld	GANterfactual-RL	54.48	1	0	22.30	**0.0003**
	RACCER-HTS	**100**	1	0	**86.50**	45.77
	RACCER-Rewind	**100**	2.78	**5.28**	85.00	6.40
	RACCER-Advance	**100**	**3.09**	1.47	84.33	6.43

With these initial results, we conclude that the performance of GANterfactual-RL is not necessarily hindered when explaining a suboptimal policy. Interestingly, GANterfactual-RL reported an increase in coverage and the percentage of realistic explanations in the stochastic gridworld, indicating that it can adapt well to explaining suboptimal policies.

5.3 Evaluating Semifactual Explanations

In this section, we evaluate SGRL-Advance and SGRL-Rewind approaches for generating semifactual explanations for RL introduced in Chapter 4.3. Firstly, Section 5.3.1 introduces the evaluation baselines. Section 5.3.2 details the research hypotheses and metrics, and Section 5.3.3 outlines the experiment design. Finally, the results are presented in Section 5.3.4.

5.3.1 Baselines

Currently, semifactuals have only been generated for supervised learning, and there are no methods for generating semifactual explanations for RL tasks. For this reason, we turn to the methods developed for supervised learning as baselines for the SGRL approach. We focus on counterfactually-free methods, as these do not require a counterfactual to be generated first, and are thus more flexible. As a baseline for semifactual generation, we choose a state-of-the-art approach, S-GEN [Kenny and Huang, 2024], as its algorithm is easy to adapt to RL and its open-source code enables benchmarking.

S-GEN

S-GEN [Kenny and Huang, 2024] is a semifactual generation approach that aims to help users optimise their decisions. It was originally developed and evaluated in supervised learning tasks. However, the approach is flexible as it requires only a training dataset and no information about the black-box policy it is explaining, making it easy to adapt to the RL framework.

To adapt S-GEN to RL, we only need a dataset of state-action pairs to serve as a representation of the training dataset. The data set of factual instances and actions chosen

in them by the RL policy being explained is used to represent the training data set. In this work, we implement and use three versions of the S-GEN algorithm depending on the diversity parameter – S-GEN1, S-GEN3 and S-GEN5, which generate one, three, and five semifactual explanations, respectively. A genetic algorithm generates and evaluates semifactual explanations based on the gain, robustness, plausibility, and diversity objectives.

The remainder of this section lists the evaluation hypothesis and metrics (Section 5.3.2), experiment design (Section 5.3.3), and presents results in Section 5.3.4.

5.3.2 Evaluation Hypotheses and Metrics

Our goal is to evaluate SGRL algorithms SGRL-Advance and SGRL-Rewind, and the baselines S-GEN1, S-GEN3 and S-GEN5 approaches on qualitative and quantitative evaluation metrics. The following sections detail the evaluation hypotheses and metrics.

Quantitative Evaluation Hypotheses

To evaluate SGRL and the baselines, we posit four quantitative evaluation hypotheses.

Firstly, our goal is to evaluate SGRL against RL-specific semifactual metrics defined in Section ?? *which ensures the semifactuals represent the most distant neighbour in the RL framework. Specifically, we compare SGRL-Advance and SGRL-Rewind with S-GEN1, S-GEN3, and S-GEN5 approaches on RL-specific semifactual properties validity, temporal distance, fidelity, stochastic uncertainty and exceptionality.*

Secondly, our goal is to evaluate whether the semifactual generation approaches generated realistic semifactual explanations. As S-GEN uses a genetic algorithm to generate a semifactual, it could produce semifactuals that are unrealistic or violate the causal constraints of the task. To that end, we evaluate SGRL-Advance and SGRL-Rewind and the baseline approaches S-GEN1, S-GEN3, and S-GEN5 on task-specific constraints. The same constraints used to evaluate whether a state is realistic for counterfactual explanations (Section 5.2.2) are used and can be found for each environment in Section 5.1.

Thirdly, our goal is to evaluate the diversity of explanations generated by SGRL algorithms and the baselines. Similar to counterfactual explanations, diversity of semifactuals is an important quality that ensures larger coverage of outcome causes and enables personalised explanations [Kenny and Huang, 2024]. To that end, we compare the explanations generated by SGRL-Advance and SGRL-Rewind to those generated by the baseline approaches S-GEN1, S-GEN3, and S-GEN5 based on two diversity metrics: 1) the number of generated counterfactuals per factual state and 2) pair-wise feature distance within the solution set. For a factual state x and a set X' of generated semifactual explanations, we measure pair-wise diversity as:

Table 5.10: A summary of quantitative evaluation goals and metrics for evaluating semifactual explanations presented in Section 5.3.2.

Hypothesis	Evaluation Goal	Evaluation Metrics
H1.1	Generating most distant world semifactuals	Validity Temporal Distance Fidelity Stochastic uncertainty Exceptionality
H1.2	Generating realistic semifactuals	Plausibility
H1.3	Generating diverse sets of semifactuals	Number of Explanations Feature Diversity
H1.4	Efficient generation of semifactuals	Coverage Generation Time

$$D(x, X') = \frac{1}{|X'|} \sum_{i \in \{1,\dots,|X'|\}} \sum_{\substack{j \in \{i+1,\dots,|X'|\} \\ j \neq i}} ||x'_i - x'_j||_2 \tag{5.3}$$

Finally, our goal is to evaluate the scalability of the SGRL approach compared to the baseline. We rely on two metrics: 1) coverage, measured as the percentage of factual queries for which a semifactual is generated, and 2) average semifactual generation time.

To summarize, we evaluate the SGRL approach on four hypotheses:

1. ***H1.1 [Semifactual Properties]*** *Semifactual explanations generated by SGRL-Advance and SGRL-Rewind will perform better on RL-specific metrics of validity, temporal distance, fidelity, stochastic uncertainty, and exceptionality compared to the baselines.*
2. ***H1.2 [Generating Realistic Explanations]*** *SGRL approaches SGRL-Advance and SGRL-Rewind will produce semifactual explanations that are realistic under the rules of the RL environment more often compared to the baselines.*
3. ***H1.3 [Diversity]*** *SGRL approaches SGRL-Advance and SGRL-Rewind will produce a more diverse set of semifactual explanations compared to the baselines.*
4. ***H1.4 [Scalability]*** *SGRL approaches SGRL-Advance and SGRL-Rewind will produce semifactual explanations in less time and achieve higher coverage compared to the baseline.*

An overview of quantitative evaluation goals and metrics is available in Table 5.10. In the next section, we introduce the qualitative evaluation hypotheses and metrics.

Table 5.11: A summary of qualitative evaluation goals and metrics for evaluating semifactual explanations presented in Section 5.3.2.

Hypothesis	Evaluation Goal	Evaluation Metrics
H1.1	Understanding Agent behaviour	Prediction accuracy
H1.2	User satisfaction	Useful Detailed Complete Satisfying Actionable Reliable Trustworthy

Qualitative Evaluation Hypotheses

Ultimately, semifactual explanations are intended to aid users in better understanding the decisions of black-box policies. For this reason, we evaluate the semifactual explanations with real users. Specifically, we compare explanations generated by SGRL-Advance and SGRL-Rewind with a baseline approach. For simplicity, we choose S-GEN1 as the baseline approach for the user study, as it only generates one semifactual explanation per factual state. The user study has two evaluation hypotheses:

1. ***H2.1 [Understanding Agent behaviour]:*** *SGRL-Advance and SGRL-Rewind approaches will produce semifactuals that can help users better understand and predict the behaviour of RL agents compared to the baseline.*
2. ***H2.2 [User Satisfaction]:*** *Semifactual explanations generated by SGRL-Advance and SGRL-Rewind will be perceived as more satisfactory by users compared to explanations generated by the baseline.*

The two hypotheses correspond to hypotheses H2.1 and H2.3 used to evaluate counterfactual explanations in Section 5.2.3. To evaluate semifactual explanations, however, we do not measure the user's ability to compare agents, as this has proven to be a difficult task for users in previous work [Huber et al., 2023], as well as our user study on counterfactual explanations.

To evaluate H2.1, we measure the user's accuracy to predict the agent's next action in a previously unseen state. To evaluate user satisfaction, we use explanation goodness metrics [Hoffman et al., 2018], which provide a 1-5 Likert scale for measuring explanations' usefulness, detailedness, completeness, user satisfaction, actionability and reliability. Additionally, we measure the user's perceived trustworthiness of the generated explanations. The user study design is detailed in Section 5.3.3, and the results are shown in Section 5.3.4. A summary of qualitative evaluation hypotheses and metrics is presented in Table 5.11.

5.3.3 Experiment Design

In this section, we detail the experiment design for evaluating semifactual explanations. Firstly, we describe the training process for RL policies, which semifactual explanations will explain. Furthermore, we provide information about factual state generation and the design of the user study used to evaluate qualitative hypotheses.

Training Black-box Policies

The goal of this section is to evaluate SGRL and S-GEN approaches on policies converged to behaviour in the environment. To that end, for each of the tasks we train a black-box policy π. Policy π is trained to a high standard and performs the task with high reward. For all tasks, we decide on a DQN model to train the policy. The same RL policies are used for semifactual and counterfactual explanations for policy π, and their training parameters are shown in Table 5.4.

Generating Factual Queries

With a trained policy π, our goal is to explain why an agent chose an action a in state x by answering the question: "Why was action a' not chosen?". For every action a, we collect a set of states where a is not chosen by the agent. As we aim to search for semifactuals in the agent's past, we only record states which are preceded by a state-action path of length k, where k is the horizon parameter used by the SGRL approach. We then search for semifactual states which reinforce the outcome of not choosing a.

For each factual state, we generate semifactual explanations using the SGRL-Advance, SGRL-Rewind, S-GEN1, S-GEN3, and S-GEN5 approaches. Examples of semifactual explanations generated in this way are shown in Appendix A2. The specific parameters used by these approaches are given in Table 5.12. In all environments, we set the horizon parameter for SGRL approaches $k = 5$. The horizon parameter is task-specific and should be chosen to be large enough that performing k actions can lead to a significant change in the agent's state, but small enough to enable a feasible search of the space of action sequences of length k. In this thesis, we find that in all environments, five actions are enough to substantially change the state (e.g. a plant can grow into a different stage in the farm environment, and an agent can change lanes and velocity in the highway driving).

The quantitative evaluation hypotheses H1.1, H1.2, H1.3 and H1.4 are evaluated on semifactual explanations generated in this way.

User-study Design

To evaluate qualitative hypotheses H2.1 and H2.2, we conducted a user study. Specifically, we compared explanations generated by SGRL-Advance, SGRL-Rewind, and S-GEN1. The study uses game examples from a stochastic gridworld task, as this task is easy to understand and requires no prior knowledge. A full description of the study is

Table 5.12: Semifactual generation algorithm parameters. The same parameters are used to generate semifactuals in all tasks.

Algorithm	Number of generations	Population size	Diversity parameter	Horizon
S-GEN1	25	24	1	NA
S-GEN3	25	24	3	NA
S-GEN5	25	24	5	NA
SGRL-Advance	25	24	NA	5
SGRL-Rewind	25	24	NA	5

available in Appendix A1.

We recruited 90 *participants from English-speaking countries (US, UK, Canada, Australia, Ireland, and New Zealand) through the Prolific platform. All participants were reimbursed for their time according to the Prolific payment policy. We divided the participants into 3 groups. The first group received explanations generated by SGRL-Advance, the second, explanations by SGRL-Rewind, and the third, by S-GEN1. Before the study, participants were introduced to the task and semifactual explanations. The study was divided into 3 parts – training, testing, and a user satisfaction survey. In the training part, users were shown examples of RL states and semifactual explanations showing why the agent did not choose actions CHOP and SHOOT in that state. We limited our examples to these two actions to reduce cognitive load, as they are most informative. Additionally, to reduce cognitive load, we offer users only one explanation chosen at random from a diverse set of explanations. Each user was shown 9 training examples during the training phase. In the testing phase, users are shown RL states and asked to predict the next action the agent will choose in the state without any explanations. The states shown to users during the training and testing phase were chosen by the HIGHLIGHTS algorithm [Amir and Amir, 2018] to ensure they capture the most important decisions. Finally, in the last stage of the study, the participants were asked to rate explanations on a 1-5 Likert scale based on the explanation goodness metrics [Hoffman et al., 2018].*

5.3.4 Evaluation Results and Analysis

In this section, we present the evaluation results for semifactual generation methods. The remainder of this section discusses the results of quantitative evaluation hypotheses H1.1, H1.2, H1.3 and H1.4, followed by the results of the user study evaluating qualitative hypotheses H2.1 and H2.2, as defined in Section 5.3.2.

Generating Semifactual Explanations for Optimal Policies

To evaluate H1.1 (Section 5.3.2), we evaluate RL-specific semifactual metrics validity, temporal distance, fidelity, stochastic uncertainty and exceptionality for generated semifactual explanations. The evaluation results are presented in Table 5.13.

Same as for evaluating GANterfactual-RL in Section 5.2.4, to evaluate these metrics on explanations generated by S-GEN, we need to find a sequence of actions leading to the semifactual. Same as discussed for counterfactual explanations, we run a genetic

algorithm to find a sequence of actions connecting the original to the semifactual state. In both frozen lake and stochastic gridworld, we find that the semifactual is reachable in less than 5% *of cases. For this reason, we include the results on the frozen lake and stochastic gridowrld, while we cannot generate farm and highway as their continuous feature spaces made the action search infeasible.*

In all environments, all approaches generate only valid explanations. This is because all approaches see validity as a hard constraint and only generate explanations which satisfy this metric.

On the temporal distance metric, SGRL-Rewind approach produces the best results in frozen lake ($TD = 0.48$*), stochastic gridworld (*$TD = 0.50$*) and farm (*$TD = 0.33$*) environments. SGRL-Rewind is followed by SGRL-Advance with temporal distance achieving* 0.67 *in the frozen lake,* 0.52 *in the stochastic gridworld and* 0.46 *in the farm environment. S-GEN approaches generate semifactuals that are, as discussed above, most often not reachable, resulting in a high average temporal distance.*

Results on the fidelity metric are low in all environments. The highest fidelity is achieved by SGRL-Rewind with 0.15 *in the frozen lake,* 0.21 *in the stochastic gridworld and* 0.12 *in the farm environment. Low values of fidelity are somewhat expected since reaching alternative states requires diverging from the agent's policy. For S-GEN approaches, fidelity is consistently low (*0.01*) as a consequence of low reachability.*

*SGRL-Advance achieves the highest results on stochastic uncertainty in frozen lake (*0.39*) and stochastic gridworld environments (*0.56*). In the farm environment, the highest stochastic uncertainty of* 0.11 *is achieved by the SGRL-Rewind. For S-GEN approaches stochastic uncertainty is low (*0.01*) as a consequence of low reachability.*

Exceptionality is highest in the frozen lake environment, where SGRL-Advance achieves $E = 0.85$ *and SGRL-Rewind* $E = 0.82$*. The high values of exceptionality indicate that on the way to the semifactual agent experienced unexpected state transitions. The frozen lake is followed by the farm environment, in which the SGRL-Advance achieves an exceptionality of* 0.19 *and SGRL-Rewind* 0.14*. In the stochastic gridworld, exceptionality is low (*0.01 *for SGRL-Advance and* 0.0 *for SGRL-Rewind), indicating no unexpected transitions on the path to the semifactual. Exceptionality for S-GEN approaches is consistently low (*0.01 *for frozen lake and* 0.02 *for stochastic gridworld) as a consequence of low reachability.*

In conclusion, SGRL-Advance and SGRL-Rewind perform similarly to each other and outperform the baseline S-GEN approaches on all RL-specific semifactual metrics, confirming H1.1.

Generating Realistic Explanations

To evaluate hypothesis H1.2 defined in Section 5.3.2, we measure the percentage of semifactuals that obey the task-specific constraints defined in Section 5.1. The results for are presented in Table 5.14.

Table 5.13: Evaluation results for semifactual properties for semifactual generation approaches S-GEN1, S-GEN3, S-GEN5, SGRL-Advance, SGRL-Rewind on the frozen lake, stochastic gridworld, farm and highway driving tasks.

Task	Algorithm	Semifactual Properties				
		Validity (=1.0)	Temporal distance (↓)	Fidelity (↑)	Stochastic uncertainty (↑)	Exceptionality (↑)
Frozen lake	S-GEN1	**1.0**	0.97	0.01	0.01	0.02
	S-GEN3	**1.0**	0.97	0.01	0.01	0.02
	S-GEN5	**1.0**	0.97	0.01	0.01	0.02
	SGRL-Advance	**1.0**	0.67	0.05	**0.39**	**0.85**
	SGRL-Rewind	**1.0**	**0.48**	**0.15**	0.37	0.82
Stochastic gridworld	S-GEN1	**1.0**	0.97	0.0	0.01	0.01
	S-GEN3	**1.0**	0.97	0.0	0.01	0.01
	S-GEN5	**1.0**	0.97	0.01	0.01	0.01
	SGRL-Advance	**1.0**	0.52	0.04	**0.56**	**0.01**
	SGRL-Rewind	**1.0**	**0.50**	**0.21**	0.40	0.0
Farm	SGRL-Advance	**1.0**	0.46	0.09	0.0	**0.19**
	SGRL-Rewind	**1.0**	**0.33**	**0.12**	**0.11**	0.14
Highway driving	SGRL-Advance	**1.0**	0.64	0.09	0.56	**0.0**
	SGRL-Rewind	**1.0**	**0.56**	**0.13**	**0.57**	**0.0**

Table 5.14: Generation of realistic instances, diversity and scalability evaluation results for semifactual generation methods S-GEN1, S-GEN3, S-GEN5, SGRL-Advance and SGRL-Rewind in the frozen lake, stochastic gridworld, farm and highway environments.

Task	Algorithm	Realistic instances (%) (↑)	Diversity		Scalability	
			Num. of expl. (↑)	Feature diversity (↑)	Coverage (↑)	Gen. time (s) (↓)
Frozen lake	S-GEN1	**100**	1	0.0	79.20	59.45
	S-GEN3	**100**	3	11.38	79.20	121.10
	S-GEN5	**100**	5	**11.69**	79.20	178.30
	SGRL-Advance	**100**	**17.43**	5.48	**100**	4.24
	SGRL-Rewind	**100**	12.02	4.92	**100**	**4.12**
Stochastic gridworld	S-GEN1	**100**	1	0.0	78.33	80.50
	S-GEN3	99.28	3	4.00	77.00	197.70
	S-GEN5	98.48	**5**	4.99	74.33	298.20
	SGRL-Advance	**100**	3.55	2.03	**100**	**8.44**
	SGRL-Rewind	**100**	3.07	**5.92**	**100**	9.08
Highway driving	S-GEN1	**100**	1	0.0	73.00	**55.98**
	S-GEN3	**100**	3	0.14	73.00	123.20
	S-GEN5	**100**	5	0.14	73.00	186.90
	SGRL-Advance	**100**	**7.61**	**0.74**	**100**	686.40
	SGRL-Rewind	**100**	7.38	0.73	**100**	685.20
Farm	S-GEN1	82.53	1	0.0	80.14	**19.16**
	S-GEN3	40.05	3	**178.30**	80.14	55.79
	S-GEN5	40.68	**5**	143.1	80.00	92.10
	SGRL-Advance	**100**	1.38	4.53	99.00	107.4
	SGRL-Rewind	**100**	1.45	2.02	**100**	84.16

We find that, in all environments, SGRL-Advance and SGRL-Rewind produce only realistic semifactual states. This is expected, as they search for semifactuals within the agent's execution and only generate semifactuals that are found in the environment. However, more surprisingly, the generative S-GEN approach also performs perfectly on this metric in frozen lake and highway environments. Specifically, S-GEN1, S-GEN3, and S-GEN5 produce only realistic semifactuals in the frozen lake task. Similarly, these approaches fare well in the stochastic gridworld task with over 98% *realistic semifactuals. However, the percentage of realistic explanations for S-GEN approaches is significantly lower for the more complex farm environment. Specifically, S-GEN1 produces realistic semifactuals for* 82.53%, *while the percentage of realistic semifactuals falls to* 40.05% *for S-GEN3 and* 40.68% *for S-GEN1.*

In summary, in all four environments, SGRL-Advance and SGRL-Rewind produce only realistic semifactuals, while the baselines fail to generate a realistic semifactual for each factual state, confirming H1.2.

Diversity Evaluation Results

To evaluate hypothesis H1.3 defined in Section 5.3.2, we measure diversity using two metrics: 1) the number of explanations generated per factual state and 2) pair-wise feature diversity within the semifactual set. The diversity evaluation results in all tasks are shown in Table 5.14.

In all environments, S-GEN1, S-GEN3, and S-GEN5 approaches produce 1, 3, and 5 explanations, respectively, as indicated by their diversity parameter. In the frozen lake environment, SGRL-Advance and SGRL-Rewind produce more than double the number of explanations of S-GEN5, with SGRL-Advance generating 17.43 *and SGRL-Rewind* 12.02 *explanations. In the stochastic gridworld, SGRL-Advance and SGRL-Rewind produce on average* 3.55 *and* 3.07 *explanations on average respectively, putting them between S-GEN3 and S-GEN5 approaches. In the farm environment, SGRL-Advance and SGRL-Rewind generate, on average, only a little over one semifactual explanation, putting them between S-GEN1 and S-GEN3 on the first diversity metric. A visualisation of the number of explanations generated by semifactual generation approaches is presented in Figure 5.5.*

*In the frozen lake environment, S-GEN5 produces the most feature-diverse explanations ($D = 11.69$), despite the lower number of explanations generated per factual state. In the stochastic gridworld, the highest feature diversity is achieved by RACCER-Rewind ($D = 5.92$), followed by S-GEN5 ($D = 4.99$) and S-GEN3 ($D = 4.00$). In the farm environment, S-GEN3 and S-GEN5 produce the highest feature diversity (*178.3 *for S-GEN3 and* 143.10 *for S-GEN5). However, this is a consequence of the S-GEN approach often generating semifactual explanations by changing the "day of the year" feature, which ranges between* 1 *and* 365*, thus generating explanations with high feature diversity that do not necessarily provide more information. SGRL, on the other hand, produces semifactuals by searching the space around the original instance using RL actions, meaning the explanations cannot differ substantially from the original instance in the "day of the year" feature.*

Diversity is an important property of semifactual explanations, which is necessary for a full understanding of different feature effects on the outcome and enables personalisation. The S-GEN approach directly optimizes the feature diversity objective, while SGRL-Advance and SGRL-Rewind enable diversity indirectly through the search algorithm. ***While SGRL approaches produce larger and more diverse semifactual explanation sets for smaller environments, in a complex highway driving task, S-GEN approaches with higher diversity parameters outperform SGRL, leading to only a partial confirmation of hypothesis H1.3.***

Scalability Evaluation Results

To evaluate hypothesis H1.4 defined in Section 5.3.2, we evaluate scalability using two measures: 1) coverage, indicating the percentage of factual queries for which a semifactual was successfully generated and 2) average semifactual generation time. The

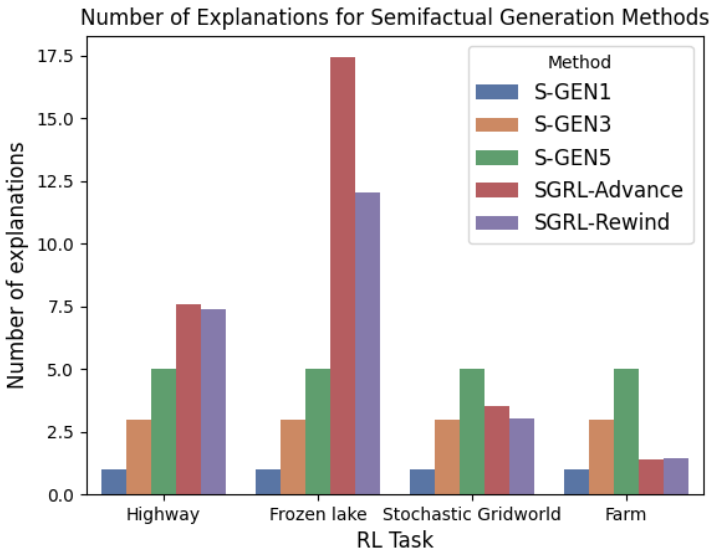


Figure 5.5: Number of explanations generated using semifactual generation methods in the frozen lake, stochastic gridworld, highway driving and farm environments.

scalability evaluation results can be found in Table 5.14.

