Explaining Learned Reward Functions with Counterfactual Trajectories*

Jan Wehner[1,*], Frans Oliehoek[2] and Luciano Calvante Siebert[2]

[1]CISPA Helmholtz Center for Information Security
[2]Delft University of Technology

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.

Keywords

Value Alignment, Reward Learning, Explainable AI, Counterfactual Explanations

1. Introduction

As Reinforcement Learning (RL) models grow in their capabilities and adoption in real-world applications [1, 2, 3], we must ensure that they are safe and aligned with human values. A core difficulty of achieving trustworthy and controllable AI [4, 5] 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 [6, 7]. Reward Learning is a set of techniques that circumvents this problem by instead learning the reward function from data. For example, Preference-based RL [8] derives a reward function from preference judgments queried from a human and has recently been applied to control the behaviour of Large Language Models [9]. Similarly, Inverse RL [10], 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 [5, 11]. 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 [12]. However, these techniques suffer from a multitude of theoretical [13, 14] and practical problems [15] that make them unable to reliably learn human values which are diverse [16], dynamic [17] and context-dependent [18].

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 [19] 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 [20],

Implementing AI Ethics through a Behavioural Lens — ECAI 2024 Workshop, October 19–20, 2024, Santiago de Compostela, Spain
*Corresponding author.
jan.wehner@cispa.de (J. Wehner)

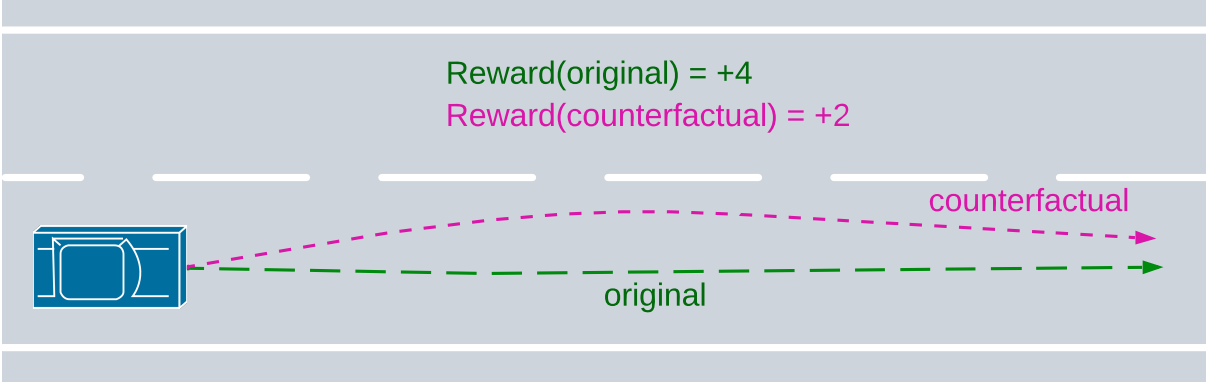


Figure 1: A car has originally taken a straight line and received a reward of +4 from the reward function. By providing a counterfactual that receives a lower reward of +2 the user can make hypotheses about how the reward function assigns rewards.

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 [20, 19]. However, there have been few attempts to interpret reward functions and only Michaud et al. [21] 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[22, 23, 24]. Thus users can intuitively reason about and learn from counterfactual explanations, which makes this an effective and user-friendly mode of explanation [25, 26, 27].

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 [26]. 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

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

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 1. *CTEs* $\{(t_{org}, \bar{r}_{org}), (t_{cf}, \bar{r}_{cf})\}$ *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 $\bar{r}_{cf}$ instead of $\bar{r}_{org}$. [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 [29]. 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 [30, 31, 32] 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 [31, 32]. This difference in outputs makes it possible to causally reason about the changes in the inputs. We maximise Validity as $\left|R_\theta(t_{org}) - R_\theta(t_{cf})\right|$.

2. Proximity: The counterfactual should be similar to the original [30, 33, 32]. Thus we minimize a measure based on the Modified Hausdorff distance [34] 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 [35, 36]. Consequently, each new CTE should establish novel information rather than repeating previously

[2] Examples of CTEs in the Emergency Environment [28] can be found in: https://drive.google.com/drive/folders/1JMjwQM24BbDwL8vRnG3pST5hlvpzRfZM?usp=sharing

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 [37] 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 [36]. 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 $-\sum_{a \in A} \pi(a|s_0) \log \pi(a|s_0)$ [36, 38].

