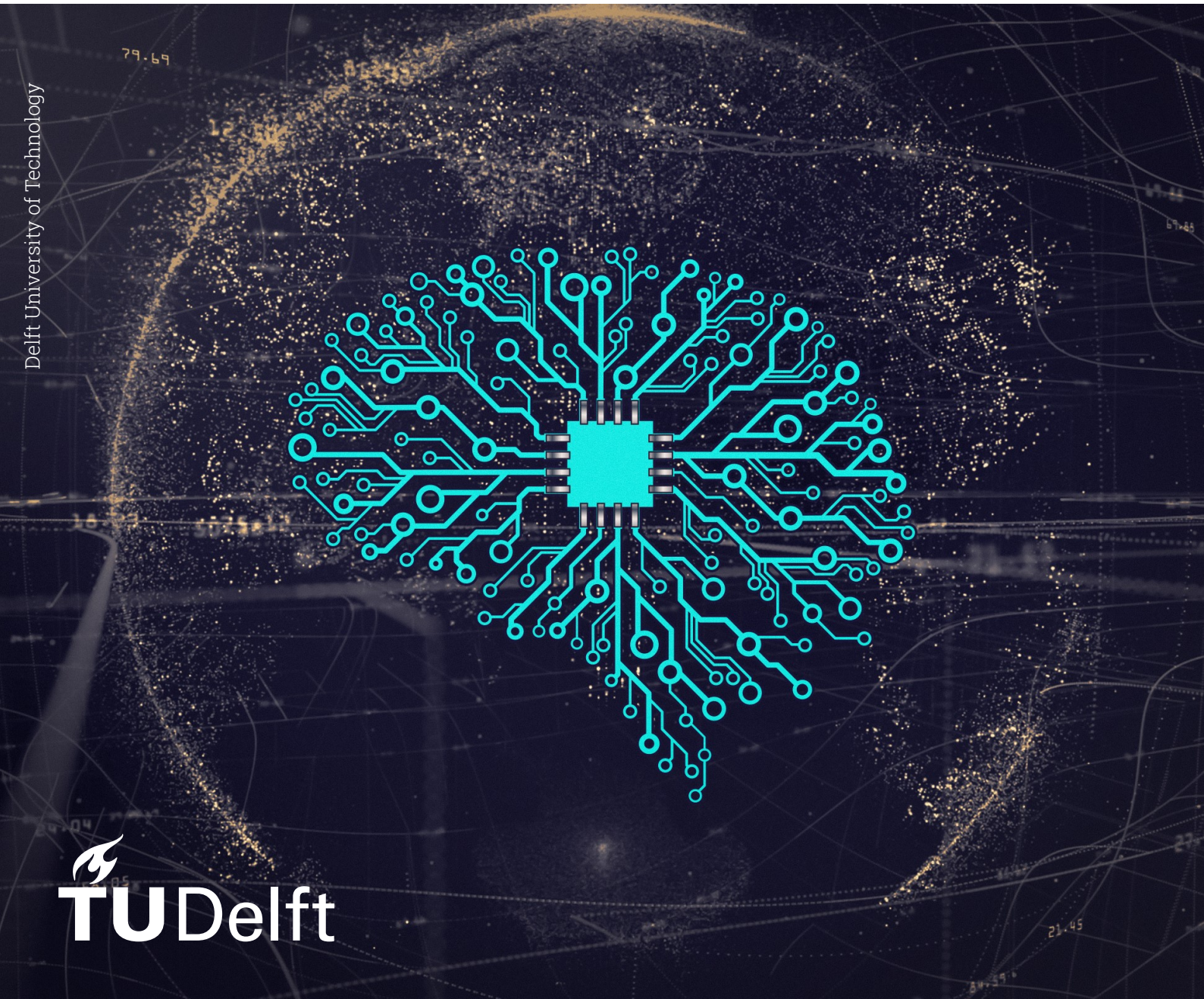
Counterfactual Explanations of Learned Reward Functions

Jan Wehner



Delft University of Technology

TU Delft

Counterfactual Explanations of Learned Reward Functions

by

Jan Wehner

Student Name	Student Number
Jan Wehner	5622735

Research group:	Interactive Intelligence
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft
Project Duration:	February 2023 - November 2023
Thesis Committee:	Prof. dr. L. C. Siebert TU Delft, supervisor
	Prof. dr. F. Oliehoek TU Delft, supervisor
	Prof. dr. U. Gadiraju TU Delft, external member

Cover: Mike MacKenzie via `www.vpnsrus.com` (CC BY 2.0)

TU Delft

Preface

As AI systems become widely employed this technology will profoundly impact society. To ensure this impact is positive it is essential to align these systems with the values and intentions of the human stakeholders. This presents a crucial open challenge in AI that needs to be solved before these systems can be employed to act autonomously in the interests of humans.

I believe learning a reward function from a human represents a promising approach to this challenge. However, when assessing the literature on reward learning and using these algorithms, it struck me that we can not simply rely on these techniques, since they are often imperfect and have considerable theoretical limitations. These problems need to be addressed before reward learning is a genuine approach to safely developing AI systems that are aligned with human values.

To address this shortcoming, I imagined an iterative process of refining the alignment between humans and AI through demonstrations and explanations. Specifically, assisting the human in finding the flaws in the objective function of an AI could help the human understand misalignments and then provide the AI agent with new demonstrations of desired behaviour. It struck me as especially fruitful to leverage techniques from the field of eXplainable AI in this setting. So this thesis is an attempt to transfer a technique from eXplainable AI, namely Counterfactual Explanations, to the setting of explaining a reward function to a human.

This report is composed of a main scientific article and a set of Appendices. The scientific article presents this research project's main motivation and results. The appendix presents a further set of experiments that support the experiments in the paper and discusses the available literature, concrete implementations and implications of the research in more depth.

This research was conducted with the Interactive Intelligence Group at the Delft University of Technology under the supervision of Prof. Dr. Luciano Cavalcante Siebert and Prof. Dr. Frans Oliehoek.

Jan Wehner
Delft, October 2023

Contents

Nomenclature

Abbreviations

Abbreviation	Definition	Context
CTE	Counterfactual Trajectory Explanation	Type of Explanations consisting of pair of original and counterfactual trajectory
MCTS	Monte Carlo Tree Search	Method for generating CTEs
DaC	Deviate and Continue	Method for generating CTEs
RL	Reinforcement Learning	Area of Machine Learning
AIRL	Adversarial Inverse Reinforcement Learning	Reward Learning Algorithm
PPO	Proximal Policy Optimization	Reinforcement Learning Algorithm
NN	Neural Network	Type of Machine Learning Model
LM	Linear Model	Type of Machine Learning Model

Symbols

Symbol	Definition
R_θ	Learned Reward Function
R^*	Ground truth Reward Function
π_θ	Policy trained on R_θ
π^*	Policy trained on R^*
τ_{exp}	Trajectories generated by π^*
τ_{org}	Trajectories generated by π_θ
t_{org}	Partial original trajectory extracted from from $t_{org} \subseteq \tau_{org}$
t_{cf}	Partial counterfactual trajectory generated by a generation method
ω	Uniformly sampled set of weights for the quality criteria
$\omega_{criterion}$	The weight assigned to one quality criterion $\omega_{criterion} \in \omega$
ρ	quality values; the normalised, weighted sum of quality criteria

1

Scientific Article

Counterfactual Explanations of Learned Reward Functions

Jan Wehner

Wednesday 25th October, 2023

Abstract:
Learning rewards from humans is a promising approach to aligning AI with human values. However, methods are not able to consistently extract the correct reward functions from demonstrations or feedback. To allow humans to understand the limitations and misalignments of a learned reward function we adopt the technique of counterfactual explanations from the field of eXplainable AI (XAI). Concretely, we propose Counterfactual Trajectory Explanations (CTEs) as an approach to contrast an original with a counterfactual partial trajectory and the rewards they receive. We devise and test 2 methods for generating CTEs of which a generation method based on Monte Carlo Tree Search proves to be the most effective. The CTEs are optimised for 6 quality criteria that were derived from the literature and tested experimentally. We found that most quality criteria are beneficial for creating more informative CTEs, while Validity stands out as contributing especially much to making explanations informative. Finally, we measure how informative the generated explanations are to a proxy-human model. While the model is not able to capture all aspects of the reward function, it does learn a substantial amount of knowledge that generalises to different trajectory distributions from the CTEs. These results present the use of counterfactuals, and more generally XAI methods, on learned reward function as a promising avenue for further inquiry.

1.1. Introduction

With the growing adoption of Reinforcement Learning (RL) models in real-world use cases like healthcare [89], autonomous driving [38] or recommender systems [2], and the increasing capabilities of these models, it is more pressing than ever to consider how we can ensure that these models are safe and aligned with human values [22, 91]. One core difficulty of achieving trustworthy [13] and controllable AI [12, 63] is to accurately capture the intentions and preferences of the designers in the reward function on which the RL agent is trained [56]. For example, an autonomous driving system trained on a misspecified reward function [39] could display undesired driving behaviour [87], cause dangerous situations [34] or make objectionable ethical trade-offs [24]. Looking further ahead, aligning the objectives that powerful AI systems are trained on with human values is a key open problem [27, 4] for mitigating catastrophic risks from AI [63].

For many tasks, it is hard to manually specify reward functions that accurately capture the intentions of the human stakeholders, preferences of users or the values of society at large [57, 4]. Reward Learning is a set of techniques that circumvent this problem by instead learning the reward function from data. We specifically focus on Inverse Reinforcement Learning (IRL) [55] which aims to retrieve the objective function of an expert from demonstrations they generate. When applied to human data this makes reward learning a promising approach for aligning the reward functions of AI systems with the intentions of humans [63, 42].

However, even if we assume the existence of one consistent expert reward function, IRL often learns flawed reward functions that differ from the learning target. This could arise because IRL is an ill-posed problem [1], wrong assumptions about the rationality of the expert [69, 5], or other failures in learning. Furthermore, learning human values via IRL is made more complicated [81] by the diverse [77], dynamic [53, 18, 61] and context-dependent [43] nature of human values. To avert this misalignment between the true and learned reward function we aim to empower humans to evaluate and correct the learned reward function [31]. To do this, humans need to be able to understand the reward function, which allows them to identify general patterns or specific situations for which it diverges from their judgement.

Currently, it is hard for humans to understand the reward function learned by IRL. This especially applies to Deep IRL [84], where a Neural Network represents the learned reward function. While Deep IRL allows the method to learn more complex reward functions, it also makes the learned reward function an uninterpretable black box, especially because the networks can be very large in scale. To address this Sanneman and Shah propose the *"Transparent Value Alignment"* framework [65] in which eXplainable AI (XAI) methods are applied to provide explanations of the learned values to the human. However, many XAI methods have not yet been adapted to explain reward functions and there have been few attempts [47, 30, 62] to develop methods for interpreting deep reward functions. Doing so requires to reconceptualise the methods so they can uncover misalignments.

Counterfactual explanations are one promising XAI technique that can be applied to this setting. Coun-

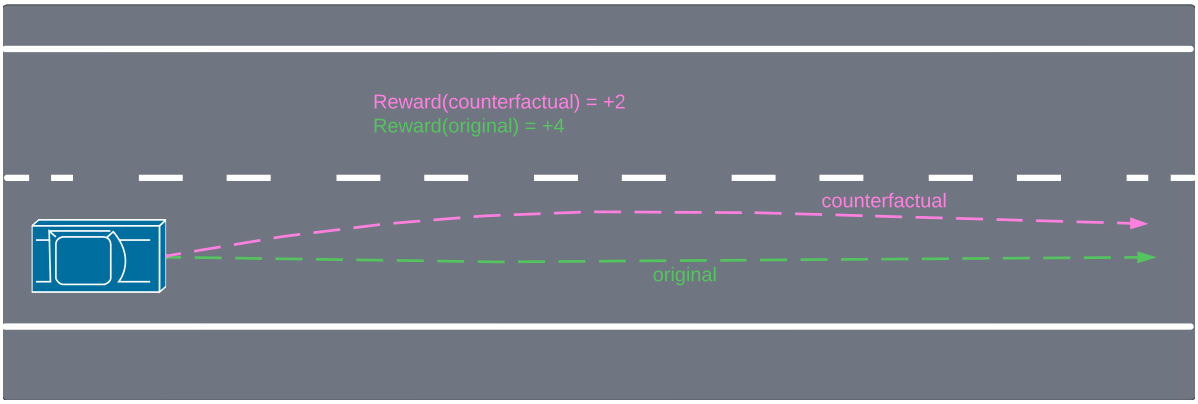


Figure 1.1: A car has originally taken a straight line and received a reward of $+4$ from the reward function. By providing a counterfactual that receives a lower reward of $+2$ we can make hypotheses about how the reward function assigns rewards.

terfactual Explanations are ”if-then” statements that contrast reality with an alternative scenario. They have been shown to enable users to better understand the decisions made by AI systems [52, 79]. These methods are also supported by a wealth of psychology research which shows how fundamental counterfactuals are to human understanding and reasoning by allowing us to reason about alternative scenarios [11, 60, 17, 46, 32, 48].

Figure 1.1 illustrates how a counterfactual trajectory can aid in understanding how a reward function rates driving behaviour. If the explanation only shows the original trajectory with the reward $+4$ assigned to it, it is hard to draw higher-level conclusions about the reward function that generalise to different contexts. Adding the counterfactual trajectory, which got the reward $+2$ allows the human to form or validate hypotheses about what behaviour the reward function (dis-)incentivises. For example, the displayed explanation could support the hypothesis that the reward function disincentivises the driver from swerving and getting close to the other lane. Instead of showing a set of independent trajectories, counterfactual trajectories allow the user to learn more efficiently from them, because they allow the user to make causal inferences [45] and thus efficiently interpret the underlying mechanism driving the decision-making [80].

While there is work that aims to explain reward functions to users [66, 64, 8, 33, 35], there have been few attempts to do this for deep reward functions [47, 70]. Furthermore, counterfactual explanations have only recently been applied to RL [23, 44, 20, 71].

This study makes a novel connection between XAI and Reward Learning by, to the best of the author’s knowledge, being the first to automatically generate counterfactual explanations about learned reward functions. We hope that such counterfactual explanations about reward functions can be used in a cyclical framework as displayed in Figure 1.2 to iteratively and interactively improve the alignment between a human’s values and the learned reward functions.

The overarching question guiding this research is: **How can we create counterfactual explanations that serve as informative explanations about deep reward functions?**

This requires us to answer several research sub-questions (RQs) that are connected to the contributions of this thesis:
RQ 1: How can counterfactual explanations be reframed to the reward learning setting? We develop the notion of Counterfactual Trajectory Explanations (CTEs) that explain reward functions. Furthermore, we survey criteria that counterfactual explanations should meet, and adapt and operationalise them in the setting of reward functions.
RQ 2: How to generate counterfactual trajectories for reward functions? Two methods that generate CTEs by optimising for the quality criteria are developed and tested.
RQ 3: How informative are the generated counterfactual explanations? We develop a novel measure for the informativeness of counterfactual explanations based on a proxy-human regression model that learns from provided explanations. With this, we evaluate which quality criteria and generation methods lead to more informative explanations.

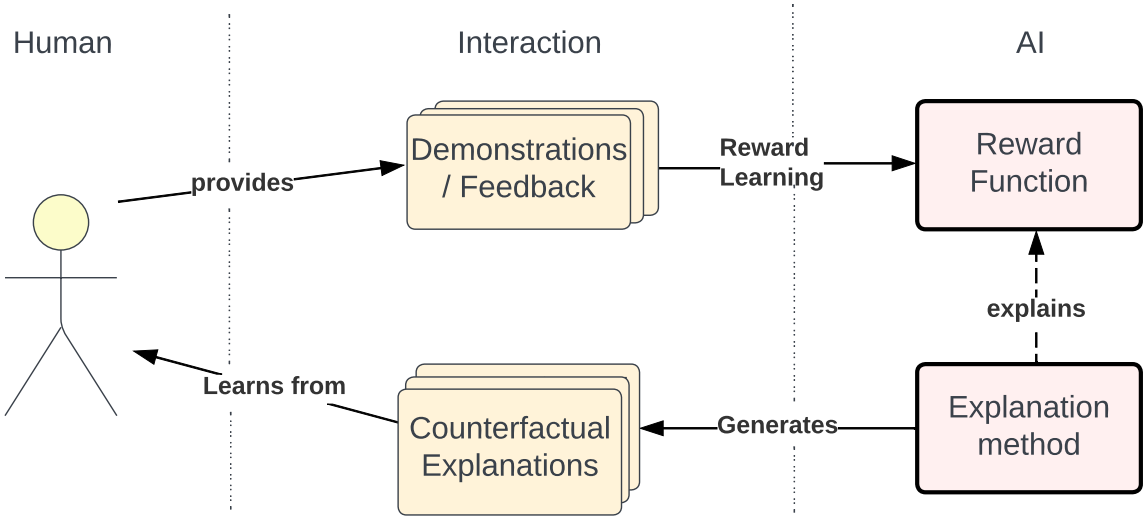


Figure 1.2: Humans provide data in the form of demonstrations or preferences, that contains information about their intentions or values. From this a reward learning algorithm learns a reward function. Next, an explanation method generates counterfactual explanations about the reward function and provides them to the human. Based on this the human can provide new data that specifically corrects on the detected misalignments

The remaining article is structured as follows. We start by surveying previous work about explaining reward functions and counterfactual explanations (Section 1.2) and set out clearly define the problem and our approach to address it (Section 1.3). Next, we describe the methods developed for creating counterfactual explanations (Section 1.4) and how they are evaluated (Section 1.5). Section 1.6 show the results of three experiments and interpret them with regard to the research questions. Finally, we conclude by laying out limitations and future work (Section 1.7). [1]

1.2. Related Work

This section covers previous work on understanding and testing learned reward functions. Further, it introduces and justifies the use of counterfactual explanations and points to some examples of previous uses of this technique. A detailed discussion of the literature can be found in Appendix A.

1.2.1. Understanding Learned Reward Functions

To position this research project we survey previous work that has the goal of helping humans to understand reward functions. First, we survey different approaches for directly evaluating the quality of a learned reward function and then survey approaches for helping humans in interpreting reward functions.

Evaluating Learned Reward Function

Evaluations of the quality of a learned reward function R_θ are often based on the true reward function R^* [92, 14, 90, 74, 9]. Several researchers propose direct evaluation by comparing the learned reward function with the true function, using methods like Equivalent-Policy Invariant Comparison (EPIC) [25] or Dynamics-Aware Reward Distance (DARD) [82]. Testing how well reward learning algorithms generally work can also be conducted through tools and benchmarks testing on specific algorithms [19, 73]. Furthermore, the policies trained on R_θ and R^* can be compared [86, 83, 85, 54].

The presented methods for evaluating learned reward function assume that there is access to the ground truth reward function or conflate policy learning with reward comparisons. Assuming access to a ground-truth reward function makes the method less applicable since reward learning is specifically useful when we cannot specify the true reward function. Conflating policy learning and reward comparison makes it is unclear whether differences are caused by misalignments of the reward function or by

[1] The full code for the project is available at: `https://github.com/TheGermanCodeMachine/Counterfactual-Trajectory-Explanations-for-Learned-Reward-Functions`

failures of an RL algorithm to retrieve the optimal policy. This necessitates the search for evaluation methods that don't rely on these assumptions.

Interpreting Reward Functions

Enabling the human to understand the reward function by making reward learning interpretable could enable them to evaluate the reward function, without being able to write down a ground truth reward function.

Intrinsically interpretable reward functions

Intrinsic Interpretability applies to Machine Learning models inherently designed for easy comprehension, albeit sometimes restricting predictive accuracy. Tree-based models are seen as intrinsically interpretable, because due to explicit decision procedures. In the literature, binary [8] and differentiable [33] decisions trees have been tested [7] and applied [70] as reward functions. Other methods, such as learning a linear temporal logic formula [35] or conducting IRL in a relational domain [54], also offer easier interpretation than deep reward function.

Our proposed method generates explanations without having to restrict the architecture of the learned reward function and thus allows for the use of more capable models.

Post-hoc interpretability for reward functions

Post-hoc interpretability methods, applied after model training, illuminate the model's decision-making process. The closest work to ours comes from Michaud et al. [47] who apply gradient salience and occlusion maps, as well as handcrafted counterfactual input to understand learned reward functions that help them to identify flaws and misalignments. In contrast, our counterfactual inputs are partial trajectories instead of single states and are automatically generated instead of handcrafted.

Jenner and Gleave [30] apply equivalence transformations that simplify the reward function. Similarly, Russel and Santos [62] train a more interpretable decision tree to replicate a deep reward function.

Lastly, Lindsey and Shah conduct experiments that measure how different explanation techniques enhance human comprehension of the reward function, while also considering the cognitive workload this places on the user [64, 66].

Our proposed method also generates Post-Hoc explanations about the reward function but does so by automatically generating counterfactual examples of how the reward function judges behaviour.

1.2.2. Counterfactual Explanations

One popular type of explanation used in XAI to help humans understand and interpret predictions by ML models is Counterfactual Explanations. After laying out a general definition for counterfactuals we lay out their advantages and their previous applications to RL.

Definition of Counterfactual Explanations:

Counterfactual explanations are a popular tool in XAI to help humans understand and interpret predictions by ML models. Counterfactuals are hypothetical scenarios that pose "what-if" questions, offering alternative instances that would have led to a different outcome. For example, an applicant for a loan might wonder why she was rejected and then receive the explanation: "If your income would be 1500/year higher, you would have gotten the loan". This gives users clear, actionable insight into why the decision was made and how they would need to change the inputs to get a different decision.

More formally, Wachter et al. [79] define counterfactual explanation as statements:

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

Why use Counterfactual Explanations?

Humans use counterfactuals to reason about alternative scenarios to learn from their experience and improve their behaviour in the future [11, 60, 17]. They also play a critical role in causal reasoning, planning and blame assignment [46, 32, 48]. The fact that humans can so readily reason about and learn from counterfactuals, makes them intuitive and easy to comprehend even for non-expert users [52]. Further, counterfactuals provide users with local and actionable explanations, allowing them to

understand why a specific decision was made and what changes would be required to achieve the desired outcome [79]. However, critics point out the risk of generating unjustified counterfactual explanations [40] and call for caution when making counterfactuals about social categories such as race or gender [36].

Previous use of and work on counterfactuals:
There is a large body of work on generating counterfactual explanations for ML models, of which the majority focuses on supervised learning problems. Multiple surveys explore methods for generating counterfactual explanations [76, 6, 26, 72] and their connection to human psychology [37, 10].

While most work on Counterfactual Explanation addresses Supervised Learning, adapting this method to Reinforcement Learning requires adaptation. When making counterfactual explanations to explain the behavior of RL agents, changes can be made to the Features, Goals, Objectives, Events, or Expectations of the agent in order to change the pursued Actions, Plans, or Policies [23]. This can be done to improve the users understanding of out-of-distribution behaviour [20], provide them with more informative demonstrations [41] or showcase how an agent's environmental beliefs influence its planning [71].

However, so far there has been no principled attempt to leverage counterfactual explanations in the setting of Reward Learning. While the work described previously aims to explain a policy π we aim to explain a reward function R.

1.3. Preliminaries

This section sets up some important methods, notations and definitions used throughout the paper. It also gives an overview of the high-level approach we follow in this paper.

Adversarial Inverse Reinforcement Learning:
The goal in IRL is to infer the reward function from a set of expert demonstrations in a Markov Decision Process (MDP). A deterministic Markov Decision Process (MDP) is represented as a tuple (S, A, R, γ), where S is the set of states, A is the set of actions, $R : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. In the IRL problem, we have access to the S, A and γ, but have to infer the reward function R^* from a set of expert trajectories $D = \{\tau_i\}_{i=1}^{M}$ under the assumption that the expert uses a policy π^* that is approximately optimal wrt. the reward function R^*. This can be formulated as the maximum likelihood problem:

$$\max_{\theta} R_{\tau \sim D}[\log p_\theta(\tau)] \tag{1.1}$$

where θ parameterizes the learned reward function $R_\theta(s, a)$.

Adversarial IRL (AIRL) [21] poses the IRL problem as a two-player adversarial game between a reward model and a policy optimizer. The discriminator aims to learn a reward function which can distinguish between trajectories from the expert demonstrations D and trajectories generated by the generator. The generator aims to learn a policy that creates trajectories that the discriminator misclassifies as coming from the expert demonstrations. Herein, the policy and the reward function are modelled as a Neural Network, which serves as a powerful function approximator. By using this adversarial training approach AIRL is able to learn robust and generalisable rewards from expert demonstrations.