In the frozen lake and stochastic gridworld environment, SGRL-Rewind and SGRL-Advance approaches generate a semifactual for each factual state. Similarly high coverage is achieved in the farm environment, with SGRL-Rewind generating a semifactual for each fact, and SGRL-Advance for 99% *of facts. S-GEN approaches, on the other hand, produce semifactuals for* 79% *of facts in a frozen lake environment and* 80.14% *of facts in the farm environment. In the stochastic gridworld, S-GEN1 generates semifactuals in* 78.33% *of cases, followed by S-GEN3 with coverage of* 77.00% *and S-GEN5 with coverage of* 74.33%*. A visualization of the coverage for semifactual generation approaches is available in Figure 5.6.*

In frozen lake and stochastic gridworld environments, SGRL-Advance and SGRL-Rewind generate the semifactual explanations on average the fastest. In the frozen lake, these approaches take around 4*s, compared to around one minute for S-GEN1, two minutes for S-GEN3 and three minutes for S-GEN5. In the stochastic gridworld, SGRL-Advance is the fastest at* 8.44 *seconds, followed by SGRL-Rewind at* 9.08*. On the other hand, S-GEN1 takes around* 10 *times longer at* 80.50*s, followed by S-GEN3 at* 197.70*s and S-GEN5 at* 298.20*s. In the farm environment, S-GEN approaches are faster compared to the SGRL approaches. S-GEN1 takes only around* 20*s, followed by S-GEN3 at* 55.79*s and S-GEN5 at* 92.10*. SGRL-Advance, on the other hand, takes on average* 107.40*s and SGRL-Rewind takes* 87.16*s per factual state. Same as for RACCER-based approaches discussed in Section 5.2.4, the average generation time of SGRL approaches depends primarily on the stochastic complexity in the environment (Table 5.1).*

SGRL approaches consistently produce semifactuals for a large number of queries across tasks, while coverage for S-GEN spans from 73% *in the highway environment to around* 80% *for the farm task. While SGRL-Advance and SGRL-Rewind boast faster execution in the simpler frozen lake and stochastic gridworld environments, S-GEN approaches*

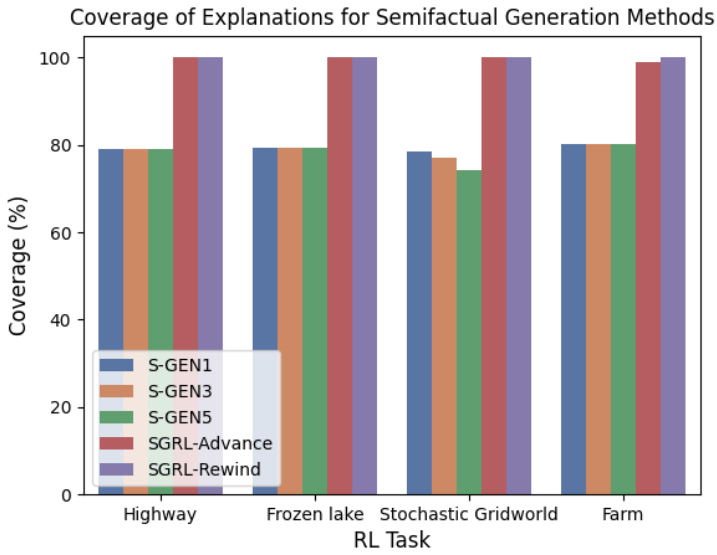


Figure 5.6: Coverage of semifactual generation algorithms in the frozen lake, stochastic gridworld, farm and highway environments.

scale better to larger environments such as farm and highway driving. Since SGRL approaches rely on estimating stochastic environment effects through simulations, their execution time increases with the complexity of the environment. ***Taking into account the better performance of SGRL approaches on coverage metrics, but also the increase in generation time in more complex tasks, we can only partially confirm H1.4.***

User Study Results

To evaluate qualitative hypotheses H2.1 and H2.2 defined in Section 5.3.2 we conducted a user study as described previously in Section 5.3.3. In this section, we outline the results gathered in this study.

Firstly, we measure the user's accuracy in predicting the agent's actions in previously unseen states, to investigate hypothesis H2.1. Users who have seen explanations generated by the baseline S-GEN1 were able to correctly predict the agent's actions in 68.62% *of cases. However, users who were presented with explanations generated by SGRL-Rewind achieved an accuracy of* 75.00%*, while users who had seen SGRL-Advance were correct in* 77.58% *of questions.* ***We performed a non-parametric one-tailed Mann-Whitney test and found no significant difference between the explanations, rejecting H2.1.***

Secondly, to evaluate hypothesis H2.2 we collect user scores on a 1-5 Likert scale on explanation goodness metrics [Hoffman et al., 2018]. The results are visualized in Figure 5.7 Regarding the perceived utility, all explanations received goodness scores above 2.5 *with no statistically significant difference between the three algorithms.* ***We find no significant difference between the explanation goodness metrics using the non-parametric one-tailed Mann-Whitney test and reject H2.2.***

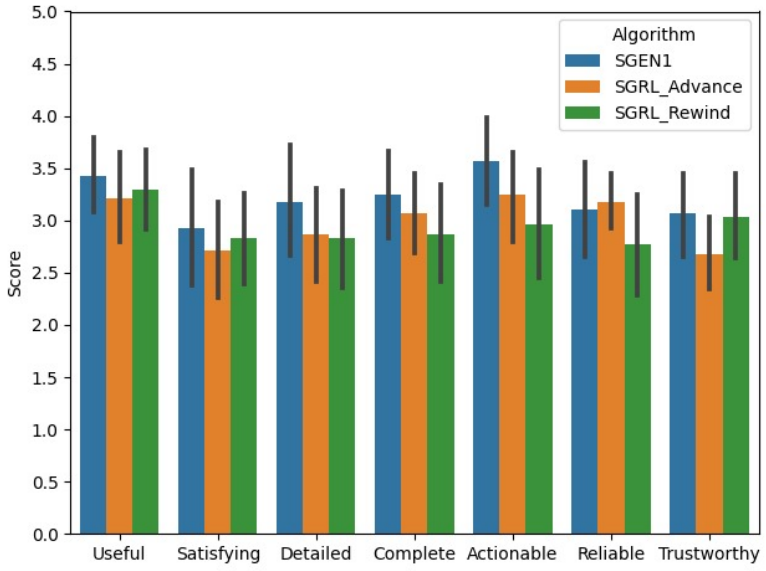


Figure 5.7: User satisfaction scores for semifactuals generated by the S-GEN1, SGRL-Advance and SGRL-Rewind approaches.

5.4 Chapter Summary and Contributions

In this chapter, we conducted an extensive evaluation of RACCER and SGRL algorithms implemented in Chapter 4.

To evaluate counterfactual explanations, we compared RACCER-HTS, RACCER-Rewind and RACCER-Advance approaches to the state-of-the-art method for counterfactual generation in RL GANterfactual-RL. We evaluated the approaches on four quantitative hypotheses and three qualitative metrics in four standard RL benchmarking tasks. To evaluate quantitative hypotheses we generated and compared over 500 *counterfactual explanations in each environment. To evaluate qualitative hypotheses we conducted a user study with over* 150 *participants.*

Our experiments confirmed all quantitative evaluation hypotheses. Specifically, we found that RACCER-HTS, RACCER-Advance and RACCER-Rewind outperform the GANterfactual-RL approach in all RL-specific counterfactual metrics. Additionally, we find that RACCER-HTS, RACCER-Rewind and RACCER-Advance consistently produce realistic, more diverse explanations compared to GANterfactual-RL. Finally, we found that RACCER-Rewind and RACCER-Advance algorithms based on evolutionary algorithms scale better to more complex environments and produce counterfactuals much faster than the tree-based RACCER-HTS.

Our user study confirmed one qualitative hypothesis, while one was rejected and one was partially confirmed. Specifically, The user study results showed that counterfactual explanations generated by RACCER-HTS help users better understand and predict the actions of an RL agent, confirming the first qualitative hypothesis. Additionally, users found these explanations more useful, detailed and complete compared to those generated by GANterfactual-RL, while there was no significant difference in other subjective metrics, partially confirming the third qualitative hypothesis. However, we find no sig-

nificant evidence that RACCER-HTS helps users better distinguish between two agents with different preferences, and reject the second qualitative hypothesis. We discuss why the agent comparison problem is so challenging for users and propose potential solutions in Chapter 7.

Our work is first to show the drawbacks of relying on the state-of-the-art approaches based on supervised learning in RL. We show that these approaches inherently cannot guarantee realistic instance generation, resulting in misleading counterfactuals and lower user performance and satisfaction. An emerging contribution of this chapter is demonstrating the insufficiency of these widely accepted explanation techniques for explaining agent behaviour in stochastic and sequential RL tasks.

To evaluate semifactual explanations, we compared SGRL-Rewind and SGRL-Advance approaches to the state-of-the-art approach S-GEN, adapted from supervised to RL. We evaluated the approaches on four quantitative and two qualitative hypotheses in four standard RL benchmark tasks. To evaluate quantitative hypotheses we generated and compared over 500 *semifactual explanations in each environment. To evaluate qualitative hypotheses we conducted a user study with over* 90 *participants.*

We find S-GEN to be a very competitive baseline, which is especially surprising as it was not developed specifically for RL framework. However, S-GEN explicitly accounts for certain properties utilised by SGRL approach, such as uncertainty and diversity. As a result, S-GEN3 and S-GEN5 approaches achieve competitive results on diversity metrics, outperforming SGRL in some environments. Unlike SGRL, S-GEN does not search for the semifactual in the agent's execution but rather "constructs" the semifactual state from features. This results in S-GEN outperforming SGRL in scalability, as it does not require running simulations and searching through agents' execution. However, this also leads to the generation of unrealistic explanations which do not conform to the rules of the environment.

Our user study results showed no significant difference between SGRL-Advance, SGRL-Rewind, and S-GEN approaches when it comes to users' ability to understand and predict agents' actions. Additionally, no significant evidence was found that users prefer explanations generated by any of the approaches over others.

We conclude that S-GEN is a competitive approach and a strong baseline for developing semifactuals for RL. Lower computational cost and diverse solutions make it an attractive choice for semifactual search. However, its reliance on feature permutation can result in unrealistic states.

6 Real-life Application Case Study: Bike-sharing System

In Chapter 4, we introduced RACCER and SGRL as algorithms for generating counterfactual and semifactual explanations in RL. We evaluated the algorithms on standard RL benchmarking tasks of increasing complexity, from a simple gridworld to a simulated driving task (Chapter 5). However, counterfactual and semifactual explanations are ultimately developed to assist users in real-life tasks. Adapting explanations to high-dimensional real-life tasks requires the explanation generation methods to scale to large state and action spaces, complex behaviours and highly stochastic conditions.

In this chapter, we demonstrate the use of counterfactual and semifactual explanations on a real-life simulated RL task, CitiBikes. CitiBikes environment consists of over 30 *state features and uses a multi-discrete action space with* 250 *individual actions, representing an increase in complexity compared to the standard RL benchmark in Section 5.1. The goal of this chapter is to evaluate how approaches introduced in this thesis scale to such a complex task and demonstrate the use of counterfactual and semifactual explanations for real-life tasks. The insights obtained from these explanations could serve as reassurance of the agent's capabilities or could indicate flaws in the agent's reasoning to developers or expert users modelling a bike-sharing system. Although we focus on a bike-sharing system in this work, insights from this study could be used to generate explanations for similar resource management tasks.*

In the remainder of this chapter, we describe the application and evaluation of counterfactual and semifactual generation methods to the CitiBikes task. Section 6.1 describes the CitiBikes environment, and Section 6.2 details the experiment design, including evaluation metrics. The evaluation results are presented and analysed in Section 6.3.

6.1 CitiBikes RL Task

CitiBikes [Jiang et al., 2020] is an RL environment based on the New York City bike share system. The CitiBikes task consists of a network of docking stations where bikes can be unlocked and returned. The demand for bikes or empty docking ports depends on multiple factors such as the station's location, time of day, or weather. Some stations are more popular, leading to overloading of bikes, while others might suffer from bike shortages. For this reason, pre-emptive bike repositioning is needed to adapt to the

Table 6.1: Training parameters and performance results for the PPO policy π on the CitiBikes task.

Training Parameters	Architecture	Number of layers	2
		Nodes in each layer	[1028, 1028]
	Learning rate		1e-4
	Training timesteps		1e6
Performance	Average reward		-109.33
	Reward standard deviation		1.71

dynamic and stochastic bike demand.

We choose the bike repositioning problem as it has been extensively investigated in RL, and has been used to develop, evaluate, and benchmark complex RL algorithms [Li et al., 2018, Xiao, 2018, Pan et al., 2019, Seo et al., 2022, Liang et al., 2024]. Additionally, this problem is sequential and stochastic, and bike repositioning actions can be influenced by the conditions of the past or the future, lending itself naturally to counterfactual and semifactual explanations. For example, an agent explaining why it decided to transfer N bikes from the city centre station to the park could explain its decision counterfactually: "Had the temperature been higher I would have transferred more bikes to the park", or semifactually "Even if there was a higher demand in the city centre, I would still transfer bikes to the park". In this way, the agent can explain that it expects the demand for bikes in the park to increase and prioritises stocking for that station.

In this work, we use a CitiBike topology consisting of five docking stations. Station S1 is a self-balancing station with approximately the same number of bikes loaned and returned to this station. Stations S2 and S5 receive more returned bikes, and stations S3 and S4 suffer a higher demand and bike shortages.

An RL agent is tasked with transferring bikes between stations to satisfy demand and prevent bike shortages. At each timestep, the agent can observe information about each of the stations, such as the number of available bikes and empty docking places, as well as the general environment features such as weather and temperature. At each timestep, the agent can transfer up to 10 bikes from one station to another. An RL action is thus multi-discrete and consists of three components: the ID of a station sending the bikes, the ID of the station receiving the bikes, and the number of bikes transferred. The agent's reward in each timestep is based on: 1) bike shortages across all stations and 2) how many bikes were transferred in that timestep. Each bike shortage is attributed a negative reward, and transferring a bike between stations incurs a small penalty as well. If action a leading from state s to s' decides to transfer n bikes from one station to another, the following reward is issued:

$$R(s, a, s') = -\frac{bike\_shortage}{trip\_requirement} - 0.001 \cdot n \tag{6.1}$$

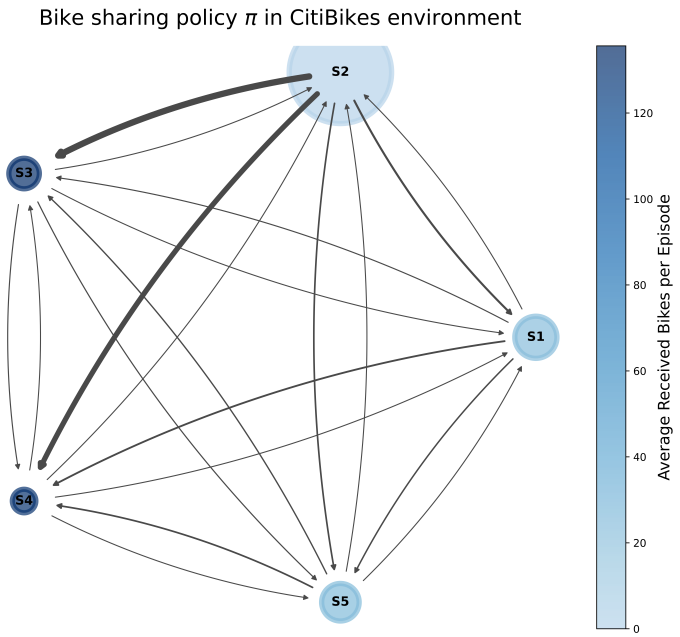


Figure 6.1: CitiBikes Repositioning Policy π: The repositioning of bikes among the five stations under policy π over 100 episodes. The size of the nodes indicates the number of sent bikes, while the colour indicates the received bikes. Stations $S3$ and $S4$ are the biggest receivers, with most of the bikes transferred from station $S2$.

6.2 Experiment Design

In this section, we detail the experiment design for evaluating counterfactual and semi-factual generation methods on the CitiBikes task. Firstly, in Section 6.2.1 we describe the training of a black-box RL policy in the CitiBikes tasks. Section 6.2.2 details the process of selecting factual states, and Section 6.2.3 summarises the approaches evaluated in this chapter. Finally, evaluation metrics are introduced in Section 6.2.4.

6.2.1 RL Policy Training

To generate explanations for CitiBikes, we start by performing the standard RL policy training process, resulting in policy π. We use the PPO algorithm [Schulman et al., 2017]. PPO is a powerful RL algorithm which uses gradient descent to perform robust updates to the policy. We choose PPO as it can be easily adapted for tasks with multi-discrete action spaces, and due to its success in learning behaviour in highly stochastic, complex tasks [Del Rio et al., 2024]. The training parameters and performance of the policy π are given in Table 6.1. Note that our goal is only to obtain a sensible policy that can be explained by counterfactual and semifactual methods, rather than to achieve state-of-the-art performance on CitiBikes tasks. Figure 6.1 illustrates the bike transfer between stations managed by policy π over 100 episodes. Under policy π, stations $S3$ and $S4$ receive the most bikes, mostly from station $S2$, indicating that π has learned to account for the bike shortages in $S3$ and $S4$.

6.2.2 Generating Factual States

The action space of the CitiBikes environment features 250 *individual actions. The policy* π *does not utilise all of them with the same frequency. For this reason, we consider semifactual and counterfactual actions only the actions that frequently appear in the agent's execution. Explaining an infrequent action by using it as a semifactual or counterfactual action might be infeasible, as a similar state where the agent would choose that action might be extremely rare. To that end, we select the four most frequent actions to explain.*

Counterfactual and semifactual explanations are local explanations that focus on explaining one decision. To compare them, we select 100 *informative states and generate counterfactual and semifactual explanations using the algorithms summarised in Section 6.2.3. Inspired by the HIGHLIGHTS algorithm [Amir and Amir, 2018] for selecting important states from the agent's execution, we select those states in which there is a large difference between choosing two actions in the factual state. As we deal with multi-discrete actions, we choose states where there is a large difference probability of choosing different actions in at least one dimension.*

Finally, to generate counterfactual explanations, we also need to define a target action. For each factual state x*, where the agent chooses an action* $a, \pi(x) = a$*, we choose a counterfactual action* $a' \neq a$ *at random from the set of frequent actions. For each factual state* x *in which an action* a *is chosen, we generate a counterfactual explanation as a state* x' *in which an alternative action* $a' \neq a$ *is chosen* ($\pi(x') = a'$).

For semifactuals, however, we use a different formulation from the one used in Chapter 5. For each factual state x *where* $\pi(x) = a$*, we generate semifactual explanation as a state* x' *where* $\pi(x') = a$*. In this way, we demonstrate how semifactuals can be used to reaffirm a specific action choice rather than reaffirming a choice to avoid an action (as demonstrated in Chapter 5).*

6.2.3 Counterfactual and Semifactual Generation Approaches

To generate counterfactual and semifactual explanations for factual states defined in the previous section, we use the RACCER and SGRL approaches proposed in this thesis (Chapter 4), and the same baselines as we used to evaluate them in standard RL benchmarks in Chapter 5.

To generate counterfactual explanations in the CitiBikes task, we compare the following algorithms:

1. ***GANterfactual-RL [Huber et al., 2023]:*** *Similar as in Chapter 5, we use this approach as the state-of-the-art baseline for counterfactual generation in RL. We limit the GANterfacutal-RL approach to learning domain translation only for states in which the agent chooses one of the frequent actions defined in Section 6.2.2. In this way, we ensure GANterfactual-RL has enough data to learn domain translation between states where different actions are chosen.*

2. ***RACCER-Rewind:*** *As implemented in Section 4.2.3, we use RACCER-Rewind to generate backwards counterfactuals that explore changes in the agent's past.*

3. ***RACCER-Advance:*** *As implemented in Section 4.2.3, we use RACCER-Advance to generate forward counterfactuals that explore changes in the agent's future.*

We do not include the RACCER-HTS algorithm in this case study. In Chapter 5, we have shown that RACCER-HTS cannot scale to more complex farm and highway environments and takes a prohibitive amount of time to generate a counterfactual. Additionally, we have shown that RACCER-Advance and RACCER-Rewind produce more diverse counterfactuals of similar quality (in RL-specific counterfactual metrics) as RACCER-HTS, but in a fraction of the time. As CitiBikes is a more complex environment, both in the state and action space size than farm or highway environments, it is infeasible to run RACCER-HTS in this task. The results of evaluating counterfactual explanations are presented in Section 6.3.1.

To generate semifactual explanations in the CitiBikes task, we compare the following algorithms:

1. ***S-GEN1 [Kenny and Huang, 2024]:*** *We use the S-GEN as the state-of-the-art baseline for semifactual generation from supervised learning. Adapting it to the RL in the CitiBikes task requires only a dataset of agents' interactions in the environment, which we generate by collecting rollouts of* 100 *episodes of the black-box policy π in the environment. S-GEN1 denotes the S-GEN algorithm with the diversity parameter set to* 1*, resulting in only one semifactual explanation per factual state.*

2. ***S-GEN3 [Kenny and Huang, 2024]:*** *Adapted in the same way as S-GEN1, but generating three diverse semifactual explanations.*

3. ***S-GEN5 [Kenny and Huang, 2024]:*** *Adapted in the same way as S-GEN1, but generating five diverse semifactual explanations.*

4. ***SGRL-Rewind****: As implemented in Section 4.2.3, we use SGRL-Rewind to generate backward semifactuals that explore changes in the agent's past.*

5. ***SGRL-Advance:*** *As implemented in Section 4.3.3, we use SGRL-Advance to generate forward semifactuals that explore changes in the agent's future.*

The evaluation results for generating semifactual explanations in the CitiBikes task are presented in Section 6.3.2.

6.2.4 Evaluation Metrics

In this chapter, we perform quantitative evaluations of counterfactual and semifactual generation methods in the CitiBikes task. We use the same metrics used to evaluate RACCER and SGRL in Chapter 5. For completeness, we summarise the evaluation metrics here:

1. ***Counterfactual and Semifactual Metrics**: We evaluate explanations according to the RL-specific semifactual and counterfactual metrics introduced in this work. Specifically, to evaluate counterfactual explanations, we measure validity, temporal distance, fidelity and stochastic certainty as defined in Section 4.2.2. To evaluate semifactual explanations, we measure validity, temporal distance, fidelity, stochastic uncertainty and exceptionality as defined in Section 4.3.2.*

2. ***Generating Realistic Instances:** For the CitiBikes task, we define three constraints that need to be satisfied to ensure a state is realistic under the rules of the environment:*

   - *A bike station cannot store more bikes than its capacity indicates.*
   - *A bike station cannot fulfil more rides than its capacity.*
   - *The shortage of bikes at a station has to be lower than its capacity.*

   Any state that fails any of these constraints is considered unrealistic.

3. ***Diversity:** We utilise two metrics, 1) the number of generated explanations and 2) pair-wise feature diversity metrics. To evaluate the pair-wise feature diversity of an explanation set X', we measure:*

$$D(x, X') = \frac{1}{|X'|} \sum_{i \in \{1,\dots,|X'|\}} \sum_{\substack{j \in \{i+1,\dots,|X'|\} \\ j \neq i}} ||x'_i - x'_j||_2 \tag{6.2}$$

4. ***Scalability:** We measure the scalability of the approaches through 1) coverage: percentage of input facts for which an explanation was successfully generated, and 2) average explanation generation time.*

The following section presents the evaluation results for counterfactual and semifactual generation methods based on these metrics.

6.3 Results

In this section, we present the evaluation results for semifactual and counterfactual generation methods for explaining an agent tasked with a bike repositioning problem in the CitiBikes environment. Counterfactual explanations are evaluated in Section 6.3.1, and semifactuals are evaluated in Section 6.3.2.

6.3.1 Evaluating Counterfactual Explanations

In this section, we report the evaluation results of counterfactual generation methods on the CitiBikes environment. We evaluate RACCER-Advance and RACCER-Rewind approaches against the baseline GANterfactual-RL approach. We start by reporting results on RL-specific counterfactual metrics, followed by the approaches' ability to generate

realistic explanations. Furthermore, we evaluate the diversity of generated explanation sets and scalability in this highly complex environment.

RL-specific Counterfactual Metrics

Firstly, we evaluate how RACCER-Rewind, RACCER-Advance, and GANterfactual-RL approaches compare on the RL-specific counterfactual metrics validity, temporal context, fidelity and stochastic certainty defined in Section 4.2.2. We report the average values of counterfactual metrics for all approaches in the CitiBikes environment in Table 6.2.

Same as discussed before for the counterfactual explanation evaluation on RL benchmarks in Section 5.2.4, we do not include the counterfactual metric results for GANterfactual, as this approach does not generate a sequence of actions leading to the counterfactual explanation, which are required to evaluate counterfactual metrics. Due to the large state and action space, it is infeasible to search for an action sequence that connects the counterfactual and original state.