5. Realisticness: The constellation of variables in a counterfactual should be likely to happen [30, 32, 31]. 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: $\overline{R_\theta(t_{cf})} - \overline{R_\theta(t_{org})}$.

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 [30, 31, 32, 33]. 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-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 [39, 40]. 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}.

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**
      $t^s_{cf} \leftarrow$ SELECTION(t_{cf})
      $t^e_{cf} \leftarrow$ EXPANSION(t^s_{cf})
      $\rho \leftarrow$ SIMULATION(t^e_{cf})
      $Q \leftarrow$ BACK-PROPAGATION(Q, ρ)
    end for
    $a^* = argmax_{a \in A}(Q(t_{cf}, a))$
    $s_n \leftarrow env$.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: $argmax_{c \in candidates} \rho(c)$

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

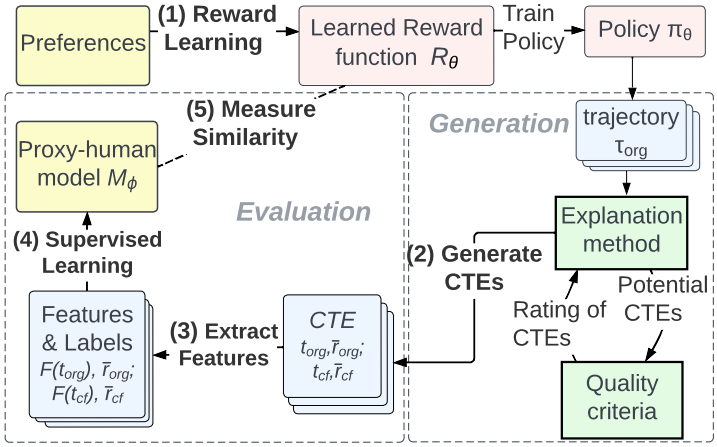


Figure 2: Schematic that describes how rewards are learned (1), explanations are generated (2) and evaluated (3,4&5).

$s_n \in \tau_{org}$, 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.

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) [41]. This policy is used to generate 1000 expert trajectories $\tau_{exp} = \{\tau_{exp_k}\}_{k=1}^{1000}$. Secondly, we use Adversarial IRL [42] 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 [28], 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.

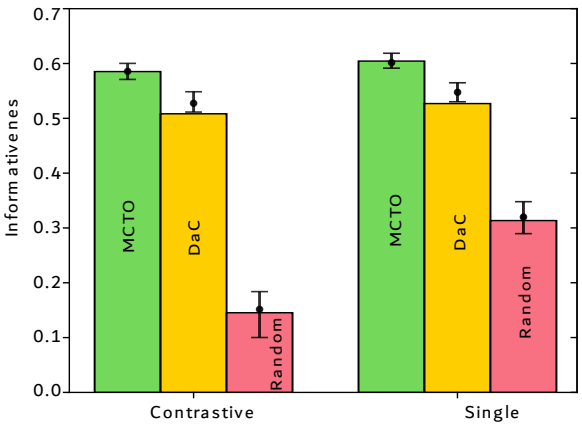
The evaluation procedure consists of three steps, as presented in Figure 2: (3) features and labels are extracted from the CTEs to form a dataset to train on, (4) a proxy-human model is trained to predict the rewards of trajectories from these features, and, lastly, (5) the similarity between the predictions 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 [43] 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.

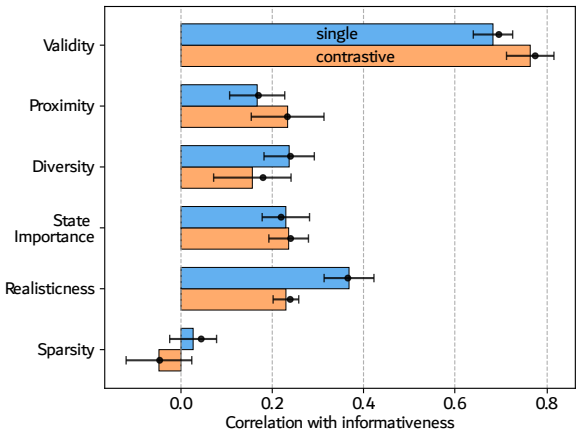
Learning a proxy-human model (4)
A proxy-human regression model M_ϕ is trained to predict the average reward $\bar{r}$ given to the partial trajectory t by R_θ from the extracted features $F(t)$. Humans learn from counterfactual explanations in a contrastive manner by looking at the difference in outputs to causally reason about the effect of the inputs [33] 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.