Reframing Counterfactual Explanations to Reward Learning:
To use counterfactual explanations in the setting of reward learning, the original definition (see 1.2.2) needs to be adapted. In the setting of supervised learning, we manipulate the input features of the model to change the outputs of the model. However, in the Reward Learning setting the variables to be manipulated could be single states or (partial) trajectories. Correspondingly, the outputs to be targeted can be seen as rewards assigned to single states or as the average reward assigned to the states in a partial trajectory.

Only altering individual state-action pairs could overlook complex multi-step plans and runs the risk of creating infeasible counterfactuals that cannot occur through any sequence of actions. In contrast, modifying trajectories by choosing a different sequence of actions can provide the user with insights

about which longer-term behaviours are incentivised by the reward function. In this way, planning over multiple steps is captured and the counterfactual is guaranteed to consist only of achievable states. We choose to focus on the average reward over a partial trajectory since this allows fair comparisons between trajectories of different lengths. Thus we define Counterfactual Trajectory Explanations (CTEs) as follows:

Definition 1.3.1. *CTEs consist of a pair* $\{(t_{org}, \overline{r_{org}}), (t_{cf}, \overline{r_{cf}})\}$ *of partial trajectories and their average rewards assigned by a reward function* R_θ *that start in the same state* s_n*, but then follow a different sequence of actions resulting in different rewards.*

From this we can argue that the reward function R_θ assigned the average reward $\overline{r_{org}}$ to the partial trajectory t_{org} , because starting in state s_n actions leading to the sequence $((s_n, a_{org_n}), ..., (s_{org_m}, a_{org_m}))$ were chosen. I instead actions leading to $((s_n, a_{cf_n}), ..., (s_{cf_k}, a_{cf_k}))$ had been chosen R_θ would have assigned the average reward $\overline{r_{cf}}$ to the counterfactual trajectory t_{cf}. [2]

Given a learned reward function R_θ, a policy π_θ trained on R_θ and a full original trajectory τ_{org} generated by π_θ, the goal is to select a part of that trajectory $t_{org} \subseteq \tau_{org}$ and produce a counterfactual t_{cf} to it that starts in the same state s_n. Throughout the report, τ denotes full trajectories generated by a policy, while t describes partial trajectories that serve as a part of a CTE.

Overview of the Approach:
Figure 1.3 gives an overview of the process in which our methods are deployed. First, a policy π^* trained on the ground-truth reward function R^* provides demonstrations $D = \{\tau_i\}_{i=1}^M$ of the desired behaviour (1) from which a reward learning algorithm (in our case, AIRL) learns a reward function R_θ (2). In the targeted applications a human would provide these demonstrations based on its values and preferences.

Next, the explanations are generated (3). A policy π_θ is trained on R_θ and is used to generate a set of original trajectories $\tau_{org} = \{\tau_{org_j}\}_{j=1}^N$, where $\tau_{org_j} = \{(s_0, a_0), ..., (s_l, a_l)\}$. Then the methods outlined in section 1.4 generate CTEs that are optimised for the quality criteria described in section 1.4.1.

Lastly, we want to evaluate how informative the generated CTEs are for interpreting R_θ. This is done by training a proxy-human model M_ϕ on the CTEs to predict rewards over trajectories. Firstly, a set of features $f_0, ..., f_p$ and the average rewards $\overline{r}$ are extracted from the original and counterfactual trajectories of the CTEs (4). These features represent plausible hypotheses a human might have about which aspects of the environment could influence the reward function predictions. Next, M_ϕ is trained to predict the separate rewards $\overline{r_{org}}$ and $\overline{r_{cf}}$, but also the difference between them $\overline{r_{org}} - \overline{r_{cf}}$, from the extracted features f_{org}, f_{cf} (5). Importantly, humans learn from counterfactuals in a contrastive way by comparing the differences in conditions and outcomes to arrive at causal conclusions [45]. However, they also learn to predict the rewards for single examples without having to rely on a second data point. Finally, the accuracy of the learned model is measured on an unseen dataset of CTEs (6). We measure how close the outputs of M_ϕ are to those of R_θ through the correlation in the reward assigned to CTEs. This gives us an indication of how informative the provided CTEs were to M_ϕ about R_θ.

1.4. Method

This section illustrates the approach used to generate Counterfactual Trajectory Explanations (CTEs). First, it describes the quality criteria that measure how good an explanation is and how they are combined into a scalar quality value. Then it introduces two methods for generating CTEs that optimise for the quality value.

1.4.1. Quality criteria for counterfactual explanations

The classic approach to creating counterfactuals is to define a loss function that determines how good a counterfactual is and optimise for that loss function [6]. This loss function is made up of different components we call ”quality criteria”. To create explanations for reward functions we need to rethink what makes a good counterfactual explanation in this setting and adapt existing quality criteria. From the plethora of knowledge from psychology about counterfactual explanations and the best-use advice

[2] A number of examples of Counterfactual Trajectory Explanations in the Emergency Environment [58] can be found in: `https://drive.google.com/drive/folders/1JMjwQM24BbDwL8vRnG3pST5hlvpzRfZM?usp=sharing`

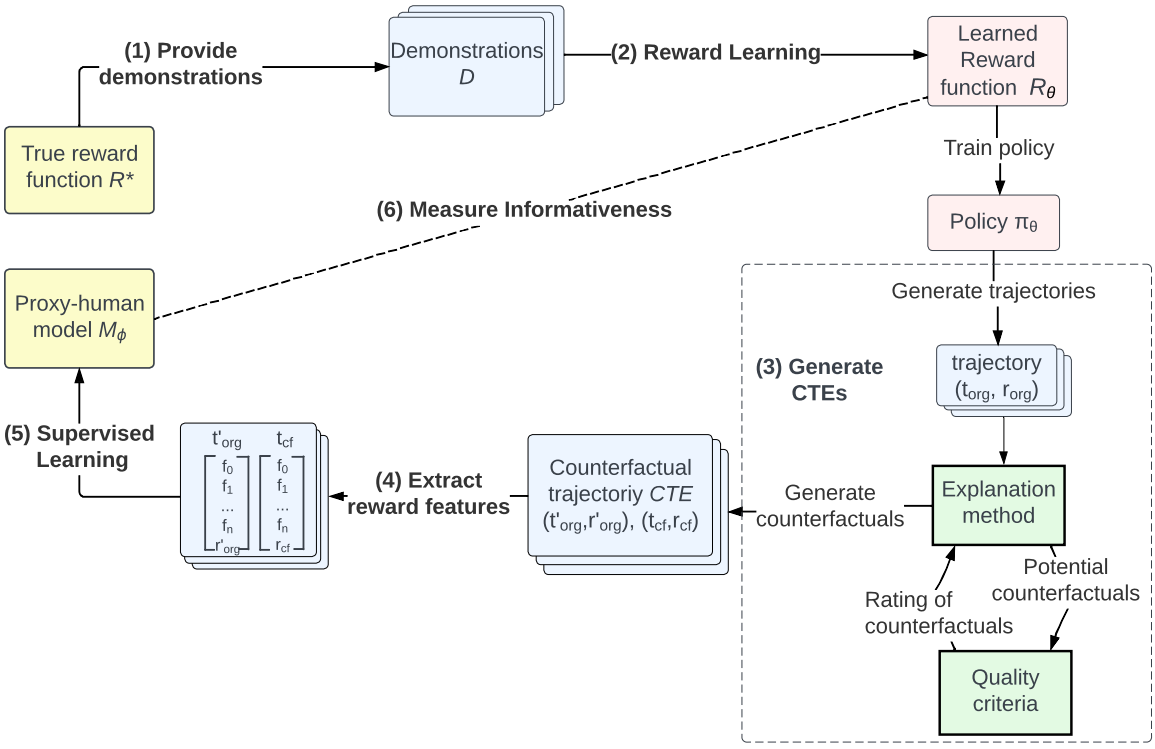


Figure 1.3: Schematic that describes how rewards are learned (1&2), explanations are generated (3) and evaluated (4,5&6).

from XAI, we derive six criteria for high-quality counterfactual explanations that are suitably adapted to explain reward functions. Details about the implementation of these criteria can be found in Appendix B.1.

Quality Criteria

1. Validity: Counterfactual examples should lead to the desired difference in the output of the model [75, 23]. This difference in outputs makes it possible to causally reason about the changes in the inputs. We calculate the Validity as the absolute difference in average rewards between t_{org} and t_{cf}.

2. Proximity: The counterfactual data point should be similar to the original data point [37, 50, 23]. Thus we implement a measure based on the Modified Hausdorff distance [16] that calculates the distance between state-action pairs and finds the closest match between the trajectories. The distance is calculated through the Manhattan distance of the player positions, whether the same action was taken and the edit distance between humans.

3. Diversity: Explanations should cover the space of possible variables as well as possible [29, 20]. Consequently, with each generation of new counterfactual trajectory explanations (CTEs), there should be an incentive to establish novel information rather than repeating previously relayed data. New CTEs are incentivised to be different compared to previously shown CTEs in the length of the trajectory, starting times, starting states and counterfactual direction (upward or downward comparisons) [59].

4. State importance: In generating counterfactual explanations, it is important to focus on important states that have a significant impact on the trajectory outcome [20]. We aim to start counterfactual trajectories in critical states, where the policy strongly favours some actions over others. This can be calculated as the negative entropy in the policy's action distribution for the starting state [20, 28].

5. Realisticness: Counterfactuals should be realistic, in that their constellation of variables is likely to happen [37, 23, 75]. Thus we prefer counterfactual trajectories that are likely to be generated by a policy that was trained on the reward function that is being explained. Such a trajectory would likely score high on the reward function. This is operationalised as the difference between the average reward of

the counterfactual and original trajectory. In this way, the criterion is normalised to differences in how much reward could be achieved at a point in the environment.

6. Sparsity: Counterfactuals should only change a few features compared to the original to make it cognitively easier for a human to process the differences [37, 75, 23, 50]. We interpret this as meaning that the counterfactual and original trajectories should be shorter so that the human does not have to compare many states and actions.

Combining quality criteria into a scalar quality value

These six quality criteria need to be scalarised into one *quality value* ρ to assign to a CTE. This is done by normalising the values and combining them into a weighted sum. Given the scores of the quality criteria in vectorised form QC, and the set of weights for quality criteria ω the quality value is calculated as $\rho = \omega \times NORM(QC)$, where $NORM(x) = \frac{x-min}{max-min}$. This weighted sum serves as the target for optimisation for the generation methods.

Normalisation: The criteria are normalised to a range of $[0.1]$. To calculate this we want to find the maximum and minimum value for each criterion. We do this in an adaptive fashion, where we generate CTEs using a set of normalisation values, record the maximum and minimum values and use them to generate new CTEs until convergence (see Appendix B.1.2)

Quality Criteria Weights: The weights ω assigned to the quality criterion correspond to their relative importance. However, this opens the question of which combination of quality criteria and weights assigned to them leads to the most informative explanations. Of course, the optimal weights to assign to the criteria depend on the preferences of the user. To find them we suggest a *calibration phase* in which N different sets of weights $\omega = \{\omega_{validity_j}, ..., \omega_{sparsity_j}\}_{j=1}^{N}$ are uniformly sampled $\omega_i \sim U(0, 1)$ and used to create CTEs. Their informativeness is tested and the set of weights that produces the most informative CTEs to a specific user is chosen for further use.

1.4.2. Generating counterfactual trajectory explanations

In order to generate CTEs (see Definition 1.2.2) we propose two methods that optimise for the aforementioned quality criteria (see Section 1.4.1) along with a random baseline method. The design decisions for the generation methods are justified by ablation studies that can be found in Appendix C.2.

Method 1 - Deviate and Continue:
The *Deviate and Continue* (DaC) method deviates from the original trajectory τ_{org} at a starting state s_n for a number of steps. From there the counterfactual trajectory is continued using the policy π_θ and ends at a randomly determined point. This process is repeated with every state in τ_{org} as a possible starting state s_n, where $n \in \{0, ..., |\tau_{org}|\}$. Finally, the CTE with the highest quality values is chosen.

The deviation is done by taking an action that leads to a different state. In s_n the original trajectory took the action a_n^{org} resulting in the next state s_{n+1}^{org}. Instead we sample a new action a_n^{cf}, according to the policy π_θ which has to lead to a different state $s_{n+1}^{cf} \neq s_{n+1}^{org}$. We do 3 consecutive deviation steps that are required to not lead to the same state as the original ($s_{n+2}^{cf} \neq s_{n+2}^{org}$ and $s_{n+3}^{cf} \neq s_{n+3}^{org}$).

After deviating the counterfactual trajectory is continued by following the policy π_θ. While continuing the counterfactual has a chance $p(terminal) = 0.55$ of ending both t_{org} and t_{cf} at every step. After having generated many potential CTEs with a different starting point each, we measure their quality criteria and choose the one with the highest quality value. Ablations and hyperparameter tuning of DaC are shown in Appendix C.2.1.

A weakness of this method is that it does not do a guided exploration of the space of possible counterfactuals.

Method 2 - Monte Carlo-based Trajectory Optimization (MCTO):
Monte Carlo Tree Search (MCTS) is a heuristic search algorithm for decision processes that has previously been used in Reinforcement Learning [68, 78]. It models the RL problem as a game tree, where states are nodes and actions taken in these states are branches. from the nodes.

MCTS works by estimating the Q-value of states by running simulations until a terminal state and then

choosing further exploration based on this estimation. Initially, MCTS starts with the starting state s_n as its root node. It then performs iterations which consist of 4 steps that are repeated [3]:

1. Selection: A state in the tree, where not all actions have been explored yet, is chosen according to a multi-armed bandit algorithm. To determine the next action we employ Upper Confidence Trees based on the estimated Q-value of the action in the state $Q(s, a)$ and the number of times this state $N(s)$ and state action pair $N(s, a)$ have already been visited:
$UCT(s, A(s)) = \text{argmax}_{a \in A(s)} Q(s, a) + 2C_p(\frac{2\ln N(s)}{N(s,a)})^{0.5}$
2. Expansion: After we select a state we start expanding its node by choosing actions and creating new nodes according to the states resulting from those actions.
3. Simulation: One full playout is completed by sampling actions randomly until the environment terminates the trajectory. The resulting CTE is evaluated according to the quality criteria.
4. Back-propagation: The accumulated rewards are backpropagated using a discount factor γ to adjust the Q-values of previous nodes.

After a set number of iterations (*n-iterations*), the action with the highest Q-value is chosen. The chosen next state becomes the new root node and MCTS is run again. This process is repeated until a leaf node is chosen.

Inspired by this technique we devise Monte Carlo-based Trajectory Optimization (MCTO), which adapts MCTS to the task of generating counterfactuals. The main difference is that nodes represent partial trajectories instead of single states. Branches still represent the actions available from the last state of that trajectory. In MCTS rewards can be received from the environment in every node. In MCTO rewards are based on the quality value of a completed partial trajectory and are only given in leaf nodes. Lastly, we do not want to favour shorter trajectories in our search, so we opt not to use a discount factor ($\gamma = 1$).

Furthermore, we give MCTO access to an additional terminal action that leads to a leaf node. During the simulation, this action gets chosen with likelihood $p(terminal) = 0.35$. As a heuristic, we only consider branches of actions a that have a probability $\pi_\theta(a|s) \geq threshold_a = 0.003$ according to the policy. To strike a balance between performance and efficiency we use MCTO to create a CTE starting in each point $s_n \in \tau_{org}$, but only perform 10 iterations in every step. Ablations show that other heuristics such as choosing actions in the simulation based on the policy π_θ or basing the expansions on an early estimate of the quality values did not improve performance. Hyperparameter tuning and ablations to improve the performance of MCTO can be found in Appendix C.2.2.

Baseline Method - Random As a weak baseline, we will compare our methods to CTEs which are randomly generated. A uniformly sampled state in the original trajectory τ_{org} is chosen as a start state s_n of the counterfactual. From there actions are uniformly sampled. Additionally, every state (except the first one) has a $p(terminal) = 0.325$ (see Appendix C.2.3) chance of ending both the original and counterfactual trajectory in that timestep.

1.5. Experimental Setup

This section describes the pipeline outlined in Figure 1.3 in detail by explaining how the reward function is learned and how CTEs are generated for it. It further details a novel method for measuring the informativeness of CTEs.

1.5.1. Environment

We employ the emergency environment developed by Peschl et al. [58] that represents a burning building. An agent, as well as 7 humans, are randomly positioned onto a 6 × 6 grid. The humans are lost and need to be escorted which is done by standing next to them and interacting with their cell. Furthermore, the bottom right corner contains a state with a fire extinguisher, which the agent uses by standing on its cell. At each step, the agent can either move in one of the four directions, interact with an adjacent cell or stand still. Lastly, there are obstacles that block a cell which the agent cannot move

[3] We follow the implementation by Miller, 2022 [51]

through. In each run, the starting position of the player and the position of the humans and obstacles are randomly initialised and the environment is run for 75 steps.

We choose this environment, since it is simple, but still requires generalisation and provides multiple sources of reward. The fact that it is a deterministic, small Gridworld environment makes it computationally cheaper to train PPO and AIRL and thus allows for faster iterations in the development cycle. Since the starting positions of the agent, humans and obstacles are randomised the policy and reward function are required to have different initialisations. Lastly, providing multiple sources of rewards means that the learned reward function needs to capture multiple aspects of the environment, which makes it more interesting to provide explanations for it.

1.5.2. Learning, Training and Generation

To start, we define a ground truth reward function R^* that assigns a reward of $+10$ for saving a human and $+1$ for using the fire extinguisher for one step. We train a policy π^* via Proximal Policy Optimization (PPO) [67]. This policy is used to generate 1000 expert trajectories $\tau_{exp} = \{\tau_{exp_k}\}_{k=1}^{1000}$ (1) which serve as demonstrations for Adversarial Inverse Reinforcement Learning (AIRL) [21]. Hyperparameters for PPO and AIRL are detailed in Appendix B.3. AIRL gives us both a learned reward function R_θ and policy π_θ learned alongside that reward function (2).

The aim is now to generate counterfactual explanations about R_θ (3). Hence, we use the policy π_θ to generate a set of original trajectories $\tau_{org} = \{\tau_{org_k}\}_{k=1}^{1000}$ from a starting state until termination of the environment. Lastly, we use the methods described in section 1.4.2 to make one counterfactual trajectory for each original trajectory resulting in 1000 Counterfactual Trajectory Explanations $CTEs = \{t_{org_k}, t_{cf_k}\}_k^{1000}$.

1.5.3. Evaluating the informativeness of CTEs

We now want to evaluate how informative the generated explanations are. We define an informative trajectory as one that helps the explainee (the entity that consumes the explanations) better understand the learned reward function. Understanding is here formalised as the ability to make predictions about a system's behaviour in unseen situations that are similar to the actual behaviour of the system. In our setting, that means an explainee assigns similar rewards as the learned reward function R_θ to an unseen (partial) trajectory.

In the envisioned application scenario the explainee is a human user. However, in this study, we use a Neural Network trained via regression as a proxy-human model to evaluate the informativeness of explanations. While humans and Neural Networks learn very differently from data, this evaluation method still gives us important insights into the functioning and effectiveness of CTEs. Testing how well the trained models generalise shows us whether it is possible to extract some general knowledge about the reward function from the provided CTEs. Furthermore, it allows us to compare different methods and quality criteria by measuring and contrasting the informativeness of CTEs they generate.

This evaluation method consists of three steps: Firstly, features and labels are extracted from the CTEs to form a dataset to train on. Secondly, a proxy-human model is trained to predict the rewards of trajectories from these features. Lastly, the similarity between the predictions of the proxy-human model and the rewards assigned by R_θ is measured to indicate how informative the CTEs were to the model.

1. Extracting features and labels:
The evaluation starts by extracting features and labels from the CTEs to serve as a dataset for later training. We extract 46 handcrafted features $F(t) = \{f_0, ..., f_{45}\}$ about the partial trajectories. These features are at the level of partial trajectories and not features for single states so it is possible to compare trajectories of different lengths and have insights into multi-step behaviour. Examples of features would be concepts such as "standing on the fire extinguisher", "distance to the next human" or "overall number of humans in danger". Features are described in detail in the Appendix B.2.

Furthermore, the reward of a partial trajectory averaged over timesteps is recorded:
$\overline{r} = \frac{1}{length(t)} \sum_{i=start...end} r(s_i)$.
This serves as the labels the proxy-human model is trained to predict. The rewards are averaged over

the trajectory to not bias the learning with the length of trajectories, which would make comparisons between methods unfair and skew results.

2. Learning a proxy-human model
A proxy-human regression model M_ϕ is trained to predict the reward given to the CTE by R_θ from the extracted features $F(t)$. M_ϕ represents a human to learn from the explanations and test how informative they are.

Our proxy-human model M_ϕ is defined as a 4-layer Neural Network that receives the features extracted from both the original and the counterfactual as a concatenated input and is trained in a multi-task fashion on two tasks. The body of the Neural Network is shared between both tasks and feeds into two separate last layers that perform the two tasks:

Single task: It is trained to separately predict the average reward for the original and the counterfactual. Giving rewards to unseen trajectories shows how similar the judgements of M_ϕ and the reward function are for trajectories. The loss on this task is calculated via the sum of Mean Squared Errors on the average rewards.

Contrastive task: It is trained to predict the *difference* between the average original and counterfactual reward. Humans learn from counterfactual explanations in a contrastive manner [50]. That means they compare the difference between the original and the counterfactual data points and see how it leads to differences in results. This allows them to draw conclusions about the causal connection between inputs and outputs from counterfactuals. We want our model M_ϕ to learn from counterfactuals in a similar way, instead of just seeing the data points as independent. To do this we adapt the MSE loss to capture the contrast between original and counterfactual:

$$ContrastiveMSE(T_{org}, T_{cf}) = \frac{1}{|T_{org}|} \sum_{(t_{org}, t_{cf}) \in (T_{org}, T_{cf})} (M_\phi(F(T_{org}), F(T_{cf})) - (R_\theta(t_{org}) - R_\theta(t_{cf})))^2$$

The losses for the two tasks are used separately to update the respective last layer. They are added into a weighted sum, where the contrastive task receives a weight of 3 and the single a weight of 1, to update the shared body of the network.

Notably, M_ϕ can also be based on a different architecture and we perform experiments using a Linear Model (see Appendix C.5).

Before training, we split the 1000 samples into an 80% training set and a 20% test set. Furthermore, hyperparameter tuning and architecture search are performed via 5-fold cross-validation. Multiple learning rates, regulations values and training epochs are tried with multiple Neural Network architectures in a grid-search fashion. We use the same hyperparameters for the settings we compare.

The Neural Network is trained with the Adam optimiser and weight decay. We perform 30 training runs with different initial seeds and average the results.

3. Measuring similarity to the reward function
To measure how similar the proxy-human model's predictions are to the reward function's outputs we measure its performance on an unseen test set. We want to compare the informativeness of CTEs generated with different settings, for example, different generation methods or quality criteria weights. To do this we cannot simply compare the performance on a held-out test set of CTEs generated by the same settings as the training set, since this would make for an unfair comparison. If we have two sets of $CTEs^A_{train}$ and $CTEs^B_{train}$ and train the model M_ϕ on each of them respectively, it might turn out that $CTEs^A_{test}$ are inherently easier to predict than $CTEs^B_{test}$. This does not mean that one is more informative than the other. Thus we need a fairer, more objective way of comparing CTEs.

To address this we use a "combined" test set which combines the test sets from all sets of CTEs generated through different settings. Thus all models are tested on the same test set and we can measure how well the patterns they learned about the reward function generalise to predicting CTEs generated by other settings. Nevertheless, we still report on the performance on the "own" test set generated by the same settings to show how well a model learned.

Our main measure of informativeness is the Pearson Correlation between the outputs of M_ϕ and the true labels on both tasks. The Pearson Correlation is well suited to comparing reward functions since

rewards are invariant under multiplication of positive numbers and addition [88]. If rewards are consistently 10 too high or too low by a factor of 7 this does not change the optimal strategy. What matters is how rewards compare relatively to the other rewards. This is well captured by the Pearson Correlation because it is insensitive to constant additions or multiplications.