On the validity metric, all approaches show a perfect score ($= 1.0$*) and produce only valid counterfactuals. This is because all approaches consider validity a hard constraint and only produce counterfactuals that satisfy validity.*

On the temporal distance metrics, all approaches show the worst possible result ($= 1.0$*). This means that all methods generate only counterfactuals that are as distant as possible from the original state. In this work, we limited the search space to the horizon* $k = 5$*, meaning all generated counterfactuals can be reached in five RL actions. RACCER-Rewind and RACCER-Advance severely underperform on this metric compared to their performance on the benchmark tasks, where these methods could generate closer counterfactuals, at between two and three actions' distance.*

On the fidelity metrics, RACCER-Advance and RACCER-Rewind perform comparatively, while the GANterfactual-RL approach achieves the lowest fidelity possible result of 0.0*. This means that counterfactuals generated by GANterfactual-RL are not reachable by paths likely under the agent's execution. RACCER-Rewind achieves fidelity of* 0.24*, comparable to its results on this metric in the RL benchmarks. Similarly, RACCER-Advance reports fidelity of* 0.20*, in line with its results on the RL benchmark tasks.*

Similar to the temporal distance metric, on the stochastic certainty, all approaches produce the lowest possible value of 0.0*. This means that all generated counterfactuals are reachable by highly uncertain paths, and the counterfactual outcome is not robust to the stochastic conditions in the environment. RACCER-Rewind and RACCER-Advance severely underperform compared to their results on this metric in RL benchmark tasks, where stochastic certainty ranged between* 0.62 *(in the frozen lake task) to* 0.25 *(in the farm task) for RACCER-Rewind and between* 0.59 *(in the frozen lake task) to* 0.28 *(in the farm task) for RACCER-Advance.*

We find that, although RACCER-Advance and RACCER-Rewind approaches outperform GANterfactual-RL on the fidelity metric, they exhibit a drop in

Table 6.2: Evaluation results of GANterfactual-RL, RACCER-Advance, and RACCER-Rewind counterfactual generation methods for RL in the CitiBikes task.

		GANterfactual-RL	RACCER-Rewind	RACCER-Advance
Counterfactual metrics	Validity (=1.00)	**1.00**	**1.00**	**1.00**
	Temporal distance (↓)	NA	1.00	1.00
	Fidelity (↑)	NA	**0.24**	0.20
	Stochastic certainty (↑)	NA	0.00	0.00
Realistic explanations		39.00	**100**	**100**
Diversity	Num. of expl.	**1**	**1**	**1**
	Feature diversity	**0**	**0**	**0**
Scalability	Coverage	**100**	**100**	**100**
	Gen. time	**0.006**	125.8	129.9

performance on all RL-specific counterfactual metrics compared to their results on less complex RL benchmarks (Chapter 5). *The quality of the generated counterfactuals could be a consequence of the increased action space compared to the RL benchmarks in Chapter 5, with* 250 *in CitiBikes vs.* 7 *actions in the farm task, the largest action space in RL benchmarks. Namely, given the substantially larger number of actions, there is a smaller proportion of the potential counterfactual states in which an alternative action would be chosen. This could affect the optimisation process and lead to a decrease in the quality of generated counterfactuals. We discuss further this issue and propose potential solutions to the increased action space in Section 7.3.*

Generating Realistic Instances

To determine whether an RL state is realistic under the rules of the CitiBikes environment, we evaluate three constraints introduced in Section 6.2.4. We report the percentage of realistic counterfactuals generated by each approach in Table 6.2.

RL-based approaches RACCER-Rewin and RACCER-Advance produce only realistic instances, achieving the perfect result of 100% *on this metric. On the other hand, GANterfactual-RL produces realistic instances for* 39% *of factual states. As RACCER-Rewind and RACCER-Advance search for counterfactuals in the agent's execution, they can only offer real RL states as counterfactual explanations, ensuring they are realistic. On the other hand, GANterfactual-RL relies on generative modelling, where a counterfactual state is generated without considering the potential causal relationships between features, resulting in a lower percentage of realistic counterfactuals.*

These results are compatible with the evaluation on standard RL benchmarks in Chapter 5. Namely, RACCER-Rewind and RACCER-Advance have generated only realistic counterfactuals in all RL benchmarks. Similarly, in the stochastic gridworld task, GANterfactual-RL produces realistic explanations in as little as 32.17% *of cases.* ***We find that consistent with evaluation on RL-benchmarks, RACCER-Advance and RACCER-Rewind generate only realistic counterfactuals, while GANterfactual-RL produces a realistic counterfactual in*** 39% ***of the cases.***

Diversity

To evaluate diversity, we measure 1) the number of generated explanations per factual state and 2) the pair-wise feature distance between counterfactual explanations. We report the results for all approaches in Table 6.2.

All approaches generate only a single counterfactual per factual state. Consequently, pair-wise feature diversity achieved by these approaches is the lowest possible at 0*, as there is not a set of counterfactual explanations that can be used to calculate it. While generating a single counterfactual is a characteristic of the GANterfactual-RL approach, RACCER-Advance and RACCER-Rewind are developed to generate a set of multiple diverse explanations.'Compared to the evaluation on RL benchmarks, where RACCER-Advance and RACCER-Rewind produced between* 1.66 *(in the farm task) to* 9.98 *explanations (in frozen lake task) per factual state, this represents a significant decrease in diversity.* ***To conclude, we find that all approaches generate only one approach per factual state in the CitiBikes task, resulting in a lack of diversity in the solution sets for all approaches.***

Similar to the decrease in the quality of the counterfactuals generated in the CitiBikes task, we see a decrease in the diversity of generated explanations. We suspect that the same reason is behind the decrease in counterfactual quality and diversity scores in the CitiBikes, compared to the RL benchmarking environments. Namely, given the substantially larger action space of CitiBikes, a smaller proportion of the search space corresponds to states where the counterfactual action would be chosen. For this reason, it is more difficult to find high-quality, diverse explanations.

Scalability

To compare the scalability of the counterfactual search approaches to the highly complex CitiBikes environment, we measure 1) coverage and 2) average counterfactual generation time. We present the results in Table 6.2.

Firstly, all approaches generate a counterfactual for each factual state, resulting in a perfect coverage score. However, when it comes to time efficiency, there is a considerable difference between the baseline and RACCER approaches. The GANterfactual-RL approach requires substantial pre-training time (around 14h). However, once trained, it can produce counterfactuals in a fraction of a second. On the other hand, RACCER-Advance and RACCER-Rewind approaches require no pre-training, but they take substantially longer to produce a counterfactual as they need to traverse the agent's execution. Specifically, both RACCER-Rewind and RACCER-Advance take around two minutes to generate a counterfactual in the CitiBikes task. ***In conclusion, we find that all approaches generate a counterfactual for each factual state, with GANterfactual-RL producing explanations in substantially less time, given substantially longer pre-training time.***

The scalability results are consistent with the results achieved on the standard RL bench-

mark tasks in Chapter 5, where GANterfactual-RL takes a fraction of a second to generate explanations, while RACCER-Rewind and RACCER-Advance require as much as 492*s (in the highway task).*

6.3.2 Evaluating Semifactual Explanations

In this section, we report the evaluation results of semifactual generation methods on the CitiBikes environment. We evaluate SGRL-Advance and SGRL-Rewind approaches against the baseline S-GEN1, S-GEN3, and S-GEN5 approaches. We start by reporting results on RL-specific semifactual metrics, followed by the approaches' ability to generate realistic explanations. Furthermore, we evaluate the diversity of generated explanation sets and scalability in this highly complex environment.

RL-specific Semifactual Metrics

We evaluate how SGRL-Rewind, SGRL-Advance and the baseline S-GEN1, S-GEN3 and S-GEN5 approaches compare on the RL-specific semifactual metrics validity, temporal context, fidelity, stochastic uncertainty and exceptionality defined in Section 4.3.2. We report the average values of semifactual metrics for all approaches in the CitiBikes environment in Table 6.3.

Similarly, as discussed above, we cannot report the results for S-GEN approaches on semifactual metrics, as these approaches do not generate a sequence of actions leading to the semifactual.

Firstly, all approaches report the highest possible validity score ($= 1.0$*) and generate only semifactuals that are valid. This is because all of the approaches consider validity as a constraint and generate only semifactuals that satisfy this constraint.*

On the temporal distance metric, all approaches achieve the worst possible value of 1.0*. This means that each semifactual generated by these approaches is at a distance of horizon* $= 5$ *RL actions from the original state. This is the same performance as that of RACCER for counterfactual generation for CitiBikes in Section 6.3.1. These results are in contrast with the results on RL benchmarks, where the SGRL-Advance achieves temporal distance between* 0.46 *(in the farm task) and* 0.67 *(in the frozen lake task), and SGRL-Rewind between* 0.33 *(in the farm task) and* 0.50 *(in the stochastic gridworld task).*

SGRL-Rewind reports an average fidelity of 0.09*, while SGRL-Advance achieves a fidelity of* 0.96*. This is consistent with the results obtained in the standard RL benchmarks (Chapter 5), where these approaches report fidelity between* 0.21 *(for SGRL-Advance in stochastic gridworld) and* 0.04 *(for SGRL-Rewind in the stochastic gridworld environment).*

On the stochastic uncertainty metric, both SGRL-Rewind and SGRL-Advance report similar results at 0.61 *and* 0.63*, respectively. These results are similar to those reported in benchmark RL environments.*

Table 6.3: Evaluation results for semifactual explanations generated using the S-GEN1, S-GEN3, S-GEN5, SGRL-Advance and SGRL-Rewind approaches in the CitiBikes task.

		S-GEN1	S-GEN3	S-GEN5	SGRL-Rewind	SGRL-Advance
Semifactual metrics	Validity (= 1.0)	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Temporal distance (↓)	NA	NA	NA	**1.0**	**1.0**
	Fidelity (↑)	NA	NA	NA	**0.09**	0.04
	Stochastic uncertainty (↑)	NA	NA	NA	0.61	**0.63**
	Exceptionality (↑)	NA	NA	NA	**0.0**	**0.0**
Realistic explanations (%) (↑)		**100**	**100**	**100**	**100**	**100**
Diversity	Num. of expl. (↑)	1	3	5	3.35	3.31
	Feature diversity (↑)	0	**32.44**	32.41	2.66	2.14
Scalability	Coverage (↑)	**100**	**100**	**100**	64	93
	Gen. time (↓)	206.3	599.3	918.4	**132.00**	136.00

On the exceptionality metric, all approaches report the worst possible result of 0.0. *This means that the generated semifactuals are generated via paths where no unexpected state transitions occur. While this is in contrast with simpler benchmarking tasks like frozen lake and stochastic gridworld, it is more similar to the results obtained for more complex farm environments (for SGRL-Rewind, for SGRL-Advance).*

We find that, although SGRL-Advance and SGRL-Rewind approaches outperform the baselines on the fidelity and stochastic uncertainty metrics, they underperform severely on all RL-specific semifactual metrics compared to their results on less complex RL benchmarks (Section 5). *Similar to the decrease in counterfactual quality for CitiBikes reported in the previous section, we find a substantial decrease in semifactual quality in this evaluation. Specifically, SGRL-Advance and SGRL-Rewind underperform on temporal distance and exceptionality metrics, compared to their performance on simpler RL benchmarking environments. Same as for counterfactuals, we suspect that the increased action space of this task (with* 250 *actions, compared to at most* 7 *in standard benchmarks) led to the decrease in explanation quality. With larger action space, a smaller proportion of potential semifactual states in which a specific action is chosen decreases, making optimisation challenging, and leading to suboptimal semifactuals.*

Generating Realistic Instances

To determine whether an RL state is realistic under the rules of the CitiBikes environment, we evaluate three constraints introduced in Section 6.2.4. We report the percentage of realistic semifactuals generated by each approach in Table 6.2.

Firstly, we find that SGRL-Advance and SGRL-Rewind produce only realistic semifactuals. This is in line with the results reported on standard RL benchmarks (Chapter 5), where these approaches achieve a perfect score of 100%*on realistic instances. Additionally, the baseline approaches also produce only realistic semifactuals, despite not directly acknowledging the causal connections between features. The percentage of realistic semifactuals of S-GEN is comparable to that achieved in frozen lake (*100%*), stochastic gridworld (over* 98% *) and the farm environments (*100%*), and higher than that achieved for the highway task (around* 60%*). ,xtbfln conclusion, we find that all approaches generate only realistic counterfactual explanations for each factual state.*

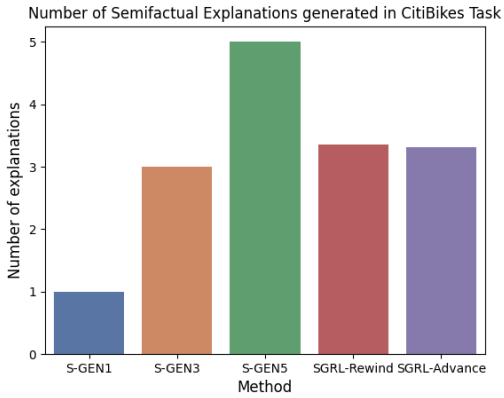


Figure 6.2: Number of semifactual explanations generated by S-GEN1, S-GEN3, S-GEN5, SGRL-Advance and SGRL-Rewind approaches in CitiBikes task.

Diversity

To evaluate diversity, we measure 1) the number of generated explanations per factual state and 2) the pair-wise feature distance between semifactual explanations. We report the results for all approaches in Table 6.3.

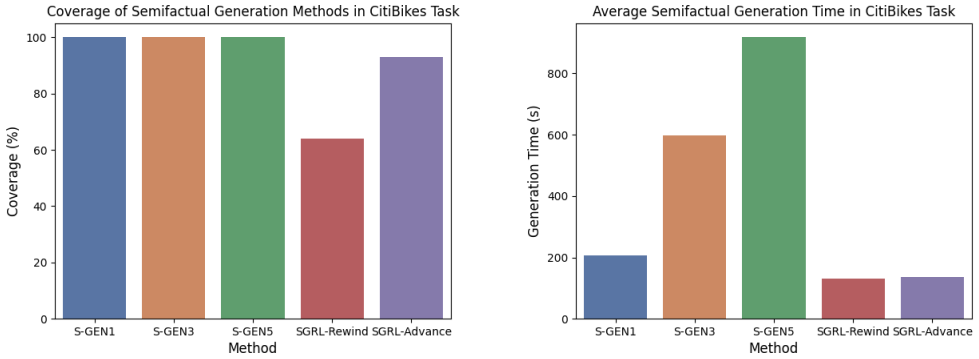
Firstly, we note that S-GEN1, S-GEN3 and S-GEN5 approaches generate one, three and five semifactual explanations respectively, as indicated by their diversity parameter. SGRL-Advance and SGRL-Rewind score somewhere between S-GEN3 and S-GEN5 on the number of explanations, with SGRL-Rewind generating on average 3.35*, and SGRL-Advance* 3.31 *semifactuals per factual state. The average number of explanations for all semifactual generation algorithms is visualised in Figure 6.2.*

On the pair-wise feature diversity metric, the best performance is achieved by S-GEN3 with a score of 33.44*, followed by S-GEN5 with a score of* 32.41*. SGRL-Rewind and SGRL-Advance perform substantially worse on this metric compared to S-GEN3 and S-GEN5, with SGRL-Rewind scoring* 2.66 *and SGRL-Advance* 2.14*. These results are similar to those reported for the frozen lake, where despite SGRL-Advance and SGRL-Rewind generating over* 15 *explanations per factual state, S-GEN3 and S-GEN5 achieved better pair-wise feature diversity in the explanation set. We showcase the number of explanations for different algorithms in Figure 6.2.* ***To conclude, we find that, despite SGRL-Advance and SGRL-Rewind producing around*** 3 ***semifactuals per factual state, S-GEN3 and S-GEN5 approaches report substantially higher feature diversity between the explanations.***

Scalability

To compare the scalability of the semifactual search approaches to the highly complex CitiBikes environment, we measure 1) coverage and 2) average semifactual generation time. We present the results in Table 6.3.

S-GEN approaches achieve perfect coverage and successfully generate a semifactual for each factual state. SGRL-Advance follows with 93% *coverage, while SGRL-Rewind produces a semifactual for only* 63% *of factual states. However, when it comes to average*



(a) Coverage of semifactual generation methods in the CitiBikes task.

(b) Average semifactual generation time in CitiBikes task.

Figure 6.3: Coverage and average generation time of S-GEN1, S-GEN3, S-GEN5, SGRL-Advance and SGRL-Rewind approaches on the CitiBikes task.

generation time, SGRL-Advance and SGRL-Rewind require, on average, around two minutes per factual state. S-GEN1, on the other hand, takes around three, S-GEN3 ten and S-GEN5 fifteen minutes per factual state. These results are similar to those reported on stochastic gridworld and frozen lake, where SGRL approaches provided semifactuals in less time (around 8s) compared to S-GEN (around 80s). However, they contradict the results achieved for farm and highway driving tasks, where SGRL approaches took substantially longer compared to S-GEN approaches to generate a semifactual. We visualise the scalability results in Figure 6.3. ***To conclude, we find that while S-GEN approaches achieve higher coverage, SGRL-Advance and SGRL-Rewind can generate semifactuals in a fraction of the time needed for S-GEN.***

6.4 Chapter Summary and Discussion

In this chapter, we evaluated counterfactual and semifactual generation approaches introduced in this thesis on a real-life simulated CitiBikes task. The CitiBikes task is a highly stochastic task containing over 30 state features and with a multi-discrete action space of over 250 individual actions.

We evaluate RACCER and SGRL against the appropriate baselines on the number of quantitative metrics, such as RL-specific counterfactual and semifactual metrics, percentage of realistic instances, diversity and scalability. We have found that, despite performing well on standard RL benchmark environments (as shown in Chapter 5), RACCER and SGRL approaches struggle with generating high-quality explanations in the highly dimensional CitiBikes task with a large action space.

We find that, on average, counterfactual and semifactual explanations in the CitiBikes task underperform on RL-specific metrics compared to the explanations generated for the standard RL benchmark. For semifactual explanations, the coverage of SGRL approaches suffers under the large action space and leads to the SGRL-Rewind approach generating semifactuals for only 64% of factual states. For counterfactual explanations, RACCER successfully scales to the CitiBikes task when observing the coverage metric, producing a

counterfactual for each state. However, the diversity of RACCER-Rewind and RACCER-Advance suffers as they only produce one counterfactual per factual state in the CitiBikes task. However, RACCER and SGRL still consistently deliver only realistic explanations in CitiBikes.

Counterfactual and semifactual explanations are ultimately intended to help users understand complex decisions in highly dimensional, large action space tasks. While RACCER and SGRL offer the first steps in producing RL-specific, realistic explanations, the scalability issue might prevent their successful integration in large tasks such as CitiBikes. We discuss the challenges around scalability and propose potential future work directions to address it in Section 7.2.

7 Conclusion

In this thesis, we investigated the problem of defining and generating user-friendly counterfactual and semifactual explanations for RL tasks. Inspired by the wide applications they found in supervised learning, this thesis aimed to adapt counterfactual and semifactual explanations to sequential and stochastic RL tasks. This thesis lays the theoretical groundwork by defining and categorising these explanations for RL and makes a practical contribution by providing a suite of algorithms for their generation in RL. In the remainder of this chapter, we first summarise the contributions of this thesis (Section 7.1), discuss its limitations (Section 7.2) and conclude by identifying future research directions emerging from this thesis (Section 7.3).

7.1 Thesis Contributions

The goal of this thesis was to provide theoretical groundwork and practical approaches for utilising counterfactual and semifactual explanations in RL. Given their success in explaining decisions of supervised learning models, our goal was to adapt these user-friendly explanations from one-step non-stochastic supervised learning tasks to sequential and stochastic RL frameworks.

To address the problem of defining, evaluating and generating counterfactual and semifactual explanations in RL, we identified four research questions (Section 1.2). Research questions RQ1 and RQ2 investigated how counterfactual and semifactual explanations can be defined for RL tasks. Specifically, in RQ1, we questioned how the counterfactual concept for the closest possible world can be translated in sequential and stochastic environments. Similarly, RQ2 dealt with redefining the semifactual concept of the most distant neighbour to RL. We address the RQ1 and RQ2 in Chapter 3. To answer RQ1 and RQ2 we first conducted an in-depth survey of these explanations in supervised and RL (Section 3.1), and identified the 1) sequential execution, 2) stochastic conditions and 3) a wide array of explanation factors in RL as the main aspects that distinguish RL from supervised learning from the perspective of these explanations. Guided by the insights of this survey, we defined a set of five counterfactual requirements – validity, reachability, recency, plausibility and certainty of outcome which provide a high-level guideline for finding the closest possible world counterfactual in stochastic and sequential environments. (Section 3.5.2). Similarly, we defined a set of six semifactual requirements – validity, reachability, recency, plausibility, unexpectedness and argument strength, which provide

high-level guidelines for choosing the most distant neighbour semifactual in sequential and stochastic environments (Section 3.5.3).

Furthermore, in Section 3.5.3 we defined counterfactual and semifactual explanations, in accordance with the RL-specific counterfactual and semifactual requirements as defined above.

Research questions RQ3 and RQ4 (defined in Section 1.2) aim to provide algorithms to generate counterfactual and semifactual explanations in RL based on their novel RL-specific definitions. Specifically, RQ3 investigated how counterfactual search can be designed and implemented in RL, in accordance with the RL-specific counterfactual requirements defined above. RQ4 investigated how semifactual search can be designed and implemented in RL, in accordance with the RL-specific semifactual requirements. We addressed RQ3 and RQ4 in Chapter 4. To address RQ3, we proposed RACCER, the first RL-specific counterfactual generation algorithm in Section 3.5.2. RACCER implements four counterfactual metrics – validity, temporal distance, fidelity and stochastic certainty, which are used to evaluate RL-specific counterfactual requirements defined in Section 3.5.2. RACCER implements three algorithms: a naive RACCER-HTS based on heuristic tree search and a more efficient and flexible RACCER-Advance and RACCER-Rewind, based on an evolutionary algorithm. In Section 3.6.2, we also proposed SGRL, the first semifactual generation algorithm for RL. SGRL implements five semifactual metrics – validity, temporal distance, fidelity, stochastic uncertainty and exceptionality, which measure RL-specific semifactual requirements posited in Section 3.6.2. SGRL implements two algorithms, SGRL-Advance and SGRL-Rewind, which use an evolutionary algorithm to evaluate action sequences leading to potential semifactuals.

We evaluated RACCER and SGRL against state-of-the-art baselines in four standard RL benchmarks in Chapter 5. We found that RACCER and SGRL produce more realistic explanations compared to baselines. Additionally, we found that in simpler tasks, such as frozen lake and stochastic gridworld, the RACCER and SGRL approaches generate explanations more consistently and in less time. However, we found that in more complex environments, RACCER and SGRL struggle to replicate the coverage and generation time performance reported in simpler tasks.

Additionally, we evaluated counterfactual explanations generated by RACCER-HTS in a user study in Section 5.2.3. We compared them to explanations generated by a state-of-the-art baseline approach, GANterfactual-RL, and measured user understanding, user's ability to compare different agents and user satisfaction. We found that explanations generated by RACCER-HTS are perceived as more useful, detailed and complete and help users better understand and predict RL agents' actions. However, we found no significant proof that RACCER-HTS helps users better compare agents with different preferences. Furthermore, in Section 5.3.3 we conducted a user study to compare SGRL-Advance, SGRL-Rewind and a state-of-the-art semifactual generation approach S-GEN1. However, we find no significant difference in the user's ability to predict the agent's actions or in the user's perceived satisfaction with the explanations.

Additionally, one goal of this thesis was to evaluate how the approaches scale to highly complex real-life tasks. To that end, in Section 6 we conducted a case study on counterfactual and semifactual explanations in a real-life simulated CitiBikes task. We evaluate RACCER and SGRL approaches, alongside the accompanying baselines, on this highly complex environment with over 30 state features and 250 actions.

In conclusion, we summarise the answers to the four research questions as defined in Section 1.2 as follows:

RQ1: ***How can the counterfactual concept of the closest possible world be defined in stochastic and sequential RL environments?***

To adapt the concept of the closest possible world to RL, we focused on defining 1) what makes two states close and 2) what makes a state plausible in stochastic and sequential RL environments.

We proposed five counterfactual requirements – validity, reachability, recency, plausibility and certainty of outcome that describe the closest possible world in RL. Each of these requirements is designed to ensure closeness (to the original instance) and the possibility of a counterfactual state in stochastic and sequential environments. Reachability and recency ensure closeness by making sure the counterfactual and original state are connected through the underlying temporal connections in the environment and can be reached in as few as possible RL actions. Plausibility ensures the state is possible under the agent's policy, while the certainty of outcome measures the possibility of a counterfactual outcome under the stochastic conditions of the environment. Validity is the only requirement that does not depend on the sequential or stochastic conditions in the environment and serves as a constraint that ensures a counterfactual outcome is achieved in the counterfactual state.

