RACCER: Towards Reachable and Certain Counterfactual Explanations for Reinforcement Learning

Jasmina Gajcin
Trinity College Dublin
Dublin, Ireland
gajcinj@tcd.ie

Ivana Dusparic
Trinity College Dublin
Dublin, Ireland
ivana.dusparic@tcd.ie

ABSTRACT

While reinforcement learning (RL) algorithms have been successfully applied to numerous tasks, their reliance on neural networks makes their behavior difficult to understand and trust. Counterfactual explanations are human-friendly explanations that offer users actionable advice on how to alter the model inputs to achieve the desired output from a black-box system. However, current approaches to generating counterfactuals in RL ignore the stochastic and sequential nature of RL tasks and can produce counterfactuals that are difficult to obtain or do not deliver the desired outcome. In this work, we propose RACCER, the first RL-specific approach to generating counterfactual explanations for the behavior of RL agents. We first propose and implement a set of RL-specific counterfactual properties that ensure easily reachable counterfactuals with highly probable desired outcomes. We use a heuristic tree search of the agent's execution trajectories to find the most suitable counterfactuals based on the defined properties. We evaluate RACCER in two tasks as well as conduct a user study to show that RL-specific counterfactuals help users better understand agents' behavior compared to the current state-of-the-art approaches.

KEYWORDS

Reinforcement Learning; Explainability; Transparency; Counterfactual Explanations

ACM Reference Format:
Jasmina Gajcin and Ivana Dusparic. 2024. RACCER: Towards Reachable and Certain Counterfactual Explanations for Reinforcement Learning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning (RL) has shown remarkable success in recent years and is being developed for high-risk areas such as healthcare and autonomous driving [2]. However, RL algorithms often use neural networks to represent their policies, making them difficult to understand and apply to real-life tasks [22].

Counterfactual explanations are user-friendly explanations for interpreting decisions of black-box algorithms [18]. In machine learning, counterfactuals are defined as an answer to the question: *"Given that the black-box model M outputs A for input features f_1, ..., f_k, how can the features change to elicit output B from M?"* [25]. They give users actionable advice on how to change their input to obtain a desired output, and are inherent to human reasoning, as we rely on them to assign blame and understand events [3].

In recent years, numerous methods for generating counterfactual explanations have been developed both for supervised [6, 9, 10, 16, 17, 19, 21, 23, 26] and RL [12, 20]. In RL, Olson et al. [20] propose a generative model for generating realistic counterfactuals that requires access to internal parameters of the black-box model. In contrast, Huber et al. [12] propose GANterfactual-RL, the only model-agnostic approach to generating counterfactuals in RL. GANterfactual-RL uses generative modeling to generate counterfactuals for visual tasks.

The majority of proposed methods for generating counterfactuals in supervised and RL search for the smallest change in features that leads to a change in outcome. However, due to the sequential nature of RL tasks, two states with similar features can be far away in terms of execution and even small changes in features can have uncertain outcomes due to stochasticity in the environment [8]. Offering users counterfactuals that are not easy to reach or do not deliver on the promised outcome can cost users substantial time, and cause them to lose trust in the AI system. Additionally, current approaches do not distinguish between the two types of counterfactual explanations that can be defined for RL – those that change causes from the past, from those that provide actionable advice for the future. For example, if a user's loan is denied, the counterfactual can either state that *"Had your income been higher you would have been approved"*, or *"If you increase your income, you will be approved in the future"* [5].

In this work, we propose RACCER (Reachable And Certain Counterfactual Explanations for Reinforcement Learning), 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 generates explanations that explore how changes to the current state can affect future outcomes, sometimes referred to *prefactual explanations* [5]. Firstly, we propose three novel RL-specific counterfactual properties – *reachability, stochastic certainty, and fidelity*. These counterfactual properties rely on the stochastic and sequential nature of RL tasks and ensure that counterfactuals are easy to reach and deliver the desired outcome with high probability. RACCER searches for the most suitable counterfactual by optimizing a loss function consisting of the three RL-specific properties using a heuristic tree search of the agent's execution tree. We evaluate RACCER in two environments and compare it to the only other

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

model-agnostic approach for RL – GANterfactual-RL [12]. We find that RACCER performs better on feature-based and RL-specific counterfactual properties when explaining both fully-trained and suboptimal models. Additionally, we conduct a user study in which we compare the effect of counterfactual explanations on user understanding of RL agents and show that RACCER generates counterfactuals that help humans better understand and predict the behavior of RL agents.

Our contributions are as follows:

(1) We design three RL-specific counterfactual properties – reachability, stochastic certainty, and fidelity, and provide metrics for their estimation.
(2) We propose RACCER, the first algorithm for generating RL-specific counterfactual explanations, which relies on the above counterfactual properties.
(3) We conduct a user study and show that RACCER can produce counterfactuals that help humans better understand an agent's behavior compared to the baseline approaches.

