Explaining Learned Reward Functions with Counterfactual Trajectories

Jan Wehner [1] **Frans Oliehoek** [2] **Luciano Cavalcante Siebert** [2]

Abstract

Learning rewards from human behavior or feedback is a promising approach to aligning AI systems with human values but fails to consistently extract correct reward functions. Interpretability tools could enable users to understand and evaluate possible flaws in learned reward functions. We propose Counterfactual Trajectory Explanations (CTEs) to interpret reward functions in Reinforcement Learning by contrasting an original and a counterfactual trajectory and the rewards they each receive. We derive six quality criteria for CTEs and propose a novel Monte-Carlo-based algorithm for generating CTEs that optimizes these quality criteria. To evaluate how informative the generated explanations are to a proxy-human model, we train it to predict rewards from CTEs. CTEs are demonstrably informative for the proxy-human model, increasing the similarity between its predictions and the reward function on unseen trajectories. Further, it learns to accurately judge differences in rewards between trajectories and generalizes to out-of-distribution examples. Although CTEs do not lead to a perfect prediction of the reward, our method, and more generally the adaptation of XAI methods, are presented as a fruitful approach for interpreting learned reward functions and thus enabling users to evaluate them.

1. Introduction

As Reinforcement Learning (RL) models grow in their capabilities and adoption in real-world applications (Yu et al., 2021; Kiran et al., 2022; Afsar et al., 2022), we must ensure that they are safe and aligned with human values. A core difficulty of achieving trustworthy and controllable AI (Cavalcante Siebert et al., 2023; Russell, 2019) is to accurately

capture human intentions and preferences in the reward function on which the RL agent is trained since the reward function will shape the agent's objectives and behaviour. For many tasks, it is hard to manually specify a reward function that accurately represents the intentions, preferences, or values of designers, users or society at large (Pan et al., 2022; Amodei et al., 2016). Reward Learning is a set of techniques that circumvents this problem by instead learning the reward function from data. For example, Preference-based RL (Christiano et al., 2017) derives a reward function from preference judgments queried from a human and has recently been applied to control the behaviour of Large Language Models (Bai et al., 2022). Similarly, Inverse RL (Ng & Russell, 2000), which is commonly used in autonomous driving and robotics, aims to retrieve the reward function of an expert from the demonstrations they generate. Reward learning is a promising approach for aligning the reward functions of AI systems with the intentions of humans (Russell, 2019; Leike et al., 2018). It has significant advantages over behavioral cloning, which learns a policy by using supervised learning on observation-action pairs since reward functions are considered the most succinct, robust, and transferable definition of a task (Abbeel & Ng, 2004). However, these techniques suffer from a multitude of theoretical (Armstrong & Mindermann, 2018; Skalse & Abate, 2022) and practical problems (Casper et al., 2023) that make them unable to reliably learn human values which are diverse (Lera-Leri et al., 2022), dynamic (van de Poel, 2022) and context-dependent (Liscio et al., 2022).

We aim to develop interpretability tools that help humans to understand learned reward functions so that they can detect misalignments with their own values. This is in line with the "Transparent Value Alignment" framework in which Sanneman and Shah (Sanneman & Shah, 2023) suggest leveraging techniques from eXplainable AI (XAI) to provide explanations about the reward function. The process of explaining reward functions can be useful for both the *understanding* and *explaining* phases of the XAI pipeline (Dwivedi et al., 2023), by enabling both developers and users to inspect reward functions. This is a relevant task for the XAI community, as it contributes to the goal of enabling human users to understand, appropriately trust, and produce more explainable models (Dwivedi et al., 2023; Sanneman & Shah, 2023). However, there have been few attempts to interpret

[1]CISPA Helmholtz Center for Information Security [2]Department of Intelligent Systems, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands. Correspondence to: Jan Wehner <jan.wehner@cispa.de>.
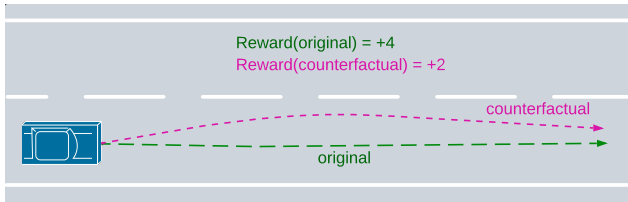
Figure 1. A car has originally taken a straight line and received a reward of $+4$ from the reward function. By providing a counterfactual that receives a lower reward of $+2$ the user can make hypotheses about how the reward function assigns rewards.

reward functions and only Michaud et al. (Michaud et al., 2020) attempt this for deep, learned reward functions. Our work makes a novel connection between XAI and reward learning by providing, to the best of our knowledge, the first principled application of counterfactual explanations to reward functions.

Counterfactual explanations are a popular XAI tool that has not yet, to the best of our knowledge, been applied to explain reward functions. It helps humans to understand the predictions of ML models by posing hypothetical "what-if" scenarios. Humans commonly use counterfactuals for decision-making, learning from past experiences, and emotional regulation(Byrne, 2016; Kahneman & Miller, 1986; Roese & Olson, 2014). Thus users can intuitively reason about and learn from counterfactual explanations, which makes this an effective and user-friendly mode of explanation (Mittelstadt et al., 2019; Wachter et al., 2018; Mandel, 2011).

We propose Counterfactual Trajectory Explanations (CTEs) that serve as informative explanations about deep reward functions. CTEs can be employed in a sequential decision-making setting by contrasting an original with a counterfactual partial trajectory along with the rewards assigned to them. This enables the user to draw inferences about what behaviours cause the reward function to assign high or low rewards. For instance, consider the domain of autonomous driving illustrated in Figure 1. While a given driving trajectory by itself might not provide much insight, adding a counterfactual trajectory along with its reward allows a user to hypothesise that the reward function negatively rewards the driving agent for swerving and getting close to the other lane.

In order to generate CTEs we identify and adapt six quality criteria for counterfactual explanations from XAI and psychology and introduce two algorithms for generating CTEs that optimise for these quality criteria. To evaluate how effective the generated CTEs are we introduce a novel measure of informativeness in which a proxy-human model learns from the provided explanations. Implementation details, ablations and further experiments can be found in the

technical appendix. [1]

2. Counterfactual Trajectory Explanations (CTEs)

This study focuses on adapting counterfactual explanations to interpret a learned reward function. Counterfactual explanations alter the inputs to a given system, which causes a change in the outputs (Wachter et al., 2018). When explaining reward functions the inputs could either be single states or (partial) trajectories. Correspondingly, the outputs to be targeted can either be seen as rewards assigned to single states or as the average reward assigned to the states in a (partial) trajectory. If we would only alter individual states, multi-step plans could be overlooked and infeasible counterfactuals that cannot occur through any sequence of actions might be created. By generating trajectories and showing their average rewards we can provide the user with insights about which multi-step behaviours are incentivized by the reward function, while also guaranteeing that counterfactuals are feasible. While it would be possible to generate multiple counterfactuals per original, we only show the user one counterfactual to be able to cover more original trajectories.

We operate in Markov Decision Processes consisting of states S, actions A, transition probabilities P and a reward function R. Further, we denote a learned reward function as $R_\theta : S \times A \Rightarrow \mathbb{R}$, a policy trained for R_θ as π_θ, full trajectories generated by a full play-through of the environment as τ and partial trajectories as $t \subseteq \tau$. Counterfactual Trajectory Explanations (CTEs) can now be defined as:

Definition 2.1. CTEs $\{(t_{org}, \overline{r}_{org}), (t_{cf}, \overline{r}_{cf})\}$ consist of an original and counterfactual partial trajectory and their average rewards assigned by a reward function R_θ. Both start in the state s_n but then follow a different sequence of actions resulting in different average rewards.

The difference in rewards can be causally explained by the difference in actions. If the agent had chosen actions $(a_{cf_n}, ..., a_{cf_k})$ instead of $(a_{org_n}, ..., a_{org_m})$ resulting in t_{cf} instead of t_{org} the reward function R_θ would have assigned an average reward $\overline{r}_{cf}$ instead of $\overline{r}_{org}$.[2]

We propose a method to address the following problem: 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 task is to select a part of that trajectory $t_{org} \subseteq \tau_{org}$ and generate a counterfactual t_{cf} to it that starts in the same state s_n so that the resulting CTE is informative for an explainee

---

to understand R_θ.

3. Method

This Section presents the method used to generate CTEs. First, quality criteria that measure the quality of an explanation are derived from the literature and combined into a scalar quality value. Then two algorithms are introduced which generate CTEs by optimising for the quality value.

3.1. Determining the quality of CTEs

Counterfactual explanations are usually generated by optimising them for a loss function that determines how good a counterfactual is (Artelt & Hammer, 2019). This loss function combines multiple aspects, which we call "quality criteria".

3.1.1. QUALITY CRITERIA

By reviewing XAI literature we were able to identify 9 quality criteria that are used for counterfactual explanations. These criteria are designed to make counterfactuals more informative to a human. Out of these Causality, Resource and Actionability (Keane et al., 2021; Verma et al., 2018; Gajcin & Dusparic, 2022) are automatically achieved by our methods. We are left with six quality criteria to optimise for which we adapt to judge the quality of CTEs.

1. Validity: Counterfactuals should lead to the desired difference in the output of the model (Verma et al., 2018; Gajcin & Dusparic, 2022). This difference in outputs makes it possible to causally reason about the changes in the inputs. We maximise Validity as $|R_\theta(t_{org}) - R_\theta(t_{cf})|$.

2. Proximity: The counterfactual should be similar to the original (Keane et al., 2021; Miller, 2019; Gajcin & Dusparic, 2022). Thus we minimize a measure based on the Modified Hausdorff distance (Dubuisson & Jain, 1994) that finds the closest match between the state-actions pairs in the two trajectories. The distance of state-action pairs is calculated as a weighted sum of the Manhattan distance of the player positions, whether the same action was taken and the edit distance between non-player objects in the environment.

3. Diversity: Explanations should cover the space of possible variables as well as possible (Huang et al., 2019; Frost et al., 2022). Consequently, each new CTE should establish novel information rather than repeating previously shown CTEs. Thus we maximize Diversity of a new CTE compared to previous CTEs. This is calculated as the sum of the average difference between the new length of the trajectory and previous lengths, the average difference in the new starting time in the environment and previous starting times, and the fraction of previous trajectories that are of the same counterfactual direction. Counterfactual direction can be upward

or downward comparisons (Roese, 1994) when the reward of the counterfactual is higher or lower than the original's reward.