RQ2: How can the semifactual concept of the most distant neighbor be defined in stochastic and sequential RL environments?

To adapt the concept of the most distant neighbour to RL, we define 1) what makes two states distant and 2) what makes a state plausible in stochastic and sequential environments. To define the most distant neighbour in stochastic and sequential RL environments, we defined six semifactual requirements that a semifactual explanation in RL should satisfy. Similar to the semifactual requirements discussed in the previous research question, we defined validity, reachability, recency and plausibility as important requirements. They have the same purpose as counterfactual explanations, ensuring that the semifactual outcome is achieved in the semifactual state and that the semifactual and original states are connected and can be reached in a few RL actions under the agent's policy. Additionally, we propose two further requirements – unexpectedness and argument strength. Unexpectedness requires semifactuals to be a product of an unexpected or even shocking event, to showcase the immutability of the outcome under the stochastic conditions. Argument strength refers to semifactuals showing situations in which an outcome change is likely, but the outcome persists, further strengthening

the outcome.

RQ3: How can we design and implement counterfactual search algorithms in stochastic and sequential RL environments?

In this work, we proposed RACCER, the first RL-specific approach to generating counterfactual explanations. RACCER implemented four counterfactual metrics – validity, temporal distance, fidelity and stochastic certainty, which evaluated the RL-specific counterfactual requirements introduced in this thesis.

To optimise these counterfactual metrics, we first proposed a heuristic tree search approach, RACCER-HTS. RACCER-HTS searched for a counterfactual in the agent's execution by building a tree of the agent's interaction and optimising a collapsed loss objective consisting of counterfactual metrics to find a suitable counterfactual. However, we identified three main drawbacks of RACCER-HTS: 1) scalability, 2) the ability to search only for forward counterfactuals and 3) the generation of only one counterfactual per factual query. To address these limitations, we implemented RACCER-Advance and RACCER-Rewind, which search for forward and backward counterfactuals, respectively, by using a genetic algorithm to evaluate and score action sequences leading to promising counterfactuals.

In our experiments, we have shown empirically that RACCER-Advance and RACCER-Rewind scale better to more complex environments and can generate counterfactuals in substantially less time than RACCER-HTS while performing similarly on counterfactual metrics. Additionally, we find that RACCER-Rewind and RACCER-Advance produce more diverse explanations compared to RACCER-HTS. When comparing RACCER-HTS, RACCER-Rewind and RACCER-Advance to GANterfactual-RL, a state-of-the-art baseline for counterfactual generation in RL, we also find that our approaches outperform GANterfactual-RL on the counterfactual metrics, the percentage of realistic instances generated and the diversity of explanations. In conclusion, we find that our RL-specific approaches outperform the current state-of-the-art counterfactual generation method, which is translated from supervised learning. One of the main contributions of this thesis is demonstrating the need for considering the stochastic and sequential nature of the RL framework for generating realistic, user-friendly counterfactual explanations.

RQ4: How can we design and implement semifactual search algorithms in stochastic and sequential RL environments?

To evaluate semifactual requirements for RL defined above, we defined five semifactual metrics – validity, temporal distance, fidelity, exceptionality and stochastic uncertainty. These metrics ensure that the semifactual is connected to the original instance in a few RL actions through the underlying temporal connections of the RL environment, likely under the agent's policy and favour semifactuals that are a product of unexpected stochastic events in the environment.

Inspired by the success of RACCER-Rewind and RACCER-Advance for counterfactual generation, we implement two semifactual generation algorithms, SGRL-Advance and

SGRL-Rewind, based on evolutionary algorithms to optimise these metrics.

Our experiments compare SGRL-Rewind and SGRL-Advance to the state-of-the-art semifactual generation method for supervised learning, S-GEN. However, we find that S-GEN provides a competitive baseline, despite being adapted from supervised learning. While SGRL approaches outperform S-GEN on scalability and diversity metrics in smaller toy environments, S-GEN generates semifactuals faster and with more diversity in higher-complexity tasks. Additionally, the user study finds no significant difference in users' ability to understand the agent's policy.

In conclusion, this thesis presented the first theoretical and practical steps towards utilising counterfactual and semifactual explanations in RL. The thesis provides a theoretical foundation for developing counterfactual and semifactual explanations in stochastic and sequential contexts and a taxonomy of explanation types elicited by a wide range of behavior motivators in RL. We provide a rich framework for future research to build on and develop a wide range of counterfactual and semifactual explanations for RL. Additionally, this thesis proposes RACCER, an algorithm for generating counterfactual explanations in RL, which outperforms the current state-of-the-art baseline and leads to higher user understanding and satisfaction in a user study. Finally, this thesis provides the first steps in utilising semifactual explanations in RL. However, the thesis finds that a current state-of-the-art approach from supervised learning provides a competitive baseline to the RL-specific algorithm SGRL.

While we have shown the potential of these explanations in user studies with real humans, there are still some limitations to these approaches. The following section identifies the main drawbacks of RACCER and SGRL. Additionally, there are a number of topics that have remained outside of the scope of this thesis and can be tackled in future research in this field. In Section 7.3, we list the research directions emerging from the work of this thesis.

7.2 Thesis Limitations

In this thesis, we offered the first RL-specific counterfactual search approach, RACCER, and the first semifactual generation approach for RL, SGRL. Through extensive evaluation in complex RL environments and by conducting two user studies, we have shown the potential of these approaches for generating counterfactual and semifactual explanations for RL. However, in this section, we cover some limitations of this work.

Explaining Continuous Outcomes

In this work, we developed and evaluated RACCER and SGRL for explaining discrete actions. Similarly, current work on generating counterfactual and semifactual explanations both in supervised and RL is limited to only discrete outcomes. This is because explaining continuous outcomes leads to an increase in the search space and makes it difficult to find a state where an exact continuous outcome is achieved. For example,

it might be difficult to answer a counterfactual question "Why was my salary not raised by $x \in \mathbb{R}$ per cent?" as finding another person with that exact salary increase is highly unlikely.

To enable RACCER and SGRL to explain continuous outcomes, firstly, we would need to redefine the notion of validity. For example, an explanation could be considered valid if it elicits an outcome that belongs to a range of values rather than being equal to a specific value. For example, to answer a question such as "Why was my salary not raised by 10.43%*?", a valid counterfactual could be any one that achieves a salary increase of* 10.43% *or higher.*

Additionally, explaining continuous outcomes comes with added complexity, which could affect the scalability of the approach. In this work, we have shown that an increase in the size of the agent's action space can lead to a decrease in scalability, as finding a state which elicits a specific action becomes more difficult. Many real-life RL problems, however, feature continuous action spaces. Future work in counterfactual and semi-factual generation is required to address the issues of scalability in continuous action spaces.

Scalability

Another drawback of the work is the scalability of the RACCER and SGRL approaches to highly stochastic environments with large action spaces. In Chapter 3, we defined reachability as an important requirement for counterfactual and semifactual explanations in RL, to ensure they are connected with the original instance through the underlying temporal state connections in the environment. This means that the counterfactual and semifactual states can be found by exploring the states surrounding the original one through RL actions. Searching for counterfactual and semifactual explanations in the agent's execution is the only way to ensure they satisfy the reachability property. However, this affects the scalability of the search as the action space and stochasticity of the environment increase. On the other hand, supervised learning approaches often search the training dataset for an explanation, and their scalability often depends only on the dataset size, rather than state or action space dimensions.

One potential improvement to the tree-based counterfactual search algorithm RACCER-HTS could be pruning the tree of the nodes that show suboptimal counterfactual metric scores. On the other hand, the genetic algorithm behind RACCER-Advance, RACCER-Rewind, SGRL-Advance and SGRL-Rewind approaches could be sped up by using a suitable initial population likely to contain counterfactual or semifactual explanations. For example, genetic search could be initialised by a population containing solutions of previously generated counterfactuals for a similar state. In this way, solutions to previous factual queries could be reused to kick-start the following calculations. Previous work on state clustering could be used to estimate state similarity for this purpose [Zahavy et al., 2016, McCalmon et al., 2022].

Estimating Stochastic Conditions

RL environments are often stochastic, and we defined counterfactual and semifactual explanations in accordance with this in this thesis. Counterfactual and semifactual metrics such as stochastic certainty, fidelity or exceptionality embody the stochasticity of the environment. In this thesis, we used simulations to approximate the stochastic conditions of the task. However, using simulations is time-consuming and applicable only to tasks where direct access to the environment is enabled.

Alternatively, stochastic conditions could be estimated from the model of the environment. This would enable explanation generation in cases when the environment is not readily available, or in offline scenarios where only the agent's transitions are available. To learn the model of the environment approaches from the field of model-based RL could be used [Moerland et al., 2023]. Additionally, this would reduce the explanation generation time as, once the model is obtained, stochastic conditions could be estimated using the environment's transition function, which encodes the stochasticity in the environment. On the other hand, using model-based RL approaches might introduce problems endemic to this field into the explanation generation process, such as model approximation errors or non-stationarity. Future research is needed to enable flexible and efficient estimation of stochastic conditions for the purpose of generating counterfactual and semifactual explanations.

7.3 Future Research Directions

In the remainder of this section, we outline the potential research directions which, although outside the scope of this thesis, could be explored by utilising counterfactual and semifactual explanations for RL.

Beyond Feature-based Explanations

To fully understand the behaviour of RL agents, we cannot rely only on feature-based explanations, as an agent's decisions are often influenced not only by the state features but also by its goals, objectives or outside events. In Chapter 3, we defined counterfactual and semifactual explanations to include a wider range of possible counterfactual variables present in RL tasks. However, in this work, we tackled only the problem of generating feature-based counterfactuals and semifactuals as the first step, utilising these explanations in RL.

Future research in the field of counterfactual and semifactual explanations needs to address the need for explanations that explain the outcome through the lens of the agent's goals, objectives, outside events or societal norms. Developing explanations that include these factors requires imagining alternative scenarios where they are different and estimating their distance from the original factor. For example, consider an autonomous vehicle making an unexpected turn because it had to change its goal of delivering the passenger to the railway station to a temporary goal of filling the tank due to low fuel.

A counterfactual explaining that if the car was going to the railway station, it would continue straight is more informative and reassuring for the user compared to one stating that it would have continued straight if it were driving to the airport. Understanding which explanation is more informative requires reasoning about the relationships between abstract concepts of goals, objectives, stochastic events and societal norms.

For goal-based explanations, the field of hierarchical RL might serve as a starting research point [Pateria et al., 2021]. Hierarchical RL splits the RL problem into a hierarchy of goals and subgoals which the agent needs to achieve. This hierarchy provides a structure to agents' goals and could allow comparison between them.

To generate counterfactual and semifactual explanations where the agent's objective is the variable, additional research is needed to understand what constitutes "close" or "plausible" reward functions in RL. Previous works on reward shaping [Ng et al., 1999], reward decompositions [Juozapaitis et al., 2019, Sukkerd et al., 2020] or reward function comparison [Gleave et al., 2020, Jenner and Gleave, 2022] could be used to provide structure to agents' objectives and allow comparison between them. These explanations could be particularly useful in multi-objective environments where an agent needs to balance between multiple, often conflicting objectives [Hayes et al., 2022].

To use stochastic events as counterfactual and semifactual variables, research is needed to compare these events. Current works in the fields of safe RL [Gu et al., 2022] could be a starting point for this research, as this field deals with understanding and ensuring robustness in the case of unexpected events. Similarly, current works in explaining unexpected events from the fields of event processing [Parra-Ullauri et al., 2022] or self-adapting systems [Ghanadbashi et al., 2024] could provide insights into detecting and comparing unexpected events.

Finally, explanations utilising social norms are not yet common in RL. However, the field of robotics and human-robot interaction could provide insights into how social norms could feature in counterfactual and semifactual explanations. Current work in human-robot interaction provides a plethora of research on managing and comparing social norms and explanations [Arnold et al., 2021, Papagni and Koeszegi, 2023] which could be adapted to RL tasks and used to generate counterfactual and semifactual explanations.

In this thesis, we have defined a broad spectrum of counterfactual and semifactual explanations that could be used to explain the behaviour of RL agents. However, given the novelty of the xRL field in general, further research is needed on implementing methods for their generation.

Explanation Diversity

Diversity is posited as one of the main requirements for counterfactual and semifactual explanations, necessary to ensure complete, detailed explanations applicable to users with different preferences [Mothilal et al., 2020, Laugel et al., 2023]. However, few works

enable diverse explanations, and even fewer evaluate whether this diversity enables better user understanding through user studies. A plethora of work on diverse explanations in xAI through the lens of psychology and social sciences could inform future work and inspire future user studies [Wang et al., 2019, Bove et al., 2023].

Additionally, a question arises on how diverse explanations should be presented to the end user. Dazeley et al. [2021c] argue that the process of explaining a black-box system should take the form of a conversation, with the user first being offered more rudimentary information, after which they can probe the system for more detailed explanations. Research in the field of human-in-the-loop RL deals with including user feedback in the RL development process and could be used as a starting point for researching how diverse explanations can be best utilised for end users. Similarly, to choose from a potentially large set of diverse explanations, previous work on integrating prior knowledge into explanations might be useful [Jeyasothy et al., 2022]. Users could then provide their preferences in advance, and these could be used to provide them with personalised explanations.

Explanations for RL Agent Comparison

In this work, we conducted user studies to investigate how counterfactual and semifactual explanations affect users' understanding of RL agents. Notably, we have found that even though counterfactuals generated by RACCER-HTS help users better understand and predict behaviour, they had no impact on users' ability to compare agents with different preferences. Our results reaffirm previous works on using local explanations to compare agents. For example, Huber et al. [2023] have found that although counterfactuals generated by the GANterfactual-RL approach help users better understand the agent compared to the approach by Olson et al. [2019], they do not aid in distinguishing between agents with different strategies in the Pacman task.

Future work should explore how counterfactual and semifactual explanations can be equipped to highlight the differences between agents of different preferences. We speculate that work in combining xRL explanation methods, such as that by Huber et al. [2021], could be useful in this task. Combining global and local explanations could enable one to understand an agent's general behaviour while focusing on its most important decisions.

Bibliography

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE access, *6:52138–52160, 2018.*

Mostafa Al-Emran. Hierarchical reinforcement learning: a survey. International journal of computing and digital systems, *4(02), 2015.*

Shuroug A Alowais, Sahar S Alghamdi, Nada Alsuhebany, Tariq Alqahtani, Abdulrahman I Alshaya, Sumaya N Almohareb, Atheer Aldairem, Mohammed Alrashed, Khalid Bin Saleh, Hisham A Badreldin, et al. Revolutionizing healthcare: the role of artificial intelligence in clinical practice. BMC medical education, *23(1):689, 2023.*

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. arXiv preprint arXiv:1806.08049, *2018.*

Dan Amir and Ofra Amir. Highlights: Summarizing agent behavior to people. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, *pages 1168–1176, 2018.*

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. arXiv preprint arXiv:1606.06565, *2016.*

Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. Explainable agents and robots: Results from a systematic literature review. In 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019, *pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems, 2019.*

Raghuram Mandyam Annasamy and Katia Sycara. Towards better interpretability in deep q-networks. In Proceedings of the AAAI conference on artificial intelligence, *volume 33, pages 4561–4569, 2019.*

Thomas Arnold, Daniel Kasenberg, and Matthias Scheutz. Explaining in time: Meeting interactive standards of explanation for robotic systems. ACM Transactions on Human-Robot Interaction (THRI), *10(3):1–23, 2021.*

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina,

Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information Fusion, *58: 82–115, 2020.*

André Artelt and Barbara Hammer. "even if..."–diverse semifactual explanations of reject. In 2022 IEEE Symposium Series on Computational Intelligence (SSCI), *pages 854–859. IEEE, 2022.*

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, *34(6): 26–38, 2017. doi: 10.1109/MSP.2017.2743240.*

Saugat Aryal and Mark T. Keane. Even if explanations: prior work, desiderata & benchmarks for semi-factual xai. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, *IJCAI '23, 2023. ISBN 978-1-956792-03-4. doi: 10.24963/ijcai.2023/732. URL https://doi.org/10.24963/ijcai.2023/732.*

Saugat Aryal and Mark T Keane. Even-ifs from if-onlys: Are the best semi-factual explanations found using counterfactuals as guides? arXiv preprint arXiv:2403.00980, *2024.*

Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpreting blackbox models via model extraction. arXiv preprint arXiv:1705.08504, *2017.*

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, *2(1):183–202, 2009.*

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In Proceedings of the European conference on computer vision (ECCV), *pages 456–473, 2018.*

Yanzhe Bekkemoen. Explainable reinforcement learning (xrl): a systematic literature review and taxonomy. Machine Learning, *113(1):355–441, 2024.*

Benjamin Beyret, Ali Shafti, and A Aldo Faisal. Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), *pages 5014–5019. IEEE, 2019.*

Emily Black, Zifan Wang, Matt Fredrikson, and Anupam Datta. Consistent counterfactuals for deep models. arXiv preprint arXiv:2110.03109, *2021.*

Clara Bove, Marie-Jeanne Lesot, Charles Albert Tijus, and Marcin Detyniecki. Investigating the intelligibility of plural counterfactual examples for non-expert users: an explanation user interface proposition and user study. In Proceedings of the 28th International Conference on Intelligent User Interfaces, *pages 188–203, 2023.*

Leo Breiman. Random forests. Machine learning, *45(1):5–32, 2001.*

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Dieter Brughmans, Pieter Leyman, and David Martens. Nice: an algorithm for nearest instance counterfactual explanations. Data mining and knowledge discovery, *38(5): 2665–2703, 2024.*

Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. Journal of Artificial Intelligence Research, *70:245–317, 2021.*

Ruth MJ Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In IJCAI, *pages 6276–6282, 2019.*

Ruth MJ Byrne and Suzanne M Egan. Counterfactual and prefactual conditionals. Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale, *58(2):113, 2004.*

Romain Cadario, Chiara Longoni, and Carey K Morewedge. Understanding, explaining, and utilizing medical artificial intelligence. Nature human behaviour, *5(12):1636–1642, 2021.*

Lenart Celar and Ruth MJ Byrne. How people reason with counterfactual and causal explanations for artificial intelligence decisions in familiar and unfamiliar domains. Memory & Cognition, *51(7):1481–1496, 2023.*

David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In Ijcai, *volume 91, pages 726–731, 1991.*

Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. Deep reinforcement learning in recommender systems: A survey and new perspectives. Knowledge-Based Systems, *264:110335, 2023.*

Ziheng Chen, Fabrizio Silvestri, Gabriele Tolomei, He Zhu, Jia Wang, and Hongshik Ahn. Relace: Reinforcement learning agent for counterfactual explanations of arbitrary predictive models. arXiv preprint arXiv:2110.11960, *2021.*

Furui Cheng, Yao Ming, and Huamin Qu. Dece: Decision explorer with counterfactual explanations for machine learning models. IEEE Transactions on Visualization and Computer Graphics, *27(2):1438–1447, 2020.*

Hao-Fei Cheng, Ruotong Wang, Zheng Zhang, Fiona O'connell, Terrance Gray, F Maxwell Harper, and Haiyi Zhu. Explaining decision-making algorithms through ui: Strategies to help non-expert stakeholders. In Proceedings of the 2019 chi conference on human factors in computing systems, *pages 1–12, 2019.*

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for

goal-oriented tasks. In Advances in Neural Information Processing Systems 36, New Orleans, LA, USA, *December 2023.*

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In Proceedings of the IEEE conference on computer vision and pattern recognition, *pages 8789–8797, 2018.*

Youri Coppens, Kyriakos Efthymiadis, Tom Lenaerts, Ann Nowé, Tim Miller, Rosina Weber, and Daniele Magazzeni. Distilling deep reinforcement learning policies in soft decision trees. In Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence, *pages 1–6, 2019.*

Lisa Cummins and Derek Bridge. Kleor: A knowledge lite approach to explanation oriented retrieval. Computing and Informatics, *25(2-3):173–193, 2006.*

Leonardo Lucio Custode and Giovanni Iacca. Interpretable pipelines with evolutionary optimized modules for reinforcement learning tasks with visual inputs. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, *pages 224–227, 2022.*

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. A survey of multilingual neural machine translation. ACM Computing Surveys (CSUR), *53(5):1–38, 2020.*

Xinyue Dai, Mark T Keane, Laurence Shalloo, Elodie Ruelle, and Ruth MJ Byrne. Counterfactual explanations for prediction and diagnosis in xai. In Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society, *pages 215–226, 2022.*

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In International Conference on Parallel Problem Solving from Nature, *pages 448–469. Springer, 2020.*

Richard Dazeley, Peter Vamplew, and Francisco Cruz. Explainable reinforcement learning for broad-xai: a conceptual framework and survey. arXiv preprint arXiv:2108.09003, *2021a.*

Richard Dazeley, Peter Vamplew, and Francisco Cruz. Explainable reinforcement learning for broad-xai: a conceptual framework and survey. arXiv preprint arXiv:2108.09003, *2021b.*

Richard Dazeley, Peter Vamplew, Cameron Foale, Charlotte Young, Sunil Aryal, and Francisco Cruz. Levels of explainable artificial intelligence for human-aligned conversational explanations. Artificial Intelligence, *299:103525, 2021c.*

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation, *6(2):182–197, 2002.*

Alberto Del Rio, David Jimenez, and Javier Serrano. Comparative analysis of a3c and ppo algorithms in reinforcement learning: A survey on general environments. IEEE Access, *2024.*

Eoin Delaney, Derek Greene, and Mark T Keane. Uncertainty estimation and out-of-distribution detection for counterfactual explanations: Pitfalls and solutions. arXiv preprint arXiv:2107.09734, *2021.*

Shripad Vilasrao Deshmukh, Arpan Dasgupta, Balaji Krishnamurthy, Nan Jiang, Chirag Agarwal, Georgios Theocharous, and Jayakumar Subramanian. Explaining rl decisions with trajectories. arXiv preprint arXiv:2305.04073, *2023.*

Ricardo Dominguez-Olmedo, Amir H Karimi, and Bernhard Schölkopf. On the adversarial robustness of causal algorithmic recourse. In International Conference on Machine Learning, *pages 5324–5342. PMLR, 2022.*

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608, *2017.*

Jeffrey Dustin. Amazon scraps secret AI recruiting tool that showed bias against women. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1M 2018. [Accessed 14-10-2024].