The implementation of RACCER and evaluation details can be found at https://github.com/jas97/RACCER.

2 RELATED WORK

In supervised learning, counterfactual explanations have been used to propose changes in input features that elicit a desired prediction from a black-box model. Various counterfactual properties have been defined to evaluate different counterfactuals [25]. For example, validity is used to measure whether counterfactual achieves the desired output, proximity is a feature-based similarity measure that ensures counterfactual features are similar to those in the original instance, and sparsity measures the number of features changed. In recent years, numerous works have proposed methods for generating counterfactual explanations in supervised learning [6, 16, 17, 19, 21, 23, 26]. The majority of these methods follow the same approach, where a loss function is defined by combining different counterfactual properties and optimized over the training data set. The methods differ in their design of the loss function and the choice of the optimization method. For example, in the first work on counterfactual explanations for supervised learning, Wachter et al. [26] use gradient descent to optimize a loss function based on proximity and validity properties. Similarly, Mothilal et al. [19] propose DICE, which introduces a diversity property to the approach of Wachter et al. [26] to ensure users are offered a set of diverse, high-quality explanations. Dandl et al. [6] poses the problem of counterfactual search as multi-objective optimization and uses a genetic algorithm to optimize validity, proximity, sparsity, and data manifold closeness of counterfactual instances.

In RL, counterfactual explanations aim to explain a decision of a black-box RL model in a specific state by proposing an alternative state in which the model would choose the desired action. Olson et al. [20] propose an approach that relies on generative modeling to create realistic counterfactuals, similar in features to the original instance, and produce a desired output. The approach is not model-agnostic and requires access to the internal parameters of the black-box model that is being explained. In contrast, Huber et al. [12] propose a model-agnostic approach GANterfactual-RL, which frames the counterfactual search as a domain translation problem, where each domain contains states in which the agent would choose a specific action. To find a suitable counterfactual, the original instance is translated to the target domain. The algorithm is based on the StarGAN architecture [4] and includes training a discriminator D and generator G. The generator receives as an input a state and target domain and produces a translated state. The role of the discriminator is to distinguish between real and fake images. The generator and discriminator are trained on states extracted from the agent's policy.

Current approaches in RL [12, 20] generate realistic counterfactuals that can help users better understand agents' decisions and even detect faulty behavior in Atari agents. However, they focus on the same feature-based counterfactual properties such as proximity and sparsity as supervised learning methods. In RL where two states can be similar in features but distant in terms of execution, feature-based metrics are not sufficient for measuring how obtainable a counterfactual is [8]. Relying only on feature-based similarity measures can produce counterfactuals that are not easily (or at all) obtainable, and decrease human trust in the system. In contrast, our work proposes the first approach for generating RL-specific counterfactuals that take into account the stochastic and sequential nature of RL tasks.

Although the purpose of counterfactual explanations is to show a path to the desired outcome, this path can be uncertain due to the environment in which the system operates. For example, even if the loan applicant fulfills all conditions stipulated in a counterfactual, the bank might change the conditions for approving a loan. Delaney et al. [7] recognize the need for estimating and presenting the uncertainty associated with counterfactuals to the user in supervised learning tasks. In this work, we estimate uncertainty from an RL perspective and use it not only as additional information for the user but as an important factor in searching for the counterfactual.

3 RACCER

In this section, we describe RACCER, our approach for generating counterfactual explanations for RL tasks. To generate a counterfactual explanation x', we require oracle access to the black-box model M being explained, the state x being explained, and the desired outcome a'. Additionally, the approach needs access to the RL environment. RACCER generates a counterfactual state x' that can be easily reached from x and in which the black-box model M chooses a' with a high probability. RACCER is fully model-agnostic, does not require information on model parameters, and can be used for generating counterfactual explanations of any RL model.

RACCER does not search for the counterfactual directly but looks for a sequence of actions to transform the original into the counterfactual instance [13, 14, 24]. This way of conducting counterfactual searches is more informative for the user, as they can be presented with not just the counterfactual instance, but also the sequence of actions they need to perform to obtain their desired outcome. To that end, we set out to find the optimal sequence of actions A that can transform x into a counterfactual state x'. In the remainder of this section, we describe how we can evaluate action sequences that lead to counterfactual states (Sections 3.1 and 3.2) and describe our approach to searching for the optimal one (Section 3.3).

3.1 Counterfactual Properties for RL

Counterfactual properties guide the counterfactual search and are used to select the most suitable counterfactual explanation. In this section, we propose three RL-specific counterfactual properties that take into account the sequential and stochastic nature of RL tasks. These properties ensure that counterfactuals are easily obtainable from the original instance, and produce desired output with high certainty. We define these properties as functions of action sequence A that transforms x into counterfactual x'.