1.6. Experiments

To address the stated Research Questions 2 and 3, three experiments were devised, executed and interpreted to test the generation methods, quality criteria and overall informativeness of the CTEs. We start by investigating how well generation methods can optimise for the quality criteria. Next, the quality criteria are tested on whether they lead to more informative CTEs. These experiments help us answer how to best generate CTEs (RQ 2). Lastly, we put those results together and measure how informative CTEs are for a proxy-human model to learn from (RQ 3). For each experiment we first describe the precise setup, then present the results and finally draw conclusions.

1.6.1. Experiment 1: Quality of Generation Methods

This experiment answers the question: How good are the generation methods at optimising for the quality criteria? To this end, we compare how well MCTO, DaC and Random score on the quality criteria.

Experimental Setup

This was tested by having each generation method produce 1000 CTEs and measuring the quality value ρ (see Section 1.4.1) they achieved. Each of these CTEs was optimised for a different, uniformly sampled set of weights: $\omega = \{\omega_{validity_j}, ..., \omega_{sparsity_j}\}_{j=1}^{1000}$, where $\omega_i \sim U(0, 1)$. This made the test independent of the set of any specific set of weights.

Furthermore, it was measured how efficiently (seconds/CTE) the methods generated CTEs, how long the CTEs were and when in the trajectory they started. These additional measures should give indications of the inductive biases of the generation methods.

Results

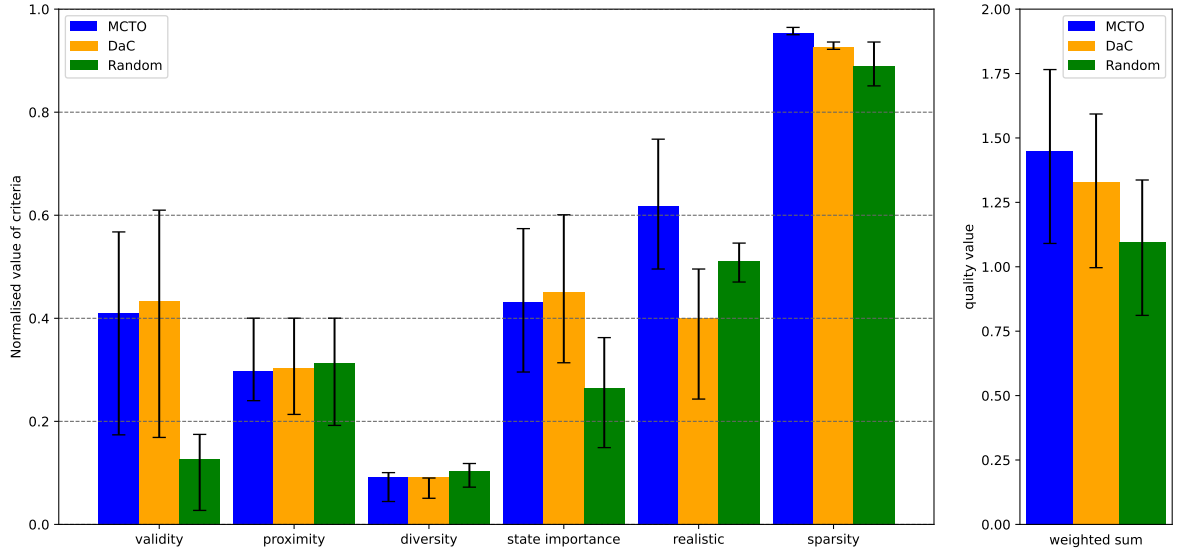


Figure 1.4: For each quality criterion (left figure) and the quality value ρ (right figure), the average normalised value and upper & lower quartile achieved by the different generation methods are shown.

From Figure 1.4 we see that MCTO achieved a higher average quality value than DaC, which again outperformed the random baseline (differences are significant with $p < 1e^{-7}$). The good performance of MCTO is largely attributed to its significantly higher scores on Realisticness, while DaC performed slightly better on Validity and State importance. Across most criteria, MCTO and DaC scored higher than Random.

	MCTO	DaC	Random
quality value ↑	**1.44**	1.32	1.1
Efficiency (s/CTE) ↓ [4]	14.86	5.46	**0.04**
Length (# steps)	2.76	4.96	7.41
Starting Points (# first step)	20.96	20.45	42.58

Table 1.1: Shows the average quality value achieved by MCTO, DaC and Random, along with the efficiency of generating CTEs, the length of the CTEs and at what step in the environment they started.

However, the higher performance came at a computational cost (see Table 1.1). MCTO needed much longer to create one CTE than the other methods since it considers more potential counterfactuals. Comparatively Random was very efficient.

On average the trajectories of Random were the longest and those of MCTO the shortest, which is reflected in the scores for Sparsity. Lastly, both MCTO and DaC tended to choose starting points earlier in the environment (20.96 and 20.45 out of 75 timesteps).

It is also notable that the average values and ranges of quality criteria differed. Sparsity has very high averages but falls in a small range of values. Comparatively, Diversity has very low average values and the scores of Validity have a large range.

Discussion

As expected, MCTO outperformed DaC since it conducts a more exhaustive search of the space of possible counterfactuals. We can conclude that MCTO was better at optimising for the quality value. However, it did not do so on every criterion, since it is not easily possible to score high on all criteria (see Appendix C.1) simultaneously. Thus methods need to strike a good trade-off between the criteria, which allows it to score a quality value.

It is likely the case that MCTO and DaC opted for earlier starting points because there had still been more humans present in the simulation environment, which made the situation ”higher stakes” and thus led to larger differences in rewards. This benefits Validity and State importance.

As a byproduct of the normalisation procedure, the value for Sparsity is much higher than other criteria, with Diversity being assigned much lower values. Appendix C.1 investigates whether this leads criteria to unevenly impact the choice of CTE. We find that Diversity does indeed have a lower influence on which CTE is chosen. Some criteria often get CTEs that they rank highly, while others often get CTEs which have a value close to their best option. Thus it is unclear whether some criteria have a stronger influence.

The difference in average values between the quality criteria can be explained by the normalisation. Each criterion is normalised to a range of $[0, 1]$ through the highest and lowest scores recorded for that criterion. While most trajectories are very short, thus scoring high on Sparsity, the longest ones can be very long. This causes most values to be very high for Sparsity. For the opposite reason, values of Diversity are normalised to be very low. For most generated CTEs there are a lot of previous CTEs, making it hard to be significantly different. However, there are some early CTEs which did score very high on Diversity and thus stretched the normalised range upwards. Values for Validity do not fall in such a big range. Thus they are left with a bigger deviation after being normalised.

We can conclude that MCTO is the most effective generation method for optimising the quality criteria. DaC also handily beats the random baseline, implying that both methods achieve significant optimisation towards the quality criteria.

1.6.2. Experiment 2: Informativeness of quality criteria

Next, we want to test which quality criteria lead to more informative explanations. To determine the influence of a quality criterion on informativeness we analyzed the correlation between the weight assigned to the criterion during the generation of CTEs and the informativeness of those CTEs.

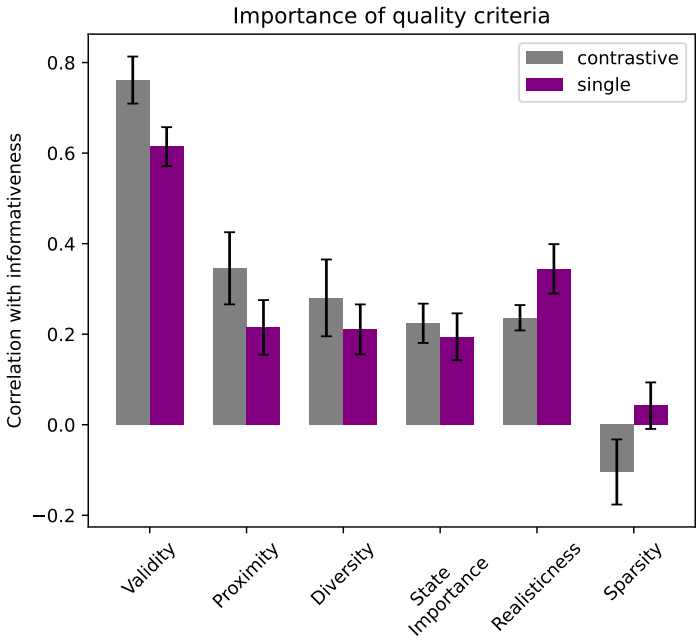


Figure 1.5: For each quality criterion the Spearman correlation between its weight used to generate CTEs via MCTO and the performance of the regression model trained on the resulting CTEs on the contrastive and single task. The results are averaged over 10 models with different initialisations and the upper and lower quartile for different models is displayed.

Experimental Setup

30 sets of weights ω were used to generate 30 sets of 1000 CTEs with MCTO. These sets of CTEs were used to train proxy-human Neural Networks (NN) (as described in Section 1.5.3) and the performance of each resulting model was measured as the Pearson correlation of its outputs to the true labels of the combined test. The combined test set is the combination of all the 30 test sets of 200 CTEs each. By measuring the Spearman correlation between the weights assigned to a criterion and the model performance we can infer the importance of a quality criterion for making CTEs informative.

Results

Figure 1.5 shows that for both contrastive and single learning, the weights of Validity ($\omega_{validity}$) correlated the strongest with the performance of the proxy-human model. This is followed by $\omega_{realisticness}$, $\omega_{proximity}$, $\omega_{diversity}$ and $\omega_{StateImportance}$ which all show a moderate correlation with the task performances. However, $\omega_{sparsity}$ was barely or even negatively correlated with informativeness.

There were differences between the two tasks, with $\omega_{validity}$ and $\omega_{proximity}$ being relatively more correlated with the performance on contrastive learning and $\omega_{StateImportance}$ and $\omega_{sparsity}$ being more correlated with the performance on single learning. However, overall the two tasks end up with a similar ordering of the correlations of quality criteria with task performance.

Discussion

It can be concluded that Validity is the most important criterion for generating informative CTEs. High weights for Validity will make the differences in rewards bigger and thus lead to a larger range of labels for contrastive predictions. Possibly, a NN can learn more information from these larger differences and is thus better informed by CTEs that are high in informativeness.

Proximity, Realisticness, Diversity and State importance are also beneficial for having the proxy-human model learn from their CTEs. While we can hypothesise about reasons for this, we do not have solid evidence for such speculations.

Sparsity is the only criterion which leads CTEs to be less informative on the contrastive task. It is also not helpful for the single task. It can be hypothesised that Sparsity is not very important for a Neural Network to learn, but would be crucial for a human. While humans can only hold a few features in their

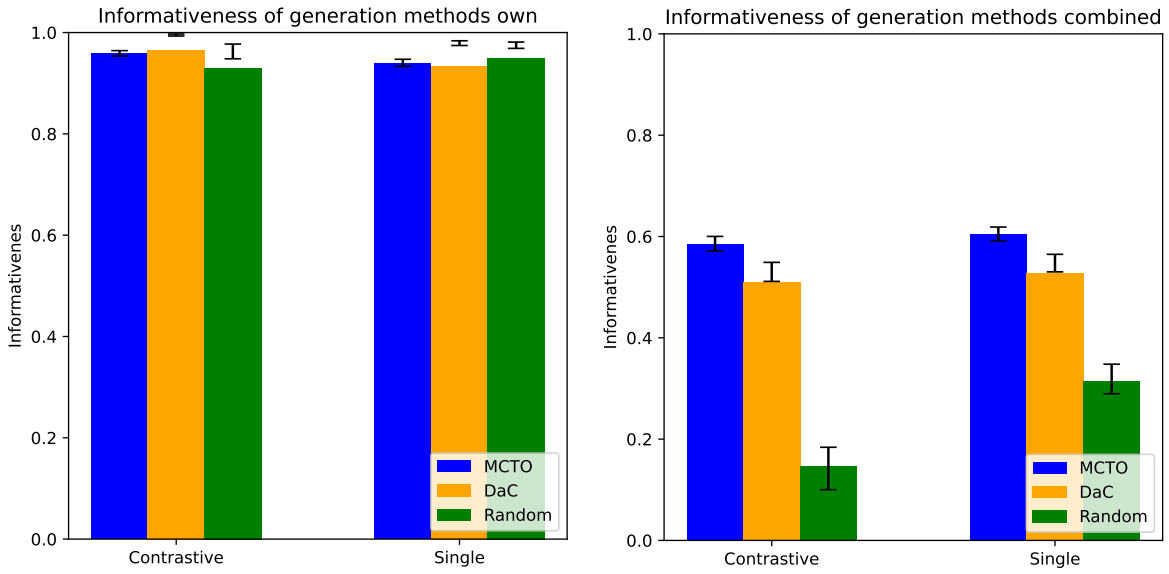


Figure 1.6: The informativeness of CTEs generated by MCTO, DaC and Random for a Neural Network. Performance for the single and contrastive task are shown on the respective *own* (left graph) dataset and the *combined* dataset (right graph).

heads to draw inferences about [49], NNs can easily compute gradients for many different features at once. Thus, in the absence of a user study, we should not conclude that Sparsity would not be important when generating CTEs for a human.

The fact that the 2 tasks largely agreed on the importance of quality criteria indicates that they complement each other. If instead, there was a trade-off between tasks we would expect to see a larger difference in their correlations with weights. This synergy likely appears because the tasks are very similar and the last layers of the network can thus benefit from the same representations developed in the body of the network. This indicates that learning contrastively from a counterfactual can help the learner to predict single examples as well. We can speculate that this would also hold for humans, making CTEs more beneficial than showing single trajectories.

Further, this experiment allowed us to identify the set of weights ω for DaC and MCTO which led them to generate the most informative CTEs. These weights are further used in Experiment 3 and discussed in detail in Appendix C.4.2.

1.6.3. Experiment 3: Informativeness of Explanations for proxy-human model

Lastly, we want to determine the success of our method in generating informative explanations for a proxy-human model. At the same time, we compare the generation methods on the downstream task.

Experimental Setup

To measure how informative the CTEs are for a learner, it was tested how well a proxy-human model based on a NN could generalise to unseen CTEs after being trained on 800 examples. Since different generation methods carry different inductive biases, we compare them by taking CTEs generated by MCTO, DaC and Random as training data for one model each. Each method generates 1000 CTEs. For this, we made use of the most informative set of weights found for DaC and MCTO in Experiment 2. While this made the comparison less biased, it does mean that the methods did not optimise for the same target.

Next, 10 models with different initialisations are trained on a set of 800 CTEs generated by one method. They were then tested on a test set of 200 unseen CTEs generated by the same method (*own*) and on a *combined* test set that has 600 examples from all three methods. We measured the Pearson Correlation between model predictions and labels.

Results

For CTEs of all three methods, a NN trained on their CTEs achieved a Pearson correlation ≥ 0.95 on both tasks on CTEs generated by the same method. That means the models are consistently able to assign rewards to trajectories as the reward function.

There were no significant differences in performances between the generation methods on the own dataset. The fact that some average values of informativeness in Figure 1.6 lie under the 25th percentile is explained by some very low outlier values.

However, on the combined test set, there were significant differences between the methods. Those trained on CTEs from MCTO performed best on the single and contrastive tasks achieving a Pearson Correlation of 0.59 (contrastive) and 0.60 (single) with the labels. Models trained on DaC CTEs were slightly, but significantly ($p < 0.001$) worse, only reaching correlations of 0.53 and 0.51. The models trained on randomly generated CTEs achieved a much lower Informativeness on the single (0.31), but especially on the contrastive task (0.15).

Models resulting from all three methods performed much better on their own test set than on the combined test set. While MCTO dropped by 0.38 (contrastive) and 0.33 (single), DaC dropped by 0.41 and 0.46, while Random even dropped by 0.64 and 0.79.

Discussion

MCTO generates the CTEs that allow the proxy-human model to learn the most amount of generalisable knowledge about the reward function. While DaC also produces quite informative CTEs, the randomly generated explanations do not allow the NN to learn a lot about the reward function.

The high performance on the own test set shows that the regression model accurately learned to predict CTEs generated by the same method. However, there is a big drop in performance on the combined test set which contains out-of-distribution examples. This indicates that CTEs generated by different methods are quite distinct from each other, making generalisation harder.

Notably, the informativeness of CTEs between different methods was similar to the ability of the methods to optimise for the quality criteria. This indicates that the method which finds the highest-quality CTEs (MCTO) also leads to the most informative CTEs and underlines that achieving high values on the quality criteria leads to high informativeness.

A Neural Network trained on CTEs is much better than random guessing at predicting rewards or judging the difference in rewards between unseen CTEs. When trained on CTEs that were appropriately optimised for the quality criteria, the Neural Network also shows a capability to generalise to out-of-distribution examples as shown by the performance on the combined test set. This indicates that they learned some aspects of the reward function which hold generally across different distributions of trajectories. However, the fact that the correlations of M_ϕ's predictions with the true labels are ≤ 0.60 clearly shows that there are aspects of the reward function, which M_ϕ did not pick up on. Else it would be able to make more similar predictions. We can conclude that CTEs are informative to a Neural Network about many aspects of the reward function, but do not enable the CTE to predict it perfectly.

1.6.4. Overall Discussion

After having tested the effectiveness of generation methods, the importance of quality criteria and the informativeness of CTEs we can draw conclusions about the best way to make CTEs and their effectiveness as a method of explanations.

MCTO is the most effective method for generating informative CTEs. This is shown by Experiment 1 which found that MCTO achieves a higher quality value than DaC and Random. Additionally, Experiment 3 indicates that it also produces the CTEs which are most informative. This means that it is the best choice for generating CTEs when efficiency is not of crucial importance.

Validity is the most important criterion for achieving informative CTEs. Furthermore, Experiment 2 shows that Realisticness, Proximity, Diversity and State importance also tend to contribute to having more informative CTEs. However, Sparsity seems unimportant or even harmful for informativeness. Thus, when generating CTEs for a NN to learn from, one should put the highest weight on Validity and ignore the Sparsity criterion. However, the prioritisation of the quality criteria might differ when showing

CTEs to a human. Thus it is crucial to conduct the calibrations phases where the most informative weights are found anew for a different learner. Furthermore, the fact that the methods which achieved higher quality values in Experiment 1 also produced more informative CTEs in Experiment 3 indicates that optimising well for the quality criteria is generally useful for making more informative CTEs.

CTEs are informative for the proxy-human model. Experiment 3 found that learning from CTEs generated by MCTO and DaC allows a Neural Network to learn a significant amount of generalisable knowledge, showing that CTEs can be a fruitful mode of explanation about reward functions. However, we cannot straightforwardly extrapolate these results to humans and there are many aspects of the reward function that the proxy-human does not learn. Furthermore, the learning was not very sample-efficient, which could pose a bottleneck, since it's not practically feasible to show a user 800 examples. Still, the fact that humans are better few-shot learners than Neural Networks provides hope that they could generalise their learnings better while needing fewer CTEs.

The fact that the best method only achieves a Pearson Correlation of 0.6 with the reward function could be caused by multiple factors. Straightforward explanations include the possibility that the Neural Network does not learn perfectly or that more training samples are necessary. Furthermore, it is the case that the reward function that is being explained is noisy, often outputting different rewards for apparently similar situations. There might be a loss of information through the feature extraction, where subtleties are lost and thus cannot be learned by the proxy-human model. Additionally, the way CTEs are currently generated does not cause them to cover all situations that a reward function might encounter, thus not enabling the proxy-human model to predict them. Finally, it might be the case that CTEs are an inherently imperfect mode of explanations about reward functions that cannot capture all it's complexity.

Although a user study remains to be performed, we can venture to say that Counterfactual Trajectory Explanations that are generated by MCTO can serve as effective explanations about learned reward functions. When optimising the CTEs for Validity, Realisticness, Proximity, Diversity and State importance it allows a proxy-human model to derive non-trivial amounts of generalisable knowledge about the reward function from them. This presents Counterfactual Trajectory Explanations as a promising tool that can be employed for interpreting and improving learned reward functions.

1.7. Conclusion

1.7.1. Summary

The aim of this study was to bridge the gap between eXplainable AI and Reward Learning by leveraging the well-proven approach of counterfactual as explanations to explain reward functions learned through Inverse Reinforcement Learning (IRL). With regards to the initial research questions (see 1.1) we make the following conclusions:

RQ 1: How can counterfactual explanations be reframed to the reward learning setting? Counterfactual Explanations can be reframed into the setting of Reward Learning through the notion of Counterfactual Trajectory Explanations (CTEs). This type of explanation is composed of an original and counterfactual partial trajectory and their average rewards that contrast each other to provide information about the reward function. To be able to measure the quality of CTEs, a set of six quality criteria, was derived from the literature and adapted to the setting.

RQ 2: How to generate counterfactual trajectories for reward functions? Two methods for generating CTEs were developed and compared to a random benchmark. Monte Carlo-based Trajectory Optimisation proved to be more effective at achieving high scores on the quality criteria than a method which deviates from the original and then continues using the same policy. Both methods handily beat the random baseline. Furthermore, in our experiments Validity stood out as the criteria that was most important for generating highly informative explanations, while Sparsity tended to be reduce the informativeness of resulting CTEs.

RQ 3: How informative are the generated counterfactual explanations? Counterfactual Trajectory Explanations enabled a Neural Network to learn a significant amount of generalisable knowledge about the reward function from CTEs. This suggests that CTEs can be a useful technique for explaining deep reward functions. However, there are many aspects of the reward function that the proxy-human

model does not learn. These finding were made through a novel evaluation method that measured the informativeness of CTEs by training and testing a proxy-human regression model on them.

This research shows that CTEs can be helpful for a learner to better understand a reward function. This better understanding can then be used to uncover *some* misalignments the learner which receives the explanations and the reward function being explained. However, the learner does not develop a perfect understanding of the reward functions and thus likely won't be able to identify *all* misalignments. Nevertheless, it is valuable to identify and address some mistakes in the reward function before employing a system widely.

1.7.2. Limitations:

Ultimately, explanations are generated to be informative to users. While using a proxy-human Neural Network gives an indication about the informativeness of CTEs and allows us to test the effectiveness of our methods, a human user study is necessary to draw conclusions about the usefulness of this technique. Suggestions for such a user study are given in the Appendix D.3. Importantly, only one learned reward function in one environment was tested. Furthermore, it is unclear how well this method scales to more complex environments and reward functions. Lastly, the specific implementations of quality criteria can be debated and we encourage more experimentation with different methods for generating counterfactuals.

1.7.3. Future Work

The necessary next step in this line of research is to conduct a user study [37] to test how effectively the technique helps humans understand flaws in the reward function. Concrete proposals for such a study are given in the Appendix D.3.

There are multiple possible improvements to the method. Firstly, the multi-objective optimisation problem of choosing CTEs according to multiple quality criteria could be solved with more principled approaches like Bayesian Optimization [3] or Evolutionary Algorithms [15]. Secondly, the measurement of Proximity and Feature Extraction should be made environment-agnostic. Lastly, a better User Interface would allow humans to learn more effectively from CTEs.

Research building on this study could leverage CTEs to investigate the differences in reward learning algorithms, generate counterfactual state-action pairs instead of counterfactual trajectories or transfer the framework to Natural Language Processing by providing counterfactual sentences about reward functions learned via RLHF.

Ultimately, we hope that this work can serve as a part of a framework, where users receive explanations about a learned reward function and provide new demonstrations or feedback on that basis [65]. To close this cycle a system must be built, in which the human takes in the CTEs, which helps them to identify flaws in the reward function, gives new demonstrations to the reward function and the reward function is updated appropriately.

References

[1] Stephen Adams, Tyler Cody, and Peter A Beling. “A survey of inverse reinforcement learning”. In: *Artificial Intelligence Review* 55.6 (2022), pp. 4307–4346.

[2] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. “Reinforcement Learning Based Recommender Systems: A Survey”. In: *ACM Comput. Surv.* 55.7 (2022). ISSN: 0360-0300. DOI: 10.1145/3543846.

[3] Apoorv Agnihotri and Nipun Batra. “Exploring Bayesian Optimization”. In: *Distill* (2020). DOI: 10.23915/distill.00026.

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. “Concrete problems in AI safety”. In: *arXiv preprint arXiv:1606.06565* (2016).

[5] Stuart Armstrong and Sören Mindermann. “Occam’s razor is insufficient to infer the preferences of irrational agents”. In: *Advances in neural information processing systems* 31 (2018).

[6] André Artelt and Barbara Hammer. “On the computation of counterfactual explanations–A survey”. In: *arXiv preprint arXiv:1911.07749* (2019).