Manuel Eberhardinger, Johannes Maucher, and Setareh Maghsudi. Learning of generalizable and interpretable knowledge in grid-based reinforcement learning environments. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, *volume 19, pages 203–214, 2023.*

Donatella Ferrante, Vittorio Girotto, Marta Straga, and Clare Walsh. Improving the past and the future: a temporal asymmetry in hypothetical thinking. Journal of Experimental Psychology: General, *142(1):23, 2013.*

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, *pages 1189–1232, 2001.*

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. arXiv preprint arXiv:1711.09784, *2017.*

Jasmina Gajcin and Ivana Dusparic. Redefining counterfactual explanations for reinforcement learning: Overview, challenges and opportunities. ACM Computing Surveys, *56 (9):1–33, 2024.*

Jasmina Gajcin, Rahul Nair, Tejaswini Pedapati, Radu Marinescu, Elizabeth Daly, and Ivana Dusparic. Contrastive explanations for comparing preferences of reinforcement learning agents. arXiv preprint arXiv:2112.09462, *2021.*

Jasmina Gajcin, James McCarthy, Rahul Nair, Radu Marinescu, Elizabeth Daly, and Ivana Dusparic. Iterative reward shaping using human feedback for correcting reward misspecification. European Conference on Artificial Intelligence, *2023.*

Luíza Caetano Garaffa, Maik Basso, Andréa Aparecida Konzen, and Edison Pignaton de Freitas. Reinforcement learning for mobile robotics exploration: A survey. IEEE Transactions on Neural Networks and Learning Systems, *34(8):3796–3810, 2021.*

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research, *16(1):1437–1480, 2015.*

Romain Gautron, Odalric-Ambrym Maillard, Philippe Preux, Marc Corbeels, and Régis Sabbadin. Reinforcement learning for crop management support: Review, prospects and challenges. Computers and Electronics in Agriculture, *200:107182, 2022.*

Saeedeh Ghanadbashi, Zahra Safavifar, Farshad Taebi, and Fatemeh Golpayegani. Handling uncertainty in self-adaptive systems: an ontology-based reinforcement learning model. Journal of Reliable Intelligent Environments, *10(1):19–44, 2024.*

Carl Ginet. In defense of a non-causal account of reasons explanations. The Journal of Ethics, *12(3):229–237, 2008.*

Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. arXiv preprint arXiv:2006.13900, *2020.*

Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. journal of Computational and Graphical Statistics, *24(1):44–65, 2015.*

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". AI magazine, *38(3):50–57, 2017.*

Brandon M Greenwell, Bradley C Boehmke, and Andrew J McCarthy. A simple and effective model-based variable importance measure. arXiv preprint arXiv:1805.04755, *2018.*

Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In International Conference on Machine Learning, *pages 1792–1801. PMLR, 2018.*

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. arXiv preprint arXiv:2205.10330, *2022.*

Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. Data Mining and Knowledge Discovery, *38(5):2770–2824, 2024.*

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. ACM computing surveys (CSUR), *51(5):1–42, 2018.*

Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Factual and counterfactual explanations for black box decision making. IEEE Intelligent Systems, *34(6):14–23, 2019.*

Wei Guo and Peng Wei. Explainable deep reinforcement learning for aircraft separation assurance. In 2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC), *pages 1–10. IEEE, 2022.*

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. Autonomous Agents and Multi-Agent Systems, *36(1):26, 2022.*

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

Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. arXiv preprint arXiv:1812.04608, *2018.*

Robert R Hoffman, Shane T Mueller, Gary Klein, Mohammadreza Jalaeian, and Connor Tate. Explainable ai: roles and stakeholders, desirements and challenges. Frontiers in Computer Science, *5:1117848, 2023.*

Sandy H Huang, Kush Bhatia, Pieter Abbeel, and Anca D Dragan. Establishing appropriate trust via critical states. In 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), *pages 3929–3936. IEEE, 2018.*

Sandy H Huang, David Held, Pieter Abbeel, and Anca D Dragan. Enabling robots to communicate their objectives. Autonomous Robots, *43:309–326, 2019.*

Tobias Huber, Dominik Schiller, and Elisabeth André. Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In KI 2019: Advances in Artificial Intelligence: 42nd German Conference on AI, Kassel, Germany, September 23–26, 2019, Proceedings 42, *pages 188–202. Springer, 2019.*

Tobias Huber, Katharina Weitz, Elisabeth André, and Ofra Amir. Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps. Artificial Intelligence, *301:103571, 2021.*

Tobias Huber, Maximilian Demmler, Silvan Mertes, Matthew L Olson, and Elisabeth André. Ganterfactual-rl: Understanding reinforcement learning agents' strategies through visual counterfactual explanations. arXiv preprint arXiv:2302.12689, *2023.*

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. The International Journal of Robotics Research, *40(4-5):698–721, 2021.*

Erik Jenner and Adam Gleave. Preprocessing reward functions for interpretability. arXiv preprint arXiv:2203.13553, *2022.*

Adulam Jeyasothy, Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. Integrating prior knowledge in post-hoc explanations. In International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, *pages 707–719. Springer, 2022.*

Arthur Jiang, Jia Zhang, Pingchao Yu, Lyuchun Huang, Yang Qiu, Jinyu Wang, Wenlei Shi, Kaiqi Li, Zhanyu Wang, Chengruidong Zhang, Tianyi Sun, Miaoran Chen, Kuanwei Yu, Xinran Wei, Michael Li, Ning Shang, Qiwei Meng, Shan Li, Jiang Bian, Biao Cheng, and Tie-Yan Liu. Maro: A multi-agent resource optimization platform, 2020. URL `https://github.com/microsoft/maro`.

Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. arXiv preprint arXiv:1907.09615, *2019.*

Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In IJCAI/ECAI Workshop on Explainable Artificial Intelligence, *2019.*

Sergey Karakovskiy and Julian Togelius. The mario ai benchmark and competitions. IEEE Transactions on Computational Intelligence and AI in Games, *4(1):55–67, 2012.*

Eoin Kenny and Weipeng Huang. The utility of "even if" semifactual explanation to optimise positive outcomes. Advances in Neural Information Processing Systems, *36, 2024.*

Eoin M Kenny and Mark T Keane. On generating plausible counterfactual and semifactual explanations for deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, *volume 35, pages 11575–11585, 2021.*

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, *2021.*

Will Knight. What uber's fatal accident could mean for the autonomous-car industry, May 2018. URL `technologyreview.com/2018/03/19/241022/what-ubers-fatal-accident-could-mean-for-the-autonomous-car-industry`.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In European conference on machine learning, *pages 282–293. Springer, 2006.*

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, *25, 2012.*

Vivian Lai, Yiming Zhang, Chacha Chen, Q Vera Liao, and Chenhao Tan. Selective explanations: Leveraging human input to align explainable ai. Proceedings of the ACM on Human-Computer Interaction, *7(CSCW2):1–35, 2023.*

Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Interpretable & explorable approximations of black box models. arXiv preprint arXiv:1707.01154, *2017.*

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning. arXiv preprint arXiv:1712.08443, *2017.*

Thibault Laugel, Adulam Jeyasothy, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. Achieving diversity in counterfactual explanations: a review and discussion. In Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, *pages 1859–1869, 2023.*

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, *2018.*

Yexin Li, Yu Zheng, and Qiang Yang. Dynamic bike reposition: A spatio-temporal reinforcement learning approach. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, *pages 1724–1733, 2018.*

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In Proceedings of the fourth ACM international conference on AI in finance, *pages 374–382, 2023.*

Yuxi Li. Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274, *2017.*

Jiaqi Liang, Sanjay Dominik Jena, Defeng Liu, and Andrea Lodi. A reinforcement learning approach for dynamic rebalancing in bike-sharing system. arXiv preprint arXiv:2402.03589, *2024.*

Roman Liessner, Jan Dohmen, and Marco A Wiering. Explainable reinforcement learning for longitudinal control. In ICAART (2), *pages 874–881, 2021.*

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, *2015.*

Zachary C Lipton. The doctor just won't accept that! arXiv preprint arXiv:1711.08037, *2017.*

Chunming Liu, Xin Xu, and Dewen Hu. Multiobjective reinforcement learning: A comprehensive overview. IEEE Transactions on Systems, Man, and Cybernetics: Systems, *45(3):385–398, 2014.*

Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. Toward interpretable deep reinforcement learning with linear model u-trees. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, *pages 414–429. Springer, 2018.*

Xiaowei Liu, Kevin McAreavey, and Weiru Liu. Contrastive visual explanations for reinforcement learning via counterfactual rewards. In World Conference on Explainable Artificial Intelligence, *pages 72–87. Springer, 2023.*

Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, *pages 650–665. Springer, 2021.*

Scott Lundberg. A unified approach to interpreting model predictions. arXiv preprint arXiv:1705.07874, *2017.*

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In Proceedings of the AAAI Conference on Artificial Intelligence, *volume 34, pages 2493–2500, 2020.*

Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. arXiv preprint arXiv:1912.03277, *2019.*

Odalric-Ambrym Maillard, Timothée Mathieu, and Debabrota Basu. Farm-gym: A modular reinforcement learning platform for stochastic agronomic games. In Artificial Intelligence for Agriculture and Food Systems (AIAFS), *2023.*

Joe McCalmon, Thai Le, Sarra Alqahtani, and Dongwon Lee. Caps: Comprehensible abstract policy summaries for explaining reinforcement learning agents. In nt'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS), *2022.*

Rachel McCloy and Ruth MJ Byrne. Semifactual "even if" thinking. Thinking & reasoning, *8(1):41–67, 2002.*

Silvan Mertes, Tobias Huber, Katharina Weitz, Alexander Heimerl, and Elisabeth André. Ganterfactual—counterfactual explanations for medical non-experts using generative adversarial learning. Frontiers in artificial intelligence, *5:825565, 2022.*

Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. ACM Computing Surveys, *56(7): 1–36, 2024.*

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence, *267:1–38, 2019.*

Tim Miller, Piers Howe, and Liz Sonenberg. Explainable ai: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences. arXiv preprint arXiv:1712.00547, *2017.*

Saumitra Mishra, Sanghamitra Dutta, Jason Long, and Daniele Magazzeni. A survey on the robustness of feature importance and counterfactual explanations. arXiv preprint arXiv:2111.00358, *2021.*

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, *2013.*

Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. Foundations and Trends® in Machine Learning, *16(1):1–118, 2023.*

Sina Mohseni, Niloofar Zarei, and Eric D Ragan. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. ACM Transactions on Interactive Intelligent Systems (TiiS), *11(3-4):1–45, 2021.*

Christoph Molnar. Interpretable Machine Learning. *2019.*

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, *pages 607–617, 2020.*

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. Advances in neural information processing systems, *32, 2019.*

Dena F. Mujtaba and Nihar R. Mahapatra. Ethical considerations in ai-based recruitment. In 2019 IEEE International Symposium on Technology and Society (ISTAS), *pages 1–7, 2019. doi:* $10.1109/ISTAS48451.2019.8937920$.

Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice Van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. ACM Computing Surveys, *55(13s):1–42, 2023.*

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In Icml, *volume 99, pages 278–287, 1999.*

Conor Nugent, Dónal Doyle, and Pádraig Cunningham. Gaining insight through case-based explanation. Journal of Intelligent Information Systems, *32:267–295, 2009.*

Matthew L Olson, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual states for atari agents via generative deep learning. arXiv preprint arXiv:1909.12969, *2019.*

Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. IEEE transactions on neural networks and learning systems, *32(2):604–624, 2020.*

Rohan Paleja, Yaru Niu, Andrew Silva, Chace Ritchie, Sugju Choi, and Matthew Gombolay. Learning interpretable, high-performing policies for autonomous driving. arXiv preprint arXiv:2202.02352, *2022.*

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. arXiv preprint arXiv:2201.03544, *2022.*

Ling Pan, Qingpeng Cai, Zhixuan Fang, Pingzhong Tang, and Longbo Huang. A deep reinforcement learning framework for rebalancing dockless bike sharing systems. In Proceedings of the AAAI conference on artificial intelligence, *volume 33, pages 1393–1400, 2019.*

Guglielmo Papagni and Sabine Koeszegi. Explaining intentional and unintentional behavior: Social norms for explainable robots. In Social Robots in Social Institutions, *pages 416–425. IOS Press, 2023.*

Juan Marcelo Parra-Ullauri, Antonio García-Domínguez, Nelly Bencomo, Changgang Zheng, Chen Zhen, Juan Boubeta-Puig, Guadalupe Ortiz, and Shufan Yang. Event-driven temporal models for explanations-etemox: explaining reinforcement learning. Software and Systems Modeling, *21(3):1091–1113, 2022.*

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. ACM Computing Surveys (CSUR), *54(5):1–35, 2021.*

Judea Pearl and Dana Mackenzie. The book of why: the new science of cause and effect. *Basic books, 2018.*

Jaime Pizarroso, José Portela, and Antonio Muñoz. Neuralsens: sensitivity analysis of neural networks. arXiv preprint arXiv:2002.11423, *2020.*

Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. ACM computing surveys (CSUR), *51(5): 1–36, 2018.*

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: feasible and actionable counterfactual explanations. In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, *pages 344–350, 2020.*

Erika Puiutta and Eric Veith. Explainable reinforcement learning: A survey. In International cross-domain conference for machine learning and knowledge extraction, *pages 77–95. Springer, 2020.*

Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. arXiv preprint arXiv:1912.12191, *2019.*

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, *pages 1135–1144, 2016.*

Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. Reinforcement learning with explainability for traffic signal control. In 2019 IEEE intelligent transportation systems conference (ITSC), *pages 3567–3572. IEEE, 2019.*

Aaron M Roth, Nicholay Topin, Pooyan Jamshidi, and Manuela Veloso. Conservative q-improvement: Reinforcement learning for an interpretable decision-tree policy. arXiv preprint arXiv:1907.01180, *2019.*

Cynthia Rudin, Caroline Wang, and Beau Coker. Broader issues surrounding model transparency in criminal justice risk scoring. Harvard Data Science Review, *2(1), 2020.*

Waddah Saeed and Christian Omlin. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. Knowledge-Based Systems, *263:110273, 2023.*

Amir Samadi, Konstantinos Koufos, Kurt Debattista, and Mehrdad Dianati. Safe-rl: Saliency-aware counterfactual explainer for deep reinforcement learning policies. IEEE Robotics and Automation Letters, *2024.*

Robert-Florian Samoilescu, Arnaud Van Looveren, and Janis Klaise. Model-agnostic and scalable counterfactual explanations via reinforcement learning. arXiv preprint arXiv:2106.02597, *2021.*

Léo Saulières, Martin Cooper, and Florence Dupin de Saint-Cyr. Reinforcement learning explained via reinforcement learning: Towards explainable policies through predictive explanation. In 15th International Conference on Agents and Artificial Intelligence (ICAART 2023), *volume 2, pages 35–44. Scitepress, 2023.*

Michele Scardi and Lawrence W Harding Jr. Developing an empirical model of phytoplankton primary production: a neural network case study. Ecological modelling, *120 (2-3):213–223, 1999.*

Maximilian Schleich, Zixuan Geng, Yihong Zhang, and Dan Suciu. Geco: Quality counterfactual explanations in real time. arXiv preprint arXiv:2101.01292, *2021.*

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, *2017.*

Young-Hyun Seo, Dong-Kyu Kim, Seungmo Kang, Young-Ji Byon, and Seung-Young Kho. Rebalancing docked bicycle sharing system with approximate dynamic programming and reinforcement learning. Journal of Advanced Transportation, *2022 (1):2780711, 2022.*

Pedro Sequeira and Melinda Gervasio. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. Artificial Intelligence, *288:103367, 2020.*

Lloyd S Shapley. 17. A value for n-person games. *Princeton University Press, 2016.*

Shubham Sharma, Jette Henderson, and Joydeep Ghosh. Certifai: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. arXiv preprint arXiv:1905.07857, *2019.*

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815, *2017.*

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, *2013.*

Julian Skirzyński, Frederic Becker, and Falk Lieder. Automatic discovery of interpretable planning strategies. Machine Learning, *110:2641–2683, 2021.*

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. Knowledge and information systems, *41(3):647–665, 2014.*

Roykrong Sukkerd, Reid Simmons, and David Garlan. Tradeoff-focused contrastive explanation for mdp planning. In 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), *pages 1041–1048. IEEE, 2020.*

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. *MIT press, 2018.*

Nicholay Topin and Manuela Veloso. Generation of policy-level explanations for reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, *volume 33, pages 2514–2521, 2019.*

Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. Towards robust and reliable algorithmic recourse. Advances in Neural Information Processing Systems, *34:16926–16937, 2021.*

William TB Uther and Manuela M Veloso. The lumberjack algorithm for learning linked decision forests. In International Symposium on Abstraction, Reformulation, and Approximation, *pages 219–232. Springer, 2000.*

Jasper van der Waa, Marcel Robeer, Jurriaan van Diggelen, Matthieu Brinkhuis, and Mark Neerincx. Contrastive explanations with local foil trees. arXiv preprint arXiv:1806.07470, *2018a.*

Jasper van der Waa, Jurriaan van Diggelen, Karel van den Bosch, and Mark Neerincx. Contrastive explanations for reinforcement learning in terms of expected consequences. arXiv preprint arXiv:1807.08706, *2018b.*

Abhinav Verma. Verifiable and interpretable reinforcement learning through program synthesis. In Proceedings of the AAAI Conference on Artificial Intelligence, *volume 33, pages 9902–9903, 2019.*

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In International Conference on Machine Learning, *pages 5045–5054. PMLR, 2018.*

Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. arXiv preprint arXiv:2010.10596, *2020.*

Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan Hines, John Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. ACM Computing Surveys, *56(12):1–42, 2024.*

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, *575 (7782):350–354, 2019.*

George A Vouros. Explainable deep reinforcement learning: state of the art and challenges. ACM Computing Surveys, *55(5):1–39, 2022.*

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. Harv. JL & Tech., *31:841, 2017.*

Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. Designing theory-driven user-centric explainable ai. In Proceedings of the 2019 CHI conference on human factors in computing systems, *pages 1–15, 2019.*

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In International conference on machine learning, *pages 1995–2003. PMLR, 2016.*

Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, *8:279–292, 1992.*

Ian Xiao. A distributed reinforcement learning solution with knowledge transfer capability for a bike rebalancing problem. arXiv preprint arXiv:1810.04058, *2018.*

Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. arXiv preprint arXiv:1810.06394, *2018.*

Jingtao Yao, Nicholas Teng, Hean-Lee Poh, and Chew Lim Tan. Forecasting and analysis of marketing data using neural networks. J. Inf. Sci. Eng., *14(4):843–862, 1998.*

Mustafa Yildirim, Barkin Dagda, and Saber Fallah. Highwayllm: Decision-making and navigation in highway driving with rl-informed language model. arXiv preprint arXiv:2405.13547, *2024.*

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. ACM Computing Surveys (CSUR), *55(1):1–36, 2021.*

Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. In International Conference on Machine Learning, *pages 1899–1908. PMLR, 2016.*

Junzhe Zhang. Designing optimal dynamic treatment regimes: A causal reinforcement learning approach. In International conference on machine learning, *pages 11012–11022. PMLR, 2020.*

Qingyuan Zhao and Trevor Hastie. Causal interpretations of black-box models. Journal of Business & Economic Statistics, *39(1):272–281, 2021.*

Ziwei Zhao, David Leake, Xiaomeng Ye, and David J Crandall. Generating counterfactual images: Towards a c2c-vae approach. In ICCBR Workshops, *pages 189–194, 2022.*

Jianlong Zhou, Amir H Gandomi, Fang Chen, and Andreas Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. Electronics, *10(5):593, 2021.*

Nir Ben Zrihem, Tom Zahavy, and Shie Mannor. Visualizing dynamics: from t-sne to semi-mdps. arXiv preprint arXiv:1606.07112, *2016.*

A1 User Study Details

In this appendix, we present the detailed description of the user studies conducted in this thesis. Section A1.1 presents the user study for evaluating counterfactual explanations conducted in Section 5.2.3. Section A1.2 covers the user study that evaluated semifactual explanations, as described in Section 5.3.3.

A1.1 User study – Counterfactual Explanations

In this Section, we describe in detail the user study that was used to evaluate counterfactual explanations generated by RACCER-HTS and GANterfactual-RL, as described in Chapter 5. Full studies can be found at https://qrxhyre44mt.typeform.com/to/cpeLrWbZ (for RACCER-HTS) and https://qrxhyre44mt.typeform.com/to/PnJXcwRw (for GANterfactual-RL). We describe the study in four parts: introductory part (Section A1.1.1), which introduces users to the task and counterfactual explanations, agent understanding part (Section A1.1.2), which measures user understanding of the behaviour of two RL agents, agent comparison part (Section A1.1.3), measuring users' ability to compare agents and user satisfaction part (Section A1.1.4) where users are asked to rate explanations based on subjective metrics.

A1.1.1 Introduction

Firstly, users were given a brief description of the study and asked to provide consent. If users declined to provide consent they were directed to leave the study. The consent form contained the following text:

> *LEAD RESEARCHER: Jasmina Gajcin*
>
> *You are invited to participate in this study which aims to investigate the effect of different types of explanations on user understanding of AI systems. As AI systems often lack transparency, understanding how their decisions can be interpreted to their users is necessary to encourage human-AI collaboration and facilitate trust. In this research we aim to investigate how different types of explanations affect user's understanding and interaction with the AI system.*
>
> *During the survey you will be asked to predict the behavior of a player in a simple grid-world game. You will be given some, but not all information*

about the rules of the game. Each question will consist of a game state for which you will be asked to predict the action the player will choose in that state. To better understand the player's reasoning, you will also be given explanations of player's behavior. The study is expected to take around 15 minutes and involve 20 - 25 different questions. At the end of the study you will be asked to evaluate the provided explanations based on their properties such as usefulness on a 1- 5 scale. Upon completion of the study you will be compensated according to the payment policy of the Prolific platform.

The results from the study will be analyzed to investigate the effects of different types of explanations on user's understanding of AI systems. We plan to submit the results of study as a conference paper for AAMAS24 conference.

As participants are anonymously sourced through the Prolific platform, there is no conflict of interest during recruitment.

Individual results may be aggregated anonymously and research reported on aggregate results.

DECLARATION:

- I am 18 years or older and am competent to provide consent.

- I have read, or had read to me, a document providing information about this research and this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.

- I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.

- I understand that if I make illicit activities known, these will be reported to appropriate authorities.

- I understand that I may refuse to answer any question and that I may withdraw at any time without penalty, until the study is submitted.

- I understand that I cannot withdraw from the study after submitting as the study is fully anonymous.

- I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.

- I understand that my participation is fully anonymous and that no personal details about me will be recorded.

- I understand that if I or anyone in my family has a history of epilepsy then I am proceeding at my own risk.

- I understand that personal information about me, including the transfer of this personal information about me outside of the EU, will be protected in accordance with the General Data Protection Regulation.

- I have received a copy of this agreement.

After providing consent, user were directed to the introductory section of the study where they were shown information about the stochastic gridworld task. The following information was provided to the users:

In this survey you will be shown images from a simple gridworld game. The images will show situations from games played by various AI agents. You will then be asked to answer questions about AI agent's behavior.

Please read the game description below carefully. There will a short quiz to check if you understood the game correctly.

Goal:

In the game, the agent moves around a grid world and its goal is to shoot the dragon by firing an arrow at it. However, the path between the agent and the dragon can be obstructed by trees or walls.

Actions:

Agent can choose between actions UP, DOWN, LEFT, RIGHT, CHOP and SHOOT at every step. Actions UP, DOWN, LEFT, RIGHT move agent one square in the appropriate direction if possible. If that square is already occupied by another object or if it would lead agent to leave the board, the action has no effect and agent remains in its place. Action SHOOT only kills the dragon if the agent and dragon are in the same row or file, and all the squares between them are empty. Action CHOP can be used to chop down a tree or a wall if agent is directly next to it. While chopping down a tree is cheap, chopping down a wall is an extremely expensive action. Trees can regrow and walls can be rebuilt at each step with some probability (this is outside of agent's control).

Game End:

The game ends when agent successfully plays the SHOOT action and kills the dragon.

The information was accompanied by an example of a RL state in the stochastic gridworld shown in Figure A1.10.

After receiving this information about the stochastic gridworld environments, the users were given a four-question test to check their knowledge of the rules of this game. The users could try to answer each question as many times as they wanted, but could not progress to the remainder of the study before clearing all questions. The questions are shown in Figure A1.2.

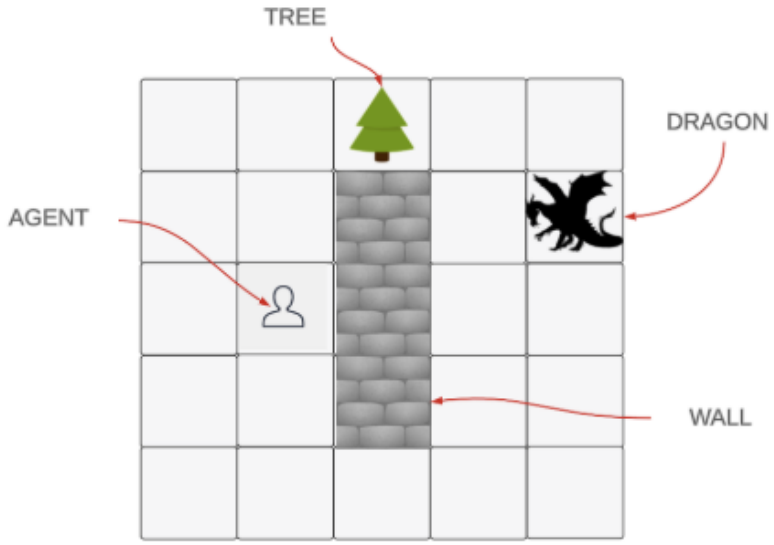
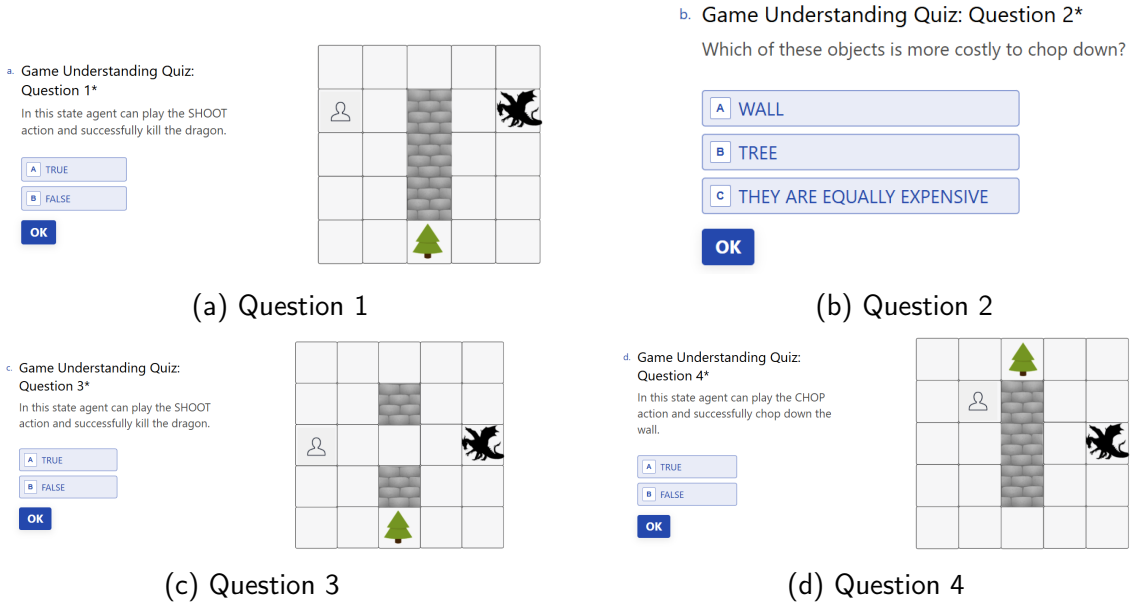


Figure A1.1: An image shown to users of the study to introduce them to the stochastic gridworld task.