3.1.1 Reachability. In RL two states can be similar in terms of state features, but far away in terms of execution. This means that, despite appearing similar, a large number of actions might be required to reach the counterfactual from the original state. Conversely, a state can be reachable by a few RL actions even if it appears different based on its feature values. Additionally, state features can be affected by stochastic processes outside of the agent's control. Relying solely on feature-based similarity measures (e.g. proximity, sparsity) could dismiss easily reachable counterfactuals where changes in features are beyond the agent's control and do not affect action choice.

To account for the sequential and stochastic nature of RL tasks, we propose measuring *reachability*. For a state x and a sequence of actions A, we define reachability as:

$$R(x, A) = len(A) \tag{1}$$

$R(x, A)$ measures the number of actions in the sequence that navigates to the counterfactual. Minimizing this property ensures counterfactuals can be reached within a small number of steps.

3.1.2 Fidelity. RACCER searches for counterfactuals by finding an optimal sequence of RL actions to transform the original instance. For the counterfactual to be representative of the agent's behavior, the sequence of actions has to be likely under the agent's policy. As an example, consider a simple grid world where an agent needs to pick up one of the two keys – red or blue and open the door. To explain why the agent did not choose to go to the door in a specific step, the counterfactual explanation might show that the agent would have gone to the door had they collected the red key first. However, if the agent's policy prefers the blue key over the red, this counterfactual is not representative of the agent's behavior and could be misleading to the user.

For this reason, RACCER prioritizes counterfactual states that can be reached under the agent's policy. We calculate the fidelity of a sequence of actions A as the probability that the agent will choose these actions from state x:

$$F(x, A) = 1 - \prod_{a \in A} softmax(Q(x, \mathcal{A}))[a] \tag{2}$$

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 optimizing fidelity, we ensure that generated counterfactuals are representative of the agent's behavior.

3.1.3 Stochastic certainty. One of the main qualities of counterfactual explanations is that they deliver the desired outcome. Asking the user to put their time and effort into changing the model inputs, only to obtain another unsatisfactory output can have detrimental effects on user trust in the system. During the time that is needed to convert the original instance into a counterfactual, the conditions of the task can change, rendering the counterfactual invalid.

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. To ensure that users are presented with counterfactuals that are likely to produce the desired output, we propose *stochastic certainty*. For instance x, a sequence of actions A, black-box model M and the desired action a' stochastic certainty is defined as:

$$S(x, A, a') = 1 - P[M(x') = a' \quad | \quad x' = A(x)] \tag{3}$$

where $A(x)$ is a state obtained by applying actions from A to state x. Intuitively, stochastic certainty measures the probability of the desired outcome still being chosen by M after the time it takes to navigate to the counterfactual state. By maximizing stochastic certainty we promote sequences of actions that more often lead to the desired outcome.

Algorithm 1 Counterfactual heuristic tree search

Input: state x, desired outcome a', black-box model M, environment E
Parameters:number of iterations T
Output: counterfactual state x'
$t = \{x\}$ {Initializing search tree}
$i = 0$
while i < T **do**
  n = select(t) {Select state n to be expanded}
  S = expand(n) {Expand n by performing available actions}
  for all $s \in S$ **do**
    $val(s) = L(x, A, a')$ {Evaluate states in S according to L}
    $t+ = s$
  end for
  backpropagate() {Propagate values back to the root}
  $i+ = 1$
end while
$p = []$
for all $s \in t$ **do**
  if $valid(s)$ **then**
    $p+ = s$ {Filter valid counterfactuals}
  end if
end for
$cf = \arg\min_{s \in p} L(x, s(A), a')$ {Select the best counterfactual}

3.2 Loss Function

To optimize the counterfactual properties, we design a weighted loss function encompassing RL-specific objectives. For a state x, sequence of actions A, desired output a', loss function is defined as:

$$L(x, A, a') = \alpha R(x, A) + \beta F(x, A) + \gamma S(x, A, a') \tag{4}$$

where α, β and γ are parameters determining the importance of different properties. By minimizing L we can find a sequence of actions that quickly and certainly leads to a counterfactual explanation. However, $L(x, A, a')$ does not verify that a' is predicted in