[7] Tom Bewley, Jonathan Lawry, Arthur Richards, Rachel Craddock, and Ian Henderson. “Reward Learning with Trees: Methods and Evaluation”. In: *arXiv preprint arXiv:2210.01007* (2022).

[8] Tom Bewley and Freddy Lecue. “Interpretable Preference-based Reinforcement Learning with Tree-Structured Reward Functions”. In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 2022, pp. 118–126.

[9] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. “Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 783–792. URL: https://proceedings.mlr.press/v97/brown19a.html.

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

[11] Ruth MJ Byrne. *The rational imagination: How people create alternatives to reality*. MIT press, 2007.

[12] Luciano Cavalcante Siebert, Maria Luce Lupetti, Evgeni Aizenberg, Niek Beckers, Arkady Zgonnikov, Herman Veluwenkamp, David Abbink, Elisa Giaccardi, Geert-Jan Houben, Catholijn M Jonker, et al. “Meaningful human control: actionable properties for AI system development”. In: *AI and Ethics* 3.1 (2023), pp. 241–255.

[13] Raja Chatila, Virginia Dignum, Michael Fisher, Fosca Giannotti, Katharina Morik, Stuart Russell, and Karen Yeung. “Trustworthy AI”. In: *Reflections on Artificial Intelligence for Humanity*. Ed. by Bertrand Braunschweig and Malik Ghallab. Cham: Springer International Publishing, 2021, pp. 13–39. ISBN: 978-3-030-69128-8. DOI: 10.1007/978-3-030-69128-8_2. URL: https://doi.org/10.1007/978-3-030-69128-8_2.

[14] Xi-liang Chen, Lei Cao, Zhi-xiong Xu, Jun Lai, Chen-xi Li, et al. “A study of continuous maximum entropy deep inverse reinforcement learning”. In: *Mathematical Problems in Engineering* 2019 (2019).

[15] Kalyanmoy Deb. “Multi-objective optimisation using evolutionary algorithms: an introduction”. In: *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 2011, pp. 3–34.

[16] M.-P. Dubuisson and A.K. Jain. “A modified Hausdorff distance for object matching”. In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. 1994, 566–568 vol.1. DOI: 10.1109/ICPR.1994.576361.

[17] Kai Epstude and Neal J Roese. “The functional theory of counterfactual thinking”. In: *Personality and social psychology review* 12.2 (2008), pp. 168–192.

[18] “Evolutionary ethics: can values change”. In: *Journal of Medical Ethics* 30.4 (2004), pp. 366–370.

[19] Pedro Freire, Adam Gleave, Sam Toyer, and Stuart Russell. “Derail: Diagnostic environments for reward and imitation learning”. In: *Proceedings of the Workshop on Deep Reinforcement Learning at NeurIPS, 2020* (2020).

[20] Julius Frost, Olivia Watkins, Eric Weiner, Pieter Abbeel, Trevor Darrell, Bryan Plummer, and Kate Saenko. “Explaining Reinforcement Learning Policies through Counterfactual Trajectories”. In: *arXiv preprint arXiv:2201.12462* (2022).

[21] Justin Fu, Katie Luo, and Sergey Levine. “Learning robust rewards with adversarial inverse reinforcement learning”. In: *arXiv preprint arXiv:1710.11248* (2017).

[22] Iason Gabriel. “Artificial intelligence, values, and alignment”. In: *Minds and machines* 30.3 (2020), pp. 411–437.

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

[24] Maximilian Geisslinger, Franziska Poszler, Johannes Betz, Christoph Lütge, and Markus Lienkamp. “Autonomous driving ethics: From trolley problem to ethics of risk”. In: *Philosophy & Technology* 34 (2021), pp. 1033–1055.

[25] Adam Gleave, Michael D Dennis, Shane Legg, Stuart Russell, and Jan Leike. “Quantifying Differences in Reward Functions”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=LwEQnp6CYev.

[26] Riccardo Guidotti. “Counterfactual explanations and how to find them: literature review and benchmarking”. In: *Data Mining and Knowledge Discovery* (2022), pp. 1–55.

[27] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. *Unsolved Problems in ML Safety*. 2022. arXiv: 2109.13916.

[28] 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)*. 2018, pp. 3929–3936. DOI: 10.1109/IROS.2018.8593649.

[29] Sandy H. Huang, David Held, Pieter Abbeel, and Anca D. Dragan. “Enabling robots to communicate their objectives”. In: *Autonomous Robots* 43.2 (2019), pp. 309–326. ISSN: 1573-7527. DOI: 10.1007/s10514-018-9771-0.

[30] Erik Jenner and Adam Gleave. “Preprocessing reward functions for interpretability”. In: *arXiv preprint arXiv:2203.13553* (2022).

[31] Erik Jenner, Joar Max Viktor Skalse, and Adam Gleave. “A general framework for reward function distances”. In: *NeurIPS ML Safety Workshop*. 2022. URL: https://openreview.net/forum?id=Hn21kZHiCK.

[32] Daniel Kahneman and Dale T Miller. “Norm theory: Comparing reality to its alternatives.” In: *Psychological review* 93.2 (1986), p. 136.

[33] Akansha Kalra and Daniel S. Brown. “Interpretable Reward Learning via Differentiable Decision Trees”. In: *NeurIPS ML Safety Workshop*. 2022. URL: https://openreview.net/forum?id=3bk40MsYjet.

[34] Athanasia Karalakou, Dimitrios Troullinos, Georgios Chalkiadakis, and Markos Papageorgiou. “Deep RL Reward Function Design for Lane-Free Autonomous Driving”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection*. Ed. by Frank Dignum, Philippe Mathieu, Juan Manuel Corchado, and Fernando De La Prieta. Cham: Springer International Publishing, 2022, pp. 254–266. ISBN: 978-3-031-18192-4.

[35] Daniel Kasenberg and Matthias Scheutz. “Interpretable apprenticeship learning with temporal logic specifications”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 4914–4921.

[36] Atoosa Kasirzadeh and Andrew Smart. “The Use and Misuse of Counterfactuals in Ethical Machine Learning”. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 228–236. ISBN: 9781450383097. DOI: 10.1145/3442188.3445886.

[37] Mark T Keane, Eoin M Kenny, Eoin Delaney, and Barry Smyth. “If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), Survey Track* (2021), pp. 4466–4474.

[38] 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”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 4909–4926. DOI: 10.1109/TITS.2021.3054625.

[39] W Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. “Reward (mis) design for autonomous driving”. In: *Artificial Intelligence* 316 (2023), p. 103829.

[40] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. “The Dangers of Post-hoc Interpretability: Unjustified Counterfactual Explanations”. In: *Twenty-Eighth International Joint Conference on Artificial Intelligence {IJCAI-19}*. Macao, Macau SAR China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 2801–2807. DOI: 10.24963/ijcai.2019/388.

[41] Michael S. Lee, Henny Admoni, and Reid Simmons. “Reasoning about Counterfactuals to Improve Human Inverse Reinforcement Learning”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 9140–9147. DOI: 10.1109/IROS47612.2022.9982062.

[42] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. “Scalable agent alignment via reward modeling: a research direction”. In: *arXiv preprint arXiv:1811.07871* (2018).

[43] Enrico Liscio, Michiel van der Meer, Luciano C Siebert, Catholijn M Jonker, Niek Mouter, and Pradeep K Murukannaiah. “Axies: Identifying and Evaluating Context-Specific Values”. In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 2021, pp. 799–808.

[44] 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* 34.03 (2020), pp. 2493–2500. DOI: 10.1609/aaai.v34i03.5631.

[45] David R Mandel. “Of Causal and Counterfactual Explanation”. In: *Understanding counterfactuals, understanding causation: Issues in philosophy and psychology* (2011), p. 147.

[46] David R Mandel, Denis J Hilton, and Patrizia Ed Catellani. *The psychology of counterfactual thinking.* Routledge, 2005.

[47] Eric J Michaud, Adam Gleave, and Stuart Russell. “Understanding learned reward functions”. In: *Deep RL Workshop, NeurIPS 2020* (2020).

[48] Dale T Miller and Saku Gunasegaram. “Temporal order and the perceived mutability of events: Implications for blame assignment.” In: *Journal of personality and social psychology* 59.6 (1990), p. 1111.

[49] George A Miller. “The magical number seven, plus or minus two: Some limits on our capacity for processing information.” In: *Psychological review* 63.2 (1956), p. 81.

[50] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[51] Tim Miller. *Monte-Carlo Tree Search (MCTS)*. 2022. URL: https://gibberblot.github.io/rl-notes/single-agent/mcts.html (visited on 09/01/2023).

[52] Brent Mittelstadt, Chris Russell, and Sandra Wachter. “Explaining explanations in AI”. In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 279–288.

[53] Md. Saddam Hossain Mukta, Mohammed Eunus Ali, and Jalal Mahmud. “Identifying and Predicting Temporal Change of Basic Human Values from Social Network Usage”. In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ASONAM ’17. Sydney, Australia: Association for Computing Machinery, 2017, pp. 619–620. ISBN: 9781450349932. DOI: 10.1145/3110025.3110087. URL: https://doi.org/10.1145/3110025.3110087.

[54] Thibaut Munzer, Bilal Piot, Matthieu Geist, Olivier Pietquin, and Manuel Lopes. “Inverse reinforcement learning in relational domains”. In: *International joint conferences on artificial intelligence*. 2015.

[55] Andrew Y Ng and Stuart J Russell. “Algorithms for Inverse Reinforcement Learning”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. 2000, pp. 663–670.

[56] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. “The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models”. In: *International Conference on Learning Representations*. 2022.

[57] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. “The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=JYtwGwIL7ye.

[58] Markus Peschl, Arkady Zgonnikov, Frans A. Oliehoek, and Luciano C. Siebert. “MORAL: Aligning AI with Human Norms through Multi-Objective Reinforced Active Learning”. In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS ’22. Virtual Event, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, 2022, pp. 1038–1046. ISBN: 9781450392136.

[59] Neal J Roese. “The functional basis of counterfactual thinking.” In: *Journal of personality and Social Psychology* 66.5 (1994), p. 805.

[60] Neal J Roese and James M Olson. *What might have been: The social psychology of counterfactual thinking*. Psychology Press, 2014.

[61] Milton Rokeach and Sandra J Ball-Rokeach. “Stability and change in American value priorities, 1968–1981.” In: *American psychologist* 44.5 (1989), p. 775.

[62] Jacob Russell and Eugene Santos. “Explaining reward functions in markov decision processes”. In: *The Thirty-Second International Flairs Conference*. 2019.

[63] Stuart Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

[64] Lindsay Sanneman and Julie Shah. “Explaining Reward Functions to Humans for Better Human-Robot Collaboration”. In: *arXiv preprint arXiv:2110.04192* (2021).

[65] Lindsay Sanneman and Julie Shah. “Transparent Value Alignment”. In: *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’23. Stockholm, Sweden: Association for Computing Machinery, 2023, pp. 557–560. ISBN: 9781450399708. DOI: 10.1145/3568294.3580147.

[66] Lindsay Sanneman and Julie A Shah. “An empirical study of reward explanations with human-robot interaction applications”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8956–8963.

[67] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[68] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[69] Joar Max Viktor Skalse and Alessandro Abate. “Misspecification in Inverse Reinforcement Learning”. In: *NeurIPS ML Safety Workshop*. 2022. URL: https://openreview.net/forum?id=rDdh2bJ4nV.

[70] Srivatsan Srinivasan and Finale Doshi-Velez. “Interpretable batch IRL to extract clinician goals in ICU hypotension management”. In: *AMIA Summits on Translational Science Proceedings* 2020 (2020), p. 636.

[71] Gregory Stein. “Generating High-Quality Explanations for Navigation in Partially-Revealed Environments”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 17493–17506. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/926ec030f29f83ce5318754fdb631a33-Paper.pdf.

[72] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. “A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence”. In: *IEEE Access* 9 (2021), pp. 11974–12001.

[73] Sam Toyer, Rohin Shah, Andrew Critch, and Stuart Russell. “The magical benchmark for robust imitation”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18284–18295.

[74] Eiji Uchibe. “Model-free deep inverse reinforcement learning by logistic regression”. In: *Neural Processing Letters* 47.3 (2018), pp. 891–905.

[75] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. “Programmatically interpretable reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5045–5054.

[76] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E Hines, John P Dickerson, and Chirag Shah. “Counterfactual Explanations and Algorithmic Recourses for Machine Learning: A Review”. In: *arXiv preprint arXiv:2010.10596* (2020).

[77] Roosevelt Vilar, James Hou-fu Liu, and Valdiney Veloso Gouveia. “Age and gender differences in human values: A 20-nation study.” In: *Psychology and aging* 35.3 (2020), p. 345.

[78] Tom Vodopivec, Spyridon Samothrakis, and Branko Ster. “On monte carlo tree search and reinforcement learning”. In: *Journal of Artificial Intelligence Research* 60 (2017), pp. 881–936.

[79] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual Explanations without opening the Black Box: Automated decisions and the GDPR”. In: *Harvard Journal of Law & Technology* 31.2 (2018).

[80] Cong Wang, Xiao-Hui Li, Haocheng Han, Shendi Wang, Luning Wang, Caleb Chen Cao, and Lei Chen. “Counterfactual Explanations in Explainable AI: A Tutorial”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD ’21. Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 4080–4081. ISBN: 9781450383325. DOI: 10.1145/3447548.3470797. URL: https://doi.org/10.1145/3447548.3470797.

[81] Jan Wehner. “Data for IRL: What is needed to learn human values?” In: *www.lesswrong.com* (2022). URL: https://www.lesswrong.com/posts/2ZKLaqKLr8TkKAxRW/data-for-irl-what-is-needed-to-learn-human-values.

[82] Blake Wulfe, Logan Michael Ellis, Jean Mercat, Rowan Thomas McAllister, and Adrien Gaidon. “Dynamics-Aware Comparison of Learned Reward Functions”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=CALFyKVs87.

[83] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. *Maximum Entropy Deep Inverse Reinforcement Learning*. 2016. arXiv: 1507.04888.

[84] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. “Maximum entropy deep inverse reinforcement learning”. In: *arXiv preprint arXiv:1507.04888* (2015).

[85] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. “Large-scale cost function learning for path planning using deep inverse reinforcement learning”. In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1073–1087.

[86] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. “Watch this: Scalable cost-function learning for path planning in urban environments”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2089–2095. DOI: 10.1109/IROS.2016.7759328.

[87] Long Xin, Shengbo Eben Li, Pin Wang, Wenhan Cao, Bingbing Nie, Ching-Yao Chan, and Bo Cheng. “Accelerated Inverse Reinforcement Learning with Randomly Pre-sampled Policies for Autonomous Driving Reward Design”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 2757–2764. DOI: 10.1109/ITSC.2019.8916952.

[88] NG A. Y. “Policy invariance under reward transformations : Theory and application to reward shaping”. In: *Proc. of the Sixteenth International Conference on Machine Learning* (1999). URL: https://cir.nii.ac.jp/crid/1570009750040818432.

[89] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. “Reinforcement Learning in Healthcare: A Survey”. In: *ACM Comput. Surv.* 55.1 (2021). ISSN: 0360-0300. DOI: 10.1145/3477600. URL: https://doi.org/10.1145/3477600.

[90] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. “Meta-Inverse Reinforcement Learning with Probabilistic Context Variables”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/30de24287a6d8f07b37c716ad51623a7-Paper.pdf.

[91] Simon Zhuang and Dylan Hadfield-Menell. “Consequences of Misaligned AI”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 15763–15773. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/b607ba543ad05417b8507ee86c54fcb7-Paper.pdf.

[92] QiJie Zou, Haoyu Li, and Rubo Zhang. “Inverse Reinforcement Learning via Neural Network in Driver Behavior Modeling”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1245–1250. DOI: 10.1109/IVS.2018.8500666.

A

Extended Background

This chapter provides a more thorough background on Reward Learning, details previous work on Interpreting Reward Functions and provides examples of work using Counterfactual Explanations in Reinforcemenet Learning.

A.1. Reward Learning

We describe Reward Learning as a set of techniques used to extract the reward function of an expert from data produced by that expert, commonly in the form of demonstrations or preference judgements. This set of techniques can be employed to learn the intentions and goals of humans from their behaviour or the feedback they give. The extracted reward function can then be used to train or fine-tune another AI system. It thus presents a promising approach to align the goals of an AI with the objectives of a human. While this technique originally comes from Reinforcement Learning it has also found recent application in Natural Language Processing in the form of Reinforcement Learning from Human Feedback [10].

This section describes multiple Reward Learning techniques with a focus on Inverse Reinforcement Learning as one technique for Reward Learning. Furthermore, fundamental problems of IRL and commonly used methods for evaluating the learned reward functions are addressed.

A.1.1. Inverse Reinforcement Learning

One prominent technique for learning reward functions is Inverse Reinforcement Learning (IRL). In IRL the goal is to infer the reward function from demonstrations generated by a teacher in a Markov Decision Process (MDP). We define a deterministic Markov Decision Process (MDP) represented as a tuple (S, A, R, γ), where S is the set of states, A is the set of actions, $R : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. For a given MDP, an optimal policy π^* maximises the expected sum of rewards in the MDP when followed.

In the IRL problem, we have access to the S, A and γ, but have to infer the reward function R^* from a set of expert trajectories $D = \{\tau_i\}_{i=1}^{M}$. We assume that each of the trajectories $\tau = \{(s_0, a_0, s_1, a_1, \ldots, s_L, a_L)\}$ is generated by an optimal policy π^* with regards to the true reward function R^*. This can be formulated as the maximum likelihood problem:

$$\max_{\theta} R_{\tau \sim D}[\log p_\theta(\tau)] \tag{A.1}$$

where θ parameterizes the learned reward function $R_\theta(s, a)$. In our setting, this learned reward function is a Neural Network, which serves as a powerful function approximator.

IRL Algorithms

Deep IRL: Human goals and values in the real world are likely very complex, thus learning them will require very expressive models. This has motivated researchers to employ Neural Networks as models

for learning reward functions.

Deep Maximum Entropy IRL is a further popular IRL method [75], which extends Maximum Entropy IRL [81] to employ a Neural Network as its reward function. It aims to maximise the log-likelihood of the observed trajectories based on the concept of maximum entropy. The reward function, represented as a neural network, is trained to predict rewards that enable the observed behaviour to have a higher likelihood under the resultant policy.

Adversarial IRL [29] poses the IRL problem as a two-player adversarial game between a reward function and a policy optimizer. The discriminator aims to learn a reward function which can distinguish between trajectories from the expert demonstrations D and trajectories generated by the generator. The generator aims to learn a policy that creates trajectories that the discriminator misclassifies as coming from the expert demonstrations. Hereby, the policy and the reward function are modelled as a Neural Network. By using this adversarial training approach AIRL is able to learn robust and generalisable rewards from expert demonstrations.

A.1.2. Problems of Inverse Reinforcement Learning

Aside from the clear strengths of IRL, there are many well-documented problems with this approach. There are multiple causes of unidentifiability, which might make it impossible to identify the correct reward function among multiple possible reward functions. Firstly, representational unidentifiability poses that if a reward function has constant added or is multiplied by a positive scalar it does not change the optimal policy [54]. Secondly, the observed behaviour could be optimal under multiple reward functions [5]. Lastly, irrationality-based unidentifiability arises since the sub-optimality of the planner and the reward function cannot be uniquely decomposed from demonstrations [7].

Furthermore, reward learning can also fall for causal confusion, where the algorithm mistakes something that is *correlated* with teacher actions with the *causes* of expert actions [21, 66]. Arnold and Kasenberg [8] argue that IRL is insufficient to align AI with human values since it is inadequate to evaluate ethical behaviour along its complex social and temporal dimensions.

A.1.3. Other Reward Learning Techniques

Aside from IRL, there are other algorithms for learning rewards or policies from human data. In preference-based Reinforcement Learning (PbRL) [72, 19] the RL agent learns a policy from human preference judgements instead of from a numerical reward. Thus it is not necessary to correctly specify the goal, instead, a human only needs to be able to give feedback on behaviour, allowing one to learn hard-to-specify objectives. This technique has recently found widespread application to Natural Language processing by finetuning Large Language Models with Reinforcement Learning from Human Feedback [10], which is based on PbRL. However, this method of learning rewards also faces many challenges and limitations [16].

Inverse Optimal Control (IOC) [1] is similar to IRL in that it aims to infer an unknown objective function from observed behaviour. IRL aims to learn a reward function for which the observed expert behaviour in an MDP is optimal. In contrast, IOC, which comes from control theory, aims to learn a cost function for which an optimal value function in a sequential control problem is stable [2].

A.1.4. Evaluating and Testing Learned Reward Functions

These weaknesses shown in Section A.1.2 make clear that a reward function learned by IRL cannot be simply trusted and thus should be evaluated before deploying it.

The most common approach for evaluating a learned reward function R_θ is to train a policy π_θ on it and measure how much ground truth R^* reward it achieves [17, 79, 68, 13]. Similarly, one can measure success on other ground truth metrics such as task success rate or win rate or devise a proxy reward [26, 68, 48].

Another approach is to compare the policies attained by training on the learned reward function and true reward function respectively. One can measure how the negative log-likelihood of the expert trajectories D on the learned policy π_θ [77, 76, 31]. Similarly, one can take the Hausdorff Distance between the original demonstration trajectories and new samples from the resulting policy [77, 76, 31]. Further, one can measure the Expected Value Difference between the value function of the two policies [74, 76, 53].

Paper	Type	Method
[32]	Distance Measure	Equivalent-Policy Invariant Comparison of Reward Functions
[73]	Distance Measure	Dynamics-Aware Reward Distance between Reward Functions
[37]	Distance Measure	General Framework for distance measures
[27]	Benchmark	Diagnostic environments for reward and imitation learning
[67]	Benchmark	Multitask Assessment of Generalisation in Imitative Control ALgorithms Benchmark
[49]	Post-hoc Interpretability	Saliency based explanations
[36]	Post-hoc Interpretability	Transforming rewards via potential-shaping
[58]	Post-hoc Interpretability	Mimicking deep reward function with a decision tree
[61, 59]	Post-hoc Interpretability	Empirical Investigations of Explanation types
[12]	Intrinsic Interpretability	Binary Decision Trees
[39]	Intrinsic Interpretability	Differentiable Decision Trees
[11]	Intrinsic Interpretability	Empirical Test of Binary Decision Trees
[62]	Intrinsic Interpretability	Deep Neural Decision Trees
[40]	Intrinsic Interpretability	Temporal Logic Formula
[53]	Intrinsic Interpretability	IRL in relational domain

Table A.1: Methods for Evaluating or Interpreting Reward Functions

Recently, a set of where it is not necessary to train a policy to evaluate the reward have been proposed and are shown in Table A.1. Using these one can determine the quality of the learned reward function directly by measuring the distance to the true reward function. Gleave et al. [32] introduce *EPIC* (Equivalent-Policy Invariant Comparison), which measures the distance between two reward functions by canonically shaping rewards to be invariant to potential-shaping transformations which lead to equivalent policies [78] and then measuring the Pearson correlation of the reward predictions between those canonically shaped rewards. However, EPIC makes the unrealistic assumption of independence between states in a transition. Wulfe et al. [73] improve upon this weakness by developing *DARD* (Dynamics-Aware Reward Distance), which is also invariant to potential shaping, but only considering predictions about the rewards of transitions which are approximately physically realizable. Finally, Jenner, Skalse and Gleave [37] formulate a general framework for measuring distances between reward functions. A reward distance measure needs to provide a *canonicalization* of the rewards with regards to an equivalence class, a *normalisation* of the canonicalized rewards and a *distance measure*.

An intuitive approach for auditing a learned reward function is to devise a suite of tests or a benchmark to measure the quality of the reward. Freire et al. [27] develop a suite of diagnostic tools that can test different algorithms for Learning from Humans to point out their strengths and weaknesses. These diagnostic tasks should test a factor in isolation, be simple enough to compare many algorithms and together cover a wider range of capabilities. Toyer et al. [67] introduce the Multitask Assessment of Generalisation in Imitative Control ALgorithms (MAGICAL) Benchmark, which tests the ability of an Imitation Learning algorithm to generalise the demonstrator's intent to substantially different deployment settings. Both of these test suites can uncover common errors that can occur in learning from human demonstrations. When auditing learned reward functions we can then specifically look out for these common errors.