(a) Question 1

(b) Question 2

(c) Question 3

(d) Question 4

Figure A1.2: Game understanding questions shown to users to verify their understanding of the rules of the stochastic gridworld game.

If the user makes a mistake on a question it is directed to the following message, and asked to try again:

> *Incorrect answer*
>
> *Please click continue to try again.*

Additionally, in the introductory part of the study, the users were demonstrated the use of counterfactual explanations. They were shown an textual definition of counterfactual explanations, as well as a visual representation of counterfactual explanations in a driving scenario. We chose this scenario as it previous user studies have used it to introduce users to counterfactual explanations in a domain that is familiar to most users [Huber et al., 2023]. Figure A1.3 shows the example counterfactual explanation given to users.

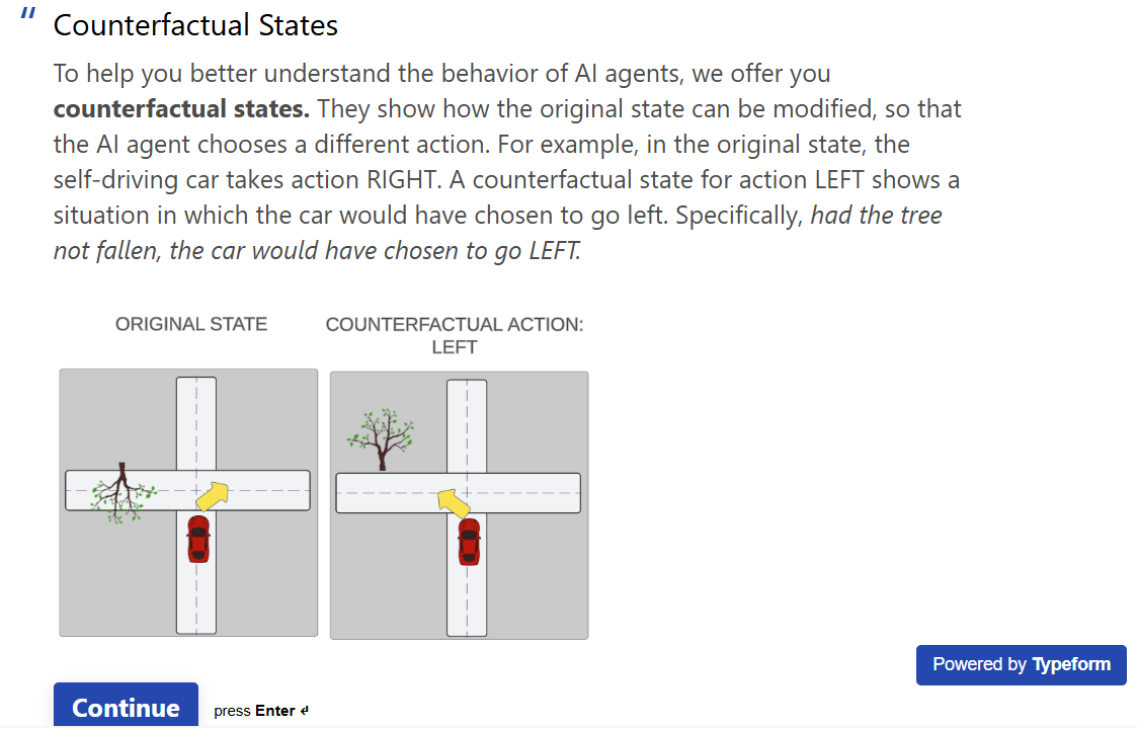


Figure A1.3: Slide used to demonstrate the use of counterfactual explanations to users.

Following the introduction of the counterfactual explanations, users were presented with a quiz consisting of two questions. The purpose of this quiz was to check their understanding of counterfactual explanations. Same as before, users could try each question as many times as they wanted, but could not progress further before answering all questions correctly. With this quiz the users concluded the introductory part of the study. We present the questions used in this quiz are shown in Figure A1.13.

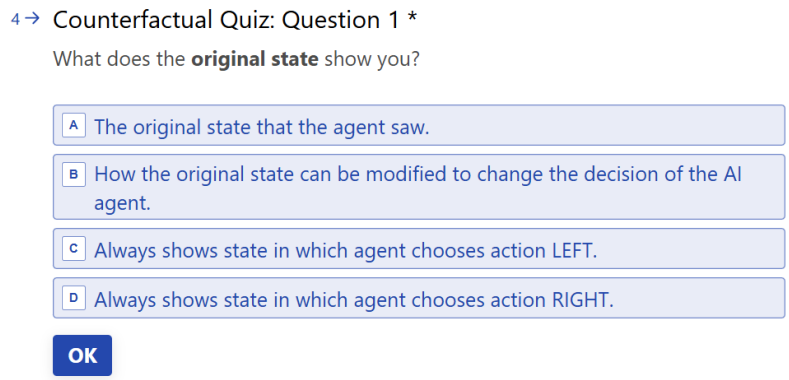
A1.1.2 Agent Understanding

After the introductory part the users were informed that the study is starting:

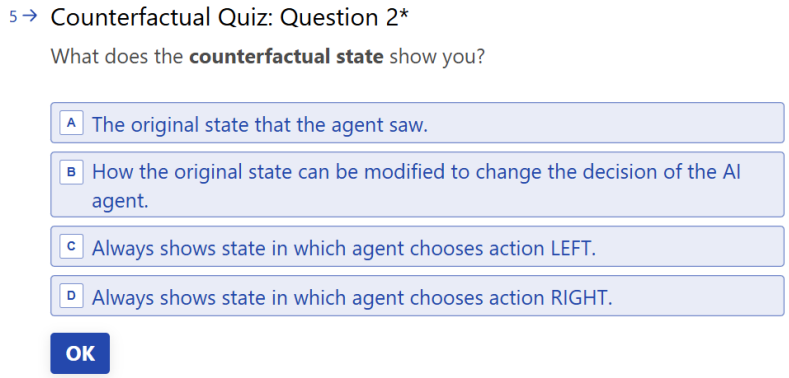
> *Study start!*
>
> *You will now be introduced to two AI agents – Agent A and Agent B playing the gridworld game. Firstly, you will be shown the behavior of agent A using counterfactual states and then you will be asked to predict its behavior in different situations. Afterwards, you will be asked to repeat the same process with agent B.*
>
> *In the end, you will be asked to compare the two agents, so pay attention to potential differences in their behavior.*

Firstly, users were shown the training phase for agent A. The were introduced to this part of the study by the following text:

> *Training Phase (Agent A)*
>
> *In this part of the study you will see some explanations of the behavior of AI agent A. Pay attention to agent's behavior, as after this you will be asked*



(a) Question 1



(b) Question 2

Figure A1.4: The quiz checking user understanding of counterfactual explanations.

to predict its behavior.

In the training phase the users were shown examples of original states, accompanied by counterfactual explanations explaining agent's behaviour. Two examples of this part of the study are given in Figure A1.5.

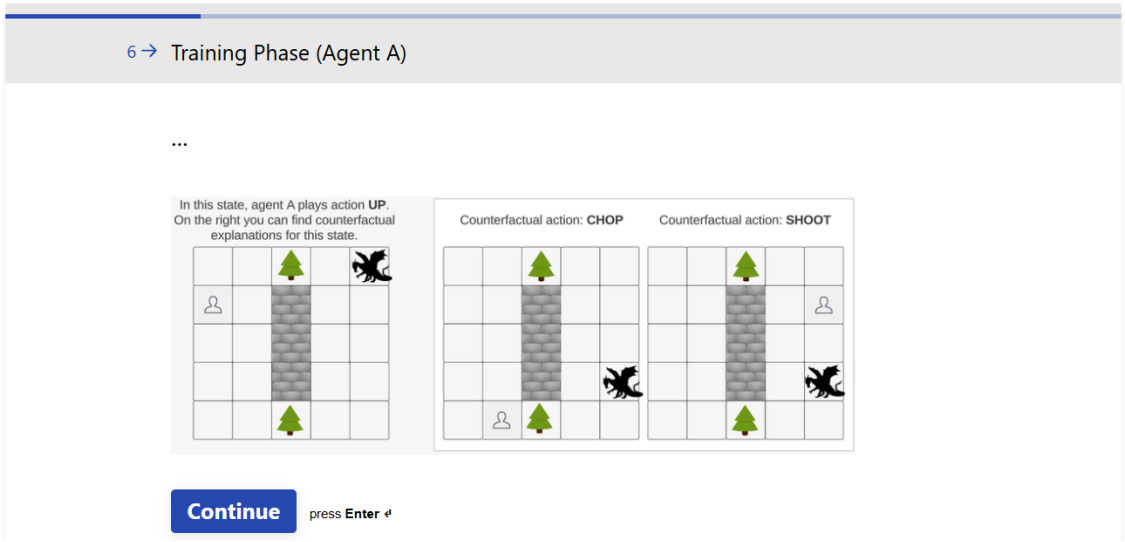
Following the training phase for agent A, users were shown the following information, leading them into the testing phase of agent A:

> *Testing phase (Agent A)*
>
> *You will now be asked to predict how AI agent A will act in specific states, based on what you learned about its behavior in the previous section.*

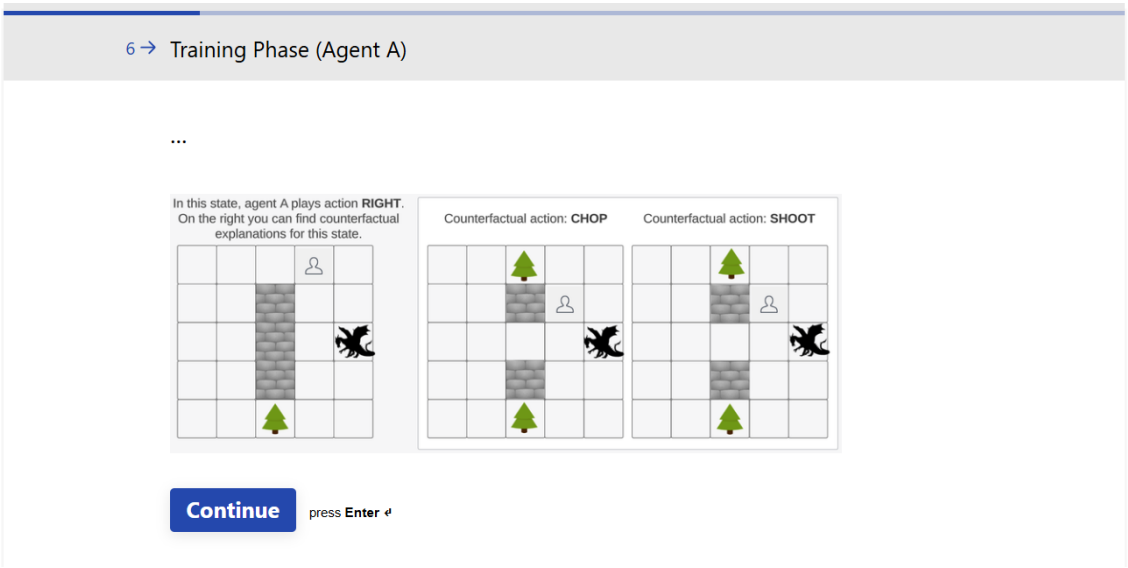
In the testing phase, users were shown previously unseen examples of RL states and asked to predict the next action of the agent A in that state. The testing phase for agent A consisted of ten questions. Examples of the questions asked during the testing phase are provided in Figure A1.6.

After finishing the testing phase of agent A, agent was informed that it will now be switching to agent B:

> *Training phase (Agent B)*
>
> *In this part of the study you will see some explanations of the behavior of a different AI agent B. Pay attention to agent's behavior, as after this you*



(a) Example 1



(b) Example 2

Figure A1.5: Two examples of training slides shown to users, containing an original state and two counterfactual states showing states in which agent would have chosen actions CHOP and SHOOT instead.

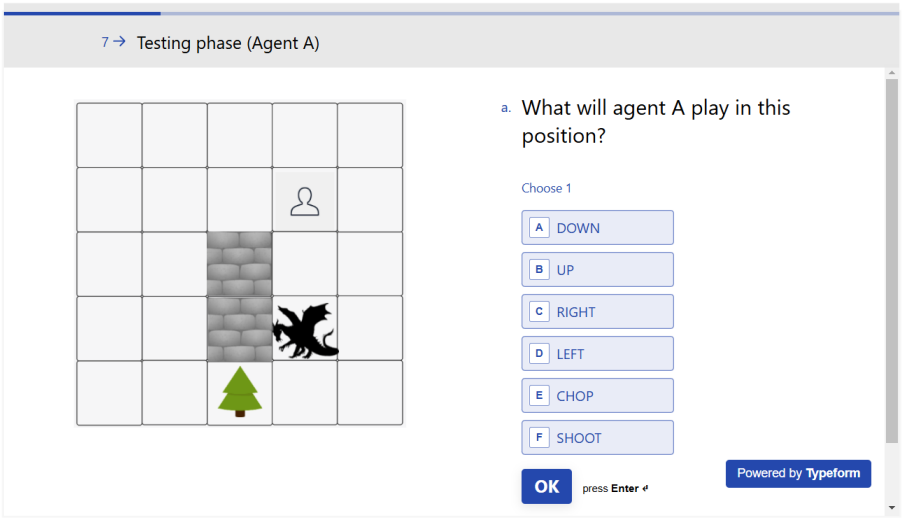
will be asked to predict its behavior.

The user is then presented with the training and testing phase for agent B, which take the exact same format as those for agent A.

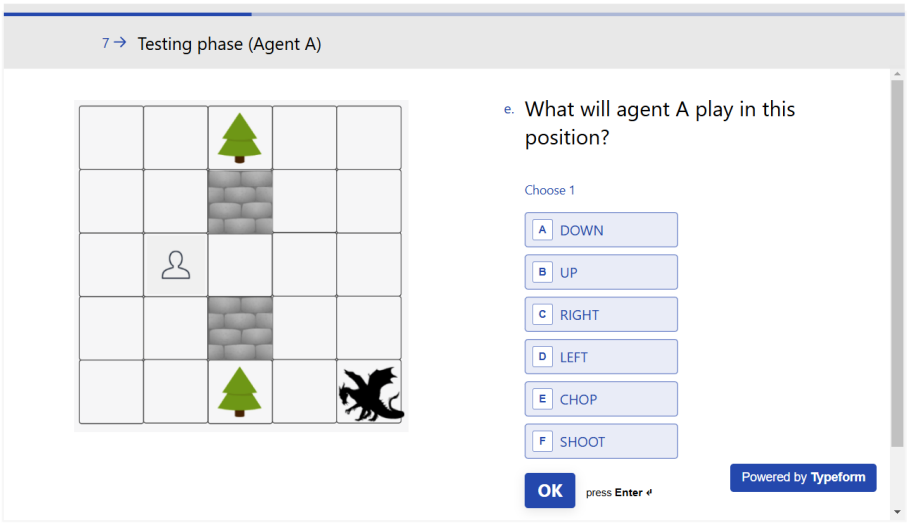
During the testing phases, users attention was also checked using three attention check questions. These attention checks were located between questions 5 and 6, and 9 and 10 for agent A test phase and between questions 7 and 8 for agent B test phase. An example attention check question is presented in Figure A1.7.

A1.1.3 Agent Comparison

After finishing the training and testing phase for both agents A and B, the user transitions to the agent comparison part. Firstly, the user is informed that a new part of the study begins:



(a) Example 1



(b) Example 2

Figure A1.6: Two examples of the test questions for evaluating user understanding of the agent.

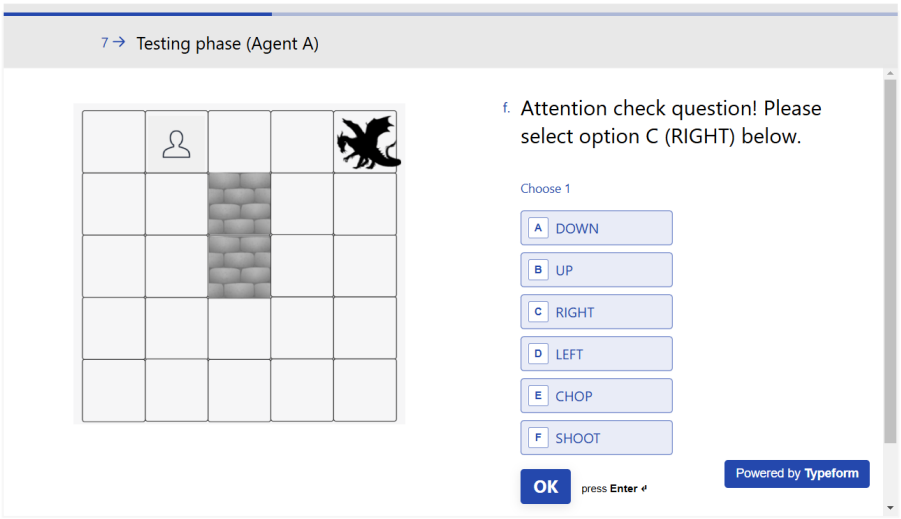
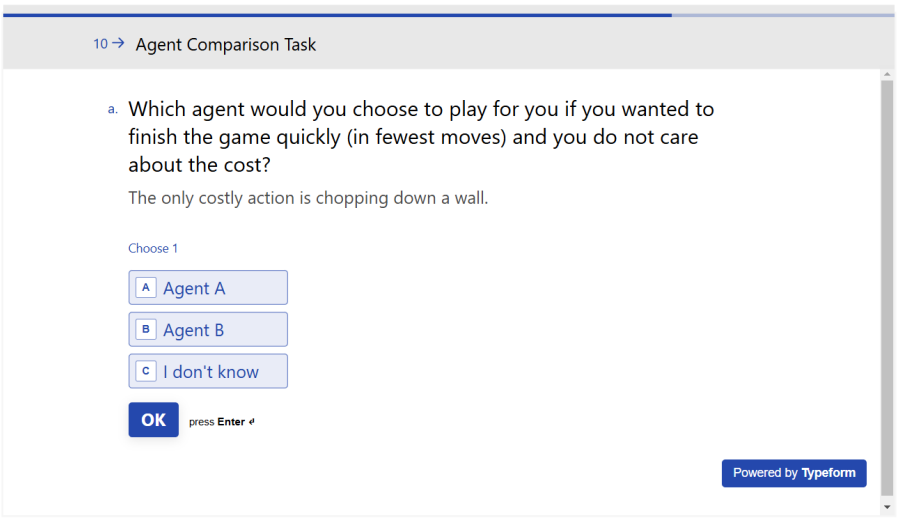


Figure A1.7: Attention check question used in the study to confirm user is paying attention on the questions

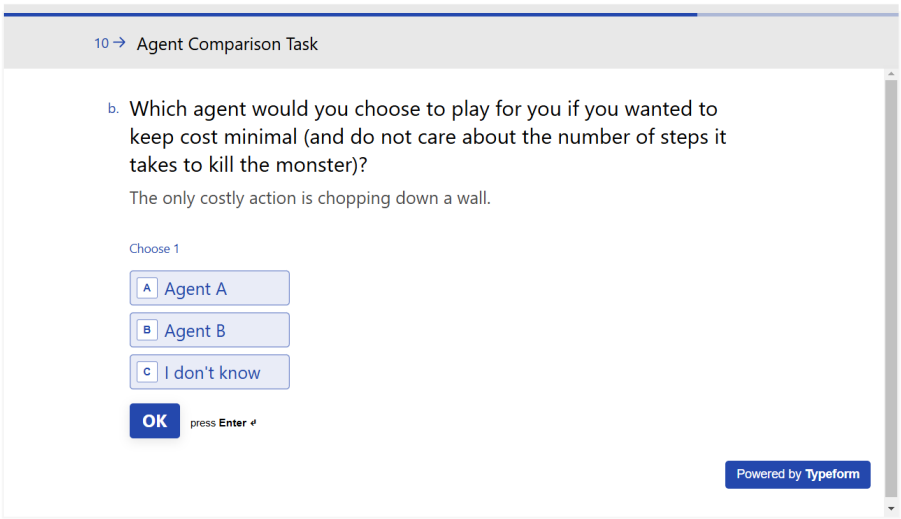
Agent Comparison Task

In this part of the study you will be asked to compare agents A and B you have previously seen.

As a part of this section, the user is asked two questions in which they should compare the behaviour of the previously seen agents A and B. We showcase the two questions in Figure A1.8.



(a) Question 1



(b) Question 2

Figure A1.8: Agent comparison questions used to assess users' ability to compare agents with different preferences.

A1.1.4 User Satisfaction

Finally, after answering agent comparison questions, user transitions to the last stage of the study, which investigates their satisfaction with the explanations:

User Satisfaction

In this section of the study you will be asked to evaluate the explanations (original states and counterfactual states) you have seen for agents A and B. Continue

Users were asked to answer seven questions in this section of the study, each on the 1-5 Likert scale. We showcase all the questions in Figure A1.9.

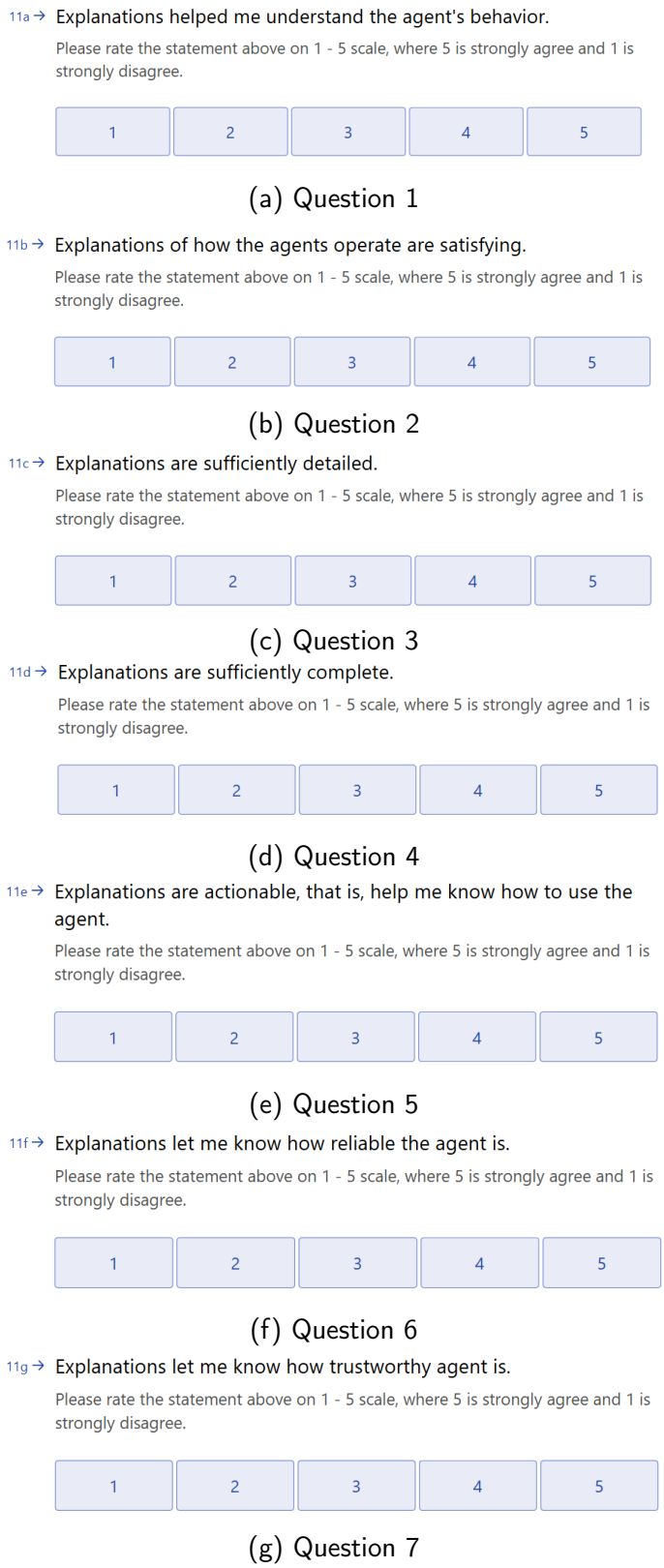


Figure A1.9: User satisfaction questions aiming to assess users satisfaction with the explanations on subjective metrics.