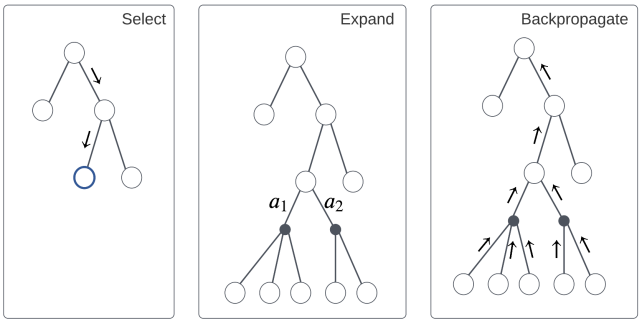

Figure 1: Heuristic tree search: in each iteration, a node is selected by navigating the tree from the root to a leaf by choosing actions according to the UCT formula. The node is expanded by performing all possible actions and appending all obtained states as children of the node. Finally, newly generated 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.

the obtained counterfactual. To that end, we additionally have to ensure that a validity constraint is satisfied:

$$V(x, x', a') = M(x') == a' \tag{5}$$

where x' is obtained by performing actions from A in x. Validity is used to filter potential counterfactual instances as is described in more detail in the next part of this section.

3.3 Counterfactual Search

Our goal is to obtain a sequence of actions A that minimizes the loss function L and satisfies the validity constraint. Unlike traditional counterfactual search which directly searches for a counterfactual in a data set, we are looking for an optimal sequence of actions that can transform the original state into a counterfactual one. This means that we cannot directly optimize L over a data set of states to find a counterfactual as this would give us no information about how difficult this counterfactual is to reach in terms of RL actions. To this end, we propose a counterfactual search algorithm that utilizes heuristic tree search to find a sequence of actions that transform the original into a counterfactual state that minimizes the loss function L. The details of the algorithm are given in Algorithm 1 and shown in Figure 1.

The proposed algorithm builds a tree to represent the agent's execution – each node corresponds to a state, and each edge to one action. 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. 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.

To calculate $val(n)$ we compute the value of $L(x, A, a')$, where A is the sequence of actions that navigates from root x 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 $L(x, A, a')$ we need to calculate the values of individual counterfactual properties of reachability, fidelity, and stochastic uncertainty for nodes in the tree. We calculate the reachability of node n as the length of the path between the root and n. Similarly, to calculate fidelity, we use the Q-values of state-action pairs on the path from the root to n according to Equation 2. Finally, to calculate stochastic certainty, we perform N simulations by unrolling the sequence of actions A from x in the environment and record the number of times a desired outcome is obtained in the resulting state. We then calculate stochastic certainty as:

$$S(x, A, a') = 1 - \frac{N(M(x') == a')}{N} \tag{6}$$

where x' is a state obtained after following A in x. We normalize the values for reachability so that they fall within the $[0, 1]$ range, while fidelity and stochastic uncertainty values naturally belong to that range. We can then evaluate a node in a tree by combining and weighting the three counterfactual properties to obtain $L(x, A, a')$ as shown in Equation 4.

At the start of the search, a tree is constructed with only the root node corresponding to the state x that is being explained. At each step of the algorithm, a node in the tree is chosen and the tree is expanded with the node's children. All actions are expanded simultaneously in the node. The resulting child nodes are then evaluated against L, and the results are propagated back to the tree root to update the value of nodes and edges. 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 [15]:

$$a^* = \arg\max_{a \in A} \left\{ Q(a, n) + C\sqrt{\frac{\ln(N(n))}{N(s, a)}} \right\} \tag{7}$$

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 paths of high value and exploring underrepresented paths through the exploration constant C. The process is repeated until a predetermined maximum number of iterations T is reached.

Once the tree is fully grown, all nodes are first filtered according to the validity constraint (Equation 5) to remain only with the states that deliver the desired output. 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.

4 EXPERIMENTS

In this section, we outline the experiment setup for evaluating RACCER. We compare RACCER to the only other model-agnostic algorithm for generating counterfactuals in RL – GANterfactual-RL [12]. In Section 4.1 we describe evaluation tasks.

Table 1: Parameters used for generating counterfactual explanations for GANterfactual-RL and RACCER approaches in Stochastic GridWorld and Frozen Lake environments.

Parameter \ Task	Stochastic GridWorld	Frozen Lake
Number of iterations (T)	300	300
Number of simulations (N)	10	10
Maximum number of actions (k)	20	20
Evaluation dataset size ($\|D\|$)	500	400
Loss parameter α	-1	-1
Loss parameter β	-1	-1
Loss parameter γ	-1	-1

4.1 Evaluation Tasks

We evaluate our approach in two environments – Stochastic GridWorld and Frozen Lake.

4.1.1 Stochastic GridWorld. Stochastic GridWorld is a simple 5×5 grid world, where the agent is tasked with shooting 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. The environment also contains trees and walls in the middle file of the grid, that 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 near it. However, trees are less costly to chop down than the walls. At each step, the agent can move one step in any direction or perform SHOOT and CHOP actions. Additionally, along the middle file of the board, trees can regrow and walls can be rebuilt with different probabilities. Actions receive a -1 penalty, and successfully shooting the dragon brings a +10 reward. The episode ends when the dragon is shot or when the maximum number of time steps is reached. We consider all states that contain an agent, a dragon, and have trees and walls only along the middle file of the grid to be realistic under the game rules.