4. State importance: Counterfactual explanations should focus on important states that have a significant impact on the trajectory outcome (Frost et al., 2022). We aim to start counterfactual trajectories in critical states, where the policy strongly favors some actions over others. We maximize the importance of a starting state which is calculated as the policies negative entropy $-\sum_{a \in A} \pi(a|s_0) \log \pi(a|s_0)$ (Frost et al., 2022; Huang et al., 2018).

5. Realisticness: The constellation of variables in a counterfactual should be likely to happen (Keane et al., 2021; Gajcin & Dusparic, 2022; Verma et al., 2018). In our setting, we want counterfactual trajectories that are likely to be generated by a policy trained on the given reward function. Such a trajectory would likely score high on the reward function. Thus we maximize: $R_\theta(t_{cf}) - R_\theta(t_{org})$.

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 (Keane et al., 2021; Verma et al., 2018; Gajcin & Dusparic, 2022; Miller, 2019). Instead of meticulously restricting the number of features that differ between states we lighten the cognitive load by incentivizing CTEs to be short by minimizing: $len(t_{org}) + len(t_{cf})$.

3.1.2. COMBINING QUALITY CRITERIA INTO A SCALAR QUALITY VALUE

After measuring the six quality criteria, we scalarise them into one *quality value* ρ to be assigned to a CTE. This is done by normalising the criteria and combining them into a weighted sum. Criteria are normalised to $[0, 1]$ by iteratively generating new CTEs with random weights and adapting the minimum and maximum value the criteria take on.

The weights ω assigned to the quality criteria correspond to their relative importance. However, this opens the question of how one should weigh the different quality criteria to generate the most informative explanations for a certain user. To find the optimal set of weights 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. The CTE's informativeness is tested and the set of weights that produces the most informative CTEs to a specific user are chosen for further use.

3.2. Generation algorithms for CTEs

In order to generate CTEs we propose two algorithms that optimise for the aforementioned quality value (see Section 3.1) along with a random baseline algorithm.

Algorithm 1 Monte Carlo Trajectory Optimization

Input: full trajectory τ_{org}, environment env, actions A

$candidates = []$ % store candidate CTEs

for s_n in τ_{org} **do**

    $Q = []$ % Q-values of trajectories

    $t_{cf} = [s_n]$

    repeat

        for i to $n_{iterations}$ **do**

            $t_{cf}^s \leftarrow$ SELECTION(t_{cf})

            $t_{cf}^e \leftarrow$ EXPANSION(t_{cf}^s)

            $\rho \leftarrow$ SIMULATION(t_{cf}^e)

            $Q \leftarrow$ BACK-PROPAGATION(Q, ρ)

        end for

        $a^* = argmax_{a \in A}(Q(t_{cf}, a))$

        $s_n \leftarrow env.\text{step}(s_n, a^*)$

        APPEND($t_{cf}, (s_n, a^*)$)

    until s_n is $terminal$

    $t_{org} =$ SUBSET($\tau_{org}, s_n, |t_{cf}|$) % Subtrajectory from

                         % s_n with same lengths as t_{cf}

    APPEND($candidates, (t_{org}, t_{cf})$)

end for

Return: $argmax_{c \in candidates} \rho(c)$

Algorithm 1 - Monte Carlo-based Trajectory Optimization (MCTO):

MCTO adapts Monte Carlo Tree Search (MCTS) to the task of generating CTEs. MCTS is a heuristic search algorithm that has been applied to RL by modelling the problem as a game tree, where states and actions are nodes and branches (Silver et al., 2016; Vodopivec et al., 2017). It uses random sampling and simulations to balance exploration and exploitation in estimating the Q-values of states and actions.

In contrast to MCTS, MCTO operates on partial trajectories instead of states, optimises for quality values instead of rewards from the environment, adds a termination action which ends the trajectory and applies domain-specific heuristics. Pseudocode 1 showcases the algorithm.

In MCTO nodes represent partial trajectories t, branches are actions a and child nodes result from parents by following the action in the connecting branch. Leaf nodes are terminated trajectories which can occur from entering a terminal state in the environments or by selecting an additional terminal action that is always available. MCTO optimises for the quality value ρ of a CTE, which is being measured at the leaf nodes. A CTE is derived by taking the partial trajectory in the leaf node as the counterfactual t_{cf} and the subtrajectory of τ_{org} from starting state s_n with the same length as t_{cf} as the original t_{org}.

Each state $s_n \in \tau_{org}$ in the original trajectory is used as a potential starting point of the CTE by setting it as the root of the tree and running MCTO. Out of these, the CTE with

the highest quality value is chosen. For a given state we choose the next action by repeating these four steps for a set number of times ($n_{iterations}$) before choosing the action a^* with the highest Q-value:

1. SELECTION: A node in the tree, which still has unexplored branches is chosen. The choice is made according to the *Upper Confidence Bounds for Trees* algorithm based on the estimated Q-value of the branches and the number of times the nodes and branches have already been visited.

2. EXPANSION: After selecting a node, we choose a branch and create the resulting child node.

3. SIMULATION: One full playout is completed by sampling actions uniformly until the environment terminates the trajectory or the terminating action is chosen. At each step, the terminal action is chosen with a probability of $p_{MCTO}(end)$. The resulting CTE's quality value ρ is evaluated according to the quality criteria.

4. BACK-PROPAGATION: ρ is back-propagated up the tree to adjust the Q-values of previous nodes t: $Q(t) = \frac{1}{N(t)}(\rho - Q(t))$.

As an efficiency-increasing heuristic, we prune off branches of actions that have a likelihood $\pi_\theta(a|s) \leq threshold_a$ to be chosen by the policy. Furthermore, we choose not to employ a discount factor ($\gamma = 1$) when back-propagating ρ, since this would incentivize shorter CTEs while this is already done by the Sparsity criterion. Ablations showed that other heuristics such as choosing actions in the simulation based on the policy π_θ or basing the decisions for expansion on an early estimate of the ρ did not improve performance.

Algorithm 2 - Deviate and Continue (DaC):

The *Deviate and Continue* (DaC) algorithm creates a counterfactual trajectory t_{cf} by deviating from the original trajectory τ_{org} before continuing by choosing actions according to policy π_θ. Starting in a state $s_n \in \tau_{org}$, the deviation is performed by sampling an action from the policy π_θ that leads to a different state than in the original trajectory. After $n_{deviations}$ such deviations t_{cf} is continued by following π_θ. During the continuation, there is a $p_{DaC}(end)$ chance per step of ending both t_{org} and t_{cf}. This process is repeated for every state $s_n \in \tau_{org}$ and the resulting CTE with the highest quality value is chosen.

Baseline Algorithm - Random As a weak baseline, we compare our algorithms to randomly generated CTEs. A start state s_n of the counterfactual is uniformly chosen from the original trajectory τ_{org}. From there actions are uniformly sampled, while the trajectories have a $p_{Random}(end)$ chance of being ended in each timestep.
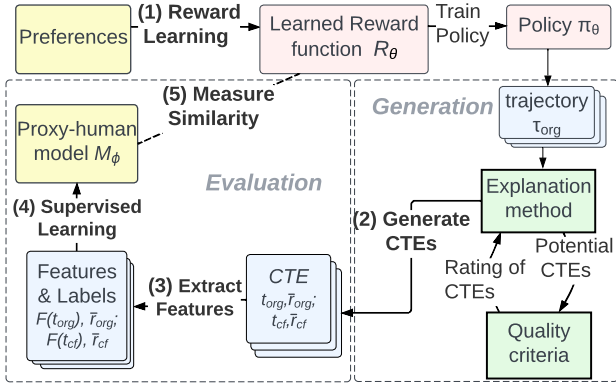
Figure 2. Schematic that describes how rewards are learned (1), explanations are generated (2) and evaluated (3,4&5).

4. Evaluation

This Section details the experimental approach we take to evaluate the informativeness of CTEs. We want to automatically measure how well an explainee can understand a reward function from explanations, while similar works perform user studies or do not offer quantitative evaluations. Since previous methods for interpreting reward functions are not applicable to our evaluation setup we can only compare our proposed methods with a baseline and criteria with each other. Our evaluation approach includes learning a reward function, generating CTEs about it and measuring how informative the CTEs are for a proxy-human model (see Figure 2).

4.1. Generating reward functions and CTEs

To learn a reward function (1) we first generate expert demonstrations. A policy π^* is trained on a ground-truth reward R^* via Proximal Policy Optimization (PPO) (Schulman et al., 2017). This policy is used to generate 1000 expert trajectories $\tau_{exp} = \{\tau_{exp_k}\}_{k=1}^{1000}$. Secondly, we use Adversarial IRL (Fu et al., 2017) which derives a robust reward function R_θ and policy π_θ from the demonstrations by posing the IRL problem as a two-player adversarial game between a reward function and a policy optimizer.

We use the Emergency environment (Peschl et al., 2022), a Gridworld environment that represents a burning building where a player needs to rescue humans and reduce the fire. The environment 7 humans that need to be rescued, a fire extinguisher which can lessen the fire and obstacles which block the agent from walking through. In each timestep, the player can walk or interact in one of the four directions. This environment is computationally cheap and simple to investigate. However, it is still interesting to study since the random initialisations require the reward function to generalise while taking into account multiple sources of reward.

To make CTEs about R_θ (2) we first generate a set of full trajectories $\tau_{org} = \{\tau_{org_k}\}_{k=1}^{1000}$ using the policy π_θ. Lastly, we use the algorithms described in Section 3.2 to optimise for the quality criteria in Section 3.1 to produce one CTE per full trajectory $CTEs = \{t_{org_k}, t_{cf_k}\}_k^{1000}$. We conducted a grid search of hyperparameters for each of the generation algorithms. Based on that we choose $p_{MCTO}(end) = 0.35$, $threshold_a = 0.003$ and $n_{iterations} = 10$ for MCTO, $p_{DaC}(end) = 0.55$ and $n_{deviaitons} = 3$ for DaC and $p_{Random}(end) = 0.15$ for Random.

4.2. Evaluating the informativeness of CTEs

We argue that informative explanations allow the explainee to better understand the learned reward function, which we formalize as the explainee's ability to assign similar average rewards to unseen trajectories as the reward function.

To evaluate the informativeness of CTEs, we employ a Neural Network (NN) as a proxy-human model to learn from the explanations and to predict the average reward assigned by R_θ for a trajectory. While humans learn differently from data than an NN, this evaluation setup still gives us important insights into the functioning and effectiveness of CTEs.

Notably, this measure only serves to evaluate the generation method and would not be used when showing CTEs to humans. It allows us to test whether extracting generalisable knowledge about the reward function from the provided CTE is possible by measuring how well the proxy-human model can predict unseen CTEs. Furthermore, it allows us to compare different algorithms and quality criteria by measuring and contrasting the informativeness of CTEs they generate.