Importantly, all of the ways to evaluate learned reward functions named in this section rely on access to the ground truth reward function or only compare policies instead of reward fucntions. However, reward learning is especially useful in domains where it is hard for humans to define a good reward function and we consequently do not have access to a ground truth reward function. Thus we should strive towards ways of evaluating learned reward functions directly and without access to a ground truth reward function.

A.2. Interpreting Reward Functions

A promising avenue to help humans evaluate learned reward functions is to give them a better understanding of the reward function through interpretability tools. The proposed methods and tools for the interpretability of reward functions can be split according to the time of information extraction into ”post-hoc” and ”intrinsic” interpretability methods. Papers are also listed in Table A.1.

A.2.1. Post-hoc Interpretability

Post-hoc interpretability methods are applied to models *after* they are trained and aim to explain how the model arrived at its predictions. They can help to understand which part of the input the model pays attention to, how it reasons about inputs and how it comes to decisions.

Michaud et al. [49] attempt to apply the Interpretability methods gradient salience and occlusion maps, as well as handcrafted counterfactual input to understand learned reward functions. They find that learned reward functions can implement surprising algorithms that fall for spurious correlations. This work showcases that interpretability tools can be used on deep, learned reward functions to identify flaws and misalignments. After learning the reward function Jenner and Gleave [36] apply equivalence transformations to it. This does not change the optimal policy resulting from the reward function but makes it easier to understand. They find that transforming reward functions via potential-shaping [78] to be more sparse and take on fewer different values does sometimes make them more understandable.

Russel and Santos [58] train an interpretable decision tree to mimic the reward function. From this, they are able to derive global and local explanations about the behaviour of the reward function.

Lindsey and Shah set out to empirically investigate which explanation techniques about learned reward function best help humans understand the reward function. They set up an experiment where they compare different explanation techniques which use features and weights of the reward function (feature space techniques) and demonstrations in the environment (policy space techniques) according to multiple metrics that measure reward understanding. They further propose that the complexity of understanding a reward in a domain can be split up into the complexity of reward functions, features, environment and situations [61]. They then follow up with an experiment that tests how effective feature space and policy space techniques are for explaining reward functions of different complexities. Their results show that more complex rewards are harder to understand and lead to a higher cognitive workload. Furthermore, directly showing humans the reward function leads to the best reward understanding while showing humans explanations with feature abstractions leads to lower cognitive workload [59].

A.2.2. Intrinsic Interpretability

Intrinsic Interpretability refers to Machine Learning models which are designed or trained to be inherently interpretable. These models are often more simple and transparent and thus easier to understand. However, this can come at a trade-off to the capability of the model to make accurate predictions.

Decision Trees: Models based on trees are often seen as intrinsically interpretable because their decision procedures can be understood by looking at how the data is split in the nodes and because feature importance can be easily extracted from them. Bewley and Lecue [12] learn the reward function as a binary decision tree which splits the state-action space into buckets via binary decision rules and then assigns rewards to these buckets. Kalra and Brown [39] extend this work by using a differentiable decision tree as a reward model, which retains the interpretability advantages but allows to learn more expressive reward functions. Later, Bewley et al. [11] show experimentally that such tree-based reward functions are broadly competitive with Deep reward functions and should be favoured since they allow analysis of the model, explaining its predictions and verify its alignment. Srinivasan and Finale [62] employ a Deep Neural Decision Tree for learning rewards via IRL in a medical domain. This allows them to learn rewards in terms of discrete concepts which domain experts can understand, allowing experts to vet the learned reward function.

Other intrinsic interpretability methods: Kasenberg and Scheutz [40] learn a linear temporal logic formula, which best explains observed behaviour. This way of specifying an agent’s intentions is easier to explain than learning a reward function. Other approaches attempt to perform IRL in a relational domain [53], leading to easier interpretation.

A.3. Counterfactual Explanations

A.3.1. What are Counterfactual Explanations?

Counterfactual explanations are a popular tool in XAI to help humans understand and interpret predictions by ML models. Counterfactuals are hypothetical scenarios that pose ”what-if” questions, offering alternative instances that would have led to a different outcome. For example, an applicant for a loan might wonder why she was rejected and then receive an explanation like: ”If your income would be 1500/year higher, you would have gotten the loan”. This gives users clear, actionable insight into why the decision was made and how they would need to change the inputs to get a different decision.

More formally, Wachter et al. [71] define counterfactual explanation as statements:

> “Score p was returned because variables V had values $(v_1, v_2, ...)$ associated with them. If V instead had values $(v'_1, v'_2, ...)$, and all other variables had remained constant, score p' would have been returned.”

A.3.2. Why use Counterfactual Explanations?

Counterfactuals are an important cognitive tool for humans and are well-studied in psychology. Humans use counterfactuals to reason about alternative scenarios to learn from their experience and improve their behaviour in the future [15, 57, 25]. They also play a critical role in causal reasoning, planning and blame assignment [47, 38, 50].

The fact that humans can so readily reason about and learn from counterfactuals, makes them intuitive and easy to comprehend even for non-expert users [52]. Further, counterfactuals provide users with local and actionable explanations, allowing them to understand why a specific decision was made and what changes would be required to achieve the desired outcome [71]. However, critics point out the risk of generating unjustified counterfactual explanations [44] and call for caution when making counterfactuals about social categories such as race or gender [41].

There is a large body of work on generating counterfactual explanations for ML models, of which the majority focuses on supervised learning problems. Multiple surveys explore methods for generating counterfactual explanations [70, 9, 33, 64] and their connection to human psychology [42, 14].

A.3.3. Counterfactual Explanations for Reinforcement Learning

The examples of counterfactual explanations most relevant to this research come from their application to explain RL agents.

Gajcin and Dusparic [30] adapt the previously mentioned definition 1.2.2 of counterfactual explanations to RL by giving multiple options on how the meaning of variable and score can be redefined. They argue that in RL the variables to be changed could be Features, Goals, Objectives, Events or Expectations and that the scores to be targeted could be Actions, Plans or Policies. Further, they make proposals about how 7 quality criteria for counterfactual explanations can be reinterpreted and measured in an RL setting.

Madumal et al. [46] learn a causal action-influence model along with the RL agent to then generate counterfactual explanations from it.

To give a user an understanding of how an RL agent will generalise Frost et al. [28] generate counterfactual trajectories that are out of the distribution of the policy. They do this by pausing a normal trajectory at a random point, continuing with an exploration policy to a counterfactual state not typically observed in the training data and then continuing with the original policy from there.

Stein [63] trains an RL agent in a partially revealed environment and generates counterfactual explanations about how the agent’s beliefs about the environment would need to change to choose a different plan.

Lee, Admoni and Simmons [45] aim to explain an agent’s decision-making by providing maximally informative demonstrations. By reasoning about counterfactual trajectories a human might consider, they are able to provide the human with more informative trajectories.

B

Implementation Details

This chapter lays out important design decisions and implementations in detail to open up this work for replication and criticism.

Firstly, we justify the choices for quality criteria and describe how they are measured (section B.1). Next, the features that are extracted from the CTEs are detailed in Section B.2. We lay out the hyperparameters used to train AIRL & PPO (Section B.3). Lastly, Section B.4 explains in detail how multi-task learning is implemented in the proxy-human model.

B.1. Detailed Implementation and Justification of Quality Criteria

This section gives the motivation and idea behind each quality criterion and details how it was implemented to measure the quality of CTEs. Furthermore, some quality criteria from the literature that were not included in this research project are discussed and the process for normalising the values of qualtiy criteria is described.

B.1.1. Implementation of quality criteria

Validity

Counterfactual examples should lead to the desired change in score [69, 30]. This difference in outputs makes it possible to attribute causation to changes in the inputs. Here that means there should be a significant amount of difference between the rewards assigned to the original and counterfactual trajectory.

I calculate the difference in rewards between t_{org} and t_{cf} as the absolute difference in average rewards: $val(t_{org}, t_{cf}) = |\overline{r_{org}} - \overline{r_{cf}}|$. The average reward attained is used to not bias the Validity with regards to the differences in length between trajectories.

Proximity

The counterfactual data point should be similar to the original data point [42, 51, 30]. Otherwise, it is simply a new unrelated data point. Furthermore, Proximity makes it cognitively easier for a human to contrastively reason about counterfactuals.

We interpret Proximity as the similarity between trajecotries, which is measured using the the Modified Hausdorff distance (MHD) [24]. MHD is a technique to calculate the similarity between two sets of points and has been used in previous works in IRL to evaluate whether generated trajectories are similar to each other. It calculates how large the distance of each point in A is to its closest point in B and vice versa and then takes the higher of those two values as the distance between the trajectories:

$f(d(A,B), d(B,A)) = max(d(A,B), d(B,A))$, where $d(A,B) = \frac{1}{N_a} \sum_{a \in A} min_{b \in B} \text{dist}(a,b)$

The distance between state action pairs $dist(a,b)$ takes into three factors:

- Manhattan distance between the player coordinates in a and b to measure how far the player is apart in these states $dist_{player}(a,b) = |x_a - x_b| + |y_a - y_b|$
- Whether the action taken was the same $dist_{action}(a,b) = \begin{cases} 0 & \text{if } a_{action} = b_{action} \\ 1 & \text{otherwise} \end{cases}$
- The edit distance between humans $dist_{humans}(a,b) = ||a_{humans} - b_{humans}||$ (Where $||.||$ indicates the edit distance operation).

.

These are then added into a weighted sum: $dist(a,b) = 1.5 \cdot dist_{player}(a,b) + 0.5 \cdot dist_{action}(a,b) + dist_{humans}(a,b)$. The weight between the factors was determined by the researcher by trying out different weights and judging how the weighted sum matched with their judgment of Proximity.

This method is not environment-agnostic, since we draw on our knowledge about the specific environment to determine which factors should go into the distance measurement of states.

Diversity

Explanations should cover the space of possible variables as well as possible [35, 28]. Thus, when generating new CTE should be incentivised to be dissimilar to previous ones to give the user new information instead of repeating old ones.

We calculate Diversity as the sum of 4 aspects of trajectories: their length, starting time, starting state and whether they are an upward or downward counterfactual. Diversity is calculated for each potential new CTE in relation to CTEs that have already been shown to the user (added to the dataset). So for the first CTE Diversity is ignored and for the 5th CTE, it considers the 4 previously generated CTEs.

To measure how similar the length of the counterfactual is to lengths of previous counterfactuals we calculate the average difference to the 3 closest lengths in previous counterfactuals. In the same way, we compare the starting time of a CTE to the starting times of previous CTEs by comparing the 3 most closest previous CTEs. We measure the similarity of the start state of the CTE to the start states of previous CTEs and report on the most similar one according to the distance metric $dist$ described above.

Counterfactuals can be divided into upward and downward counterfactuals, depending on whether the counterfactual leads to a better (up) or worse (down) outcome than the original [56]. Thus we add a reward based on how often the type of counterfactual has already been shown previously. If the currently considered CTE is an upward CTE we reward it if few of the previous CTEs were upwards $\frac{down}{down+up}$ and if it is downwards we do the opposite $\frac{up}{down+up}$.

Finally, these criteria are added together to form one Diversity criterion.

State importance

When creating counterfactuals one should start the trajectory in especially important states. Previous research has found that “critical states” are more informative for explaining RL policies to humans [28]. Critical states are those in which a policy (or human) greatly prefers a small set of possible actions over all others. Intuitively these are more important because they matter for the final outcome of the trajectory.

States can be considered critical if the variance in the Q-function for the actions or the entropy in the policies distribution over actions could serve as heuristics for interesting states. Amir and Amir [6] calculate the importance of a state as the difference in Q-values between the best and the worst action: $I(s) = max_a Q^{\pi}_{(s,a)} - min_a Q^{\pi}_{(s,a)}$, while Huang et al. [34] take the difference between the best and average action $I(s) = max_a Q^{\pi}_{(s,a)} - \frac{1}{|A|} \sum_{a \in A} Q^{\pi}_{(s,a)}$. Both Forst et al [28] and Huang et al. [34] use the negative entropy in the probability distribution over actions in that state $I(s) = -\sum_{a \in A} \pi(a|s) \log \pi(a|s)$. Since our method does not produce Q-values, we rely on the negative entropy version to measure how critical the starting state of a CTE is and select CTEs that score high on this criterion.

Realisticness
This should ensure that the generated counterfactuals are realistic. In supervised learning this is often operationalised as "Data Manifold Closeness" [42, 30, 69]. This makes sure that the composition of variables in the counterfactual could realistically appear. If this is not the case the counterfactual can be less relevant.

We interpret Realisticness as meaning that a trajectory is likely to be generated by a policy trained on the reward function that is being explained. Intuitively a trajectory is more likely if it achieves a high reward on the reward function. To make this unbiased to the length of the trajectory we take the average reward per step. Further, to not bias the criteria to parts of the environment where high rewards are easily achieved we use the reward achieved by the original trajectory as a comparison: $\overline{R_\theta(t_{cf})} - \overline{R_\theta(t_{org})}$

Sparsity:
This states that only a few features should be changed in the counterfactuals compared to the original [42, 69, 30, 51]. We interpret this as meaning that the counterfactual and original trajectories should be shorter so that the human does not have to compare many behaviours and states. Thus we sum up the length of the original and counterfactual: $sparsity(t_{org}, t_{cf}) = len(t_{org}) + len(t_{cf})$

Other criteria that are automatically fulfilled by the method:
Due to the nature of the Reward Learning setting and our approach to generating CTEs there some quality criteria mentioned in the literature are automatically fulfilled.

Causality:
The causal connections between features need to be maintained [69, 30]. When features are altered the causal relationship between the feature and others should not be violated. Since we create counterfactuals by taking legal actions in the environment and do not alter state features, this criterion is automatically fulfilled in our method since we do not generate new states but only take actions to legal actions to get to new states.

Resource:
In RL environments counterfactual explanations should not be based on actions that do not obey the environment's rules [30]. When creating counterfactual trajectories this can simply be ensured by only taking legal actions.

Actionable:
In supervised learning many methods allow specifying features that can be altered (e.g. income) and features that can't be altered (e.g. ethnicity) [42, 69, 30]. However, since we do not alter features, this requirement does not suit our method.

Other criteria
Lastly, there are some criteria which were not implemented in this research project.

Relative Distance:
When creating a new counterfactual for a datapoint, the counterfactual should be closer to the original datapoint, than other points in the dataset. This should capture whether counterfactuals are instance-guided [42].

Depend on beliefs:
Good explanations are presented relative to the explainer's beliefs about the eplainee's beliefs [51, 35]. Ideally, the explainer tracks what the explainee might learn from the provided explanations and thus be able to generate more informative explanations.

Furthermore, Verma et al. [69] note Efficiency, ability to work with Black-box access and model agnosticity as desirable properties of algorithms for creating counterfactuals.

B.1.2. Normalisation of quality criteria

After we have measured the quality criteria of a CTE, we want to combine them into a scalar by taking the weighted sum. Before doing so we want to normalise each criterionto a range of $[0, 1]$. This requires finding a maximum ($norm^c_{max}$) and minimum ($norm^c_{min}$) possible value each criterion c can take on. We do this approximately in an adaptive fashion, by

1. Starting with $norm^c_{max} = 1$ and $norm^c_{min} = 0$
2. Using these normalisation values DaC and MCTO generate 20 CTEs for a random set of weights.
3. Finding the minimum $norm^c_{min}\prime = min$ and maximum $norm^c_{max}\prime = 1$ values for c in the CTEs.
4. Updating the normalisation parameters if more extreme values were found $norm^c_{max} = \max(norm^c_{max}, norm^c_{max}\prime)$ and $norm^c_{min} = \max(norm^c_{min}, norm^c_{min}\prime)$.
5. If the normalisation values were adapted for any c: Repeat from Step 2.

Furthermore, hyperparameters and settings of the generation methods are altered between repeats to account for the fact that some settings can lead to more extreme values.

B.2. Implementation of Feature and Label Extraction

After the CTEs were generated we extract features from them, which are then used to train the proxy-human model. For both the original and counterfactual partial trajectory we extract 46 handcrafted features $F(t) = \{f_0, ..., f_{45}\}$ and the average rewards $\overline{r}$.

We only extract features on the level of partial trajectories, not for single states. One can expect that a significant amount of detail gets lost when only considering features on the trajectory level since some aspects which factor into the rewards for single states get lost. However, we want the proxy-human model to make predictions about partial trajectories and not only single states so that it can evaluate multi-step behaviour. Since trajectories can have different lengths we cannot simply feed it all single-state features, but need to accumulate them into a feature vector of the same dimensionality.

The features are normalised across the set of CTEs to have a mean of 0 and a standard deviation of 1.

List of Features:
Features about humans:

1. Humans saved: How many humans did the agent save during the partial trajectory?
2. Unsaved humans: On average how many humans were still in danger
3. Final number of unsaved humans: At the end of the partial trajectory, how many humans are still in danger?
4. Number of humans: In how many of the steps in the trajectory was there $n \in \{0, ..., 7\}$ humans still in danger?
5. Average Distance between humans: How far are the humans apart in Euclidean distance averaged over all pairs of humans?

Features about the fire extinguisher:

6. Standing on fire-extinguisher: For how many steps did the agent use the fire-extinguisher
7. Distance to fire-extinguisher: On average what is the Euclidean distance between the agent and fire-extinguisher?

Features about the actions of the agent:

8. Could have saved: Sums up over the trajectory in how many states the agent stood next to a human and could have saved them, but failed to do so
9. Moved towards closest citizen: How often did the agent move towards the closest citizen or save a citizen?
10. Type of action: How often did the agent take a type of action walking, interacting or standing still?

Features about the distance of the agent to humans:

11. Average Distance to human: On average how close was the agent to the closest robot?
12. Distances to humans: In how many steps in the trajectory was the agent $d \in \{0, ..., 10\}$ steps away from the closest human
13. Direction of citizen: How often was the citizen right, left, down or up from the agent?

Features about the position of the agent?

14. Position of the agent: Average x and y position of the agent
15. Closeness to the middle: For how many steps was the agent in the middle, on the outside next to the wall or in the space between the middle and wall?
16. Time spent in quadrant: For how many steps was the agent in the top-left, top-right, bottom-left or bottom-right quadrant?

Other:

17. Length: Number of steps in the partial trajectory

Labels:
In this step, we also extract the rewards which serve as labels for the regression models to predict. For the single task these are simply the average reward of the original and counterfactual trajectory: $\overline{r_{org}} = \frac{1}{length(t_{org})} \sum_{i=start...end} r(s_{org_i})$ and $\overline{r_{cf}} = \frac{1}{length(t_{cf})} \sum_{i=start...end} r(s_{cf_i})$. For the contrastive task we simply subtract the average rewards: $\overline{r_{org}} - \overline{r_{cf}}$.

B.3. Chosen hyperparameters for AIRL & RL

Table B.1 shows the hyperparameters that were used to train PPO on the ground-truth reward function R^* and generated the demonstrations which R_θ was learned from.

Table B.2 displays the hyperparameters for Adversarial Inverse Reinforcement Learning that is used to learn a reward function R_θ from the demonstrations. Furthermore, it also learns a policy π_θ alongside the reward function, which is used to generate original trajectories and is sometimes referred to during the generation of CTEs.

Hyperparameter	Value
Environment Steps	6e6
Learning Rate	1e-4
Batch Size	12
epochs	5
γ	0.999
Entropy Regularization	0.1
ϵ-clip	0.1

Table B.1: Hyperparameters of PPO used to generate the demonstrations

Hyperparameter	Value
Environment Steps	3e6
Learning Rate Discriminator	5e-4
Learning Rate PPO	1e-5
Batch Size Discriminator	12
Batch Size PPO	12
PPO epochs	5
γ	0.999
ϵ-clip	0.1

Table B.2: Hyperparameters for Adversarial Inverse Reinforcement Learning

B.4. Multi-task Learning in the Proxy-human regression model

The proxy-human regression model M_ϕ makes predictions about the difference in rewards between the original and contrastive trajectory (contrastive) and about the rewards of the original and counterfactual trajectory separately (single) (see Section 1.5.3). To achieve this with one Neural Network we use Multi-task learning [80]. The Neural Network has two heads $h_{con}(x) \rightarrow y'_{con}$ and $h_{sin}(x) \rightarrow y'_{sin}$ (two separate last layers) which make predictions for their respective tasks based on the output of the base model. These layers are updated with the losses of their respective tasks. The base model $B(x) \rightarrow x'$ takes the inputs and returns a multi-dimensional representation. It is shared between the two tasks and thus the losses for both tasks need to be taken into account when updating the parameters. This is done by taking a weighted sum of the two losses

$$Loss(y, y) = w_{con} \times Loss_{con}(y_{con}, h_{con}(B(x))) + w_{con} \times Loss_{sin}(y_{sin}, h_{sin}(B(x)))$$

, where w_{con} and w_{sin} denote the relative importance of the tasks. These task weights are ablated in Appendix C.3.2

When using a Linear Model we simply train two separate Linear Models. The first has one output and predicts the contrastive rewards, while the second one uses two outputs to predict single rewards.

C

Further Experiments

To extend and deepen the analysis provided in the Experiments of the Main Paper (see Section 1.6, further experiments, their results and discussions are presented here. Section C.1 investigates the trade-offs between quality criteria and draws conclusions about their influence on the chosen CTEs. This sheds further light on the generation of CTEs and the results in Experiment 2 (see section 1.6.2). Extensive Ablations for the Generation Methods are given in Experiment C.2. This provides us with a deeper understanding of these methods and lends legitimacy to the results in Experiments 1 and 3 (see Sections 1.6.1 and 1.6.3). Furthermore, we validate the results from Experiment 2 (1.6.2) by also considering CTEs generated by DaC (Section C.4.1. Section C.5 also shows experimental results similar to Experiment 2 & 3 if a Linear Model is used as a proxy-human model instead of a Neural Network.

C.1. Influence and Trade-offs in quality criteria

This section explores what trade-offs and synergies exist between the quality criteria and how this influences the choice of CTEs.

C.1.1. Trade-offs and Synergies between quality criteria

Maximising the quality value, which combines six quality criteria, is a multi-objective optimization problem, where the different aspects of the objective are combined into a decision by taking the normalised, weighted sum of criteria. Thus we want to analyse the trade-offs and synergies between criteria. To study this we want to investigate the correlations between quality criteria over all candidate CTEs.

Experimental Setup:
DaC is employed to generate 1 CTE for each of the 1000 sets of weights ω_{random}. We record the quality criteria for the created CTEs, but also for all candidate CTEs that were not chosen. Based on this we record the correlations between quality criteria for candidate CTEs.

Results
There are significant negative correlations between Validity and Proximity, Diversity and Sparsity. Diversity is further negatively correlated with State Importance and Sparsity. Lastly, there is a positive correlation between Validity and State Importance.

Discussion
We hypothesise that the trade-off of Validity with Sparsity and Proximity appears because Validity pushes for longer and more dissimilar trajectories that can result in larger differences in rewards, while Sparsity pushes for shorter and Proximity for more similar trajectories. However, Validity does synergise with State Importance, likely because both criteria prefer phases in the environment, where differences in rewards can be higher. Diversity trades off against State-importance and Sparsity, because these criteria

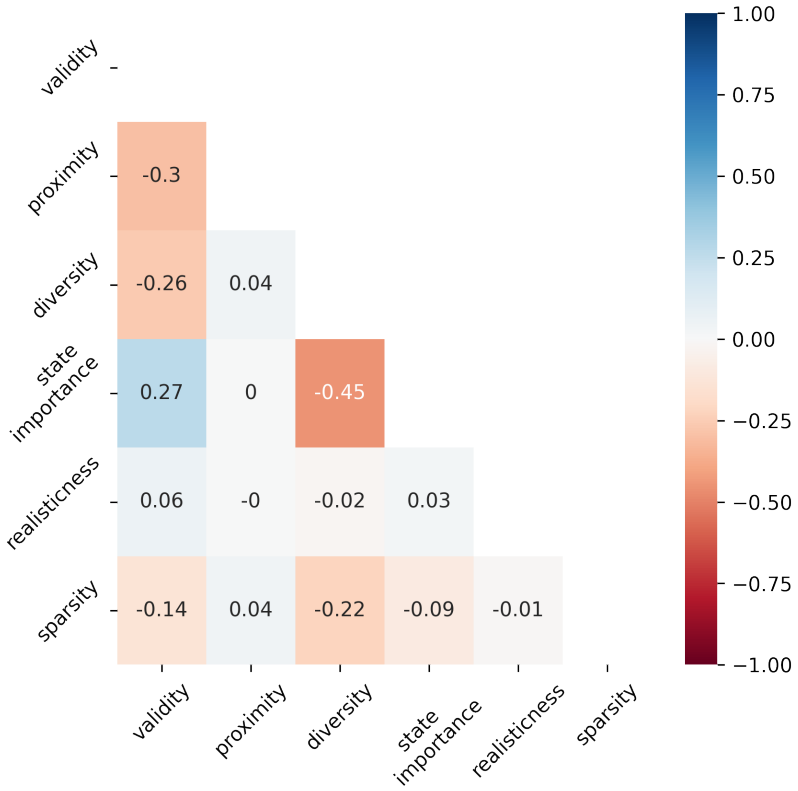


Figure C.1: The correlations between quality criteria on a set of 1000 randomly generated CTEs.

prefer certain start points and lengths of trajectories, while Diversity pushes for different start points and lengths. Furthermore, Diversity is negatively correlated with Validity.