(a) 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.



(b) 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.

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 [44]. 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.

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 3a 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 of CTEs were recorded.

	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.

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 3b 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 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 3b 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

[3]Efficiency differs depending on the hardware used.

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.

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 [45]. 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 [46, 47, 48] or in logical domains [49, 50]. Attempts have been made to make deep reward functions more interpretable by simplifying them through equivalence transformation [51] or by imitating a Neural Network with a decision tree [52]. 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 [53, 54] 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. [21] 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 [55, 29, 56, 57] and their relation to human psychology [30, 58], 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 [26]. 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 [32]. This can improve users' understanding of out-of-distribution behaviour [36], provide them with more informative demonstrations [59] or showcase how an agent's environmental beliefs influence its planning [60]. 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.

Acknowledgments

The project on which this report is based was funded by the Federal Ministry of Education and Research under the funding code 16KIS2012. The responsibility for the content of this publication lies with the author. Further, This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

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

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

[3] M. M. Afsar, T. Crump, B. Far, Reinforcement learning based recommender systems: A survey, ACM Computing Surveys 55 (2022).

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

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

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

[7] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, D. Mané, Concrete problems in ai safety, arXiv preprint arXiv:1606.06565 (2016).

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

[9] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, et al., Training a helpful and harmless assistant with reinforcement learning from human feedback, arXiv preprint arXiv:2204.05862 (2022). arXiv:2204.05862.

[10] A. Y. Ng, S. J. Russell, Algorithms for inverse reinforcement learning, in: Proceedings of the Seventeenth International Conference on Machine Learning, 2000, pp. 663–670.

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

[12] P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the twenty-first international conference on Machine learning, 2004, p. 1.

[13] S. Armstrong, S. Mindermann, Occam's razor is insufficient to infer the preferences of irrational agents, Advances in neural information processing systems 31 (2018).

[14] J. M. V. Skalse, A. Abate, Misspecification in inverse reinforcement learning, in: NeurIPS ML Safety Workshop, 2022.

[15] S. Casper, X. Davies, C. Shi, T. K. Gilbert, J. Scheurer, et al., Open problems and fundamental limitations of reinforcement learning from human feedback, arXiv preprint arXiv:2307.15217 (2023). arXiv:2307.15217.

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

[17] I. van de Poel, Understanding value change, Prometheus 38 (2022) 7–24.

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

[19] L. Sanneman, J. Shah, Transparent value alignment, in: Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction, HRI '23, New York, NY, USA, 2023, p. 557–560.

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

[21] E. J. Michaud, A. Gleave, S. Russell, Understanding learned reward functions, Deep RL Workshop, NeurIPS 2020 (2020).

[22] R. M. Byrne, Counterfactual thought, Annual review of psychology 67 (2016) 135–157.

[23] D. Kahneman, D. T. Miller, Norm theory: Comparing reality to its alternatives., Psychological review 93 (1986) 136.

