Explanation-Guided Reward Alignment

Saaduddin Mahmud[1] , **Sandhya Saisubramanian**[2] , **Shlomo Zilberstein**[1]
[1]University of Massachusetts Amherst, USA
[2]Oregon State University, USA
smahmud@umass.edu, sandhya.sai@oregonstate.edu, shlomo@umass.edu

Abstract

Agents often need to infer a reward function from observations in order to learn desired behaviors. However, agents may infer a reward function that does not align with the original intent, as there can be multiple reward functions consistent with their observations. Operating based on such misaligned rewards can be risky. Furthermore, black-box representations make it difficult to verify the learned reward functions and prevent harmful behavior. We present a framework for verifying and improving reward alignment using explanations, and we show how explanations can help detect misalignment and reveal failure cases in novel scenarios. The problem is formulated as inverse reinforcement learning from ranked trajectories. Verification tests created from the trajectory dataset are used to iteratively verify and improve reward alignment. The agent explains its learned reward, and a tester signals whether the explanation passes the test. In cases where the explanation fails, the agent offers alternative explanations to gather feedback, which is then used to improve the learned reward. We analyze the efficiency of our approach in improving reward alignment using different types of explanations and demonstrate its effectiveness in five domains.

1 Introduction

Autonomous agents are increasingly being deployed in the real world to complete complex and nuanced tasks [Zilberstein, 2015; Dietterich, 2017]. A predominant approach to train such agents in the absence of a reward function is learning from demonstrations (LfD) [Argall *et al.*, 2009]. Inverse reinforcement learning (IRL) is a form of LfD that is designed to retrieve a reward function that motivates the observed behavior [Sutton and Barto, 1998], enabling agents to learn and generalize observed behavior to unseen situations.

However, the generalization capability is often affected by the quality and size of the trajectory dataset used to learn the reward function. In particular, we focus on situations where (1) the reward function is learned from ranked sub-optimal trajectories, and (2) the trajectory dataset covers only a subset of states, providing no direct information about acceptable behavior in other states. Additionally, the dataset may also contain spurious state feature correlations—multiple features always co-occur—causing the agent to learn a misaligned reward. This can occur, for example, when training an autonomous vehicle (AV) using data collected at a particular geo-location and then deploying it in multiple different areas.

Figure 1 illustrates this challenge with an AV approaching a pedestrian walking two dogs. Suppose that in the trajectory dataset, the driver always stops when a human is crossing the street with dogs. Since dogs are often accompanied by humans, the rare case of encountering dogs alone might be missing from the dataset. Consider four different reward functions consistent with the trajectory dataset, $(R_i, 1 \leq i \leq 4)$, each with the same negative reward for not stopping in this case. R_1 does not account for dogs, R_2 rewards stopping for pedestrians with dogs, R_3 rewards stopping for pedestrians or dogs, and R_4 rewards stopping for all objects, including leaves or a plastic bag on the road. Without additional information, the AV may randomly learn one of these reward functions (say R_2), however, R_3 represents the intended reward. When operating based on R_2, the AV may not stop for dogs unaccompanied by humans. This example illustrates the reward ambiguity stemming from the incomplete trajectory dataset and the consequences of operating based on a misaligned reward. While increasing the diversity of the trajectories in the dataset may help to some extent, it is often practically infeasible or risky to demonstrate certain trajectories.

We propose to address this issue by using explanations to verify and improve reward alignment. Explaining the learned model reveals not only what the agent knows but also the potential errors in its reasoning. Building on this insight, we present a framework for **re**ward **ve**rification and re-**al**ignment using **e**xplanations (REVEALE) that consists of a *verification phase* and an *improvement phase*. Initially, the posterior over the reward function is calculated from the trajectory dataset using a Bayesian IRL method [Brown *et al.*, 2020]. In the verification phase, the tester verifies the maximum a posteriori (MAP) reward function using verification tests in the form of queries to the agent. The agent responds by explaining its reward, and the tester signals whether the explanation passes the verification test. If it fails, the agent presents additional explanations from an alternative sample of the current posterior. The tester provides feedback by selecting the explanation that most closely matches the correct reasoning. This is followed by the improvement phase, in which the agent updates