Overall these results show that there are important trade-offs between quality criteria. To navigate these trade-offs it is important to carefully assign priorities between weights, that are adapted to the users' preferences.

C.1.2. Influence of quality criteria on the choice of CTE

The previous section C.1.1 indicated that there are trade-offs between quality criteria which need to be navigated in the multi-objective optimization problem of finding the CTE with the highest quality value. We want to find out whether some quality criteria have a stronger influence on which CTE gets chosen and which criteria are disadvantaged in the choice.

Experimental Setup:
Similarly to the previous Experiment, DaC is employed to generate 1 CTE for each of the 1000 sets of weights ω_{random}. We record the quality criteria for the created CTEs, but also for all candidate CTEs that were not chosen. Based on this we record 1) how high the chosen CTE ranks amongst candidates according to that criterion (*percentile ranking*), 2) the value of the criterion compared to the criterion's highest candidate CTE (*relative value*) and 3) the average correlation over CTEs between the criterion and the quality values ρ (ρ*-correlation*).

Results:
Looking at table C.1 it stands out that Validity and State Importance often get an option they rank very highly (better than 79.6% and 82.7% respectively), but that has a much lower value relative to their favourite options (0.61 and 0.6 times as high as their favourite). Their values do have the strongest correlation with the quality values (0.34 and 0.32). On the other hand, Proximity, Realisticness and Sparsity often have to content themselves with a CTE that they don't rank highly (55.7%, 69.7% and 52.9%), but get a value close to their highest rated CTE (0.9, 0.9 and 0.92 times as good as their favourite). Further,

quality criteria	percentile ranking ↑ mean | median	relative value ↑ mean | median	ρ-correlation ↑
Validity	79.6% | **95.9%**	0.61 | 0.68	**0.34**
Proximity	55.7% | 64.4%	0.9 | 0.91	0.25
Diversity	36.6% | 28.8%	0.37 | 0.27	-0.1
State Importance	**82.7%** | 91.8%	0.6 | 0.58	0.32
Realisticness	69.7% | 93.2%	0.9 | 0.93	0.12
Sparsity	52.9% | 56.2%	**0.92** | **0.96**	0.18

Table C.1: Statistics that describe how influential quality criteria are for determining the CTE in DaC. 1000 chosen CTEs are compared to their respective candidate CTEs and for each criterion we display on average how high the chosen CTE ranks, the value of the chosen CTE compared to the criterion's favorite and its correlation with the quality value.

they have lower correlations (0.25, 0.12 and 0.18) with the quality values. Lastly, Diversity scores lowest on all three measurements.

Discussion:
Table C.1 shows us that criteria often did not get their most preferred outcome. This is shown by the fact that criteria mostly didn't get their top ranked CTE chosen. We can conclude are real trade-offs between the criteria, which means there is no easy choice which satisfies all criteria.

It is interesting that Validity and State Importance score high on percentile ranking and qc-correlation, but low on relative values, while Proximity, Realisticness and Sparsity show the opposite trend. This phenomenon emerges because values decrease steeply when moving down the ranking of CTEs according to Validity and State Importance. Thus, CTEs that rank relatively high are already much worse than their highest rated option. This means these criteria often don't achieve high absolute values. However, the large differences in values mean they have "strong opinions" about the choice of CTEs. Thus they have an outsized impact on the weighted sum of quality criteria and the final decision. On the other hand Proximity, Realisticness and Sparsity have a less steep distribution of values, thus "weaker opinions" about the choice of CTEs and end up influencing the weighted sum less.

It stands out that Diversity does not have a strong influence on the choice of CTE. This might be because Diversity is negatively correlated with multiple other criteria and thus gets crowded out when calculating the weighted sum. Furthermore, Experiment 1.6.1 showed that Diversity has a small range of values, which might further contribute to its weak influence.

C.2. Ablations of Generation Methods

To confirm the design choices that were made for the generation methods and to find good hyperparameters, we perform ablations on DaC and MCTO. To make this analysis independent of the chosen weights for the quality criteria, we uniformly sample 200 sets of weights $\omega_{random} = \{\omega_{Validity_j}, ..., \omega_{Sparsity_j}\}_{j=1}^{200}$ that are uniformly sampled $\omega_i \sim U(0,1)$. We then use different settings for the generation methods to generate 200 CTEs, where each sample is based on a different original trajectory and is optimised for a different set of weights. Finally, we measure and compare the quality criteria achieved by the settings.

C.2.1. Ablations of Deviate and Continue

There are three key elements in DaC that we want to ablate since they represent key design decisions influencing how the method operates:

1. Ending point: The point where the counterfactual and original methods meet can be determined as the point where they cross paths (*"end-meet"*) or can be determined randomly by including a probability $p(terminate)$ after each step with which the original and counterfactual are ended (*"end-random"*).
2. Action selection: After deviating from the original trajectory the actions in the counterfactual trajectory can either be chosen by following the policy π_θ (*"follow-policy"*) or by uniformly sampling random actions (*"follow-random"*).

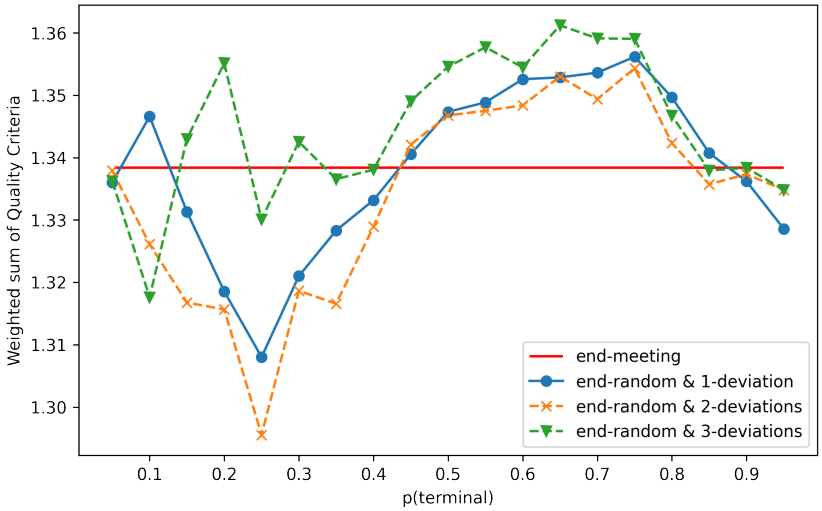


Figure C.2: Quality values achieved by different methods. For "end-random" values for $p(terminal) \in \{0.05, ...0.95\}$ are used in combination with "1-deviation" and "2-deviation". "End-meeting" is not influenced by $p(terminal)$

3. Number of deviations: The counterfactual trajectory can either perform one (*"1-deviation"*) or multiple (*"n-deviations"*) deviations [1].

Experimental Setup:
200 CTEs each are generated using DaC for a uniformly sampled set of weights for multiple settings. The possible combinations of action selection and ending point are tested using "1-deviation" and $p(terminal) = 0.5$. Further, we test different combinations of "n-deviations" ($n \in \{1, 2, 3\}$) with $p(terminal) \in \{0.05, 0.1, ..., 1.0\}$ against "end-meet".

Results:

	avg. quality value	p-value
1-deviation	1.339	1
2-deviations	1.334	0.67
3-deviations	1.346	0.52

Table C.2: Average quality value achieved across all $p(terminal)$ for different numbers of deviations, along with p-values indicating the significance of results.

	follow_policy	random_actions
random_end	1.347 | 1.0	1.351|0.95
ending_meeting	1.338|0.85	1.114|4.22e-07

Table C.3: Average quality value and statistical significance (compared to follow_policy + random_end) for the combinations of options for action selection and ending points.

Table C.2 describes the quality values achieved when performing different numbers of deviation steps averaged across many values for $p(terminal)$. The differences in results are small and don't allow for significant conclusions. This leads to the assumption that the number of deviations is not essential. However, from Figure C.2 we can make out that "3-deviations" outperforms methods with fewer steps on most values for $p(terminal)$.

In Table C.3 the combinations of options for choosing the ending point and selecting actions are shown according to their achieved quality values. The only significant result shows that a combination of choosing actions randomly and ending trajectories on meeting performs very poorly. This can be attributed to especially low values in Validity and Proximity.

Figure C.2 shows a trend that high-quality values are achieved for low $p(terminal)$ values. As $p(terminal)$ rises, quality values first drop until $p(terminal) = 0.25$ and then steadily increase to a maximum of around 0.7, before dropping off again. As a result, the "end-random" method outperforms "end-meeting" for low and moderately high (0.45-0.85) values of $p(terminal)$. Moderately high values score well because they produce short CTEs that are rated high on Sparsity. Lower values perform well on

[1] We use the same process for later deviations as for the first deviation. An action is chosen according to the policy, but actions which would lead the counterfactual to meet the original trajectory are excluded.

Diversity because they can lead to a wide range of lengths of CTEs. The extra degree of freedom for "end-random" provided through choosing a value for $p(terminal)$ allows us to find a better setup than "end-meeting".

From Figure C.2 it can also be concluded that "1-deviation" works similarly well to "2-deviations", but is outperformed by "3-deviations" for most values. This is largely because "3-deviations" regularly achieve higher values for Proximity compared to "1-deviation" ($p = 0.001$) and "2-deviations" ($p = 0.02$). The highest values are achieved by "3-deviations" between $0.5 \leq p(terminal) \leq 0.75$.

Design Decisions and Hyperparameters:
Performing "3-deviations" is preferable to "1-" or "2-deviations", since it scores higher for most values of $p(terminal)$. This might be the case, because it allows for a wider exploration of the search space or because it causes CTEs to have a length which tends to score high on the quality criteria. Using "end-random" achieves higher quality values than "end-meeting" for $0.5 \leq p(terminal) \leq 0.85$. There is no big difference between following the policy and choosing actions at random. Thus we choose to conduct further experiments "follow-policy", "3-deviation", "end-random" and $p(terminal) = 0.55$.

C.2.2. Ablations of Monte Carlo-based Trajectory Optimisation

In MCTO there are multiple design decisions to be made and hyperparameters to be adjusted:

- Number of starting points: MCTO is started at a specific point of the original trajectory. MCTO can be started multiple times from different points in the original trajectory. For each of the *"n-starts"* starting points a CTE is generated and the best one is selected. The *n-starts* starting points are selected as the n states which are rated highest by the State Importance criterion.
- Number of iterations: From a given state MCTO starts to explore the tree through selection, expanding, simulation and back-propagating. *"n-iterations"* denotes the number of iteartions in a state before a decision for the next action has to be made.
- Number of simulations: After expanding, the value of a node is determined through random*backpropagated through the tree. During backpropagation a discount factor γ can be employed to value far-away rewards less. To switch discounting off the discount factor is set to $\gamma = 1$.

Experimental Setup:
We test all different options by using MCTO to generate 100 CTEs each for a set of weights $\omega = \{\omega_{Validity_j}, ..., \omega_{Sparsity_j}\}_{j=1}^{100}$ that is uniformly sampled $\omega_i \sim U(0, 1)$. We measure the quality criteria and report on their quality values to compare options. As a default the following values are used, except for the parameters adapted in an experiment: $threshold_a = 0.003$, $p(terminal) = 0.2$, $\gamma = 0.85$, *n-iterations* $= 10$, *n-sims* $= 1$, *qc-expansion* and *policy-action* and *n-starts* $= 2$. Aside from altering the hyperparameters and measuring the resulting quality values, we also record the efficiency as the average time needed to generate a CTE ($\frac{seconds}{CTE}$).

Results:
Number of starting points: Figure C.3 shows how the quality value of CTEs increases when MCTO gets to pick from a larger number of starting points.

It displays a clear trend that increasing the number of starting points increases the quality of the resulting CTE. Further, the graph seems to be convex. This makes sense since having more different options to select the best from is strictly better, but marginal returns are hit quickly. However, this comes at the cost of efficiency, with the time needed to compute a CTE growing linearly with the number of starting points.

Getting to choose from more starting point leads is strictly better for performance. However, there are diminishing returns on investing more computing time into finding a better solution.

Number of iterations: We test multiple amounts of iterations per step for a setting using "random-expansion" and "qc-expansion".

Figure C.4 shows a slight improvement in quality values with increasing *n-iterations*. This is backed by the small correlation between *n-iterations* and quality values of 0.092 ($p = 0.01$) for *random-expansion*

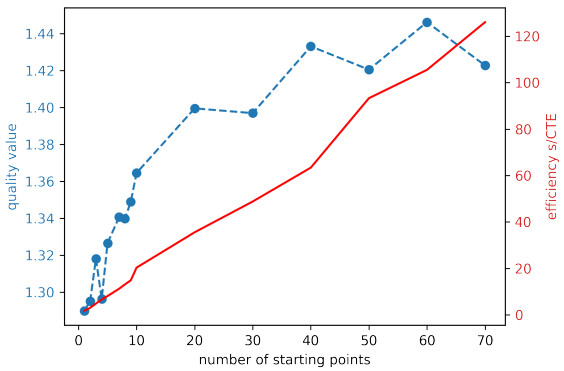


Figure C.3: Average quality values of quality criteria for CTEs created by MCTO with differing number of starting points along with the efficiency in $\frac{seconds}{CTE}$.

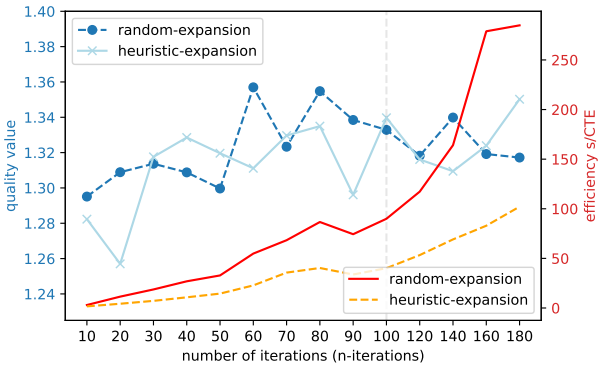


Figure C.4: Avg. Quality Value (left y-axis and blue graphs) for CTEs generated by MCTO and efficiency of generation (right y-axis with red graphs) using different numbers of iterations across a setting with (*heuristic-expansion*) and without (*random-expansion*) the expansion heuristic.

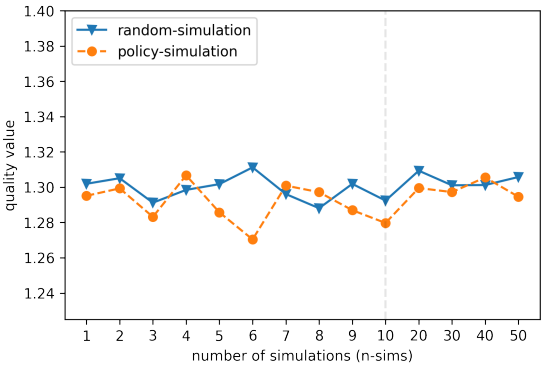


Figure C.5: Avg. Quality Value for CTEs generated by MCTO using different numbers of simulations across a setting using the policy (*policy-simulation*) and without (*random-simulation*) the expansion heuristic. At *n-sims* $= 10$ the increment of the x-axis changes from 1 to 10.

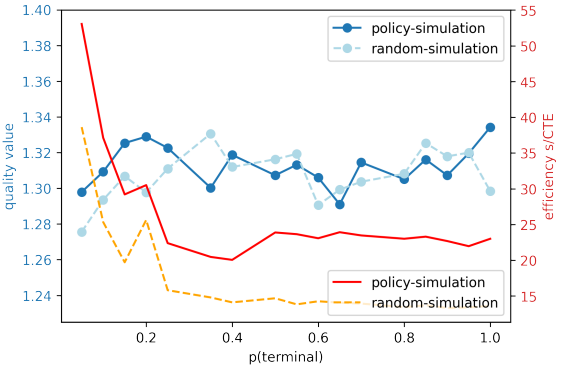


Figure C.6: Avg. Quality Value for CTEs generated by MCTO using different numbers of simulations across a setting using the policy (*policy-simulation*) and without (*random-simulation*) the expansion heuristic. At *n-sims* $= 10$ the increment of the x-axis changes from 1 to 10.

and 0.04 ($p = 0.18$) for *heuristic-expansion*. However, with every additional iteration new simulations need to be run leading to a roughly linear decrease in efficiency for *random-expansions*. This presents us with another trade-off between quality and computational efficiency.

Number of simulations: We test multiple amounts of simulations per extension with one setting sampling actions randomly (*random-simulation*) and one sampling according to the policy (*policy-simulation*) during the simulation.

Figure C.5 indicates that averaging estimates over more simulations does not improve the performance of MCTO. Both settings don't show a significant correlation between the *n-sims* and quality values. However, increasing *n-sims* comes at an approximately linear increase in computational cost.

Likelihood to terminate: Values between $[0.05, ..., 1.0]$ are used as the likelihood of the simulation to terminate in a given step.

Figure C.6 shows no clear trend in quality values when increasing *p(terminal)*. One can make out an increase in the early values ($0.05 - 0.2$) and a dip around $p(terminal) = 0.65$. However, these observations might be measurement inaccuracies rather than consistent trends. For low likelihoods of ending ($p(terminal) \leq 0.2$) there are significant increases in the computational cost for generating CTEs. This is because individual simulations become longer.

Action threshold: Multiple values for $threshold_a$, which determine which actions are viable as branches based on their likelihood according to the policy $\pi_\theta(s, a)$.

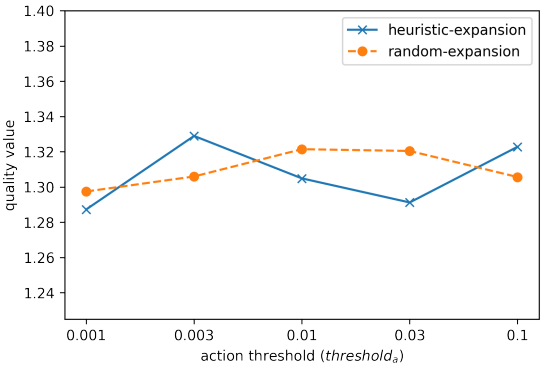


Figure C.7: Avg. Quality Value for CTEs generated by MCTO using different numbers of simulations across a setting using the policy (*policy-simulation*) and without (*random-simulation*) the expansion heuristic. At *n-sims* $= 10$ the increment of the x-axis changes from 1 to 10.

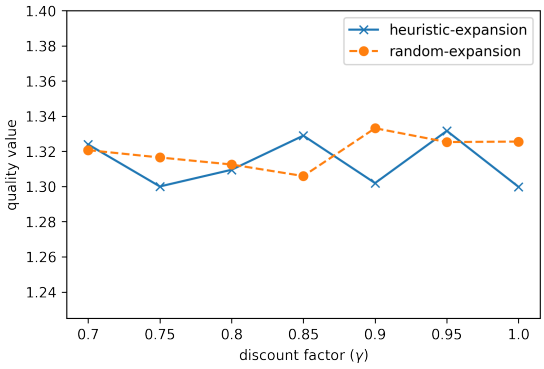


Figure C.8: Avg. Quality Value for CTEs generated by MCTO using different numbers of simulations across a setting using the policy (*policy-simulation*) and without (*random-simulation*) the expansion heuristic. At *n-sims* $= 10$ the increment of the x-axis changes from 1 to 10.

	$threshold_a$	γ	*n-iterations*
heuristic-expansion	1.307	1.314	1.315
random-expansion	1.310	1.320	1.323
p-value	0.912	0.793	0.640

Table C.4: The average quality value achieved by MCTO using *heuristic-expansion* or *random-expansions* across multiple values from $threshold_a$, γ and *n-iterations* respectively, along with the p-value indicating the statistical significance in differences between them.

There is no clearly best $threshold_a$ recognisable from Figure C.7. None of the differences between the action thresholds is significant and it can be concluded that this hyperparameter is not important for the functioning of MCTO.

Discount factor: We test values for the discount factor between $[0.7, .., 1.0]$ applied to the result of a simulation.

Figure C.8 does not show a significant impact of the discount factor on the quality value of resulting CTEs. There is also no significant correlation between those values. We conclude that adding a discount factor does not help or hinder MCTO in its search and we thus choose $\gamma = 1.0$ for simplicity.

Expansion Heuristic: To understand how the method for choosing which branch to extend influences the quality criteria of the resulting CTEs, we deploy *heuristic-expansion* and *random-expansion* across different values for $threshold_a$, γ and *n-iterations* and average the results.

For all three test cases, *random-expansion* slightly outperforms *heuristic-expansion*. However, none of the differences are statistically significant.

This indicates that an early estimation of the quality criteria is not a good heuristic to guide the expansions in the search tree. This might be because the early estimations do not accurately reflect how promising a branch is to extend. Furthermore, using the heuristic comes at a significant computational cost. Across all *n-iterations heuristic-expansion* needs $61.3\frac{s}{CTE}$, while *random-expansion* only takes $23.2\frac{s}{CTE}$ (see Figure C.4.

Action selection during simulation: We compare the two proposed methods for selection actions during the simulation, *policy-simulation* and *random-simulation*, across different values for *n-sims* and $p(terminal)$ and average the quality values of the resulting CTEs.

Table C.5 shows that there is no significant difference between *policy-simulation* and *random-simulation*. However, Figure C.6 shows us that following the policy is less efficient than choosing randomly.

	n-sims	$p(terminal)$
policy-simulation	1.327	1.295
random-simulation	1.312	1.302
p-value	0.251	0.657

Table C.5: The average quality value achieved by MCTO using *policy-simulation* or *random-simulation* across multiple values from *n-sims* and $p(terminal)$ respectively, along with the p-value indicating the statistical significance in differences between them.

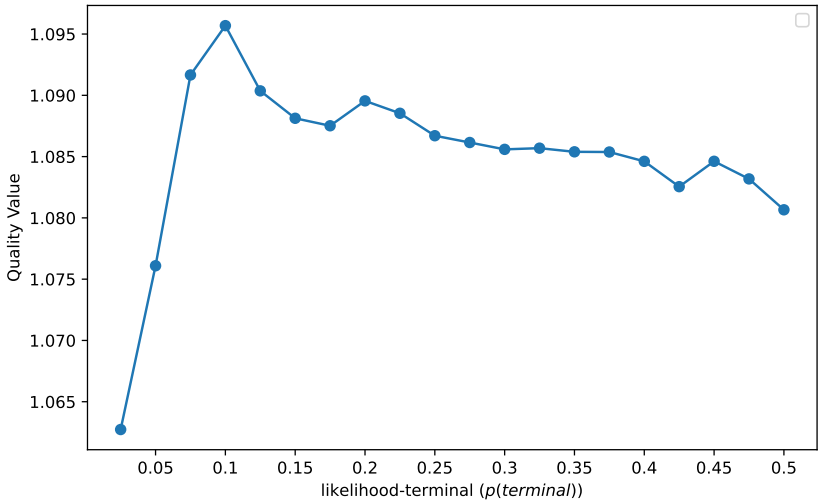


Figure C.9: Average Quality Value achieved by Random with different likelihoods of termination

Design Decisions and Hyperparameters:
n-starts$= 70$ is selected, because the authors deem the increased quality values worth the lower efficiency in this setting. However, increasing the number of iterations per choice only leads to small improvements and thus *n-iterations*$= 10$ is chosen. $p(terminal) = 0.35$ is chosen since it leads to the best performance for *random-simulation* and is computationally efficient, despite acknowledging that the high quality value is likely due to noise in the measurement. *Expansion-random* is clearly preferable, since *expansion-heuristic* worsens performance and efficiency. Thus, in further experiments, expansions should be chosen randomly instead. Additionally, random simulations are performed due to their efficiency. Considering their negligible effect on the performance, we set $threshold_a = 0.003$ and $\gamma = 1.0$.