Finally, the final slide of the study thanks the users and instructs them on how to claim their remuneration for participating in the study:

> *Survey Completed! Thank you for participating!*

IMPORTANT!

1) Make a note of this completion code: XXXXXX

2) Click 'Submit' on this page to record your response.

If you do not complete the second step, we will not receive your data and will be unable to reward you.

3) Enter the completion code on Prolific to register your submission

A1.2 User study – Semifactual Explanations

In this Section, we describe in detail the user study that was used to evaluate semifactual explanations generated by SGRL-Advance, SGRL-Rewind and S-GEN1 approaches, as described in Chapter 5. Full user studies, depending on the semifactual search algorithm used to generate explanations, are available at: https://qrxhyre44mt.typeform.com/to/lULdyTWW (for SGRL-Advance), https://qrxhyre44mt.typeform.com/to/V4uJQZX6 (for SGRL-Rewind) and https://qrxhyre44mt.typeform.com/to/XNnNDTiW (for S-GEN1). We describe the study in three parts: introductory part (Section A1.2.1), which introduces users to the task and semifactual explanations, agent understanding part (Section A1.2.2), which measures user understanding of the behaviour of an RL agent, and user satisfaction part (Section A1.2.3) where users are asked to rate explanations based on subjective metrics.

A1.2.1 Introduction

Firstly, users were given a brief description of the study and asked to provide consent. If users declined to provide consent they were directed to leave the study. The consent form contained the following text:

LEAD RESEARCHER: Jasmina Gajcin

You are invited to participate in this study which aims to investigate the effect of different types of explanations on user understanding of AI systems. As AI systems often lack transparency, understanding how their decisions can be interpreted to their users is necessary to encourage human-AI collaboration and facilitate trust. In this research we aim to investigate how different types of explanations affect user's understanding and interaction with the AI system.

During the survey you will be asked to predict the behavior of a player in a simple grid-world game. You will be given some, but not all information about the rules of the game. Each question will consist of a game state for which you will be asked to predict the action the player will choose in that state. To better understand the player's reasoning, you will also be given explanations of player's behavior. The study is expected to take around 10

minutes and involve 10 - 15 questions. At the end of the study you will be asked to evaluate the provided explanations based on their properties such as usefulness on a 1- 5 scale. Upon completion of the study you will be compensated according to the payment policy of the Prolific platform.

The results from the study will be analyzed to investigate the effects of different types of explanations on user's understanding of AI systems. We plan to submit the results of study as a conference paper for HAI24 conference.

As participants are anonymously sourced through the Prolific platform, there is no conflict of interest during recruitment.

Individual results may be aggregated anonymously and research reported on aggregate results.

DECLARATION:

- I am 18 years or older and am competent to provide consent.

- I have read, or had read to me, a document providing information about this research and this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.

- I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.

- I understand that if I make illicit activities known, these will be reported to appropriate authorities.

- I understand that I may refuse to answer any question and that I may withdraw at any time without penalty, until the study is submitted.

- I understand that I cannot withdraw from the study after submitting as the study is fully anonymous.

- I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.

- I understand that my participation is fully anonymous and that no personal details about me will be recorded.

- I understand that if I or anyone in my family has a history of epilepsy then I am proceeding at my own risk.

- I understand that personal information about me, including the transfer of this personal information about me outside of the EU, will be protected in accordance with the General Data Protection Regulation.

- I have received a copy of this agreement.

After providing consent, user were directed to the introductory section of the study where they were shown information about the stochastic gridworld task. The following information was provided to the users:

> *In this survey you will be shown images from a simple gridworld game. The images will show situations from games played by an AI agent. You will then be asked to answer questions about the AI agent's behavior.*
>
> *Please read the game description below carefully. There will a short quiz to check if you understood the game correctly.*
>
> ***Goal**: In the game, the agent moves around a grid world and its goal is to shoot the dragon by firing an arrow at it. However, the path between the agent and the dragon can be obstructed by trees or walls.*
>
> ***Actions:** Agent can choose between actions UP, DOWN, LEFT, RIGHT, CHOP and SHOOT at every step. Actions UP, DOWN, LEFT, RIGHT move agent one square in the appropriate direction if possible. If that square is already occupied by another object or if it would lead agent to leave the board, the action has no effect and agent remains in its place. Action SHOOT only kills the dragon if the agent and dragon are in the same row or file, and all the squares between them are empty. Action CHOP can be used to chop down a tree or a wall if agent is directly next to it. While chopping down a tree is cheap, chopping down a wall is an extremely expensive action. Trees can regrow and walls can be rebuilt at each step with some probability (this is outside of agent's control).*
>
> ***Game End:** The game ends when agent successfully plays the SHOOT action and kills the dragon.*

Additionally, users were provided with an example of an RL state in the stochastic gridworld environment, which identified the main features present in the environment. This example is shown in Figure A1.10.

After receiving this information about the stochastic gridworld environments, the users were given a four-question test to check their knowledge of the rules of this game. The users could try to answer each question as many times as they wanted, but could not progress to the remainder of the study before clearing all questions. The questions are shown in Figure A1.11.

Same as before, if the user made a mistake on a question in the quiz above, they are directed to the following message, and asked to try again:

> *Incorrect answer*
>
> *Please click continue to try again.*

Only after all questions were answered correctly users were allowed to progress in the study, ensuring understanding of the RL environment.

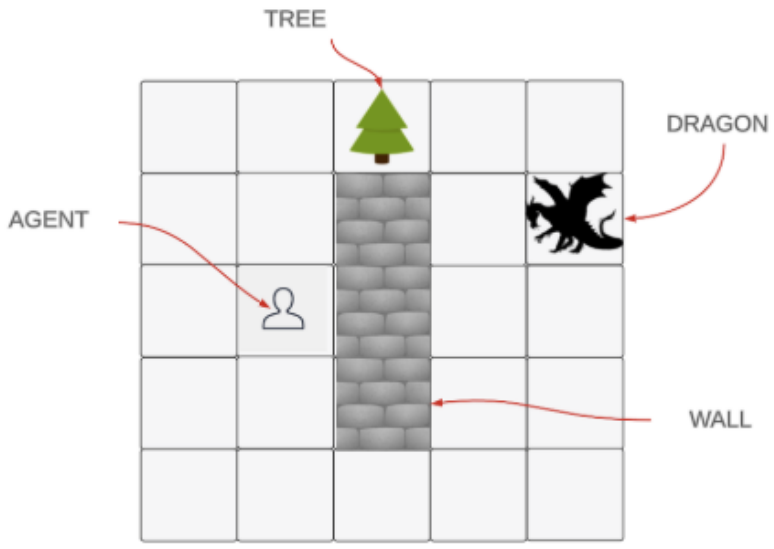
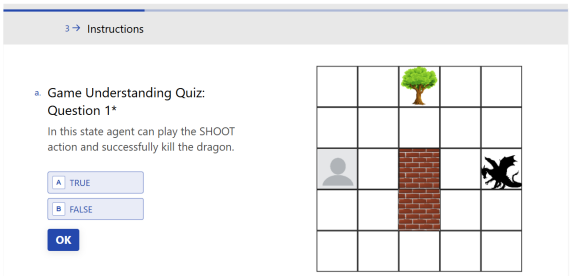
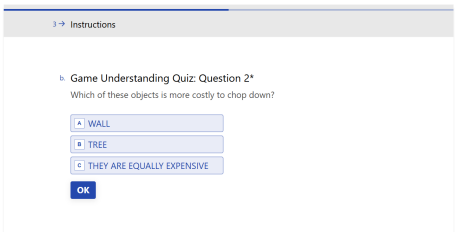


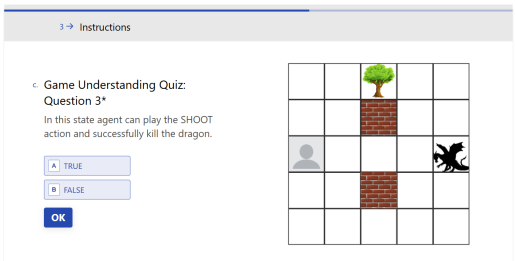
Figure A1.10: An image shown to users of the study to introduce them to the stochastic gridworld task.



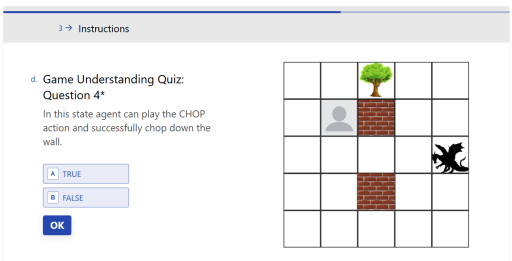
(a) Question 1



(b) Question 2



(c) Question 3



(d) Question 4

Figure A1.11: Game understanding questions shown to users to verify their understanding of the rules of the stochastic gridworld game.

Additionally, in the introductory part of the study, the users were shown a demonstration of the use of semifactual explanations. They were shown an textual definition of semifactual explanations, as well as a visual representation of semifactual explanations in the stochastic gridworld. We provide the example of semifactual explanations in the stochastic gridworld environment, as one common feedback from running the study on counterfactual explanations (Section A1.16g) was that users struggled to switch between the driving example used to demonstrate counterfactuals and stochastic gridworld used in the remainder of the study. An example of the semifactual explanation and the textual description provided to users are shown in Figure A1.12.

Following the introduction of the semifactual explanations, users were presented with a

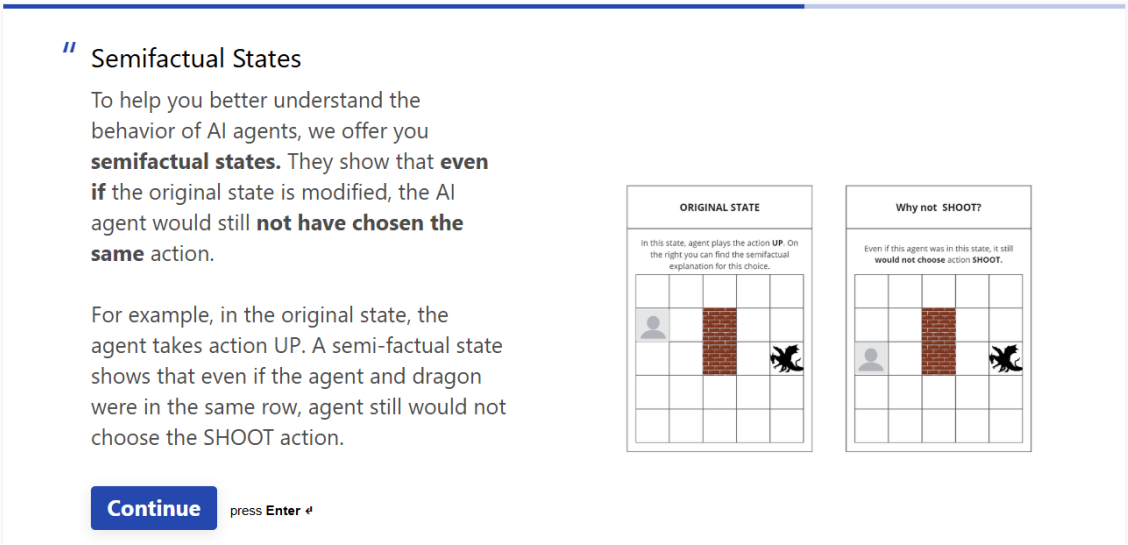
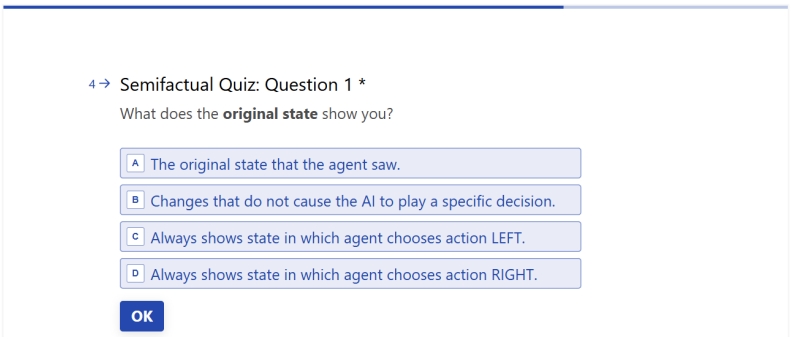
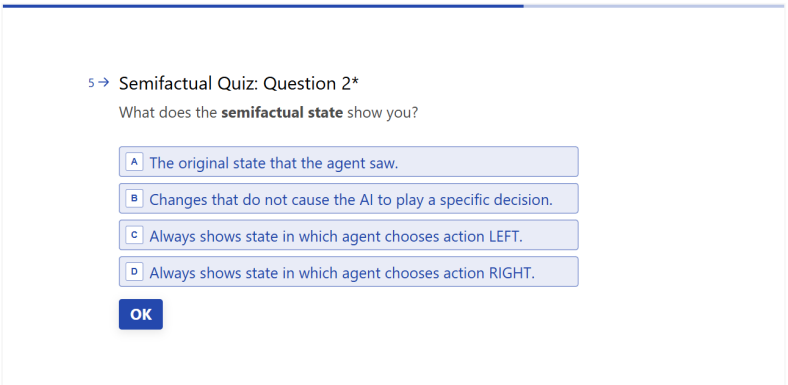


Figure A1.12: The slide used to demonstrate the use of semifactual explanations in a stochastic gridworld task to users.

quiz consisting of two questions. The purpose of this quiz was to check their understanding of semifactual explanations. Same as before, users could try each question as many times as they wanted, but could not progress further before answering all questions correctly. With this quiz the users concluded the introductory part of the study. We present the questions used in this quiz are shown in Figure A1.13.



(a) Question 1



(b) Question 2

Figure A1.13: The quiz checking user understanding of semifactual explanations.

A1.2.2 Agent Understanding

After the introductory part of the study, the users are informed that the main part of the study is starting:

> *Study start!*
>
> *You will now be* ***introduced to an AI agent playing the grid world game****. Firstly, you will be shown the behavior of the agent using semifactual states, and then* ***you will be asked to predict*** *its behavior in different situations.*

Firstly, users were shown the training phase for an agent. The were introduced to this part of the study by the following text:

> *Training Phase*
>
> *In this part of the study* ***you will see some explanations of the behavior of AI agent****. Pay attention to agent's behavior, as* ***after this you will be asked to predict its behavior.***

In this stage the users were shown examples of original states and semifactual explanations explaining agent's behaviour. Two examples shown to the users are given in Figure A1.14.

Following the training phase, users were shown the following information, leading them into the testing phase:

> *Testing phase You will now be* ***asked to predict how AI agent A will act*** *in specific states, based on what you learned about its behavior in the previous section.*

In the testing phase, users were shown previously unseen examples of RL states and asked to predict the next action of the agent in that state. The testing phase consisted of ten questions. This concluded the agent understanding task. Examples of the questions asked during the testing phase are provided in Figure A1.15.

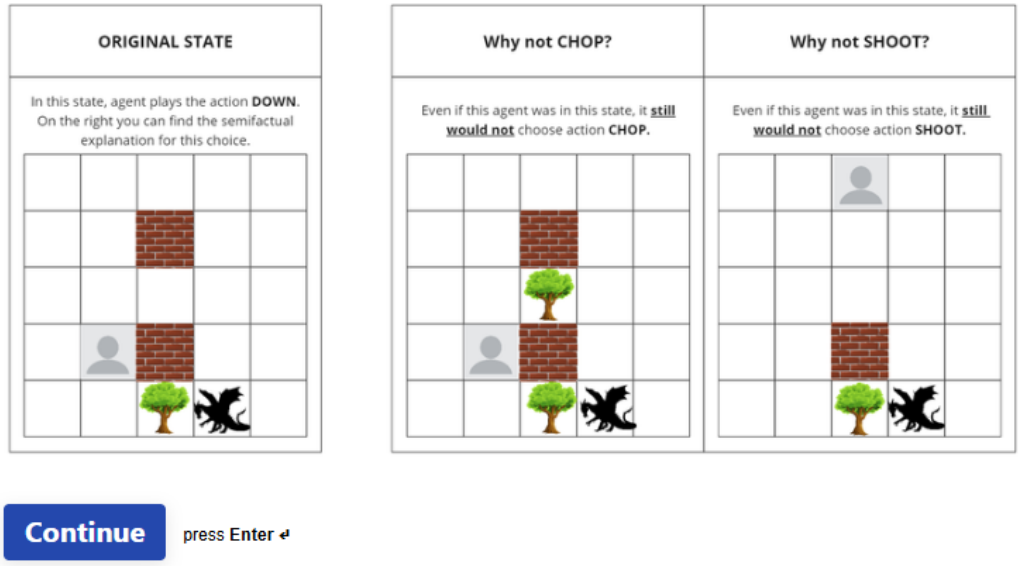
A1.2.3 User Satisfaction

Finally, after answering agent comparison questions, user transitions to the last stage of the study, which investigates their satisfaction with the explanations:

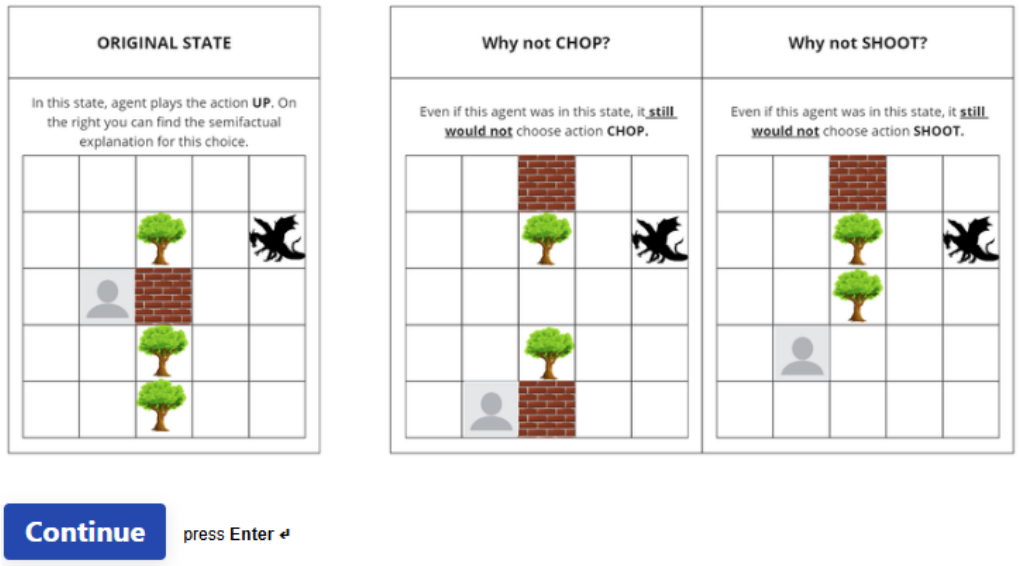
> *In this section of the study you will be asked to evaluate the explanations (original states and semifactual states) you have seen.*

Users were asked to answer seven questions in this section of the study, each on the 1-5 Likert scale. We showcase all the questions in Figure A1.16.

Finally, the final slide of the study thanks the users and instructs them on how to claim their remuneration for participating in the study:



(a) Example 1



(b) Example 2

Figure A1.14: Two examples of training slides shown to users, containing an original state and two semifactual states showing states in which agent would still not have chosen actions CHOP and SHOOT.

Survey Completed! Thank you for participating!

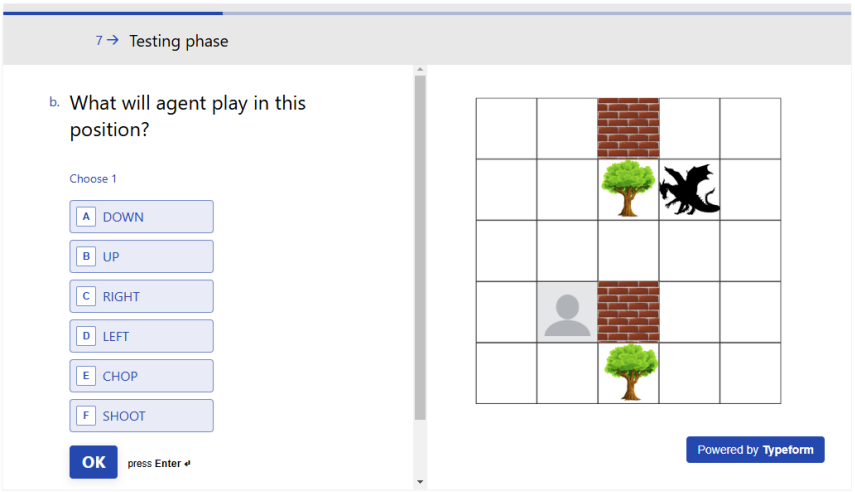
IMPORTANT!

1) Make a note of this completion code: XXXXXX

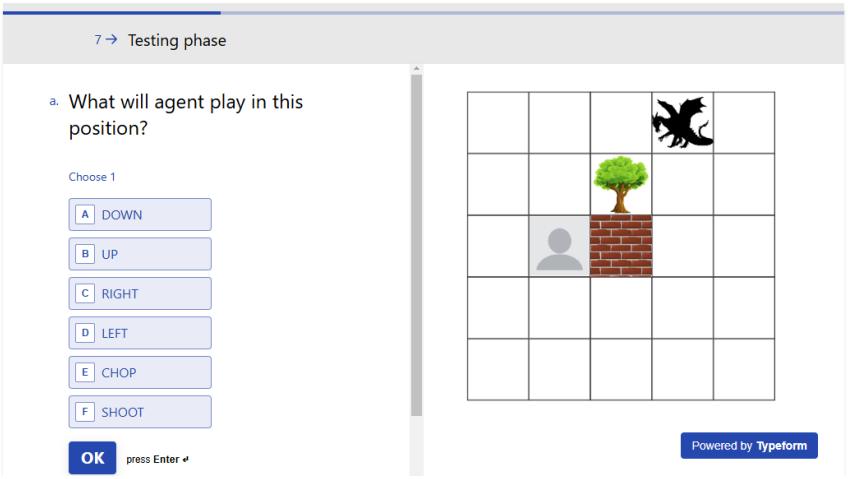
2) Click 'Submit' on this page to record your response.

If you do not complete the second step, we will not receive your data and will be unable to reward you.

3) Enter the completion code on Prolific to register your submission

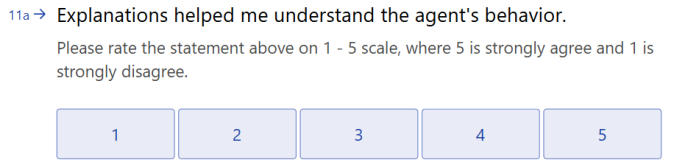


(a) Example 1

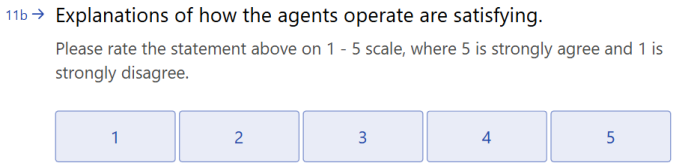


(b) Example 2

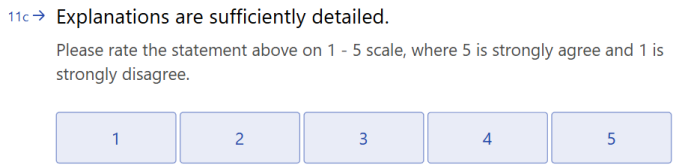
Figure A1.15: Two test questions used for evaluating user understanding.



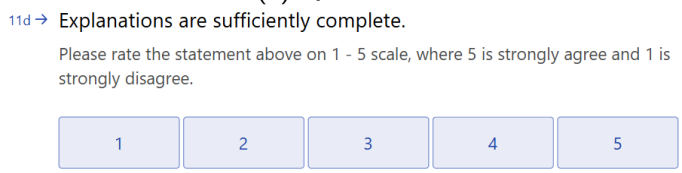
(a) Question 1



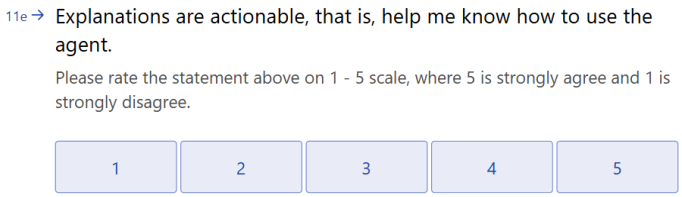
(b) Question 2



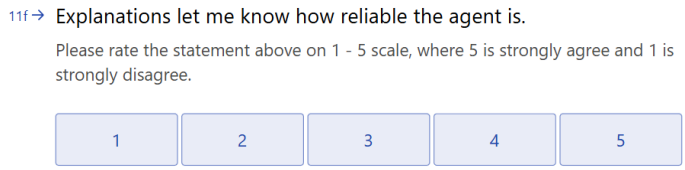
(c) Question 3



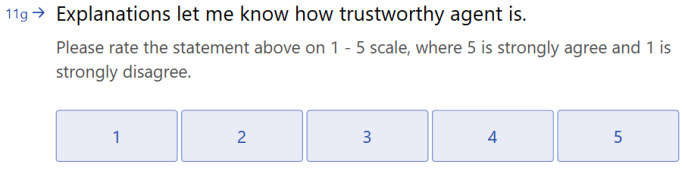
(d) Question 4



(e) Question 5



(f) Question 6



(g) Question 7

Figure A1.16: User satisfaction questions aiming to assess users satisfaction with the explanations on subjective metrics.