The evaluation procedure consists of three steps, as presented in Figure 2: (3) features and labels are extracted from the CTEs to form a dataset to train on, (4) a proxy-human model is trained to predict the rewards of trajectories from these features, and, lastly, (5) 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.

Extracting features and labels (3)
We extract 46 handcrafted features $F(t) = \{f_0, ..., f_{45}\}$ about the partial trajectories. These features represent concepts that the reward function might consider in its decision-making, for example of the form "time spent using item X" or "average distance from object Y". We opted against methods for automatic feature (**?**) extraction to avoid introducing more moving parts in the evaluation. The average reward for the states in a partial trajectory serves as the label for the proxy-human model $\bar{r} = \frac{1}{|t|}\Sigma_{s \in t} R_\theta(s)$. By averaging the reward we avoid biasing the learning to the length of partial

trajectories.

Learning a proxy-human model (4)

A proxy-human regression model M_ϕ is trained to predict the average reward $\bar{r}$ given to the partial trajectory t by R_θ from the extracted features $F(t)$. Humans learn from counterfactual explanations in a contrastive manner by looking at the difference in outputs to causally reason about the effect of the inputs (Miller, 2019) but also learn from the individual data points. Since we aim to make M_ϕ learn in a similar way to a human we train M_ϕ on two tasks. In the *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 R_θ are for trajectories. The loss on one CTE for this task is the sum: $L_{single}(t_{org}, t_{cf}) = (M_\phi(t_{org}) - R_\theta(t_{org}))^2 + (M_\phi(t_{cf}) - R_\theta(t_{cf}))^2$.

In the *contrastive task*, M_ϕ is trained to predict the difference between the average original and counterfactual reward. By doing this we train M_ϕ to reason about how the difference in inputs causes the outputs instead of only learning from data points independently: $L_{contrastive}(t_{org}, t_{cf}) = [(M_\phi(t_{org}) - (M_\phi(t_{cf})) - (R_\theta(t_{org}) - R_\theta(t_{cf})]^2$.

M_ϕ is defined as a 4-layer NN 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 single and contrastive tasks. The body of the NN is shared between both tasks and feeds into two separate last layers that perform the two tasks separately. The losses of both tasks are used separately to update their respective last layer and are added into a weighted sum to update the shared body of the network.

We train the NN on 800 samples with the Adam optimiser and weight decay and results are averaged over 30 random initialisations. We perform hyperparameter tuning using 5-fold cross-validation for the learning rate, regularisation values, number of training epochs and dimensionality of hidden layers.

Measuring similarity to the reward function (5)

To measure how similar the proxy-human model's predictions are to the reward function's outputs we measure the Pearson Correlation between them on unseen CTEs. Reward functions are invariant under multiplication of positive numbers and addition (NG, 1999). This is well captured by the Pearson Correlation because it is insensitive to constant additions or multiplications. To ensure a fair comparison between different settings we test how well a model trained on CTEs from one setting generalises to a combined test set that contains CTEs from all settings.
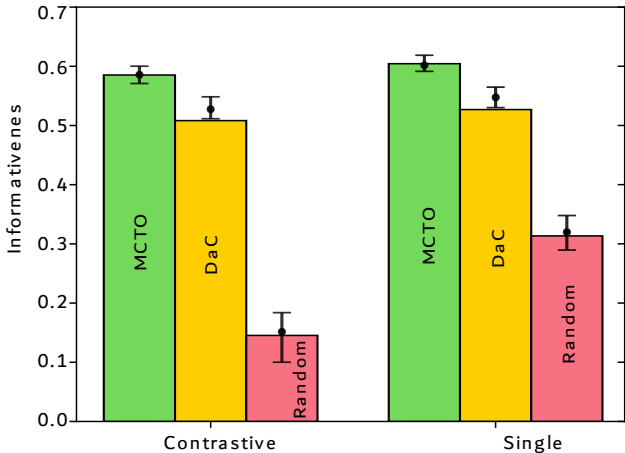

Figure 3. The average informativeness of CTEs generated by MCTO, DaC and Random for a NN trained for *single* and *contrastive* predictions, along with median, upper and lower quartile.

5. Experiments

This Section describes the results of three experiments that test the overall informativeness of CTEs, compare the generation algorithms and evaluate the quality criteria.

5.1. Experiment 1: Informativeness of Explanations for proxy-human model

Experimental Setup: We want to determine the success of our methods in generating informative explanations for a proxy-human model M_ϕ, while also comparing the generation algorithms on the downstream task. As described in Section 4.2 each generation algorithm produced 800 CTEs on which we trained 10 M_ϕs each, before testing the Pearson Correlation between their predictions and the average rewards on a combined test set of 600 CTEs. We use the weights from Table 2 for the quality criteria.

Results: Figure 3 shows that M_ϕs trained on CTEs from MCTO achieved on average higher correlation values. M_ϕs trained on DaC's CTEs were significantly ($p < 0.001$) worse, while the models trained on randomly generated CTEs achieved a much lower correlation on both tasks.

5.2. Experiment 2: Quality of Generation Algorithms

Experimental Setup: This experiment tests how good the generation algorithms are at optimising for the quality value. Each generation algorithm produced 1000 CTEs and their quality value ρ was measured. To make this test independent of the weights for quality criteria, each CTE is optimised for a different uniformly sampled set of weights: $\omega = \{\omega_{Validity_j}, ..., \omega_{sparsity_j}\}_{j=1}^{1000}$, where $\omega_i \sim U(0, 1)$. Furthermore, the efficiency of algorithms (seconds/generated CTE) and the length and starting time

	MCTO	DaC	Random
Avg quality value $\rho \uparrow$	**1.44**	1.32	1.1
Std quality value ρ	0.47	0.49	0.37
Efficiency (s/CTE) $\downarrow$ [3]	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. Shows the average quality value ρ and its variance 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.

Validity	Proximity	Diversity	State Importance	Realisticness	Sparsity
0.982	0.98	0.576	0.528	0.303	0.851

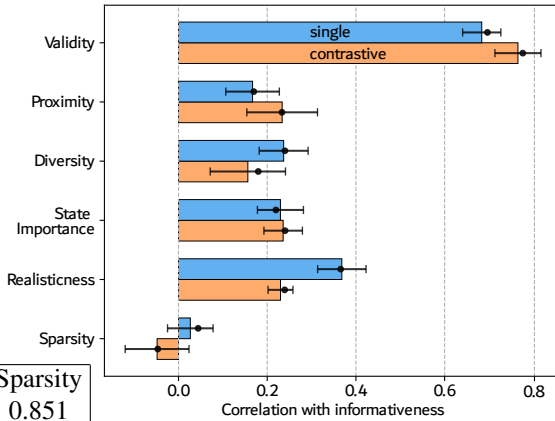*Table 2.* Most informative set of weights for MCTO and DaC.

Figure 4. Spearman correlation between weights for the quality criteria and the informativeness of the resulting CTEs for M_ϕ for the contrastive and single task. Averaged over 10 models along with the median and upper and lower quartile.

of CTEs were recorded.

Results: From Table 1 we see that MCTO achieved a higher average quality value than DaC, which again outperformed the random baseline (differences are significant with $p < 1e^{-7}$). However, the higher performance came at a computational cost, since MCTO was slower, while Random was very efficient. On average the trajectories of Random were the longest and those of MCTO the shortest. Lastly, both MCTO and DaC tended to choose starting times earlier in the environment (20.96 and 20.45 out of 75 timesteps).

5.3. Experiment 3: Informativeness of quality criteria

Experimental Setup: Finally, we wanted to determine the influence of a quality criterion on informativeness. For this, we analyzed the Spearman correlation between the weight assigned to the criterion during the generation of a set of CTEs and the informativeness of this set of CTEs. Simultaneously we carried out the calibration phase to determine the set of weights which leads to the most informative CTEs for an explainee and generation algorithm.

Thirty sets of weights ω were each used to generate one set of 1000 CTEs with MCTO. 800 CTEs were used to train 10 M_ϕs as described in Section 4.2. The performances of the resulting 30 sets of M_ϕs were evaluated on a test set that combines the remaining 200 samples from each of the 30 sets of CTEs. This indicates the informativeness of the CTEs they were trained on. By measuring the Spearman correlation between the weights assigned to a criterion and the informativeness of the resulting CTEs for M_ϕ, we can infer the importance of that criterion for making CTEs informative. Furthermore, we record the set of weights which leads to the most informative CTEs for each generation algorithm except Random which is independent of weights.

Results: Figure 4 shows that for both contrastive and

single learning, the weights of Validity ($\omega_{Validity}$) correlated the strongest with the informativeness for M_ϕ. This is followed by $\omega_{Realisticness}$, $\omega_{Proximity}$, $\omega_{Diversity}$ and $\omega_{StateImportance}$ which all show a moderate correlation with the informativeness, while $\omega_{Sparsity}$ was barely or even negatively correlated with the informativeness. While there are differences between the importance of criteria for the two tasks, they end up with similar results.

Furthermore, we find that the same set of weights leads to the most informative CTEs for both MCTO and DaC. It assigns very high weights to Validity and Proximity, while Realisticness is weighted low. Contrary to Figure 4 Sparsity is highly weighted.

5.4. Discussion

CTEs are informative for the proxy-human model. Experiment 1 shows that an NN-based model trained on CTEs is much better than random guessing at predicting rewards or judging the difference in rewards between unseen CTEs. It also shows a capability to generalise to out-of-distribution examples when predicting CTEs generated by other algorithms. This indicates that CTEs enable an explainee to learn 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. This could be explained by a lack of training samples, a loss of information during the feature extraction or insufficient coverage of different situations in the environment. Furthermore, the studied reward function is noisy, often outputting different rewards for apparently similar situations and is thus hard to understand.

---

[3] Efficiency differs depending on the hardware used.

MCTO generated the most informative CTEs, while the CTEs from Random were less informative.

Similarly, we find that **MCTO is the most effective generation algorithm for optimising the quality value**, while DaC outperforms Random. The fact that the algorithms which achieved higher quality values in Experiment 2 also produced more informative CTEs in Experiment 1 indicates that optimising well for the quality value is generally useful for making more informative CTEs. Table 1 shows a trade-off, between the performance and efficiency of the generation algorithms, which likely appears because a more exhaustive search finds higher-scoring CTEs. Furthermore, MCTO and DaC selected CTEs with earlier starting times. This is because the environment had higher fluctuations in rewards early on, which benefits Validity and State importance. This shows that they are able to select CTEs in more interesting parts of the environment. They also tend to choose shorter trajectories, which score higher on Sparsity.