In this task, two states can appear similar but be far away in terms of execution, due to the obstacles on the grid. Chopping down a tree to be able to shoot the dragon might be less preferable than going around it, and suggesting this to the user could save them time and effort. Similarly, due to the stochastic nature of the task, during the time needed to obtain a counterfactual, new trees and walls can regrow and block the agent's path to the dragon.

4.1.2 Frozen Lake. The frozen lake is a well-known stochastic grid world environment, in which the agent is tasked with reaching the goal while navigating a grid where some squares are covered in ice. Making an action in icy states can either lead the agent to a desired state or leave them in the same one with some probability. All actions carry a -1 reward, while successfully navigating to the goal brings the agent +10. We consider all states that contain an agent and the goal to be realistic under the rules of the game.

In this environment, two states with very similar features can be far away. For example, even if there is only one square difference between the agent's locations in two states, it might still be difficult to reach one from the other given the stochastic nature of the environment.

5 EVALUATION

To evaluate RACCER, we compare it to the baseline approach GANterfactual-RL [12] in Stochastic GridWorld and Frozen Lake environments. We establish 5 hypotheses for evaluating RACCER:

- **H1:** Counterfactual explanations generated by RACCER will perform better on RL-specific metrics of fidelity, stochastic uncertainty, and reachability compared to the baseline.
- **H2:** RACCER is more suitable for producing counterfactuals for explaining suboptimal agents compared to the baseline.
- **H3:** RACCER will produce counterfactuals that can help users better understand and predict the behavior of RL agents compared to the baseline.
- **H4:** RACCER will produce counterfactuals that help users better choose between agents with different preferences compared to the baseline.
- **H5:** Counterfactual explanations generated by RACCER will be perceived as more satisfactory by users compared to explanations generated by the baseline.

To evaluate H1 and H2, we evaluate the counterfactual properties of explanations generated by RACCER and GANterfactual-RL. Hypothesis H1 is explored in Section 5.1.1, and H2 in Section 5.1.2. Hypotheses, H3, H4 and H5 are evaluated through a user study described in detail in Section 5.2. Hypothesis H3 is evaluated in Section 5.2.1, H4 in Section 5.2.2 and H5 in Section 5.2.3.

5.1 Evaluating Counterfactual Properties

We evaluate RACCER and GANterfactual-RL based on both feature-based counterfactual properties (proximity, sparsity, validity, and realistic counterfactual) and RL-specific properties (reachability, fidelity, stochastic uncertainty).

To evaluate proximity, sparsity, and validity we use metrics defined in Huber et al. [12] originally used to evaluate GANterfactual-RL. For proximity, we use the L1 distance between instances:

$$P(x, x') = 1 - ||x - x'||_1 \tag{8}$$

Sparsity is calculated as the number of non-modified features when transforming the original instance x into a counterfactual x':

$$S(s, s') = \frac{||x - x'||_0}{S} \tag{9}$$

where S is the total number of features.

Validity denotes whether the target action a' is chosen by the black-box model B in the counterfactual instance x':

$$V(x') = B(x') == a' \tag{10}$$

Additionally, we also evaluate whether the resulting counterfactuals are realistic. What constitutes a realistic counterfactual is task-specific and is described in more detail in Section 4.1.

Furthermore, we evaluate RACCER and GANterfactual-RL according to RL-specific properties presented in Section 3.1. Evaluating these properties for RACCER is straightforward as it uses tree search to navigate to the counterfactual. Properties can be calculated by analyzing the sequence of actions leading from the root to the counterfactual. GANterfactual-RL, however, generates counterfactual using generative models and uses no notion of actions. To

Table 2: The average values of counterfactual properties for counterfactual explanations of a fully-trained agent generated using GANterfactual-RL and *RACCER* approaches in Stochastic GridWorld and Frozen Lake.

Task	Stochastic Gridworld		Frozen Lake	
Metric \ Approach	GANterfactual-RL	RACCER	GANterfactual-RL	RACCER
Generated counterfactuals (%)	**100**	75.4	**100**	80.75
Realistic counterfactuals (%)	76.0	**100**	**100**	**100**
Proximity (↑)	0.98	**0.99**	0.90	**0.96**
Sparsity (↓)	0.19	**0.11**	0.61	**0.14**
Validity (↑)	0.58	**1.0**	0.46	**1.0**
Reachability (↓)	0.98	**0.13**	1.0	**0.15**
Fidelity (↓)	1.0	**0.79**	1.0	**0.6**
Stochastic uncertainty (↓)	0.99	**0.18**	1.0	**0.08**

measure reachability, fidelity, and stochastic certainty for a counterfactual x' generated by GANterfactual-RL, we build a tree of the agent's execution of length k rooted in x and find x' in it. That way, we estimate properties that rely on actions even for explanations generated through the direct search for counterfactual states. If x' cannot be found in the tree, it is assigned the least desirable value for an RL-specific counterfactual property which is 1.