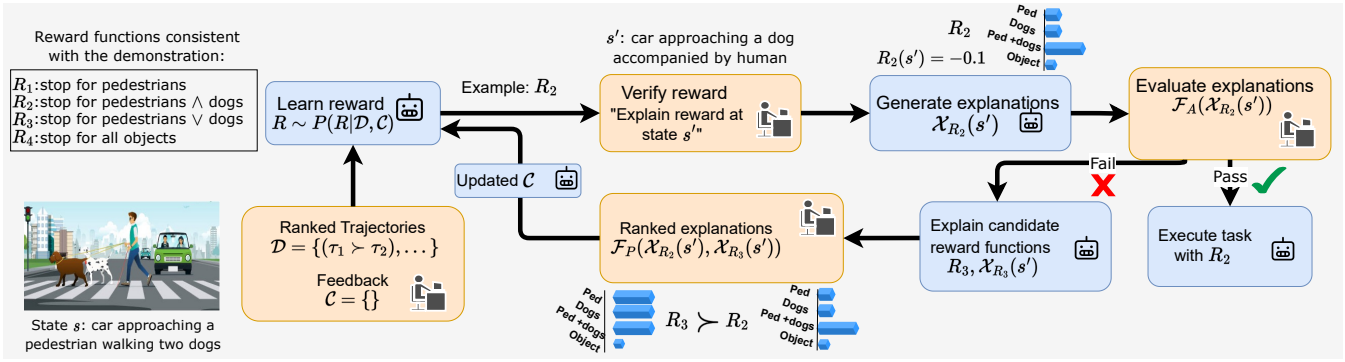


Figure 1: An example of reward verification and learning with REVEALE in autonomous navigation.

its posterior, based on the additional feedback.

We use verification tests in the form of a query: "explain the reward at state s," and explanations in the form of feature attribution (e.g., saliency map [Simonyan *et al.*, 2014], the gradient of the function [Tayyub *et al.*, 2022], and LIME [Ribeiro *et al.*, 2016]). An example of a verification test for the scenario described in Figure 1 is "explain the reward when the AV encounters a dog accompanied by humans," to which the agent would respond with its reward value and feature attributions (for R_2) indicating a low weight for the 'dogs' feature. This reveals a potential weakness of the model in the counterfactual scenario in which the dog is not accompanied by a human (missing from the dataset). When this verification test fails, the agent explains another candidate reward function (for example, R_3). The tester then selects an explanation that is similar to the explanation that their intended reward would generate (R_3 in this case), indicating that R_3 is ranked over R_2 and the desired behavior is to stop for pedestrians or dogs. This example highlights two key advantages of our method: (1) REVEALE *exposes wrong reward estimation* in novel situations that do not appear in the dataset, and (2) it *improves the alignment* of the reward function offline without requiring additional trajectory samples containing the novel situations.

2 Background and Related Work

Markov decision process (MDP). An MDP M is represented by the tuple $M{=}(S, A, T, R, S_0, \gamma)$ where S is a finite set of states, A is a finite set of actions, $T : S{\times}A{\times}S \to [0, 1]$ is the transition function, $R : S \to \mathbb{R}$ is the reward function (bounded in absolute value by R_{max}), S_0 is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : S \to A$ is a mapping from states to actions. The state values of a policy π are defined as $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0, \pi], \forall s \in S$. The optimal values are denoted by $V^*(s) = \max_\pi V^\pi(s)$. While in this paper we describe our approach using state-based reward functions, it can be naturally extended to rewards defined as $R(s_i, a_i)$ or $R(s_i, a_i, s_{i+1})$.