Among the criteria **Validity is the most important criterion for generating informative CTEs** as shown in Experiment 3. High weights for Validity lead to higher differences in rewards and lead to a larger range of labels for contrastive predictions. Possibly, an NN can learn more information from these larger differences and is thus better informed by CTEs that are high in Validity. Proximity, Realisticness, Diversity and State importance are also beneficial for having the proxy-human model learn from CTEs, but we are less certain about why they are beneficial. Although prioritising Sparsity does not correlate with informativeness, the most informative set of weights does give it a high weight. However, this high weight might be a fluke since we only tried 30 sets of weights. In any case, we should not conclude that humans would not benefit from sparse explanations. While NNs can easily compute gradients over many different features simultaneously, humans can only draw inferences about a few features at once (Miller, 1956). This clarifies that the prioritisation of quality criteria will likely differ for a human.

The fact that the two tasks largely agreed on the importance of quality criteria indicates that they complement each other. This might be because the two tasks are similar and thus benefit from developing similar representations in the shared body of the network. Furthermore, because the same set of weights out of 30 options led to the most informative CTEs when using MCTO and DaC we can speculate that the relative importance of quality criteria for an explained is similar, independent of the generation algorithm used.

Limitations: Since we do not measure the informativeness of CTEs for a human user, our experiments do not prove that CTEs are informative for humans or show how important the criteria would be to a user. Furthermore, we only conduct experiments on a single learned reward function in

a single environment, making it unclear how our findings will generalise to other settings. The method might especially struggle with large and complex environments where it is difficult to achieve high coverage of the environment with CTEs. Further, depends on the ability to reset the environment to previous states, which is not given in some environments. Lastly, our evaluation measure depends on hand-crafted features which limits its applicability.

6. Related Work

This Section covers previous work on the interpretability of reward functions and counterfactual explanations for AI.

6.1. Interpretability of Learned Reward Functions

Reward functions can be made intrinsically more interpretable by learning them as decision trees (Bewley & Lecue, 2022; Kalra & Brown, 2022; Srinivasan & Doshi-Velez, 2020) or in logical domains (Kasenberg & Scheutz, 2017; Munzer et al., 2015). Attempts have been made to make deep reward functions more interpretable by simplifying them through equivalence transformation (Jenner & Gleave, 2022) or by imitating a Neural Network with a decision tree (Russell & Santos, 2019). However, such interpretable representations can negatively impact the performance of the method.

To avoid this drawback, we interpret learned reward functions via post-hoc explanations. Post-hoc methods are applied after the model has been trained to explain the model's decision-making process. Lindsey and Shah (Sanneman & Shah, 2021; 2022) test the effectiveness and required cognitive workload of simple explanation techniques about linear reward functions. While their work requires linear reward functions our method is applicable to any representation of a reward function.

The closest work to ours comes from Michaud et al. (Michaud et al., 2020) who apply gradient salience and occlusion maps to identify flaws in a learned reward function and employ handcrafted counterfactual inputs to validate their findings. Our work focuses on counterfactuals and automatically generates them to be of high quality.

6.2. Counterfactual Explanations

Despite a large body of work on generating counterfactual explanations about ML models in supervised learning problems (Verma et al., 2020; Artelt & Hammer, 2019; Guidotti, 2022; Stepin et al., 2021) and their relation to human psychology (Keane et al., 2021; Byrne, 2019), this approach has only recently been adapted to explain RL policies. Counterfactuals consist of a change in certain input variables which cause a change in outputs (Wachter et al., 2018). In the RL setting, counterfactual explanations can be changes in

Features, Goals, Objectives, Events, or Expectations that cause the agent to change its pursued Actions, Plans, or Policies (Gajcin & Dusparic, 2022). This can improve users' understanding of out-of-distribution behaviour (Frost et al., 2022), provide them with more informative demonstrations (Lee et al., 2022) or showcase how an agent's environmental beliefs influence its planning (Stein, 2021). Instead of explaining a policy π this paper presents the first principled attempt to use them to use counterfactuals to explain a reward function R.

7. Conclusion

While reward learning presents a promising approach for aligning AI systems with human values, there is a lack of methods to interpret the resulting reward functions. To address this we formulate the notion of Counterfactual Trajectory Explanations (CTEs) and propose algorithms to generate them. Our results show that CTEs are informative for an explainee, but do not lead to a perfect understanding of the reward function. Further, they validate our MCTO algorithm to be effective at generating CTEs and imply that the difference in outcomes between an original and counterfactual trajectory is especially important to achieve informative explanations. This research demonstrates that it is fruitful to apply techniques from XAI to interpret learned reward functions.

Future work should carry out a user study to test the informativeness of CTEs for humans. Furthermore, the method should be evaluated in more complex environments and on a range of reward functions produced by different reward learning algorithms. Ultimately, we hope that CTEs will be used in practice to allow users to understand the misalignments between their values and a reward function, thus enabling them to improve the reward function with new demonstrations or feedback.

8. Acknowledgements

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Afsar, M. M., Crump, T., and Far, B. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7), 2022.

Amir, D. and Amir, O. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 1168–1176, Richland, SC, 2018.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Armstrong, S. and Mindermann, S. Occam's razor is insufficient to infer the preferences of irrational agents. *Advances in neural information processing systems*, 31, 2018.

Artelt, A. and Hammer, B. On the computation of counterfactual explanations – a survey. *arXiv preprint arXiv:1911.07749*, 2019.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

Bewley, T. and Lecue, F. Interpretable preference-based reinforcement learning with tree-structured reward functions. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 118–126, 2022.

Byrne, R. M. Counterfactual thought. *Annual review of psychology*, 67:135–157, 2016.

Byrne, R. M. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 6276–6282, 2019.

Casper, S., Davies, X., Shi, C., Gilbert, T. K., Scheurer, J., et al. Open problems and fundamental limitations of reinforcement learning from human feedback, 2023.

Cavalcante Siebert, L., Lupetti, M. L., Aizenberg, E., Beckers, N., Zgonnikov, A., et al. Meaningful human control: actionable properties for ai system development. *AI and Ethics*, 3(1):241–255, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Dubuisson, M.-P. and Jain, A. A modified hausdorff distance for object matching. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pp. 566–568 vol.1, 1994.

Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., et al. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.

Frost, J., Watkins, O., Weiner, E., Abbeel, P., Darrell, T., Plummer, B., and Saenko, K. Explaining reinforcement learning policies through counterfactual trajectories. *arXiv preprint arXiv:2201.12462*, 2022.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Gajcin, J. and Dusparic, I. Counterfactual explanations for reinforcement learning. *arXiv preprint arXiv:2210.11846*, 2022.

Guidotti, R. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pp. 1–55, 2022.

Huang, S. H., Bhatia, K., Abbeel, P., and Dragan, A. D. Establishing appropriate trust via critical states. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3929–3936, 2018.

Huang, S. H., Held, D., Abbeel, P., and Dragan, A. D. Enabling robots to communicate their objectives. *Autonomous Robots*, 43(2):309–326, 2019.

Jenner, E. and Gleave, A. Preprocessing reward functions for interpretability. *arXiv preprint arXiv:2203.13553*, 2022.

Kahneman, D. and Miller, D. T. Norm theory: Comparing reality to its alternatives. *Psychological review*, 93(2): 136, 1986.

Kalra, A. and Brown, D. S. Interpretable reward learning via differentiable decision trees. In *NeurIPS ML Safety Workshop*, 2022.

Kasenberg, D. and Scheutz, M. Interpretable apprenticeship learning with temporal logic specifications. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 4914–4921. IEEE, 2017.

Keane, M. T., Kenny, E. M., Delaney, E., and Smyth, B. If only we had better counterfactual explanations: Five key deficits to rectify in the evaluation of counterfactual xai techniques. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), Survey Track*, pp. 4466–4474, 2021.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., et al. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2022.

Lee, M. S., Admoni, H., and Simmons, R. Reasoning about counterfactuals to improve human inverse reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9140–9147, 2022.

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Lera-Leri, R., Bistaffa, F., Serramia, M., Lopez-Sanchez, M., and Rodriguez-Aguilar, J. Towards pluralistic value alignment: Aggregating value systems through ℓp-regression. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 780–788, 2022.

Liscio, E., van der Meer, M., Siebert, L. C., Jonker, C. M., and Murukannaiah, P. K. What values should an agent align with? an empirical comparison of general and context-specific values. *Autonomous Agents and Multi-Agent Systems*, 36(1):23, 2022.

Mandel, D. R. Of causal and counterfactual explanation. In *Understanding counterfactuals, understanding causation: Issues in philosophy and psychology*, pp. 147. Oxford University Press, 2011.

Michaud, E. J., Gleave, A., and Russell, S. Understanding learned reward functions. *Deep RL Workshop, NeurIPS 2020*, 2020.

Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

Mittelstadt, B., Russell, C., and Wachter, S. Explaining explanations in ai. In *Proceedings of the conference on fairness, accountability, and transparency*, pp. 279–288, 2019.

Munzer, T., Piot, B., Geist, M., Pietquin, O., and Lopes, M. Inverse reinforcement learning in relational domains. In *International joint conferences on artificial intelligence*, 2015.

NG, A. Y. Policy invariance under reward transformations : Theory and application to reward shaping. *Proc. of the Sixteenth International Conference on Machine Learning*, 1999. URL https://cir.nii.ac.jp/crid/1570009750040818432.

Ng, A. Y. and Russell, S. J. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 663–670, 2000.

Pan, A., Bhatia, K., and Steinhardt, J. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022.

Peschl, M., Zgonnikov, A., Oliehoek, F. A., and Siebert, L. C. 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, pp. 1038–1046, Richland, SC, 2022.

Roese, N. J. The functional basis of counterfactual thinking. *Journal of personality and Social Psychology*, 66(5):805, 1994.

Roese, N. J. and Olson, J. M. *What might have been: The social psychology of counterfactual thinking*. Psychology Press, 2014.

Russell, J. and Santos, E. Explaining reward functions in markov decision processes. In *The Thirty-Second International Flairs Conference*, 2019.

Russell, S. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

Sanneman, L. and Shah, J. Explaining reward functions to humans for better human-robot collaboration. *arXiv preprint arXiv:2110.04192*, 2021.

Sanneman, L. and Shah, J. Transparent value alignment. In *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '23, pp. 557–560, New York, NY, USA, 2023.

Sanneman, L. and Shah, J. A. An empirical study of reward explanations with human-robot interaction applications. *IEEE Robotics and Automation Letters*, 7(4):8956–8963, 2022.

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

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Skalse, J. M. V. and Abate, A. Misspecification in inverse reinforcement learning. In *NeurIPS ML Safety Workshop*, 2022.