C.2.3. Ablations of Random CTE Generation

To make Random a stronger baseline comparison we perform a hyperparameter search to find the best way of choosing when to end the random trajectory.

Experimental Setup:
Similarly to DaC and MCTO Random has a likelihood of $p(terminal)$ ending the counterfactual and original trajectory at every step. For multiple values of $p(terminal)$ 1000 CTEs are generated and their quality criteria are measured.

Results
Figure C.9 shows a clear trend where the average quality values achieved by Random first rise until $p(terminal) = 0.2$ and then steadily fall. However, the performance of Random does not differ greatly with quality values only differing in a range of < 0.04.

C.3. Ablations of Informativeness Measure

To validate that our Informativeness Measure using a proxy-human model works well we investigate the feature extraction and the multi-task learning.

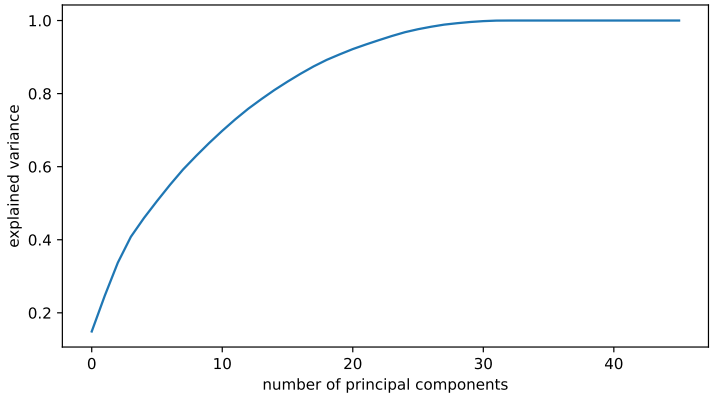


Figure C.10: PCA of the extracted features

Number of features	10	20	30	40	46
Mean Squared Error	0.469	0.360	0.237	0.216	0.217

Table C.6: Mean Squared Error on unseen CTEs when a Stepwise Linear Model is trained on CTEs, but can only use a set number of the 46 available features.

C.3.1. Importance of Extracted Features

In order for the proxy-human model to learn from the CTEs, 46 handcrafted features are extracted from the original and counterfactual trajectory (see Section B.2). We want to better understand the data and make sure it does not cause problems when training.

Experimental Setup:

Firstly, in order to find out whether the extracted features contain much redundant information we perform Principal component analysis (PCA). By increasing the number of principal components from 1-46 and measuring how much of the variance in the data is explained by these principal components we can get an understanding of how much redundant information is in the extracted features.

Secondly, the curse of dimensionality describes problems caused by high-dimensional spaces [43] and relates to ML by stating that adding more features without adding more data points can hurt the performance of an ML model [23]. To make sure the proxy-human regression model does not suffer from having too many features we train a step-wise linear model on different amounts of features. This tests whether all features are important for model performance or are better left out. To test this we train a stepwise linear model on 800 CTEs to extract the most important features for the number of features 10, 20, 30, 40, and 46 and measure its performance on the test set using Mean Squared Error (MSE).

Results & Discussion:

Figure C.10 shows that almost all of the variance in the data is explained with 30 principal components. This indicates that there is some redundant information in the features. However, this is unsurprising, since many of the features will be correlated with each other, while still contributing some relevant information.

Table C.6 showed that there was no improvement in the performance of a linear model when training on fewer features. While the last remaining features do not add much performance, they also don't hurt it. Thus we use the full 46 features.

C.3.2. Task weights in Multi-Task Learning

The proxy-human model is trained in a multi-task fashion to achieve 2 tasks (see Section B.4. It learns to predict the difference in rewards (contrastive) and the reward individually (single). The weight assigned to each task indicates its importance.

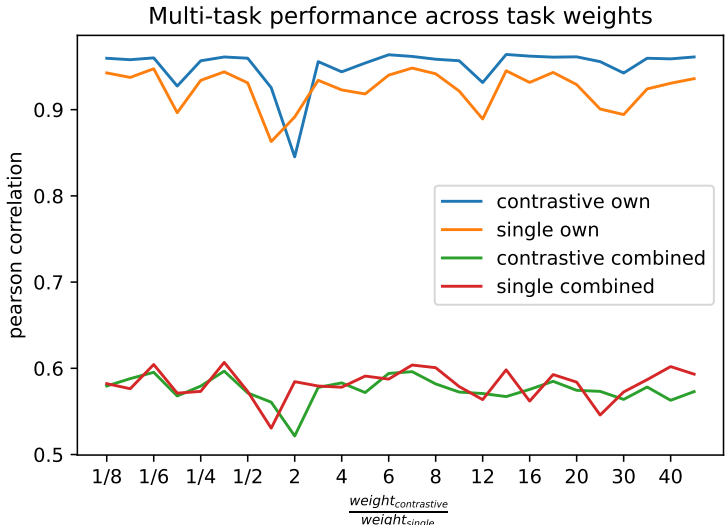


Figure C.11: Pearson correlation of model predictions with true labels on the own and combined test set for the tasks contrastive and single along different relative weights between those 2 tasks.

Experimental Setup

In order to find a good trade-off between the two tasks we try different weights for them and record the Pearson Correlation of the tasks on an unseen combined dataset. Notably, what matters here is the relative weight between contrastive and single and not their absolute weight. Thus we use the relative weights $\frac{weight_{contrastive}}{weight_{single}} \in \{\frac{1}{8}, \frac{1}{7}, ...45\}$.

Results & Discussion:

Surprisingly, Figure C.11 does not show a trend of performance changing with different task weights. For both tasks, the performance on the own test set and combined test set are steady with some noise.

Although it could have been expected that increasing the relative importance correlates with its performance, this is not the case. It seems there is enough shared information between the tasks that they benefit each other instead of trading off against each other. This makes sense since the tasks really are quite similar. Illustratively, calculating the contrastive reward can simply be done by taking the difference of the single reward. This can raise our confidence that providing the contrastive perspective really does help to predict rewards for single trajectories.

C.4. Influence of Quality Criteria on DaC

This section explores the correlations between criterion weights and informativeness using different generation methods and regression models than Experiment 3 in the main paper.

C.4.1. Informativeness of quality criteria using DaC

Experimental Setup:

The same experimental setup as described in Experiment 2 of the main paper (see Section 1.6.2) is used to evaluate the influence of the weights ω of quality criteria for the informativeness of CTEs. In contrast to section 1.6.2 the CTEs are generated by DaC instead of MCTO.

Results:

The weights for Validity $\omega_{Validity}$ correlate the strongest with contrastive performance and also strongly with single performance. $\omega_{Realisticness}$ stands out as highly correlated with single, despite being only slightly correlated with contrastive. Higher weights for Diversity also indicate higher performance on both tasks. $\omega_{StateImportance}$ is not strongly correlated with both task performances.

For both contrastive and single learning Validity is the most important criterion. $\omega_{Proximity}$ and $\omega_{Diversity}$ also correlate with the performance of the regression model on both tasks. Interestingly, $\omega_{Realisticness}$

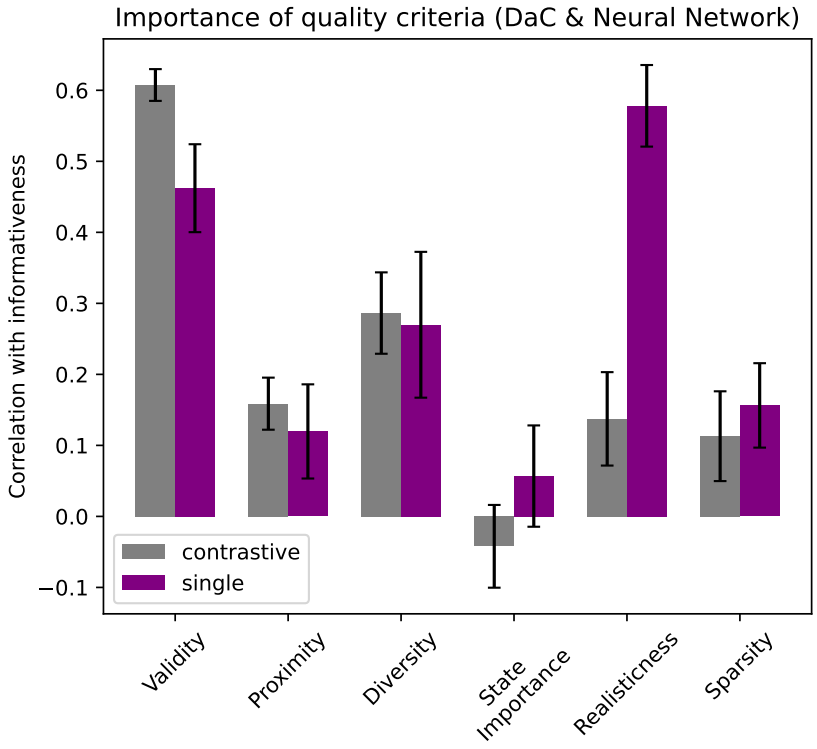


Figure C.12: For each quality criterion the Spearman correlation between its weight used to generate CTEs via DaC and the performance of the regression model trained on the resulting CTEs on the contrastive and single task. The results are averaged over 10 models with different initialisations and the standard deviation for different models is displayed.

has a strong influence on single but is less important for contrastive learning. $\omega_{Sparsity}$ do not strongly correlate with the performance of the model on the tasks, while $\omega_{StateImportance}$ even negatively correlate with the performance of the model on the contrastive task. The two tasks largely agree on how important each criterion is to their performance. This is not the case for Realisticness and it's unclear why, especially because this difference did not appear for CTEs generated by MCTO.

Discussions:
The results for DaC present notable differences from those for MCTO (see Experiment 2 in section 1.6.2). The importance of Realisticness on DaC is much higher than on MCTO. State Importance and Proximity are generally less important for DaC, while Sparsity is more important for MCTO. However, there are also similarities. Validity is very important for both tasks, and the correlations for Diversity are similar for the two settings.

These results can increase our confidence that Validity and Diversity are generally important criteria for a Neural Network to learn from. The results also show that the priority between quality criteria does depend on the specific generation method that is used. Different generation methods have different ways of navigating the trade-offs between quality criteria. Thus it is unsurprising that the correlation between criteria weights and downstream performance differs between generation methods.

C.4.2. Most and Least informative set of weights

To provide further insight into which quality criteria are important to create informative CTEs, we show the set of weights that lead to the most and least informative CTEs. The most informative weights described here are also those used to generate CTEs in Experiment 3 (see Section 1.6.3).

Experimental Setup:
For experiment 2 (see Section 1.6.2) we collect the single best and worst performing set of weights $\omega \in \omega = \{\omega_{Validity_j}, ..., \omega_{Sparsity_j}\}_{j=1}^{30}$. We use both MCTO and Dac to generate CTEs and a Neural Network and Linear Models to learn from them.

	Validity	Proximity	Diversity	State Importance	Realisticness	Sparsity
best weight contrastive	**0.982**	**0.98**	0.576	0.528	0.303	**0.851**
worst weight contrastive	0.126	0.213	0.492	**0.943**	0.052	**0.752**
best weight single	**0.878**	**0.92**	**0.915**	0.674	0.639	0.657
worst weight single	0.17	0.496	0.205	**0.968**	0.203	**0.633**

Table C.7: The sets of weights (out of 30) that lead to the most and least informative CTEs for a Neural Network when optimised by MCTS for the contrastive and single task.

	Validity	Proximity	Diversity	State Importance	Realisticness	Sparsity
best weight contrastive	**0.902**	0.658	0.405	0.587	0.12	**0.988**
worst weight contrastive	0.006	0.267	**0.891**	**0.924**	0.353	0.579
best weight single	**0.878**	**0.92**	**0.915**	0.674	0.639	0.657
worst weight single	0.412	**0.836**	0.38	0.087	0.042	**0.748**

Table C.8: The sets of weights (out of 30) that lead to the most and least informative CTEs for a Neural Network when optimised by DaC for the contrastive and single task.

Results:
Table C.7 shows that the most informative CTEs generated by MCTO for a NN have high Validity and Proximity, while the least informative ones have high State Importance and Sparsity. For the most informative weights for contrastive $\omega_{Sparsity}$ was high, while $\omega_{Diversity}$ was high for single.

When generating highly informative CTEs with DaC (see Table C.8) for the contrastive task $\omega_{Validity}$ and $\omega_{Sparsity}$ were high, while the least informative CTEs were generated with high $\omega_{Diversitiy}$ and $\omega_{StateImportance}$. To score high on the single task, $\omega_{Validity}$, $\omega_{Proximity}$ and $\omega_{Diversity}$ were high, while $\omega_{Sparsity}$ and $\omega_{Proximity}$ were high for the lowest scores.

While DaC and MCTO agree which set of weights leads to the most informative CTEs for the single task, they disagree on the three other settings.

Discussion:
These results again highlight the importance of Validity for creating CTEs that are informative to a Neural Network. Furthermore, they indicate that State Importance can be harmful. For MCTO Proximity and Diversity are clearly helpful. On DaC there are less clear claims that can be made as Proximity, Diversity and Sparsity appear with high weights when informativeness was high and when it was low.

The fact that MCTO and DaC agree on one out of four settings, reinforces the claim that different generation methods do not converge to the same priorities between criteria. However, all results should be taken lightly, since they are only based on 30 different options for weights.

C.5. Using a Linear Model as proxy-human model

In the main experiments, we use a Neural Network to learn from the CTEs. To test whether our results are robust to using a different type of learner we rerun the experiments using a Linear Regression Model (LM). If results are similar this can give us some indication that they are not only specific to a Neural Network but might also be applicable to other learners. This could even possibly to humans However, humans are not mathematical models optimised via Gradient Descent, which necessitates a user study before making such conclusions (see Appendix D.3).

The multi-task learning of the proxy-human model is adopted by training two Linear Models. While they receive the same input, the first has one output that predicts contrastive rewards, the other has two outputs that predict the single rewards.

C.5.1. Importance of Quality Criteria for a Linear Model

This experiment aims to test whether the previous results about the importance of quality criteria hold up when using Linear Regression Models (LMs) instead of a Neural Networks (NNs) as a proxy-human model. It repeats the Experimental Setup described in Experiment 2 (see Section 1.6.2), except that

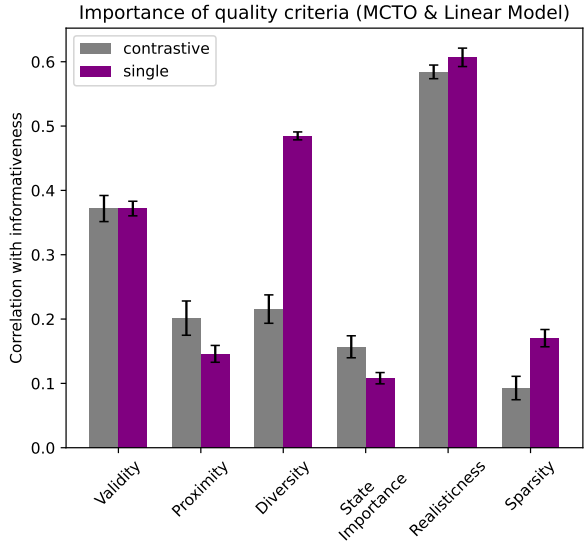


Figure C.13: Spearman correlation between the weights of quality criteria and the performance of the Linear Regression Model trained on CTEs generated by MCTO on the contrastive and single task averaged over 10 seeds.

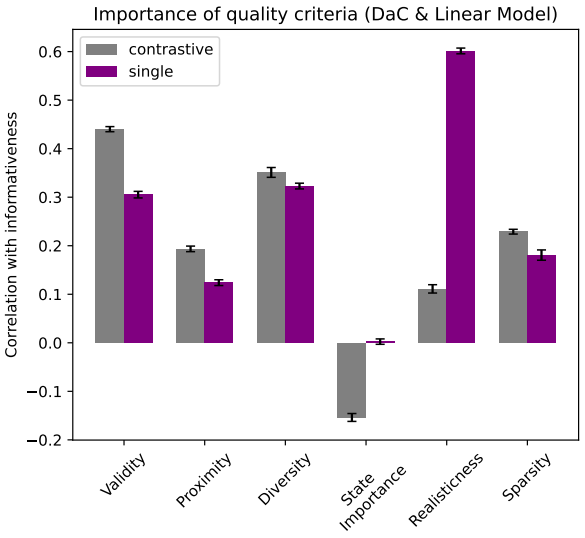


Figure C.14: Spearman correlation between the weights of quality criteria and the performance of the Linear Regression Model trained on CTEs generated by DaC on the contrastive and single task averaged over 10 seeds.

the proxy-human model is two LMs instead of a Neural Networks. Specifically, we use Simple Linear Models with a bias term trained via regression.

Experimental Setup:
To determine the influence of quality criteria on the informativeness of the generated CTEs, 30 sets of weights $\omega = \{\omega_{Validity_j}, ..., \omega_{Sparsity_j}\}_{j=1}^{30}$ were uniformly sampled $\omega \sim U(0,1)$ and used to generate 30 sets of 800 CTEs per generation method. These sets of CTEs were used to train two Linear Models and the performance of each resulting model was measured as the Pearson correlation of its outputs to the true labels of the combined test. For each method, this allows us to approximately find the set of weights which lead to the most and least informative CTEs. By analysing the correlations between the $\omega_{criterion}$ and model performance we can derive the relative importance of each quality criterion. A higher Spearman correlation between the weights and the informativeness indicates that the quality criterion is more important.

The Linear Models was trained with learning rate$= 0.1$ and weight decay regularisation of 0.01.

Results:
For the Linear Model trained on CTEs generated by MCTO Realisticness was the most important criterion for both tasks (see Figure C.13). Furthermore, it stands out that Diversity is especially important for single, while Validity is important for both tasks. For all other criteria, their weights are slightly positively correlated with the performance of the proxy-human Linear Model.

Figure C.13 also shows Realisticness as the most important criterion for single, while the weights for Validity $\omega_{Validity}$ correlate strongest with the Linear Models performance on contrastive. $\omega_{Diversity}$ proves important for both tasks, while $\omega_{StateImportance}$ is negatively correlated with contrastive performance.

Discussion:
Overall the results for the importance of quality criteria for informativeness are similar when using a Linear Model or Neural Network.

Notable differences for MCTO are that Realisticness is considerably more important for both tasks and Diversity is more important for single. Furthermore, State Importance and Sparsity now have a slight correlation with performance. Validity is relatively less important for the Linear Model than for the Neural Network. For DaC the importance of quality criteria for informativeness is very similar for the Linear Model and Neural Network.

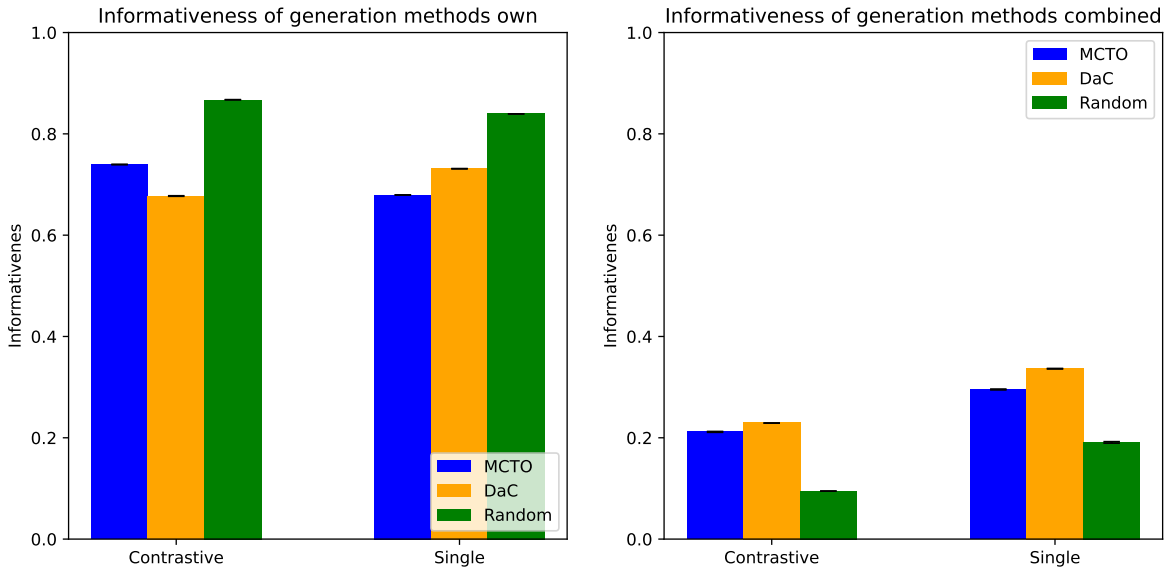


Figure C.15: The informativeness of CTEs generated by MCTO, DaC and random for a Linear Model. Performance for the single and contrastive task are shown on the respective own (left graph) dataset and the combined dataset (right graph).

Overall there are some differences in importance when using MCTO and very few for DaC. This can give us more confidence that the results are not extremely influenced by which model is used as a learner. However, we should not simply extrapolate that results will be similar for all receivers to explanations, this at least indicates some level of generality and robustness about our findings about the importance of quality criteria.

C.5.2. Informativeness of Explanations for a Linear Model

It's unclear whether CTEs are especially informative when using a Neural Network as a proxy-human model or whether less complex architectures benefit equally from CTEs. Thus we repeat Experiment 3 (see Section 1.6.3) with Linear Models as proxy-human models.

Experimental Setup:

10 pairs of Linear Models were trained on 800 CTEs generated by MCTO, DaC and Random. Each of these methods was optimised for the set of weights found to lead to the most informative CTEs. They were then tested on a test set of 200 unseen CTEs generated by the same method (*own*) and on a *combined* test set that has 600 examples from all three methods. The main measurement is the Pearson Correlation between model predictions and labels.

Results:

On the "own" test-set the LM learns best from the randomly generated CTEs for both tasks. For contrastive learning CTEs from MCTO are more informative for the LM than those from DaC. For the single predictions, DaC outperforms MCTO. Overall the correlations to the true labels are between $0.65 - 0.9$.

On the combined dataset average performances range between $0.6 - 0.38$. DaC performs best on both tasks, while MCTO outperforms Random. Further, there is very little deviation between models with different initialisations.

Discussion:

In contrast to the Experiment using a NN DaC generalises better than MCTO. Overall the LM achieves a significantly lower performance on the own dataset and that on the combined dataset. Likely, the added complexity of the Neural Network enables it to learn more complex functions that explain the rewards. This also indicates that the Neural Network learned more complex knowledge about the reward function which is not straightforwardly learned by a LM.

The reason why Random performs best is likely because the dataset is easier to predict. While the LMs trained on randomly generated CTEs perform well on-distribution, they are worse at generalising off-distribution. This indicates that they learned less generalizable knowledge about the reward function.

It should be added that the Linear Model does not benefit from the same cross-task learning as the Neural Network. While the NN trains the same body for the contrastive and single labels, thus making use of the information in both tasks, the LMs are trained separately for the different tasks. Furthermore, the fact that there is little deviation between model initialisations hints at the fact that these models converge to the same local minimum.

The fact that DaC produces more informative CTEs for a Linear Model than MCTO, calls into question the conclusion that MCTO is the most effective generation method. However, the generally bad performance of the LMs makes it a relatively weak piece of evidence. While MCTO still achieves higher quality values and produces CTEs that are more informative for a more complex model, we cannot claim that MCTO is the best choice for any learner.

C.6. Summary of Further Experiments

This further set of experiments underlines our takeaways presented in the main article.

In section C.1 we explored the trade-offs between quality criteria and how much influence different quality criteria have on the final CTE. There are clear trade-offs between some quality criteria making this a hard Multi-objective optimization problem, which requires the generation methods to adjust accordingly. On different measures of influence. We discuss that this effect is caused by the fact that the distribution of values are different between quality criteria.

We conduct rigorous ablations and hyperparameter searches for the generation methods MCTO, DaC and Random. These ensure that the methods perform at their maximum capacity and are not held back by a bad choice of hyperparameters. For MCTO we also explore the trade-offs between effectiveness and efficiency across multiple parameters. Further, we test multiple heuristics for MCTO, which do not lead to significant improvements in the method.