5.1.1 Explaining Fully-trained Agents. We start by comparing explanations generated by RACCER and GANterfactual-RL when explaining a fully trained agent. We obtain a fully trained black-box model M by training a DQN on the task until convergence. We then apply RACCER and GANterfactual-RL approaches to generate counterfactuals for explaining M. The parameters for generating counterfactuals using both algorithms are given in Table 1. Both RL-specific and feature-based counterfactual properties are evaluated for the generated counterfactuals. We also record what percentage of counterfactuals were successfully created and if they were realistic. The results for both tasks are recorded in Table 2.

In both environments, RACCER performs better on both feature-based and RL-specific counterfactual metrics. While we expected RACCER to perform better on RL-specific properties, it is surprising that it outperforms GANterfactual-RL in feature-based metrics, as GANterfactual-RL has been trained to optimize these. We speculate that this is because the GANterfactual-RL approach has been optimized for visual tasks, unlike discrete environments used in this work. RACCER also produces only realistic counterfactuals as it follows the rules of the environment. GANterfactual-RL, on the other hand, often changes features outside of the agent's control such as adding or removing tree and wall features in the Stochastic GridWorld environment, resulting in fewer realistic states. Finally, RACCER generates counterfactuals that are more often valid.

One metric in which RACCER performs worse compared to the baseline is the number of generated counterfactuals. Due to its underlying generative model, GANterfactual-RL can generate a counterfactual for each fact and target action. RACCER, however, searches the space of the agent's interactions to find a counterfactual. If the agent is very unlikely to play a certain action in the environment, RACCER will not be able to generate a counterfactual for this action. For example, if the dragon in the Stochastic GridWorld is located in the rightmost file of the grid, a well-trained agent will never need to play action LEFT. By examining the factual states and target actions for which RACCER does not generate a counterfactual we find that a large majority of them correspond to states where the target action would never be played by a well-trained agent. Specifically, in Stochastic GridWorld out of 123 situations where RACCER does not generate a counterfactual, in 80 of them (65.04%) target action would never be played by an agent. This means that RACCER fails to find a counterfactual in a situation where that is possible only 43 times, or for 8.6% of situations. Similarly, in Frozen Lake, out of 77 situations in which RACCER does not find a counterfactual, 72 corresponds to such impossible situations. In the Frozen Lake task, RACCER fails to generate counterfactuals where that is possible only 5 times, or for 0.0125% situations.

5.1.2 Explaining Suboptimal Agents. Most often, counterfactual explanations have been applied to explain the behavior of fully-trained agents. However, understanding suboptimal agents is necessary for verification and debugging. For example, Olson et al. [20] have used counterfactuals to help users recognize 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 hypothesize 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 train a suboptimal agent M_{sub} in both tasks, we use a DQN model, but train it for one-tenth of the time used to train the fully-trained agent. We evaluate RACCER and GANterfactual-RL on proximity, sparsity, and validity, as well as reachability, fidelity, and stochastic uncertainty. Additionally, we record the percentage

Table 3: The average values of counterfactual properties for counterfactual explanations for a sub-optimal agent generated using GANterfactual-RL and *RACCER* approaches in Stochastic GridWorld and Frozen Lake for a suboptimal agent M_{sub}.

Task	Stochastic Gridworld		Frozen Lake	
Metric \ Approach	GANterfactual-RL	RACCER	GANterfactual-RL	RACCER
Generated counterfactuals (%)	**100**	77.6	**100**	43.50
Realistic counterfactuals (%)	47.00	**100**	**100**	**100**
Proximity (↑)	0.98	**0.99**	0.87	**0.97**
Sparsity (↓)	0.24	**0.11**	0.81	**0.14**
Validity (↑)	0.55	**1.0**	0.2	**1.0**
Reachability (↓)	0.99	**0.14**	1.0	**0.10**
Fidelity (↓)	1.0	**0.82**	1.0	**0.76**
Stochastic uncertainty (↓)	1.0	**0.26**	1.0	**0.07**

of generated counterfactuals as well as the percentage of realistic counterfactuals. The results are presented in Table 3.