Srinivasan, S. and Doshi-Velez, F. Interpretable batch irl to extract clinician goals in icu hypotension management. *AMIA Summits on Translational Science Proceedings*, 2020:636, 2020.

Stein, G. Generating high-quality explanations for navigation in partially-revealed environments. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17493–17506, 2021.

Stepin, I., Alonso, J. M., Catala, A., and Pereira-Fariña, M. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001, 2021.

van de Poel, I. Understanding value change. *Prometheus*, 38(1):7–24, 2022. URL https://www.jstor.org/stable/48676463.

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pp. 5045–5054. PMLR, 2018.

Verma, S., Boonsanong, V., Hoang, M., Hines, K. E., Dickerson, J. P., and Shah, C. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

Vodopivec, T., Samothrakis, S., and Ster, B. On monte carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, 60:881–936, 2017.

Wachter, S., Mittelstadt, B., and Russell, C. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.

Yu, C., Liu, J., Nemati, S., and Yin, G. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys*, 55(1), 2021.

A. Environment

For our experiments, we employ the *Emergency* environment developed by Peschl et al. (2022) which is a Gridworld environment that represents a burning building where a player needs to rescue humans and reduce the fire. The environment contains a player, 7 humans that need to be rescued, a fire extinguisher which can lessen the fire and obstacles which block the agent from walking through (see Figure 5). Humans can be rescued by standing next to them and interacting with their cells. The fire extinguisher is placed in the bottom right and used 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. In each run, the starting position of the player, humans and obstacles are randomly initialised and the environment is run for 75 steps. A ground truth reward function R^* assigns a reward of $+10$ per human saved and $+1$ per step the fire extinguisher is used.

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.

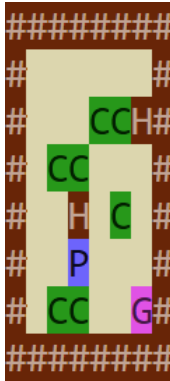The environment is available at: https://github.com/mlpeschl/moral_rl

Figure 5. A random initialisation of the Emergency environment by Peschl et al. (2022). Shows the player *P* in blue, the humans *C* in green, the obstacles *H* in brown, the fire-extinguisher *G* in pink and the borders of the environment *#* in brown.

B. Additional implementation details

B.1. Implementation of quality criteria

In this appendix, we lay out the implementation of the quality criteria we use. For the motivation of the criteria see Section 3.1.1.

B.1.1. VALIDITY

We calculate the difference in rewards between t_{org} and t_{cf} as the absolute difference in average rewards: Validity as $|R_\theta(t_{org}) - R_\theta(t_{cf})|$, where $R_\theta(t) = \frac{1}{|t|}\Sigma_{s \in t}R_\theta(s)$. The average reward attained is used to not bias the Validity with regards to the differences in length between trajectories.

B.1.2. PROXIMITY

We interpret Proximity as the similarity between trajectories, which is measured using the Modified Hausdorff distance (MHD) (Dubuisson & Jain, 1994). 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. For two trajectories A and B, 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:

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

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 $dist_{player}(a, b) = |x_a - x_b| + |y_a - y_b|$, where x and y denote the players x- and y-coordinates in the environment in that state.

- 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 and b_{humans} denotes the x- and y-coordinates of all humans in the environment.

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 trying out different weights and subjectively judging how the weighted sum matched with the researcher's judgment about the similarity between trajectories. Furthermore, the distance metric 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.

B.1.3. DIVERSITY

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 previous CTEs that have already been shown to the user. So for the first CTE Diversity $= 0$ and for the 5th CTE, it is measured with regards to the 4 previously generated CTEs.

To measure how similar the length of the counterfactual is to the lengths of previous counterfactuals we take the three previous CTEs which are closest in length (smallest difference) and calculate their average difference in length to the new CTE. In the same way, we compare the starting time of a CTE to the starting times of previous CTEs by comparing the 3 closest previous CTEs and calculating their average difference in starting times.

Further, 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 (Roese, 1994). To add more diversity 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 summed together to form one Diversity criterion.

B.1.4. STATE IMPORTANCE

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 (Amir & Amir, 2018) 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. (Huang et al., 2018) 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 (Frost et al., 2022) and Huang et al. (Huang et al., 2018) use the negative entropy in the probability distribution over actions in that state State Importance$(s) = -\sum_{a \in A} \pi(a|s) \log \pi(a|s)$. 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.

B.1.5. REALISTICNESS

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: Realisticness$= \overline{R_\theta(t_{cf})} - \overline{R_\theta(t_{org})}$.

B.1.6. SPARSITY:

We interpret this as meaning that the counterfactual and original trajectories should be shorter so that humans do not have to compare many behaviours and states. Thus we sum up the length of the original and counterfactual: $\mathsf{Sparsity}(t_{org}, t_{cf}) = len(t_{org}) + len(t_{cf})$.

B.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 normalise each criterion to a range of $[0, 1]$. This requires finding a maximum ($norm_{max}^c$) and minimum ($norm_{min}^c$) possible value that each criterion c can take on. We do this approximately in an adaptive fashion, by

1. Starting with $norm_{max}^c = 1$ and $norm_{min}^c = 0$.

2. Using these normalisation values DaC and MCTO generate 20 CTEs for a random set of weights.

3. Finding the minimum $norm_{min}^{c\prime} = min$ and maximum $norm_{max}^{c\prime} = 1$ values for c in the CTEs.

4. Updating the normalisation parameters if more extreme values were found
   $norm_{max}^c = \max(norm_{max}^c, norm_{max}^{c\prime})$ and $norm_{min}^c = \max(norm_{min}^c, norm_{min}^{c\prime})$.

5. If the normalisation values were adapted for any c: Repeat from Step 2.

When performing this procedure for our experiments, we also manually altered hyperparameters and settings of the generation algorithms between repeats to account for the fact that some settings can lead to more extreme values.

B.3. Pseudocode for Deviate and Continue

Pseudocode 2 shows how DaC is implemented.

B.4. Implementation of Extracted Features and Labels

After the CTEs were generated we extracted 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 $\bar{r}$. Here $t = \{(s_n, a_n), ..., (s_m, a_n)\}$ denotes a partial trajectory with state action pairs with length $|t| = m - n$. Furthermore, in a state s we denote $humans(s)$ as the number of humans that are still in danger, $H(s)$ as a list of the x-y-coordinates of all unsaved humans and $p_x(s)$ and $p_y(s)$ as the x- and y-coordinate of the player.

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? $humans(s_m) - humans(s_n)$.

2. Unsaved humans: On average over the steps in the trajectory how many humans were not saved yet? $\frac{1}{|t|}\Sigma_{i=n}^m humans(s_i)$.

3. Final number of unsaved humans: At the end of the partial trajectory, how many humans are not saved yet? $humans(s_m)$.

4. Number of humans: In how many of the steps in the trajectory were there $n \in \{0, ..., 7\}$ humans still in danger? For each n: $\Sigma_{i=n}^m \begin{cases} 1 & \text{if } humans(s_i) = n \\ 0 & \text{otherwise} \end{cases}$.

14

Algorithm 2 Deciate and Continue

Input: full trajectory τ_{org}, environment env, actions A, policy π_θ, $n_{deviations}$

$candidates = []$ % store candidate CTEs

for (s^n_{org}, a^n_{org}) in τ_{org} **do**

    $s_{cf} = s^n_{org}$

    $t_{cf} = [s_{cf}]$

    for i in $[0, n_{deviations})$ **do**

         % DEVIATE from the original

        repeat

            $a_{cf} = \pi_\theta(a \in A | s_{cf})$ % Resample a_{cf} until it leads to a different state

            $s_{cf} = env.\text{step}(s_{cf}, a_{cf})$

        until $s_{cf} \neq s^{n+i+1}_{org}$

        APPEND$(t_{cf}, (s_{cf}, a_{cf}))$

    end for

    $a_{cf} = \{a_{end}$ with $p_{DaC}(end) \mid \pi_\theta(a|s)$ otherwise $\}$ % CONTINUE using the policy π_θ

    while $a_{cf} \neq a_{end}$ **do**

        $s_{cf} = env.\text{step}(s_{cf}, a_{cf})$

        APPEND$(t_{cf}, (s_{cf}, a_{cf}))$

        $a_{cf} = \{a_{end}$ with $p_{DaC}(end) \mid \pi_\theta(a|s)$ otherwise $\}$ % End trajectory with likelihood $p_{DaC}(end)$

    end while

    $t_{org} = \text{SUBSET}(\tau_{org}, s^n_{org}, |t_{cf}|)$ % Subtrajectory from s^n_{org} with same lengths as t_{cf}

    APPEND$(candidates, (t_{org}, t_{cf}))$

end for

Return: $argmax_{c \in candidates} \rho(c)$ % Return the highest rated candidate

---

5. Average Distance between humans: How far are the humans apart in Euclidean distance averaged over all pairs of humans? $\frac{1}{humans(s)^2} \Sigma_{(h_x, h_y) \in H(s)} \Sigma_{(h'_x, h'_y) \in H(s)} |h_x - h'_x| + |h_y - h'_y|$.

Features about the fire extinguisher, which is located at the x-y-coordinates $(6, 6)$:

6. Standing on fire extinguisher: For how many steps did the agent use the fire extinguisher? $\Sigma^m_{i=n} \begin{cases} 1 & \text{if } p_x(s_i) = 6 \wedge p_y(s_i) = 6 \\ 0 & \text{otherwise} \end{cases}$

7. Distance to fire-extinguisher: On average what is the Euclidean distance between the agent and fire-extinguisher? $\frac{1}{|t|} \Sigma^m_{i=n} |p_x(s_i) - 6| + |p_y(s_i) - 6|$

Features about the actions of the agent:

8. Could have saved: Sums over the trajectory in how many states the agent stood next to a human and could have saved them, but failed to do so. This means in the state s_i the Euclidian distance of the player to a human was exactly 1 and in the next state s_{i+1} the amount of unsaved humans is still the same as in s_i.

9. Moved towards the closest human: How often did the agent move towards the closest human or save a human? This is true if the Euclidian distance to the closest human s_i is larger than in the following state s_{i+1}. There is a special case when the player is standing on the human since it can only interact with the human while standing *next* to them. Since it would take 2 actions to save this human (step aside and interact with them) we calculate this as a distance of 2 to this human. Furthermore, it is true if the number of humans in s_i is larger than in s_{i+1}.

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, where:

11. Average Distance to humans: On average how close was the agent to the closest human? $\frac{1}{|t|}\Sigma_{i=n}^{m} min_{(h_x,h_y)\in H(s_i)}|p_x(s_i) - h_x| + |p_y(s_i) - h_y|$.