To validate our measure of informativeness we explore whether using too many related features could reduce model performance. We find that although there is significant shared information between the features, training on fewer features does not improve performance. This ensures that the experiments are not using too many features, hampering the performance. Further, we explore the trade-offs between contrastive and single learning. We find that there are no significant trade-offs between these tasks, indicating that they do synergise very well together.

Further, we validate the Experiments about the importance of quality criteria (1.6.2) by using DaC instead of MCTO to create CTEs. Despite Realisticness being more important and State Importance being the two settings give similar priorities to the quality criteria. This raises our confidence that the findings presented in the main paper generalise to different methods.

Lastly, we validate our results in Experiments 2 and 3 (see section 1.6.2 and 1.6.3) by using Linear Models instead of Neural Networks as proxy-human models. There are only a few differences in which criteria are important to create CTEs that are informative to an LM and an NN. When comparing generation methods DaC stands out as slightly more informative than MCTO. This decreases our confidence in the claim that MCTO is the most effective generation method. Overall the higher complexity of the NN makes it much better at predicting the rewards.

D

Extended Challenges and Future Directions

D.1. Limitations

There are a number of limitations to the design and execution of this study, which limit the conclusions we can draw from it.

This report uses proxy-human models based on Neural Networks or Linear Models to learn from the explanations. However, in the envisioned applications a human user would consume the CTEs. While results are promising, esp. for the Neural Networks, we cannot simply extrapolate this to claim that humans will also be able to learn well from CTEs. Human brains are not mathematical models trained on handcrafted features via Gradient Descent. This will likely mean human users will prefer a different prioritisation between quality criteria. Furthermore, this could mean that humans learn better or worse from the CTEs than a Neural Network. Thus a user study will be necessary to prove the efficacy of this method. In section D.3 we outline some considerations for a user study.

The experiments use a fairly simple 6x6 grid world environment with only 5 different types of cells. Reinforcement Learning agents are often employed in highly complex environments. In a more complex environment, there would be more complicated and interconnected aspects a reward function has to consider and there would be more different possible situations the reward function could encounter. Thus a more complex environment would likely require a more complex model as a reward function. It is not clear how well the method of CTEs will scale to explain more complex reward functions in more complex environments. One specific issue might be that a lot of different situations would need to be covered to get a full picture of the reward function.

The quality criteria were identified by a literature search as criteria that are often emphasised as important when creating counterfactuals. While the justification for each criterion is solid, their specific implementation was based on the researchers' choice. When possible we tried to stay close to previous interpretations but otherwise attempted to capture the idea behind the quality criterion faithfully. However, some of these design choices are subjective and can be debated. Thus, we would encourage more exploration and empirical investigations into the design of quality criteria in the setting of Reward Learning and Reinforcement Learning.

Compared to DaC or the random baseline, MCTO is the most effective generation method. MCTO performs a much deeper exploration of the space of possible trajectories than DaC and Random making it an unfair comparison. Other optimisation methods should be applied and tested against MCTO.

While the combined test set makes for a fair comparison between different settings for creating CTEs, there might be more comprehensive test sets that could give more general insights. An ideal test set would cover most possible situations the reward function could be in and thus give a comprehensive overview of how well the proxy-human model learned to predict similarly in a wide range of situations.

The features we extracted to then train the proxy-human model on are often averaged or summed over the trajectory. This loses some details present in single states which are a cause of rewards. Thus this method of extracting features inherently causes a loss of information about the reward function that the proxy-human model will miss.

Lastly, the policy π_θ which was trained on R_θ plays a central role in generating CTEs. It creates the trajectories τ_{org} from which a partial trajectory is used as t_{org} in the CTE. Furthermore, it aids in guiding the generation of counterfactuals and rating the quality of CTEs. This has two problems. Firstly, it biases the provided explanation to be similar to the policy and thus not cover a wider space of possible situations. Secondly, this means it's necessary to first train a policy in the environment before we can explain the reward function. In some safety-critical situations where misalignments of a policy might cause harm, we would like to directly explain the reward function before using it to train a policy. However, this is not possible if the policy is needed to generate the explanations.

D.2. Implications

This section contains speculations about the implications of this research for the ability to identify and address misalignments between a learned reward function and the human's objectives. Claims here should not be taken as deeply justified research results but as loose thoughts on the applicability of this method.

Our approach of generating Couterfactual Trajectory Explanations that help a user to interpret a reward function fits into the Transparent Value Alignment Framework [60]. This presents Value Alignment as an iterative process where the user provides information to an AI, from which the AI learns a reward function and then provides the user with Explanations to aid them in identifying misalignments of the reward function with their own objectives.

Our method learns some, but not all aspects of the reward function. While this can help uncover *some* misalignments between the human values and the reward function, it cannot be expected to find *all* such misalignments. Even if the misalignments found by this method are addressed we should not be confident that the reward function is now fully aligned with the human's intentions.

Anecdotally, the researchers and a few informal test subjects found it helpful to look at CTEs and see examples where the reward functions were not making choices that were aligned with the ground-truth objective function. For example, we found that there were situations where the reward function being tested did not value using the fire extinguisher positively despite the ground-truth reward function giving a reward of +1 for using it. Further, the reward function often assigned higher rewards if the player was standing on a human, even in situations where this brought no apparent benefit. These insights would be very valuable to fix the misalignments by providing new demonstrations. For example, the demonstrator could show behaviour which makes the value of the fire extinguisher clear.

For a successful application of this method to humans, it would be necessary that humans are able to learn in a more sample-efficient way from CTEs since it is not feasible to show a human 800 examples. Likely humans do better at this than a Neural Network, since they are generally better at few-shot learning.

D.3. Future Work

This section outlines concrete proposals for extensions to this study and points to some interesting applications of the framework and method.

User study:
The obvious next step in this line of research is to test how informative the generated CTEs are for humans. Ultimately, the purpose of such interpretability techniques is to help humans understand the model. Research on counterfactual explanations has been criticised for neglecting to test methods with user studies [42].

Concretely, a group of humans should be shown a set of CTEs generated via MCTO. This would be contrasted to a group, which would be shown twice the amount of single trajectories [35] or state-action pairs along with their rewards. Further, a comparison could be made to [49] by also providing a group

of users with Feature Attributions.

The informativeness of the different techniques could be measured in multiple ways. The experiment could test participants' ability to predict the rewards the reward function would assign to an unseen trajectory [4]. Additionally, the metrics of complexity and understanding developed by Sanneman and Shah [59] should be considered. Alternatively, they could be asked to identify a deliberately introduced flaw in the reward function [55] or to distinguish between a reward function they received explanations about and other reward functions [28]. Further, subjective ratings about the informativeness, satisfaction and mental effort required can be collected from the user [45, 46]. Lastly, improvements in a downstream task like collaboration between the AI and a human [65] or the quality of further feedback provided by the human can be taken as measures [63]. Lastly, it will be crucial to test how efficiently humans can learn from CTEs.

Improvements to the method:
The current way CTEs are displayed in the Emergency environment makes them not easily digestible to users and thus limits their effectiveness. Designing a User Interface which allows humans to easily digest the provided information could improve the effectiveness of the method.

Improvements can be made when solving the multi-objective optimization problem of choosing the best CTE based on multiple quality criteria. Future work could explore the use of Pareto-based techniques methods to ensure that the chosen solution lies on the Pareto front and is not dominated by another. Further, during the calibration phase, a more efficient way of sampling weights that takes into account the results of previous sets of weights can be introduced. Candidates for this would be Bayesian Optimization [3] or Evolutionary Algorithms [22].

The method could be made more environment-agnostic. Currently, the measurement of Proximity and Feature Extraction rely on specific knowledge about the environment e.g. humans and obstacles are important aspects of the environment. To make the method easily adaptable to other RL environments, these aspects should be generalised to not rely on domain knowledge.

Alternative Framings: While this work chose to create counterfactuals for *trajectories*, future work could opt to create counterfactuals for single *state-action pairs*. Although trajectories do give insight into longer behaviour and multi-step planning, counterfactual state explanations might be easier to understand and generate. This would require a new set of generation methods. In the initial exploration of this research project, we implemented a method which introduced random permutations to a state-action pair by adding or removing objects. Out of the generated permutations, the one which led to the largest change was shown to the user. This provided explanations which were anecdotally useful to the researchers but lacked Diversity. Further, it would require redefining the quality criteria, since current criteria are specifically designed for rating counterfactual trajectories. As an advantage during evaluation, it might be possible to skip the feature-extracting step and let a Neural Network learn directly from state-action pairs.

The use of reward learning has recently become a key driver of progress in NLP through the use of Reinforcement Learning from Human Feedback (RLHF) [10]. Using the framework presented in this paper, the reward functions learned by RLHF could be explained with pairs of original and counterfactual sentences. In this setting the original sentence generated by the Language Model that was trained on the reward function receives a reward and is compared to a counterfactual sentence. This would require the development of new generation methods and new implementations of the quality criteria. While Validity could still refer to the difference in average rewards, Proximity could be measured as the cosine similarity of the sentence embedding.

Applications of the explainability tool:
Lastly, it would be great to see the tools developed in this study to actually investigate learned reward functions and find out more about their inner workings. For example, the tool could be used on multiple reward learning algorithms to gain a more nuanced understanding of the differences in the reward functions they produce. For example, AIRL[29], PbRL [18] and MaxEnt IRL [75] could all be trained based on the same ground-truth reward function and policy and then be investigated deeply using CTEs. This could produce insights into the workings and differences of these algorithms.

Furthermore, this method of generating counterfactual trajectories might have applications in Preferences-

based Reinforcement Learning when deciding which queries should be shown to a user to collect their preferences. While queries are currently often chosen to according to the uncertainty the reward function has about them [18, 20], it could prove beneficial to also consider the quality criteria derived in this study to adapt the queries to a user.

Ultimately we hope this explainability technique can be employed in the larger framework of Transparent Value Alignment [60]. In this framework, a user provides information in the form of demonstrations or feedback. from which a reward function is learned to capture the human's values. Next, the Counterfactual Trajectory Explanations about the learned reward function can be generated and shown to the user. Based on these the user can identify differences between the values they hold and those captured by the reward function and provide additional information to address these differences (see Figure 1.2). To make this work, future research could:

1. Improve upon the explainability techniques presented in this paper to help humans identify flaws in the reward function,
2. Help the human to effectively choose new information to provide to the reward function, which addresses the identified flaws or
3. Update the reward function appropriately with the newly provided information.

D.4. Acknowledgments

First and foremost I would like to thank my Daily Supervisor Luciano Siebert for many hours of valuable advice, insightful discussions and supportive guidance throughout the last 9 months. Additionally, discussions with Frans Olliehoek provided many valuable ideas and proved extremely useful. Thanks also go out to Ujwal Gadiraju for being on my thesis committee. Markus Peschl and Adam Gleave gave valuable comments early on, which helped to shape the direction this project took. Lastly, I want to give thanks to all my co-working partners who motivated and inspired me throughout this project and made long working days fun. Especially Anton von Hünerbein, Pierre Bongard, Koen van Pelt and Anitha Koshy deserve to be highlighted.

References in Appendix

[1] Nematollah Ab Azar, Aref Shahmansoorian, and Mohsen Davoudi. “From inverse optimal control to inverse reinforcement learning: A historical review”. In: *Annual Reviews in Control* 50 (2020), pp. 119–138.

[2] Stephen Adams, Tyler Cody, and Peter A Beling. “A survey of inverse reinforcement learning”. In: *Artificial Intelligence Review* 55.6 (2022), pp. 4307–4346.

[3] Apoorv Agnihotri and Nipun Batra. “Exploring Bayesian Optimization”. In: *Distill* (2020). DOI: 10.23915/distill.00026.

[4] Amal Alabdulkarim, Gennie Mansi, Kaely Hall, and Mark O. Riedl. *Experiential Explanations for Reinforcement Learning*. 2023. arXiv: 2210.04723.

[5] Kareem Amin and Satinder Singh. “Towards resolving unidentifiability in inverse reinforcement learning”. In: *arXiv preprint arXiv:1601.06569* (2016).

[6] Dan Amir and Ofra Amir. “HIGHLIGHTS: Summarizing Agent Behavior to People”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ’18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1168–1176.

[7] Stuart Armstrong and Sören Mindermann. “Occam’s razor is insufficient to infer the preferences of irrational agents”. In: *Advances in neural information processing systems* 31 (2018).

[8] Thomas Arnold and Daniel Kasenberg. “Value Alignment or Misalignment - What Will Keep Systems Accountable?” In: *AAAI Workshop on AI, Ethics, and Society*. 2017.

[9] André Artelt and Barbara Hammer. “On the computation of counterfactual explanations–A survey”. In: *arXiv preprint arXiv:1911.07749* (2019).

[10] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. 2022. arXiv: 2204.05862.

[11] Tom Bewley, Jonathan Lawry, Arthur Richards, Rachel Craddock, and Ian Henderson. “Reward Learning with Trees: Methods and Evaluation”. In: *arXiv preprint arXiv:2210.01007* (2022).

[12] Tom Bewley and Freddy Lecue. “Interpretable Preference-based Reinforcement Learning with Tree-Structured Reward Functions”. In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 2022, pp. 118–126.

[13] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. “Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 783–792. URL: https://proceedings.mlr.press/v97/brown19a.html.

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

[15] Ruth MJ Byrne. *The rational imagination: How people create alternatives to reality*. MIT press, 2007.

[16] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. *Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback*. 2023. arXiv: 2307.15217.

[17] Xi-liang Chen, Lei Cao, Zhi-xiong Xu, Jun Lai, Chen-xi Li, et al. "A study of continuous maximum entropy deep inverse reinforcement learning". In: *Mathematical Problems in Engineering* 2019 (2019).

[18] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

[19] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[20] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. "Active Reward Learning". In: *Robotics: Science and Systems X* (2014).

[21] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. "Causal confusion in imitation learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[22] Kalyanmoy Deb. "Multi-objective optimisation using evolutionary algorithms: an introduction". In: *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 2011, pp. 3–34.

[23] Essam Debie and Kamran Shafi. "Implications of the curse of dimensionality for supervised learning classifier systems: theoretical and empirical analyses". In: *Pattern Analysis and Applications* 22 (May 2019). DOI: 10.1007/s10044-017-0649-0.

[24] M.-P. Dubuisson and A.K. Jain. "A modified Hausdorff distance for object matching". In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. 1994, 566–568 vol.1. DOI: 10.1109/ICPR.1994.576361.

[25] Kai Epstude and Neal J Roese. "The functional theory of counterfactual thinking". In: *Personality and social psychology review* 12.2 (2008), pp. 168–192.

[26] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 49–58. URL: https://proceedings.mlr.press/v48/finn16.html.

[27] Pedro Freire, Adam Gleave, Sam Toyer, and Stuart Russell. "Derail: Diagnostic environments for reward and imitation learning". In: *Proceedings of the Workshop on Deep Reinforcement Learning at NeurIPS, 2020* (2020).

[28] Julius Frost, Olivia Watkins, Eric Weiner, Pieter Abbeel, Trevor Darrell, Bryan Plummer, and Kate Saenko. "Explaining Reinforcement Learning Policies through Counterfactual Trajectories". In: *arXiv preprint arXiv:2201.12462* (2022).

[29] Justin Fu, Katie Luo, and Sergey Levine. "Learning robust rewards with adversarial inverse reinforcement learning". In: *arXiv preprint arXiv:1710.11248* (2017).

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

[31] Lu Gan, Jessy W. Grizzle, Ryan M. Eustice, and Maani Ghaffari. "Energy-Based Legged Robots Terrain Traversability Modeling via Deep Inverse Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8807–8814. DOI: 10.1109/LRA.2022.3188100.

[32] Adam Gleave, Michael D Dennis, Shane Legg, Stuart Russell, and Jan Leike. “Quantifying Differences in Reward Functions”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=LwEQnp6CYev.

[33] Riccardo Guidotti. “Counterfactual explanations and how to find them: literature review and benchmarking”. In: *Data Mining and Knowledge Discovery* (2022), pp. 1–55.

[34] 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)*. 2018, pp. 3929–3936. DOI: 10.1109/IROS.2018.8593649.

[35] Sandy H. Huang, David Held, Pieter Abbeel, and Anca D. Dragan. “Enabling robots to communicate their objectives”. In: *Autonomous Robots* 43.2 (2019), pp. 309–326. ISSN: 1573-7527. DOI: 10.1007/s10514-018-9771-0.

[36] Erik Jenner and Adam Gleave. “Preprocessing reward functions for interpretability”. In: *arXiv preprint arXiv:2203.13553* (2022).

[37] Erik Jenner, Joar Max Viktor Skalse, and Adam Gleave. “A general framework for reward function distances”. In: *NeurIPS ML Safety Workshop*. 2022. URL: https://openreview.net/forum?id=Hn21kZHiCK.

[38] Daniel Kahneman and Dale T Miller. “Norm theory: Comparing reality to its alternatives.” In: *Psychological review* 93.2 (1986), p. 136.

[39] Akansha Kalra and Daniel S. Brown. “Interpretable Reward Learning via Differentiable Decision Trees”. In: *NeurIPS ML Safety Workshop*. 2022. URL: https://openreview.net/forum?id=3bk40MsYjet.

[40] Daniel Kasenberg and Matthias Scheutz. “Interpretable apprenticeship learning with temporal logic specifications”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 4914–4921.

[41] Atoosa Kasirzadeh and Andrew Smart. “The Use and Misuse of Counterfactuals in Ethical Machine Learning”. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 228–236. ISBN: 9781450383097. DOI: 10.1145/3442188.3445886. URL: https://doi.org/10.1145/3442188.3445886.

[42] Mark T Keane, Eoin M Kenny, Eoin Delaney, and Barry Smyth. “If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), Survey Track* (2021), pp. 4466–4474.

[43] Eamonn Keogh and Abdullah Mueen. “Curse of Dimensionality”. In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2017, pp. 314–315. ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_192. URL: https://doi.org/10.1007/978-1-4899-7687-1_192.

[44] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. “The Dangers of Post-hoc Interpretability: Unjustified Counterfactual Explanations”. In: *Twenty-Eighth International Joint Conference on Artificial Intelligence {IJCAI-19}*. Macao, Macau SAR China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 2801–2807. DOI: 10.24963/ijcai.2019/388. URL: https://hal.sorbonne-universite.fr/hal-02275308.

[45] Michael S. Lee, Henny Admoni, and Reid Simmons. “Reasoning about Counterfactuals to Improve Human Inverse Reinforcement Learning”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 9140–9147. DOI: 10.1109/IROS47612.2022.9982062.

[46] 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* 34.03 (2020), pp. 2493–2500. DOI: 10.1609/aaai.v34i03.5631. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5631.

[47] David R Mandel, Denis J Hilton, and Patrizia Ed Catellani. *The psychology of counterfactual thinking.* Routledge, 2005.

[48] Farzan Memarian, Zhe Xu, Bo Wu, Min Wen, and Ufuk Topcu. "Active Task-Inference-Guided Deep Inverse Reinforcement Learning". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 1932–1938. DOI: 10.1109/CDC42340.2020.9304190.

[49] Eric J Michaud, Adam Gleave, and Stuart Russell. "Understanding learned reward functions". In: *Deep RL Workshop, NeurIPS 2020* (2020).

[50] Dale T Miller and Saku Gunasegaram. "Temporal order and the perceived mutability of events: Implications for blame assignment." In: *Journal of personality and social psychology* 59.6 (1990), p. 1111.

[51] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[52] Brent Mittelstadt, Chris Russell, and Sandra Wachter. "Explaining explanations in AI". In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 279–288.

[53] Thibaut Munzer, Bilal Piot, Matthieu Geist, Olivier Pietquin, and Manuel Lopes. "Inverse reinforcement learning in relational domains". In: *International joint conferences on artificial intelligence*. 2015.

[54] Andrew Y Ng and Stuart J Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. 2000, pp. 663–670.

[55] Matthew L. Olson, Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. "Counterfactual state explanations for reinforcement learning agents via generative deep learning". In: *Artificial Intelligence* 295 (2021), p. 103455. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2021.103455. URL: https://www.sciencedirect.com/science/article/pii/S0004370221000060.

[56] Neal J Roese. "The functional basis of counterfactual thinking." In: *Journal of personality and Social Psychology* 66.5 (1994), p. 805.

[57] Neal J Roese and James M Olson. *What might have been: The social psychology of counterfactual thinking*. Psychology Press, 2014.

[58] Jacob Russell and Eugene Santos. "Explaining reward functions in markov decision processes". In: *The Thirty-Second International Flairs Conference*. 2019.

[59] Lindsay Sanneman and Julie Shah. "Explaining Reward Functions to Humans for Better Human-Robot Collaboration". In: *arXiv preprint arXiv:2110.04192* (2021).

[60] Lindsay Sanneman and Julie Shah. "Transparent Value Alignment". In: *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '23. Stockholm, Sweden: Association for Computing Machinery, 2023, pp. 557–560. ISBN: 9781450399708. DOI: 10.1145/3568294.3580147. URL: https://doi.org/10.1145/3568294.3580147.

[61] Lindsay Sanneman and Julie A Shah. "An empirical study of reward explanations with human-robot interaction applications". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8956–8963.

[62] Srivatsan Srinivasan and Finale Doshi-Velez. "Interpretable batch IRL to extract clinician goals in ICU hypotension management". In: *AMIA Summits on Translational Science Proceedings* 2020 (2020), p. 636.

[63] Gregory Stein. "Generating High-Quality Explanations for Navigation in Partially-Revealed Environments". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 17493–17506. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/926ec030f29f83ce5318754fdb631a33-Paper.pdf.

[64] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. “A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence”. In: *IEEE Access* 9 (2021), pp. 11974–12001.

[65] Aaquib Tabrez, Shivendra Agrawal, and Bradley Hayes. “Explanation-Based Reward Coaching to Improve Human Performance via Reinforcement Learning”. In: *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2019, pp. 249–257. DOI: 10.1109/HRI.2019.8673104.

[66] Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca Dragan, and Daniel S Brown. “A Study of Causal Confusion in Preference-Based Reward Learning”. In: *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*.

[67] Sam Toyer, Rohin Shah, Andrew Critch, and Stuart Russell. “The magical benchmark for robust imitation”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18284–18295.

[68] Eiji Uchibe. “Model-free deep inverse reinforcement learning by logistic regression”. In: *Neural Processing Letters* 47.3 (2018), pp. 891–905.

[69] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. “Programmatically interpretable reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5045–5054.

[70] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E Hines, John P Dickerson, and Chirag Shah. “Counterfactual Explanations and Algorithmic Recourses for Machine Learning: A Review”. In: *arXiv preprint arXiv:2010.10596* (2020).

[71] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual Explanations without opening the Black Box: Automated decisions and the GDPR”. In: *Harvard Journal of Law & Technology* 31.2 (2018).

[72] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. “A survey of preference-based reinforcement learning methods”. In: *Journal of Machine Learning Research* 18.136 (2017), pp. 1–46.

[73] Blake Wulfe, Logan Michael Ellis, Jean Mercat, Rowan Thomas McAllister, and Adrien Gaidon. “Dynamics-Aware Comparison of Learned Reward Functions”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=CALFyKVs87.

[74] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. *Maximum Entropy Deep Inverse Reinforcement Learning*. 2016. arXiv: 1507.04888.

[75] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. “Maximum entropy deep inverse reinforcement learning”. In: *arXiv preprint arXiv:1507.04888* (2015).

[76] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. “Large-scale cost function learning for path planning using deep inverse reinforcement learning”. In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1073–1087.

[77] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. “Watch this: Scalable cost-function learning for path planning in urban environments”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2089–2095. DOI: 10.1109/IROS.2016.7759328.

[78] NG A. Y. “Policy invariance under reward transformations : Theory and application to reward shaping”. In: *Proc. of the Sixteenth International Conference on Machine Learning* (1999). URL: https://cir.nii.ac.jp/crid/1570009750040818432.

[79] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. “Meta-Inverse Reinforcement Learning with Probabilistic Context Variables”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/30de24287a6d8f07b37c716ad51623a7-Paper.pdf.

[80] Yu Zhang and Qiang Yang. “An overview of multi-task learning”. In: *National Science Review* 5.1 (2018), pp. 30–43.

[81] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. “Maximum Entropy Inverse Reinforcement Learning”. In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI’08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 9781577353683.