RACCER performs comparably when explaining a sub-optimal agent M_{sub} (Table 3) and the fully-trained agent M (Table 2) in both tasks. In contrast, counterfactuals generated by GANterfactual-RL show a decline in counterfactual properties when explaining a suboptimal model compared to a fully trained model. In the Frozen Lake environment, GANterfactual-RL achieves lower validity when explaining a suboptimal model compared to a fully-trained model. Similarly, in the Stochastic Gridworld task, GANterfactual-RL generates counterfactuals that are far less realistic compared to those generated for a fully-trained model.

5.2 User Study

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 evaluate hypotheses H3, H4, and H5 we conducted a user study to compare the counterfactual explanations produced by GANterfactual-RL and RACCER. We conducted the study in the Stochastic GridWorld environment, as it has simple rules, and requires no prior knowledge from users.

We sourced 153 participants through the Prolific platform 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 GANterfactual-RL and the second counterfactuals produced by RACCER. After filtering participants for those who had passed attention checks, 58 participants remained in the first and 63 in the second group. Participants were remunerated for their time according to the Prolific payment policy.

The study consisted of 3 parts – evaluating user understanding of the agent's behavior, evaluating user understanding of the agent's preferences, and evaluating user satisfaction. The study design follows that used to evaluate the GANterfactual-RL algorithm in Huber et al. [12]. 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. Section 5.2.1 covers the evaluation of agent's behavior, Section 5.2.2 understanding of agents' preferences, and Section 5.2.3 user satisfaction.

5.2.1 Agent Understanding. To evaluate how well users understand agent's behavior we use a user study setup similar to that of Huber et al. [12]. Users are shown the behavior of two agents A and B with different policies, described in more detail in Section 5.2.2. For each agent users go through two stages – training and testing stage. 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, an example of which is shown in Figure 2. 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.

The factual states in the training and testing phase are selected by the HIGHLIGHTS-DIV algorithm [1], inspired by the setup from Huber et al. [12]. 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 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 game-play.

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 behavior. Users who have seen counterfactuals generated by RACCER 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

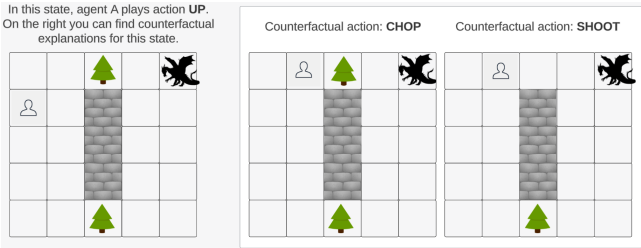

Figure 2: Example of the counterfactual explanation shown to the user during the training phase.

Mann-Whitney U test we find a significant difference between the prediction accuracy of the two approaches (p = 0.0185). This proves H3 and **indicates that RL-specific counterfactuals help users better understand and predict the behavior of RL agents.**

5.2.2 Agent Comparison. In the study, users were asked to evaluate agents A and B one after the other. Agents A and B are fully trained on the task. However, they are trained on different reward functions and have different preferences for completing the task. Agent A prefers to take the longer, but cheaper paths in the environment, while Agent B does not care about the cost and wants to finish the task as quickly as possible. The difference in the behavior between the two agents is exhibited most clearly in their interaction with the wall features. When faced with a wall obstacle, Agent A chooses to go around it to reduce costs, while Agent B chooses to chop down the wall despite the high cost to finish the task quicker.

Users were presented with the training and testing phase for one agent, followed by the training and testing phase for the second agent. The users are informed when they will be switching from one agent to the other. After seeing both agents, users are 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 would they choose to finish the task quickly, regardless of the cost.

Users presented with RACCER 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 (p = 0.6509). This indicates that **contrary to H4, RACCER is not better at helping users distinguish between agents with different policies compared to GANterfactual-RL.**

5.2.3 User Satisfaction. At the end of the study, users were asked to rank the explanations based on the *explanation goodness metrics* [11] 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.

The results of this part of the study are presented in Figure 3. After conducting a non-parametric one-tailed Mann-Whitney U test we find that users perceive explanations generated by **RACCER 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 GANterfactual-RL approach.** However, there is no significant difference between the approaches in the perceived trustworthiness (p = 0.7901), reliability (p = 0.1446),

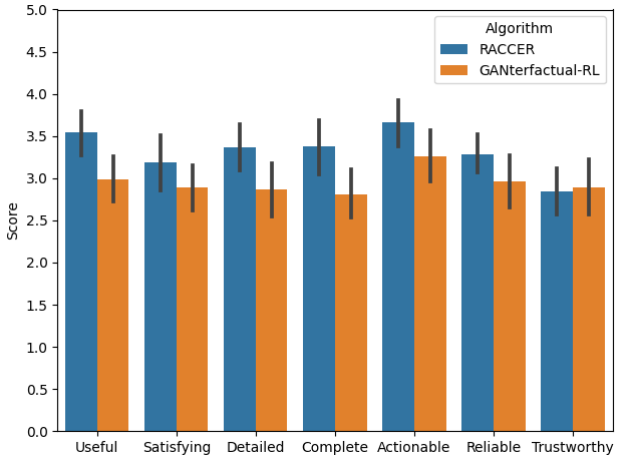

Figure 3: Users' scores on explanation goodness metrics [11] for counterfactual explanations generated using RACCER and GANterfactual-RL algorithms.

and actionability (p = 0.0729) of explanations, resulting in H5 being only partially confirmed by our experiments.