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 human: How often was the closest human right, left, down or up from the agent? For each of the directions, it counts in how many states the closest human was in that direction. If the closest human is to the top-left it counts towards both up and left.

Features about the position of the agent?

14. Position of the agent: Average x and y position of the agent. $\frac{1}{|t|}\Sigma_{i=n}^{m} p_x(s_i)$ (equivalent for y).

15. Steps next to the wall: For how many steps was the agent next to the wall? Counts for how many of the states the agent was positioned next to the wall as the number of states $s \in t$ which fulfil: $p_x(s) = 1 \lor p_x(s) = 6 \lor p_y(s) = 1 \lor p_y(s) = 6$.

16. Steps in the middle: Counts for how many states $s \in t$ the agent stood in the middle, thus fulfilling: $(p_x(s), p_y(s)) \in \{(3,3), (3,4), (4,3), (4,4)\}$.

17. Steps between middle and wall: For how many states $s \in t$ was the agent not in the middle or next to the wall?

18. Time spent in quadrant: For how many steps was the agent in the top-left ($p_x \leq 3 \land p_y \leq 3$), top-right ($p_x > 3 \land p_y \leq 3$), bottom-left ($p_x \leq 3 \land p_y > 3$) or bottom-right quadrant ($p_x > 3 \land p_y > 3$)?

Other:

19. Length: Number of steps in the partial trajectory $|t|$.

B.5. Hyperparameters for AIRL & PPO

Table 3 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 4 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 3. 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 4. Hyperparameters for Adversarial Inverse Reinforcement Learning

C. Influence and Trade-offs in quality criteria

This section explores what trade-offs and synergies exist between the quality criteria, how these influence the choice of CTEs and how the different generation algorithms perform on the quality criteria.
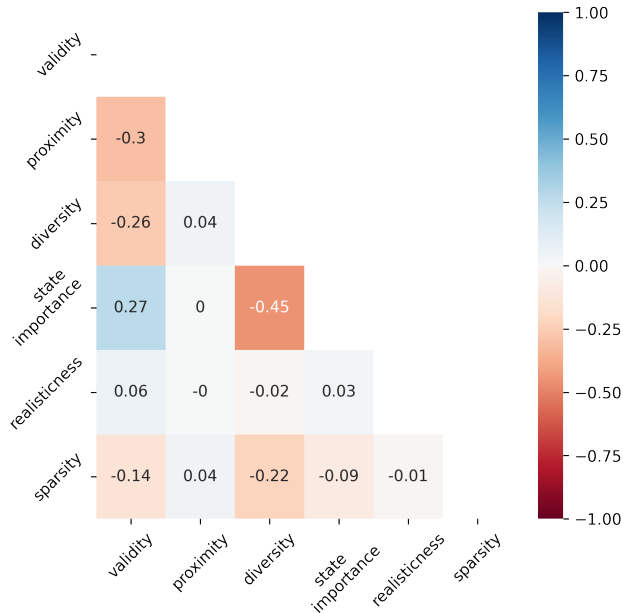
Figure 6. The average Pearson correlations between quality criteria of candidate CTEs in DaC.

C.1. Trade-offs and influence of quality criteria

Maximising the quality value ρ, 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 (see Appendix B.2). Thus we want to analyse the trade-offs and synergies between criteria. To study this we consider the correlations between quality criteria of the candidate CTEs, not only over the CTEs that are finally chosen.

Experimental Setup: DaC is employed to generate 1 CTE for each of the 1000 sets of weights ω_{random}. For each run of DaC we measure the quality criteria and calculate their Pearson correlation between each quality criterion. We then average the resulting correlations over the 1000 runs.

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 prefer certain start points and lengths of trajectories, while Diversity pushes for different start points and lengths. Possibly for the same reason, 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. Further, these trade-offs pose a require the generation algorithms to find good compromises between the criteria.

C.2. Performance of Generation Algorithms on Quality Criteria

This Appendix provides the scores achieved on the individual quality criteria by each generation algorithm in Experiment 2 (see Section 5.2).

Figure 7 shows that MCTO achieved the highest average quality value, but did not perform best on every quality criterion. 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. This

indicates that MCTO was best able to navigate the trade-offs between quality criteria allowing it to score high on the quality value.

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. This is a byproduct of the normalisation procedure. 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.

Importantly, a quality criterion with higher average values does not have a larger influence on which CTE. This is investigated in Appendix C.3.
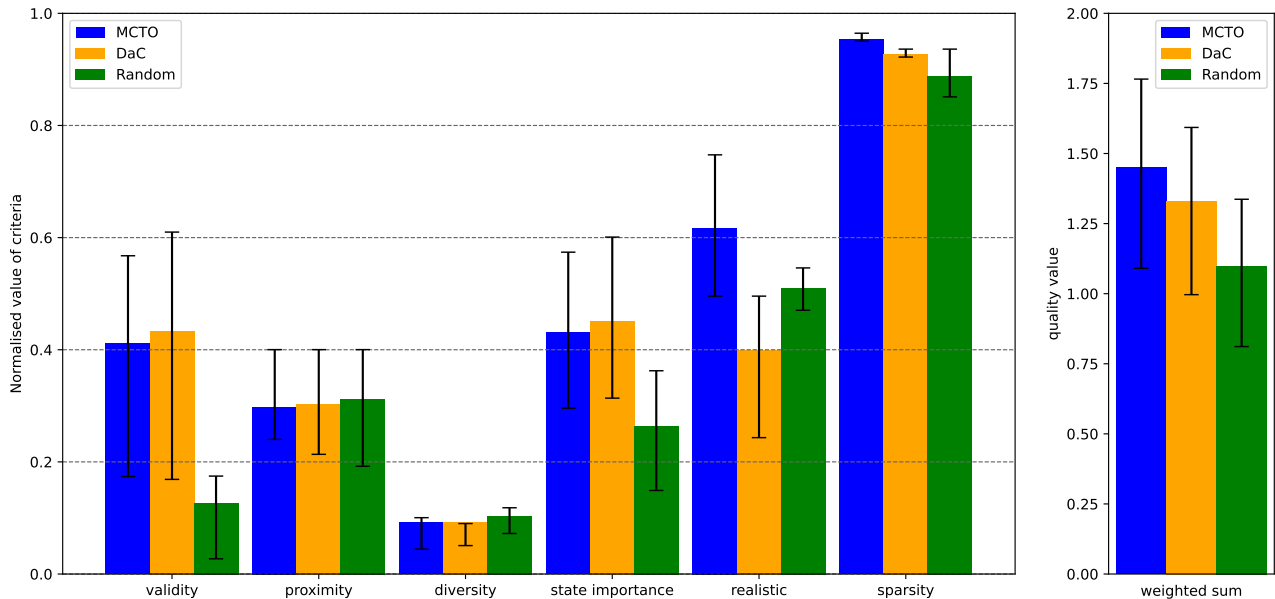

Figure 7. For each quality criterion (left figure) and the quality value ρ (right figure), the average normalised value and upper & lower quartile achieved by the different generation methods are shown.

C.3. Influence of quality criteria on the choice of CTE

The previous section C.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 more or less influence on which CTE gets chosen.

Experimental Setup: Similarly to the Experiment in Appendix C.1, DaC is employed to generate 1 CTE for each of the 1000 random sets of weights ω_{random}. We record the quality criteria for the chosen CTE (the one that is shown to the user), but also for all candidate CTEs that were not chosen. Based on this we record for each criterion and each chosen CTE 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 correlation between the criterion and the quality values ρ (*ρ-correlation*).

Results: Looking at Table 5 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

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 5. 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.

	ω	Validity	Proximity	Diversity	State Importance	Realisticness	Sparsity	
contrastive	best ↑	**0.982**	**0.98**	0.576	0.528	0.303	**0.851**	MCTO
	worst ↓	0.126	0.213	0.492	**0.943**	0.052	**0.752**	
single	best ↑	**0.878**	**0.92**	**0.915**	0.674	0.639	0.657	
	worst ↓	0.17	0.496	0.205	**0.968**	0.203	**0.633**	
contrastive	best ↑	**0.902**	0.658	0.405	0.587	0.12	**0.988**	DaC
	worst ↓	0.006	0.267	**0.891**	**0.924**	0.353	0.579	
single	best ↑	**0.878**	**0.92**	**0.915**	0.674	0.639	0.657	
	worst ↓	0.412	**0.836**	0.38	0.087	0.042	**0.748**	

Table 6. The sets of weights ω (out of 30) that lead to the most and least informative CTEs for a Neural Network when optimised by MCTO or DaC for the contrastive and single task.

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, 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 5 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. This supports our conclusion from Appendix C.1 that there 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 of these criteria cause large differences in the weighted sum. Thus they have an outsized impact on the quality value 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, Appendix C.2 showed that Diversity has a small range of values, which might further contribute to its weak influence.

C.4. Most and Least informative set of weights

Experimental Setup: To provide further insight into which quality criteria are important to create informative CTEs, we find the set of weights that lead to the most and least informative CTEs. Additionally, the most informative weights found here for each generation algorithm are also those used to generate CTEs in Experiment 1 (see Section 5.1).

In Experiment 3 (see Section 5.3) we test the informativeness of 30 sets of CTEs generated by 30 different sets of weights for the quality criteria $\omega \in \omega = \{\omega_{Validity_j}, ..., \omega_{Sparsity_j}\}_{j=1}^{30}$. We use both MCTO and DaC to generate CTEs and train 10 M_ϕs on each set of CTEs. From that, we identify the best and worst performing set of weights ω on the single and contrastive task.

Results: Table 6 shows that the most informative CTEs generated by MCTO for an 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 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 on which set of weights leads to the most informative CTEs for the single task, they disagree on the three other measures.

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 only one out of four settings, indicates that different generation algorithms do not converge to the same priorities between criteria. However, since we only sample 30 weights these results might not be highly significant.

D. Ablations and hyperparameters of generation algorithms

D.1. Ablations and hyperparameters of MCTO

In MCTO there are multiple hyperparameters to be adjusted:

- Number of starting points n_{starts}: MCTO is started at a specific point s_n of the original trajectory. We rerun MCTO n_{starts} times for different starting states $s_n \in \tau_{org}$ and select the CTE with the highest quality value. The n_{starts} starting points are selected as the n_{start} states which are rated highest by the State Importance criterion.

- Number of iterations $n_{iterations}$: From a given state MCTO starts to explore the tree through selection, expanding, simulation and back-propagating. $n_{iterations}$ denotes the number of iterations of these four steps in a state before a decision for the next action has to be made.