[24] N. J. Roese, J. M. Olson, What might have been: The social psychology of counterfactual thinking, Psychology Press, 2014.
[25] B. Mittelstadt, C. Russell, S. Wachter, Explaining explanations in ai, in: Proceedings of the conference on fairness, accountability, and transparency, 2019, pp. 279–288.
[26] S. Wachter, B. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the gdpr, Harvard Journal of Law & Technology 31 (2018).
[27] D. R. Mandel, Of causal and counterfactual explanation, in: Understanding counterfactuals, understanding causation: Issues in philosophy and psychology, Oxford University Press, 2011, p. 147.
[28] M. Peschl, A. Zgonnikov, F. A. Oliehoek, L. C. Siebert, Moral: Aligning ai with human norms through multi-objective reinforced active learning, in: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS '22, Richland, SC, 2022, p. 1038–1046.
[29] A. Artelt, B. Hammer, On the computation of counterfactual explanations – a survey, arXiv preprint arXiv:1911.07749 (2019).
[30] M. T. Keane, E. M. Kenny, E. Delaney, B. Smyth, 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 (2021) 4466–4474.
[31] A. Verma, V. Murali, R. Singh, P. Kohli, S. Chaudhuri, Programmatically interpretable reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 5045–5054.
[32] J. Gajcin, I. Dusparic, Counterfactual explanations for reinforcement learning, arXiv preprint arXiv:2210.11846 (2022).
[33] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38.
[34] M.-P. Dubuisson, A. Jain, A modified hausdorff distance for object matching, in: Proceedings of 12th International Conference on Pattern Recognition, volume 1, 1994, pp. 566–568 vol.1.
[35] S. H. Huang, D. Held, P. Abbeel, A. D. Dragan, Enabling robots to communicate their objectives, Autonomous Robots 43 (2019) 309–326.
[36] J. Frost, O. Watkins, E. Weiner, P. Abbeel, T. Darrell, B. Plummer, K. Saenko, Explaining reinforcement learning policies through counterfactual trajectories, arXiv preprint arXiv:2201.12462 (2022).
[37] N. J. Roese, The functional basis of counterfactual thinking., Journal of personality and Social Psychology 66 (1994) 805.
[38] S. H. Huang, K. Bhatia, P. Abbeel, A. D. Dragan, Establishing appropriate trust via critical states, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 3929–3936.
[39] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[40] T. Vodopivec, S. Samothrakis, B. Ster, On monte carlo tree search and reinforcement learning, Journal of Artificial Intelligence Research 60 (2017) 881–936.
[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).
[42] J. Fu, K. Luo, S. Levine, Learning robust rewards with adversarial inverse reinforcement learning, arXiv preprint arXiv:1710.11248 (2017).
[43] A. O. Salau, S. Jain, Feature extraction: A survey of the types, techniques, applications, in: 2019 International Conference on Signal Processing and Communication (ICSC), 2019, pp. 158–164.
[44] A. Y. NG, Policy invariance under reward transformations : Theory and application to reward shaping, Proc. of the Sixteenth International Conference on Machine Learning (1999).
[45] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information., Psychological review 63 (1956) 81.

[46] T. Bewley, F. Lecue, Interpretable preference-based reinforcement learning with tree-structured reward functions, in: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, 2022, pp. 118–126.
[47] A. Kalra, D. S. Brown, Interpretable reward learning via differentiable decision trees, in: NeurIPS ML Safety Workshop, 2022.
[48] S. Srinivasan, F. Doshi-Velez, Interpretable batch irl to extract clinician goals in icu hypotension management, AMIA Summits on Translational Science Proceedings 2020 (2020) 636.
[49] D. Kasenberg, M. Scheutz, Interpretable apprenticeship learning with temporal logic specifications, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, 2017, pp. 4914–4921.
[50] T. Munzer, B. Piot, M. Geist, O. Pietquin, M. Lopes, Inverse reinforcement learning in relational domains, in: International joint conferences on artificial intelligence, 2015.
[51] E. Jenner, A. Gleave, Preprocessing reward functions for interpretability, arXiv preprint arXiv:2203.13553 (2022).
[52] J. Russell, E. Santos, Explaining reward functions in markov decision processes, in: The Thirty-Second International Flairs Conference, 2019.
[53] L. Sanneman, J. Shah, Explaining reward functions to humans for better human-robot collaboration, arXiv preprint arXiv:2110.04192 (2021).
[54] L. Sanneman, J. A. Shah, An empirical study of reward explanations with human-robot interaction applications, IEEE Robotics and Automation Letters 7 (2022) 8956–8963.
[55] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, C. Shah, Counterfactual explanations and algorithmic recourses for machine learning: A review, arXiv preprint arXiv:2010.10596 (2020).
[56] R. Guidotti, Counterfactual explanations and how to find them: literature review and benchmarking, Data Mining and Knowledge Discovery (2022) 1–55.
[57] I. Stepin, J. M. Alonso, A. Catala, M. Pereira-Fariña, A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence, IEEE Access 9 (2021) 11974–12001.
[58] R. M. Byrne, Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning., in: Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019, pp. 6276–6282.
[59] M. S. Lee, H. Admoni, R. Simmons, Reasoning about counterfactuals to improve human inverse reinforcement learning, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 9140–9147.
[60] G. Stein, Generating high-quality explanations for navigation in partially-revealed environments, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, volume 34, 2021, pp. 17493–17506.