A2 Examples of Counterfactual and Semifactual Explanations

In this chapter, we provide some examples of counterfactual and semifactual explanations as generated by the approaches discussed in this work. In Section A2.1 we provide some examples of counterfactual explanations, while in Section A2.2 we showcase examples of semifactual explanations. We choose the demonstrative examples from the set of explanations generated in Chapter 5. We select to showcase those examples where a few features have been changed, in order to reduce cognitive load and emphasize the explanations.

A2.1 Counterfactual Explanation Examples

In this section, we showcase and discuss some counterfactual explanations generated using RACCER-Advance and RACCER-Rewind algorithms. We showcase explanations generated for the frozen lake (Section A2.1.1), stochastic gridworld (Section A2.1.2) and farm environments (Section A2.1.3).

A2.1.1 Frozen Lake

In Figure A2.1 we show an example of a counterfactual explanation generated for a frozen lake task. The explanation is generated using RACCER-Advance algorithm.

In the factual state the agent chooses the action RIGHT. A counterfactual question "Why did the agent not choose an alternative action DOWN?" to investigate whether agent made the action RIGHT to navigate to the exit. The counterfactual explanation is presented as a state in which the agent takes the action DOWN. With this explanation, we can see that the agent chooses to enter the frozen square to reach the goal, despite a solid path without frozen squares is also available. This could indicate that the agent is not fully trained, or that it is not able to fully estimate the consequences of stepping on a frozen squares. These insights could be used to further train the agent or adjust its reward function.

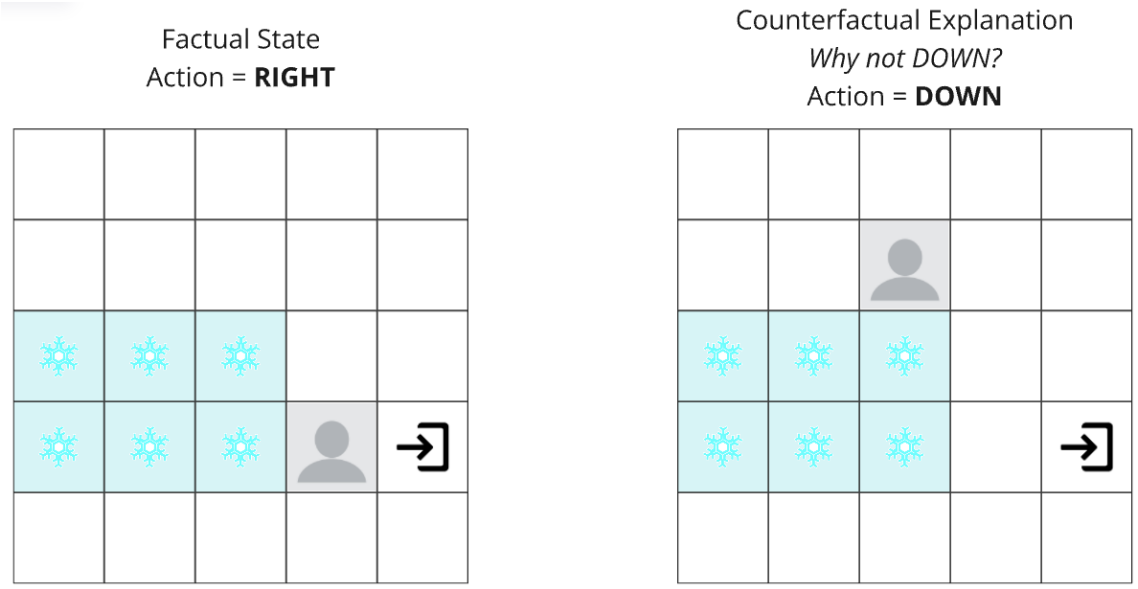


Figure A2.1: An example of a counterfactual explanation generated by RACCER-Advance algorithm in the frozen lake task. Left: Factual state in which agent chooses action RIGHT. The counterfactual question *"Why not action DOWN?"* is posed. Right: the counterfactual explanation showing an alternative state in which agent would have chosen the action DOWN.

A2.1.2 Stochastic Gridworld

In Figure A2.2 we show an example of a counterfactual explanations for the stochastic gridworld task. The explanation is generated using the RACCER-Advance algorithm.

In the factual state, agent chooses the CHOP action. As the agent is standing next to a wall, the CHOP action will result in destroying this wall. Additionally, this wall is an obstacle on the agent's way to the monster – if the wall was not there, agent could be able to shoot the monster. To investigate whether agent has learned this rule of the environment, we could pose a counterfactual question "Why not choose SHOOT in the factual state?"

The counterfactual explanation shows the state where the wall is not longer blocking the agent's path to the monster. In this counterfactual state the agent chooses the action SHOOT. This explanation confirms that the agent learned to see the walls as barriers on the way to the dragon.

A2.1.3 Farm

In Figure A2.3 we showcase an example of a counterfactual explanation for the farming environment. This explanation is generated using the RACCER-Rewind algorithm.

In the factual state the agent chooses to apply 5L of water to the field growing a tomato plant. The plant is the growth stage in the factual state. A counterfactual question "Why not apply more water to the growing plant? " could be posed by a user consulting this RL system to grow their plant.

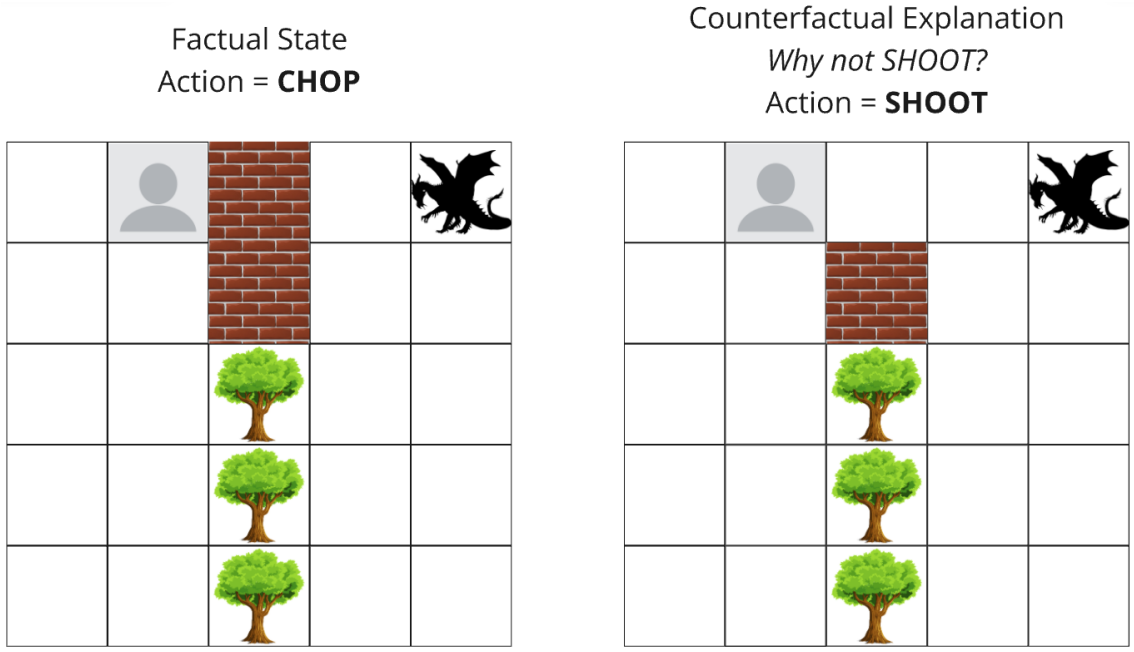


Figure A2.2: An example counterfactual explanation generated using RACCER-Advance algorithm in the stochastic gridworld task. Left: A factual state in which agent chooses action CHOP, resulting in destroying a wall next to the agent. A counterfactual explanation *"Why not choose action SHOOT?"* is posed. Right: counterfactual explanation showing an alternative state in which the agent would have chosen the action SHOOT.

To answer this question, we generate a counterfactual state in which agent chooses to apply 5L of water, the largest possible amount. In the counterfactual state, the plant has progressed to a different growth stage and has entered bloom. This indicates that the agent learned that a plants in the higher growth stages require more water. Additionally, factual and counterfactual states differ slightly (0.2cm) in the size of plant. This is a consequence causal relationships between features, which are maintained by RACCER algorithms. Channing the growth stage is likely to have an effect on the size of the plant, which is demonstrated in the counterfactual explanation.

A2.1.4 Highway Driving

In Figure A2.4 we showcase an example of a counterfactual explanations for the highway driving environment. This explanation is generated using the RACCER-Advance algorithm.

In the factual state the agent chooses to change lanes by executing an action LANE RIGHT. A counterfactual explanation could be generated to explain why the agent chose to change lane to the right rather than left using an action LANE LEFT.

To answer this question we generate a counterfactual explanation using RACCER-Rewind algorithm. The counterfactual explanation is a state in which the agent chooses LANE LEFT as the action. In the factual state the agent is close to another vehicle in the left lane. However, in the counterfactual state the agent is more advanced than the vehicle in the left lane, enabling the agent to change lanes to the left. This indicates that the agent has learnt that it cannot change lanes if there is a vehicle occupying that lane in

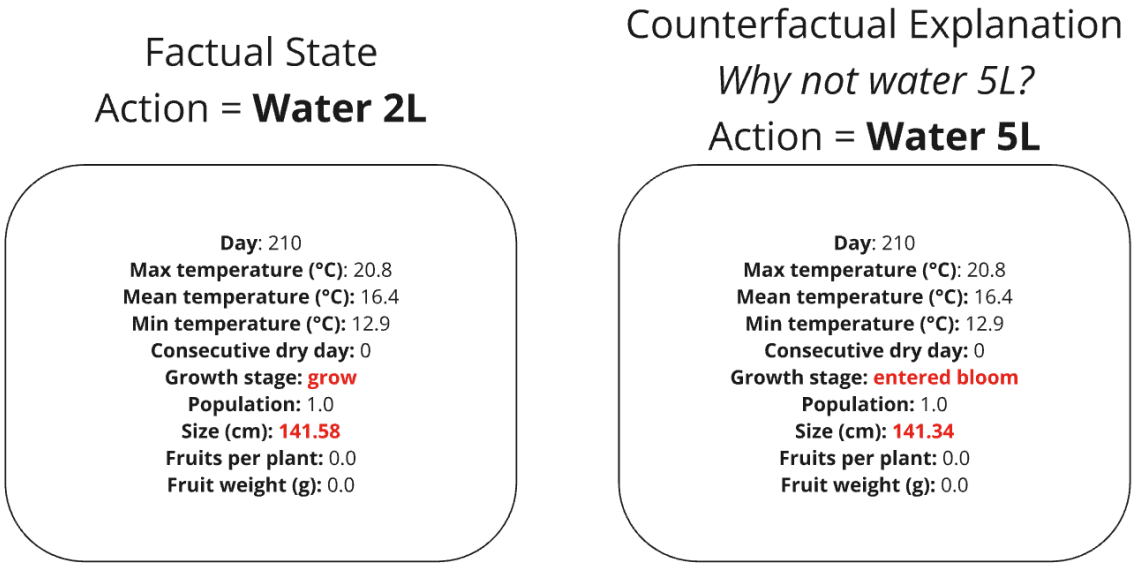


Figure A2.3: An example of a counterfactual explanation generation by RACCER-Rewind algorithm in the farm task. Left: Factual state in which agent chooses to apply 2L of water to the tomato plant. A counterfactual question "Why not apply more water to the growing plant?" is posed. Right: Counterfactual explaantion showing an alternative state, differing in growth stage and plant size, in which the agent would have chosen to apply the highest possible amount of 5L of water.

the close vicinity of the agent.

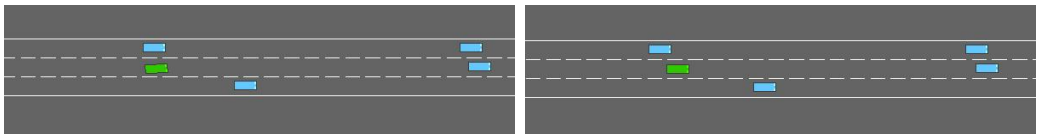


Figure A2.4: An example of a counterfactual state generated by RACCER-Advance algorithm in highway driving task. Left: Factual state in which agent chooses action LANE RIGHT. A counterfactual question *"Why not action LANE LEFT?"* is posed. Right: Counterfactual explanation showing an alternative state in which agent chooses to action LANE LEFT.

A2.2 Semifactual Explanation Examples

In this section, we showcase semifactual explanations as generated by SGRL-Advance and SGRL-Rewind approaches introduced in Chapter 4. The following sections showcase and discuss sample explanations in the frozen lake (Section A2.2.1), stochastic gridworld (Section A2.2.2) and farm (Section A2.2.3) environments.

A2.2.1 Frozen Lake

In Figure A2.5 we showcase a semifactual explanation in the frozen lake environment. The explanation is generated using the SGRL-Advance algorithm.

In the factual state the agent chooses an action UP. A semifactual explanation could be asked to investigate why agent did not choose to exit the griworld by executing the action EXIT. To answer this question, we generate a semifactual state in which agent

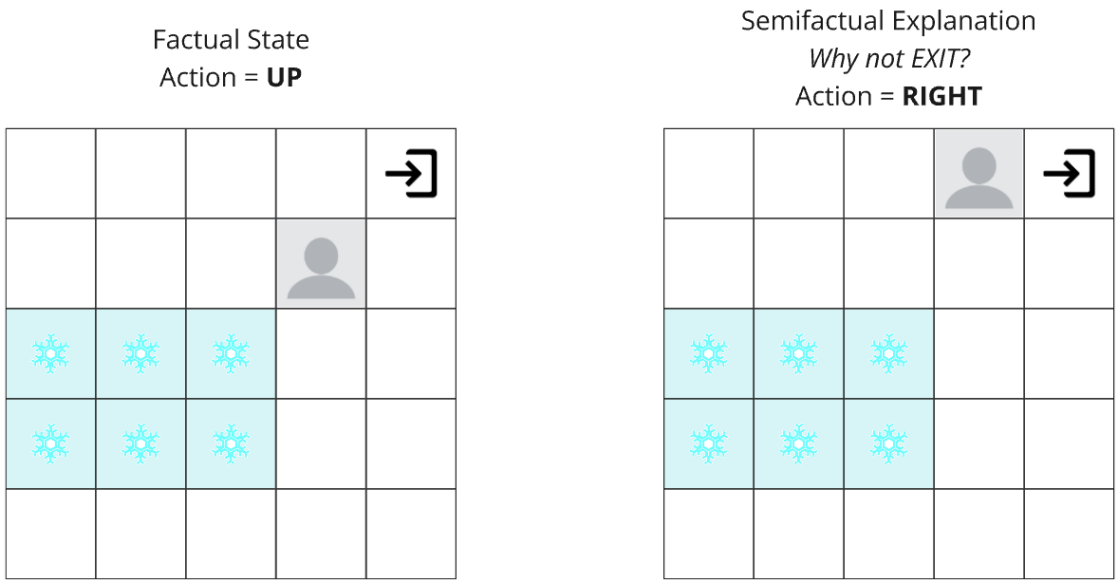


Figure A2.5: An example of a semifactual explanation generated by SGRL-Advance in the frozen lake environment. Left: A factual state in which the agent chooses action UP. A semifactual question asking "Why did the agent not choose EXIT?" is posed for the factual state. Right: a semifactual explanation showing an alternative state in which the agent still does not chose the action EXIT.

also does not choose action EXIT. The semifactual state shows the agent closer, but still not at the exit. In this semifactual state, agent still does not choose the action EXIT, and rather moves RIGHT. In this way, the semifactual state reaffirms why exiting is still not the best course of action for the agent.

A2.2.2 Stochastic Gridworld

In Figure A2.6 we show an example of a semifactual explanation in a stochastic gridworld environment. The explanation is generated using the SGRL-Advance algorithm.

In the factual state the agent chooses the CHOP action. To better understand why the user chose this action, we can constrast it with the SHOOT action. To answer the question "Why did the agent not choose the action SHOOT in the factual state?", we generate a semifactual state which reaffirms the agent's decision not to shoot.

In the semifactual state the agent does not choose to SHOOT, but instead chooses action CHOP. The difference between the factual and semifactual state is in the object blocking the path to the monster. The semifactual explanations shows that the agent would not choose to SHOOT even if a different obstacle was in its way.

A2.2.3 Farm

In Figure A2.7 we show an example of a semifactual explanation in the farm environment. The explanation is generated using the RACCER-Rewind algorithm.

In the factual state, the plant is in the blooming stage, and there are still no fruits. The agent chooses to apply 2L of water to the plant in the factual state. To better understand

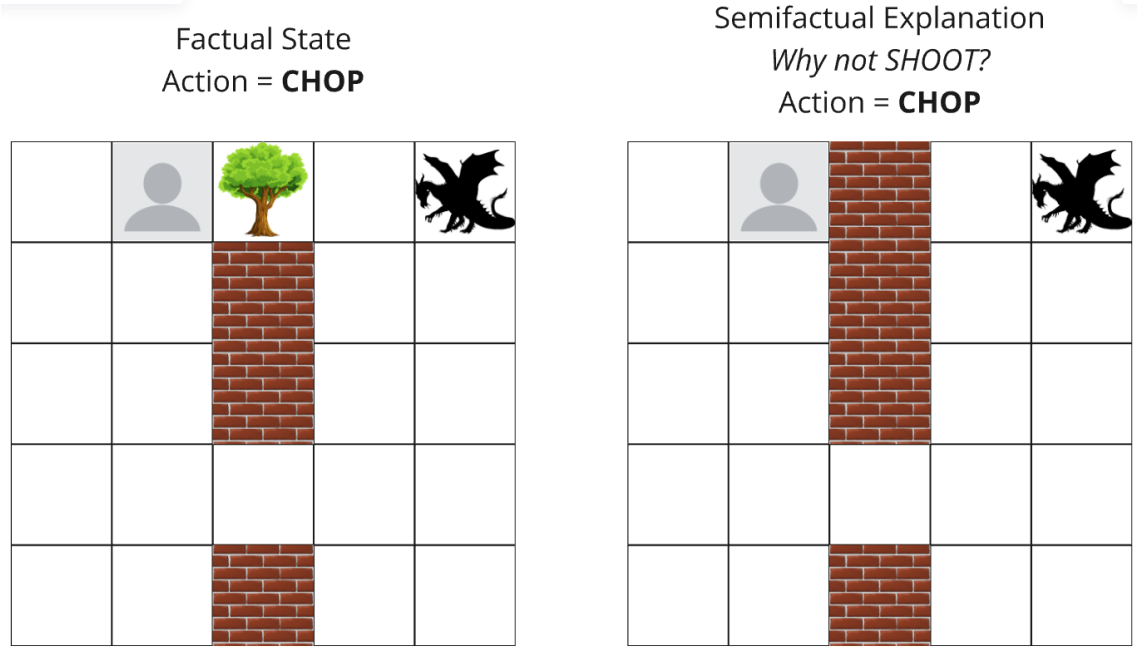


Figure A2.6: An example of a semifactual state in the stochastic gridworld environment, generated using the RACCER-Advance algorithm. Left: A factual state in which the agent chooses to chop down the tree. Right: A semifactual explanation reaffirming agent's decision not to shoot in the factual state. In the semifactual state the agent chooses to CHOP again, despite the change in the obscale in front of it.

the agent's behaviour, one might ask why the agent has not chosen to harvest the plant in the factual state.

To answer this question, we generate a semifactual explanation that reaffirms the agent's decision not to harvest the plant. In the semifactual state the agent chooses to apply 5L of water, rather than harvest. However, the semifactual state shows a far more progressed plant, which has entered a ripe stage and has fruit. In this way, the semifactual reaffirms the decision not to harvest an early-stage plant by showing it would not choose to harvest even a more progressed plant. Additionally, this semifactual showcases how SGRL algorithms maintain the causal relationships between feature. Namely, the change in the growth stage is associated with the number and weight of fruits on the plant, which is demonstrated in the semifactual state.

A2.2.4 Highway Driving

In Figure A2.8 we show an example of a semifactual explanation in a highway driving environment. The explanation is generated using the SGRL-Advance algorithm.

In the factual state, the agent chooses to increase its speed using the FASTER action. To better understand the agent's behaviour we can contrast it with the LANE LEFT action, which would cause the agent to change lanes to the left. To answer the question "Why did the user not choose the action LEFT LANE in the factual state?" we generate a semifactual state which reaffirms the agent's decision not to change lanes.

In the semifactual state, the agent does not choose to change lanes, but still chooses to increase speed via the FASTER action. From this semifactual, we can see that the

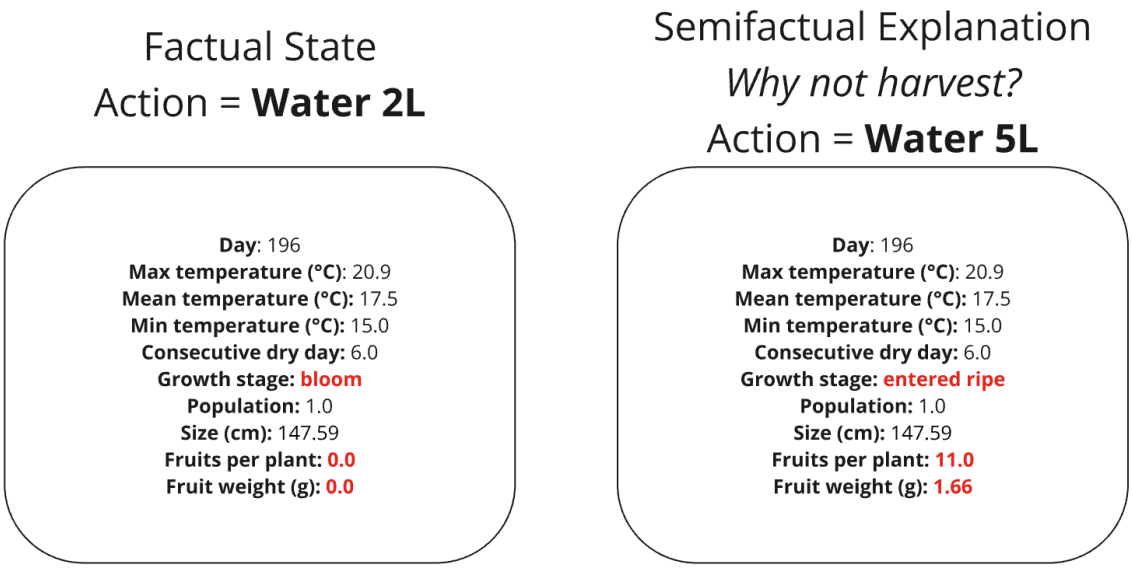


Figure A2.7: An example of a semifactual explanation for the farming environment generated using the RACCER-Rewind algorithm. Left: A factual state showing a sample day in the growing processes of the tomato plant. In the factual state the agent chooses to apply 2L of water to the field with the plant. Right: a semifactual explanation reaffirming the choice to not harvest. In the semifactual state agent does not harvest but applies 5L of water, even though the plant has progressed in growth and has fruit.

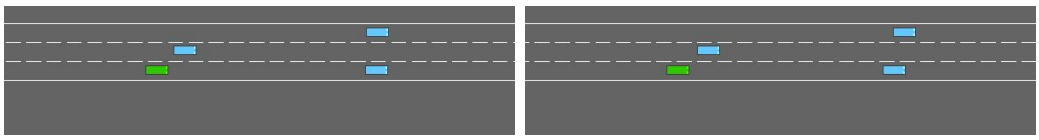


Figure A2.8: An example of a semifactual explanation in the highway driving environment. Left: A factual state in which the agent chooses the action FASTER. Right: A semifactual explanation reaffirming agents decision not to change lanes using LANE LEFT action. In the semifactual state, the agent chooses FASTER action.

agent would not choose to change lanes even if the distance between it and the vehicle in the left lane was bigger.