- Likelihood to terminate $p_{MCTO}(end)$: During each step of the simulation the trajectory is ended with a likelihood of $p_{MCTO}(end)$.

- Action threshold $threshold_a$: As a heuristic, we do not consider every action as a possible branch in the tree, but only those actions a that are sufficiently likely according to the policy so that $\pi_\theta \geq threshold_a$.

- Discount factor γ: During backpropagation, a factor γ can be employed to discount the rewards at each node. To switch discounting off the discount factor is set to $\gamma = 1$.

Furthermore, we experiment with two heuristics:

- Expansion heuristic: When deciding which branch to extend, we can choose through uniform sampling (*random-expansion*) or according to a heuristic that takes into account an early estimate of the quality value (*heuristic-expansion*). This heuristic tries each possible action, assumes the resulting trajectory as a leaf node and calculates the quality value of the resulting CTE. Then the action which achieved the highest quality value is chosen.

- Action selection during simulation: During the simulation, we can sample actions uniformly (*random-simulation*) or according to the policy π_θ (*policy-simulation*).

D.1.1. 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. We keep the following values for all experiments and only alter the parameter in question: $threshold_a = 0.003$, $p_{MCTO}(end) = 0.2$, $\gamma = 0.85$, $n_{iterations}, = 10$, *heuristic-expansion*, *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 ($\frac{seconds}{CTE}$).
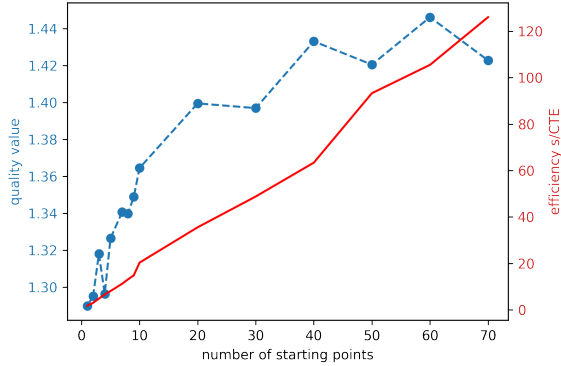
Figure 8. Average quality values of quality criteria for CTEs created by MCTO with differing number of starting points along with the efficiency in $\frac{seconds}{CTE}$.
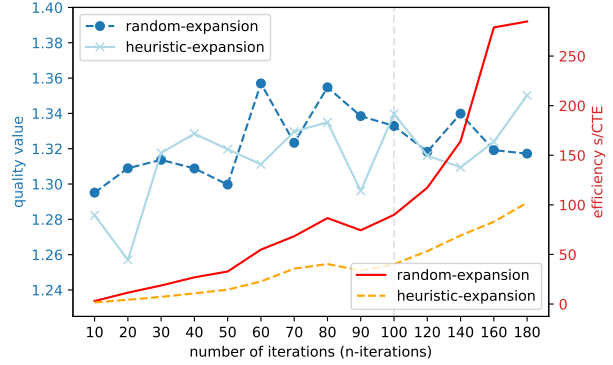

Figure 9. Avg. Quality Value (left y-axis and blue graphs) for CTEs generated by MCTO and efficiency of generation (right y-axis with red graphs) using different numbers of iterations across a setting with (*heuristic-expansion*) and without (*random-expansion*) the expansion heuristic.

D.1.2. RESULTS:

Number of starting points: Figure 8 shows how the quality value of CTEs increases when we choose from more candidate CTEs produced by MCTO 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 with some noise in measurements. This makes sense since having more different options to select the best from is strictly better, but has diminishing returns when considering more options. However, this comes at the cost of efficiency, with the time needed to compute a CTE growing linearly with the number of starting points.

Despite the increased computing time and diminishing returns we deem the increase in performance as significant. Thus we choose to run MCTO for every possible starting state, setting $n_{starts} = 70$ for our experiments.

Number of iterations: We test multiple amounts of iterations per step for a setting using "random-expansion" and "qc-expansion". Figure 9 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* and 0.04 ($p = 0.18$) for *heuristic-expansion*. However, with every additional iteration a new simulation needs 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. Since the benefits of increasing the number of iterations are less clear, we choose $n_{iterations} = 10$ for our experiments.

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

Figure 10 shows no clear trend in quality values when increasing $p_{MCTO}(end)$. One can make out an increase in the early values $(0.05 - 0.2)$ and a dip around $p_{MCTO}(end) = 0.65$. However, these observations might be measurement inaccuracies rather than consistent trends. For low likelihoods of ending ($p_{MCTO}(end) \leq 0.2$) there are significant increases in the computational cost for generating CTEs. This is because individual simulations become longer.

For our experiments, we decide to use $p_{MCTO}(end) = 0.35$ since it achieves the highest quality value in combination with *random-simulation* and is reasonably efficient.

Action threshold: We try a range of values for $threshold_a \in \{0.001, 0.003, 0.01, 0.03, 0.1\}$. There is no clearly best $threshold_a$ recognisable from Figure 11. 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. For our experiments, we choose $threshold_a = 0.03$.

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

Figure 12 does not show a significant impact of the discount factor on the quality value of resulting CTEs. There is also no
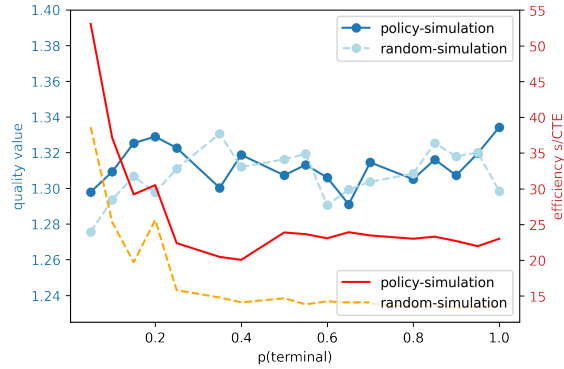
Figure 10. Avg. quality value for CTEs generated by MCTO for different likelihoods of termination during the simulation $p_{MCTO}(end)$. We show this for *policy-* and *random-simulation* along with the efficiency in $\frac{seconds}{CTE}$.
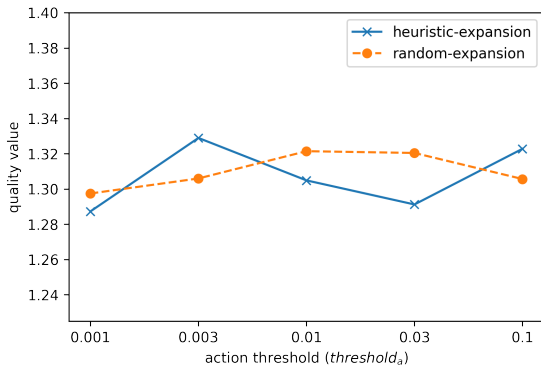

Figure 11. Avg. quality value for CTEs generated by MCTO for different action thresholds $threshold_a$ using both *random-* and *heuristic-expansion*.
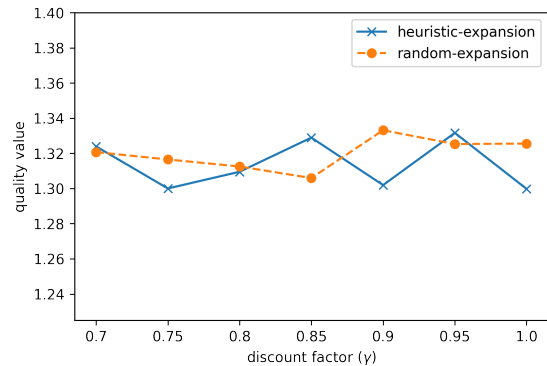

Figure 12. Avg. quality value for CTEs generated by MCTO for different discount factors γ using both *random-* and *heuristic-expansion*.

significant correlation linking the discount factor and quality value. We conclude that adding a discount factor does not help or hinder MCTO in its search and thus choose $\gamma = 1.0$ for simplicity.

Expansion Heuristic: To understand how the method for choosing which branch to extend influences the quality value 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 9). Thus we choose to use *random-expansion* for our experiments.

	$threshold_a$	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 7. The average quality value of CTEs generated 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.

Action selection during simulation: We compare the two proposed methods for selecting actions during the simulation, *policy-simulation* and *random-simulation*, across different values for $p_{MCTO}(end)$ and average the quality values of the resulting CTEs.

There is no statistically significant difference between the two methods with *policy-simulation* averaging a quality value of 1.295 and *random-simulation* achieving 1.302. Additionally, Figure 10 shows us that following the policy is less efficient than choosing randomly. Thus we decide to use *random-simulation* for our experiments.

D.2. Ablations of Deviate and Continue

We ablate three aspects of DaC:

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

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 (*"continue-policy"*) or by uniformly sampling random actions (*"continue-random"*).

3. Likelihood to terminate: At each step of the continuation phase the trajectories have a chance $p_{DaC}(end)$ of ending.

D.2.1. EXPERIMENTAL SETUP:

For each setting DaC is used to generate 200 CTEs each for a set of weights $\omega = \{\omega_{Validity_j}, ..., \omega_{Sparsity_j}\}_{j=1}^{200}$ that is uniformly sampled $\omega_i \sim U(0, 1)$. We test the combination of values for *n-deviations* $\in \{1, 2, 3\}$ and $p_{DaC}(end) \in \{0.05, 0.1, ..., 1.0\}$. Furthermore, we compare *continue-policy* and *continue-random*, while using "1-deviation" and $p_{DaC}(end) = 0.5$.

D.2.2. RESULTS:

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

Table 8. Average quality value achieved across all values for $p_{DaC}(end)$ for different numbers of deviations.

Number of deviations: Table 8 shows that the differences in quality values between different amounts of deviation steps are small. In fact, the differences are statistically not significant indicating that the number of deviations is not essential. However, from Figure 13 we can make out that "3-deviations" outperforms methods with fewer steps on most values for $p_{DaC}(end)$. This is largely because "3-deviations" regularly achieve higher values for Proximity ($p < 0.05$). The highest values are achieved by "3-deviations" between $0.5 \le p_{DaC}(end) \le 0.75$. We thus choose to use "3-deviations".

Action selection: Averaged over multiple values of $p_{DaC}(end)$, continuing with the policy achieves an average quality value of 1.347, while *random-continuation* achieved 1.351. However, these results are not statistically significant. Since *policy-continuation* led to more informative CTEs according to the researchers' subjective judgement, we choose to use *policy-continuation* in our experiments.