6 CONCLUSION AND FUTURE WORK

In this work, we presented RACCER, the first RL-specific approach to generating counterfactual explanations. We designed and implemented three novel counterfactual properties that reflect the sequential and stochastic nature of RL tasks, and provided a heuristic tree search approach for optimizing these properties. We evaluated our approach in Stochastic GridWorld and Frozen Lake environments and showed that RACCER generates counterfactuals that are easier to reach and provide the desired outcomes more often compared to baseline approaches. We have also conducted a user study, and shown that RACCER helps users better predict the behavior of RL agents, and produces explanations that are perceived as more useful, detailed,d and complete compared to GANterfactual-RL.

In this work, we have limited our search to only the best counterfactual. In future work, we hope to expand our search to include a set of diverse counterfactual explanations optimizing different counterfactual properties. In this way, users would have a wider choice of potentially actionable advice. Additionally, we have only explored the prefactual explanations which explore how changes in the current state can lead to different outcomes. In future work, we hope to investigate counterfactuals that explore past decisions and compare them to prefactuals in RL.

ACKNOWLEDGEMENTS

This publication has emanated from research supported in part by grants from Science Foundation Ireland under grant number 18/CRT/6223 and SFI Frontiers for the Future grant number 21/FFP-A/8957. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] Dan Amir and Ofra Amir. 2018. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 1168–1176.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[3] Ruth MJ Byrne. 2019. Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning.. In *IJCAI*. 6276–6282.

[4] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8789–8797.

[5] Xinyue Dai, Mark T Keane, Laurence Shalloo, Elodie Ruelle, and Ruth MJ Byrne. 2022. Counterfactual explanations for prediction and diagnosis in XAI. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*. 215–226.

[6] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. 2020. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*. Springer, 448–469.

[7] Eoin Delaney, Derek Greene, and Mark T Keane. 2021. Uncertainty estimation and out-of-distribution detection for counterfactual explanations: Pitfalls and solutions. *arXiv preprint arXiv:2107.09734* (2021).

[8] Jasmina Gajcin and Ivana Dusparic. 2022. Counterfactual Explanations for Reinforcement Learning. *arXiv preprint arXiv:2210.11846* (2022).

[9] Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. 2019. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems* 34, 6 (2019), 14–23.

[10] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–42.

[11] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for explainable AI: Challenges and prospects. *arXiv preprint arXiv:1812.04608* (2018).

[12] Tobias Huber, Maximilian Demmler, Silvan Mertes, Matthew L Olson, and Elisabeth André. 2023. GANterfactual-RL: Understanding Reinforcement Learning Agents' Strategies through Visual Counterfactual Explanations. *arXiv preprint arXiv:2302.12689* (2023).

[13] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. 2020. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050* (2020).

[14] Amir-Hossein Karimi, Julius Von Kügelgen, Bernhard Schölkopf, and Isabel Valera. 2020. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. *Advances in Neural Information Processing Systems* 33 (2020), 265–277.

[15] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[16] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2017. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443* (2017).

[17] Arnaud Van Looveren and Janis Klaise. 2021. Interpretable counterfactual explanations guided by prototypes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 650–665.

[18] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence* 267 (2019), 1–38.

[19] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 607–617.

[20] Matthew L Olson, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. 2019. Counterfactual states for atari agents via generative deep learning. *arXiv preprint arXiv:1909.12969* (2019).

[21] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. FACE: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 344–350.

[22] Erika Puiutta and Eric Veith. 2020. Explainable reinforcement learning: A survey. In *International cross-domain conference for machine learning and knowledge extraction*. Springer, 77–95.

[23] Robert-Florian Samoilescu, Arnaud Van Looveren, and Janis Klaise. 2021. Model-agnostic and Scalable Counterfactual Explanations via Reinforcement Learning. *arXiv preprint arXiv:2106.02597* (2021).

[24] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency*. 10–19.

[25] Sahil Verma, John Dickerson, and Keegan Hines. 2021. Counterfactual Explanations for Machine Learning: Challenges Revisited. *arXiv preprint arXiv:2106.07756* (2021).

[26] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.