Likelihood to terminate: Figure 13 shows a trend that high-quality values are achieved for low $p_{DaC}(end)$ values. As $p_{DaC}(end)$ rises, quality values first drop until $p_{DaC}(end) = 0.25$ and then steadily increase to a maximum of around 0.7, before dropping off again. Moderately high values score well because they produce short CTEs that are rated high on Sparsity. Lower values perform well on Diversity because they can lead to a wide range of lengths of CTEs.

For our experiments, we decide to choose "3-deviations" with $p_{DaC}(end) = 0.55$.

D.3. Hyperparameters 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.
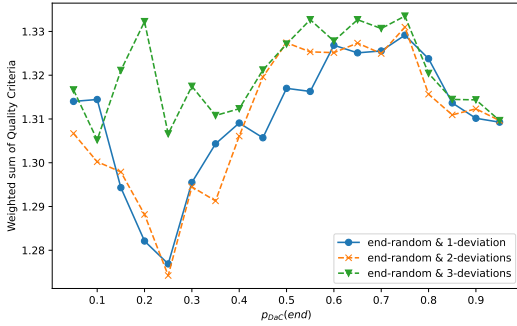
Figure 13. Quality values achieved by different methods for different values of $p_{DaC}(end) \in \{0.05, ...0.95\}$.
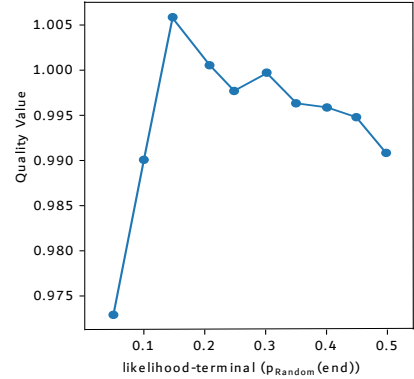

Figure 14. Average Quality Value achieved by Random with different likelihoods of termination

D.3.1. EXPERIMENTAL SETUP:

Similarly to DaC and MCTO, at every step in the trajectory Random has a likelihood of $p_{Random}(end)$ ending the counterfactual and original trajectory. For multiple values of $p_{Random}(end)$, 1000 CTEs are generated for 1000 randomly sampled sets of weights and their quality values ρ are measured.

D.3.2. RESULTS

Figure 14 shows a clear trend where the average quality values achieved by Random first rise until $p_{Random}(end) = 0.15$ and then steadily fall. However, $p_{Random}(end)$ does not strongly influence the performance of Random with quality values only differing in a range of < 0.04. Consequently, we choose $p_{Random}(end) = 0.15$ for our experiments.

E. Ablation for informativeness of quality criteria using DaC

Experimental Setup: The same experimental setup as described in Experiment 3 of the main paper (see Section 5.3) is used to evaluate the influence of the weights ω of quality criteria for the informativeness of CTEs. In contrast to section 5.3 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.

Discussions: The results for DaC present notable differences from those for MCTO (see Experiment 3 in section 5.3). Relatively, Realisticness and Sparsity are more beneficial for making informative CTEs for DaC and Proximity and State Importance are less important than 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 algorithm that is used. This might be because different generation algorithms have different ways of navigating the trade-offs between quality criteria.

F. Ablation using Linear Model as proxy-human model

In the main experiments, we use a Neural Network as a proxy-human model M_ϕ that receives and learns from the CTEs. To test whether our results are robust to using a different type of explainee 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 be generally applicable to other explainees.
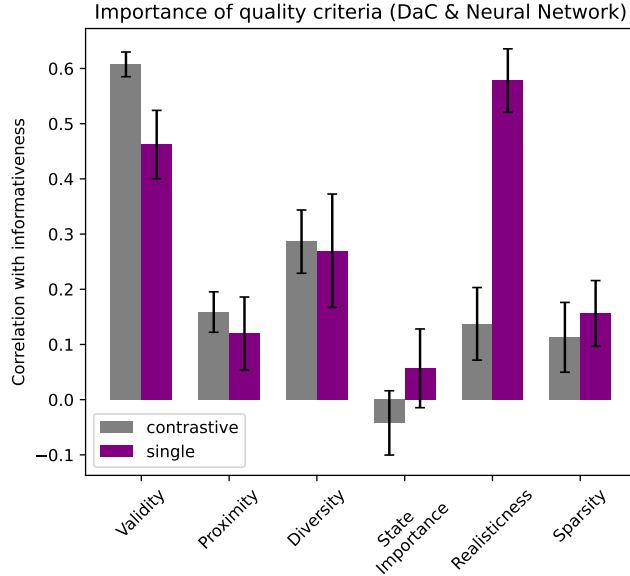
Figure 15. Spearman correlation between weights for the quality criteria and the informativeness of the CTEs generated by DaC for M_ϕ for the contrastive and single task. Averaged over 10 models along with the median and upper and lower quartile.

F.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 M_ϕ.

It repeats the Experimental Setup described in Experiment 3 (see Section 5.3), except that M_ϕ is represented via LMs instead of NNs. Specifically, we use Simple Linear Regression Models with a bias term trained via regression. Instead of performing multi-task learning, as we did with the Neural Network, we simply train two independent Linear Models on the single and contrastive task respectively.

Experimental Setup: We repeat the experimental setup described in Experiment 3 (see Section 5.3), except that M_ϕ is represented via LMs instead of NNs. Specifically, we use Simple Linear Models with a bias term trained via regression. Instead of performing multi-task learning, as we did with the Neural Network, we simply train two independent Linear Models on the single and contrastive task respectively. 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 16). 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 informativeness to the proxy-human Linear Model.

Similarly, for the informativeness of CTEs generated by DaC Figure 17 shows that Realisticness is the most important criterion for single, while the weights for Validity $\omega_{Validity}$ correlate strongest with informativeness on the contrastive task. $\omega_{Diversity}$ proves important for both tasks, while $\omega_{StateImportance}$ is negatively correlated with informativeness for contrastive.

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

Notable differences for MCTO are that Realisticness is considerably more important for an LM on 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.

Overall there are some differences in importance when using MCTO and very few for DaC. This can give us more confidence
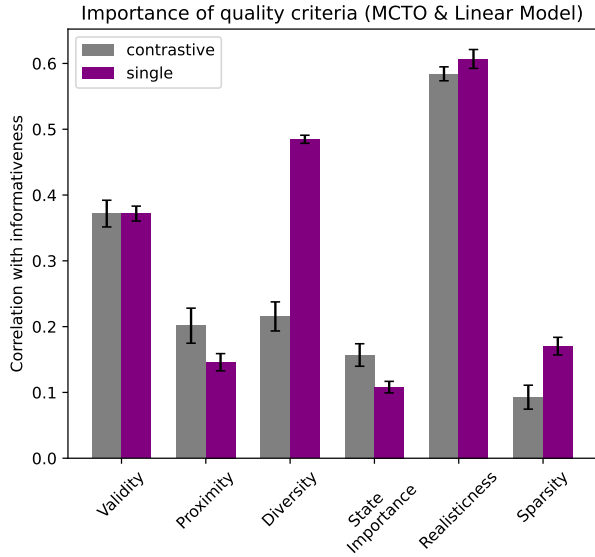
Figure 16. Spearman correlation between the weights of quality criteria and the performance of the Linear Regression Model trained on CTEs generated by MCTO on the contrastive and single task averaged over 10 seeds.
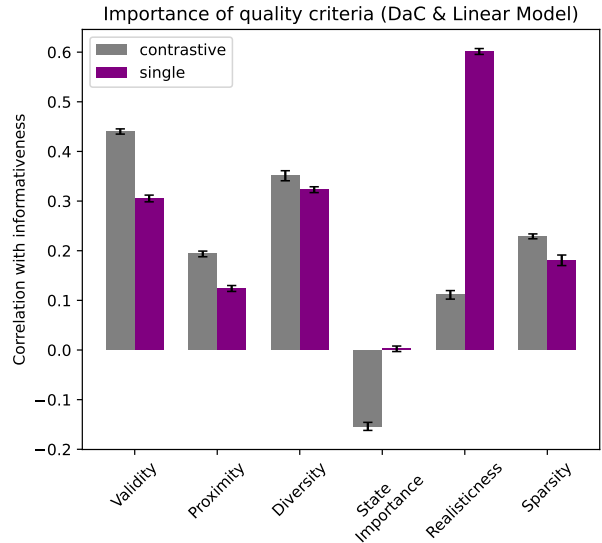

Figure 17. Spearman correlation between the weights of quality criteria and the performance of the Linear Regression Model trained on CTEs generated by DaC on the contrastive and single task averaged over 10 seeds.

that the importance of different quality criteria for generating informative CTEs is similar for different explainees. However, we cannot simply extrapolate these results to all possible explainees but need to reevaluate which criteria should take priority. This applies especially when showing CTEs to humans instead of ML models.

F.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 ablate Experiment 1 (see Section 5.1) by using Linear Models instead of Neural Networks as the proxy-human models M_ϕ.

Experimental Setup: We repeat the experimental setup from Experiment 1 (see section 5.1). We train the Linear Models as described in Appendix F.1.
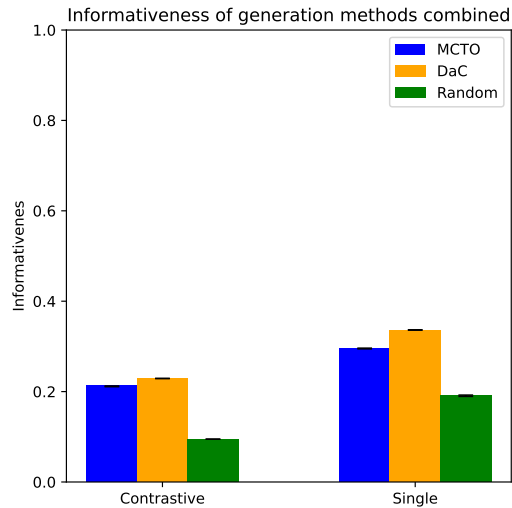

Figure 18. The informativeness of CTEs generated by MCTO, DaC and Random for a Linear Model on the single and contrastive task.

26

Results: DaC performs best on both tasks, while MCTO outperforms Random. Overall informativeness of CTEs is low for the LMs, scoring a correlation of $0.1 - 0.4$ with the labels Further, there is very little deviation between models with different initialisations.

Discussion: In contrast to the Experiment on NNs, LMs are better able to learn from CTEs generated by DaC than by MCTO. This calls into question the conclusion that MCTO is the most effective generation algorithm. However, the generally bad performance of the LMs makes it relatively weak evidence about the effectiveness of methods.

Overall the LM achieves a significantly lower performance on both tasks for all algorithms. Likely, the flexibility 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. Furthermore, LMs do not benefit from cross-task learning, while the NN had a body that was shared between the contrastive and single tasks.