Reward learning. Most IRL algorithms learn a reward function using expert trajectories [Ng and Russell, 2000; Abbeel and Ng, 2004; Ziebart *et al.*, 2008]. Recent algorithms utilize additional information to improve reward learning, such as preferences over sub-optimal trajectories [Brown *et al.*, 2020; Palan *et al.*, 2019], prior over reward functions [Ramachandran and Amir, 2007], or feature queries [Basu *et al.*, 2018]. However, most of these methods do not generalize well when there is a mismatch between the training and deployment settings. Further, unlike [Basu *et al.*, 2018] that uses human feedback to identify a feature that affects trajectory preferences, we use feedback to identify which automatically generated explanation provides correct reasoning about reward estimation, thereby reducing reward ambiguity. While the former approach is limited to linear rewards, our approach generalizes to nonlinear cases. Finally, none of these previous methods perform reward alignment verification.

Bayesian IRL. A Bayesian framework for IRL defines a probability distribution of reward functions given a dataset $\mathcal{D}$ using Bayes rule, $\mathcal{P}(R|\mathcal{D}) \propto P(\mathcal{D}|R)P(R)$. The form of $P(\mathcal{D}|R)$ may vary depending on the specifics of the information available in $\mathcal{D}$. For scalability reasons, we utilize the definition provided by BREX [Brown *et al.*, 2020]. Let τ_i^1 and τ_i^2 denote two different trajectories[1] and $\tau_i^1 \succ \tau_i^2$, indicating that τ_i^1 is ranked over τ_i^2. Then the trajectory dataset is denoted by $\mathcal{D} = \{(\tau_1^1 \succ \tau_1^2), (\tau_2^1 \succ \tau_2^2), ..., (\tau_n^1 \succ \tau_n^2)\}$. BREX defines $\mathcal{P}(\mathcal{D}|R)$ as:

$$\mathcal{P}(\mathcal{D}|R) = \prod_{(\tau_i^1 \succ \tau_i^2) \in \mathcal{D}} \frac{e^{\beta R(\tau_i^1)}}{e^{\beta R(\tau_i^1)} + e^{\beta R(\tau_i^2)}} \quad (1)$$

where, $R(\tau) = \sum_{s \in \tau} R(s)$ and $\beta \in [0, \infty)$.

Value alignment. Value alignment focuses on ensuring that an agent's behavior is aligned with its user's intentions. It has also been explored in efforts to avoid *negative side effects* of AI systems [Saisubramanian *et al.*, 2022]. Notably, the inverse reward design approach [Hadfield-Menell *et al.*, 2017] aims to retrieve the intended reward function by treating the specified reward function as a proxy. On the other hand, we learn the true reward function using human feedback on automatically generated explanations of potential reward functions. While some recent work focused on value

[1]Here a trajectory is a sequence of consecutive states visited when following some policy π.

alignment verification (VAV) with a minimum number of queries [Brown *et al.*, 2021], our work differs in that (1) we use feedback in the form of ranking of reward explanations to verify the reward, while VAV uses reward weights, value weights or trajectory ranking for verification, and (2) VAV can detect but *cannot* amend misaligned reward, while our approach verifies and rectifies incorrect reward. Furthermore, VAV can only check the consistency of the value function in situations that occur during training and cannot verify performance in novel situations.

Explainable AI. Explainable AI methods often use feature attribution to explain the relationship between input features and the output of a learned model. Examples include LIME [Ribeiro *et al.*, 2016], gradient as explanation (GaE) [Tayyub *et al.*, 2022] for tabular/feature-based data, and saliency maps [Simonyan *et al.*, 2014] for image data [Alqaraawi *et al.*, 2020]. GaE is a core component of saliency maps and provides insight into the local structure of the function. LIME learns a linear regression model for states sampled in the neighborhood of the state being explained. Besides feature attribution, there are other broad classes of automated explanation generation methods such as model reconciliation [Chakraborti *et al.*, 2017] and policy summarization [Amir *et al.*, 2019]. While many works in this area can be considered as methods for reward alignment verification [Huang *et al.*, 2018; Tabrez *et al.*, 2020] none of them improve reward alignment. Finally, [Guan *et al.*, 2021] focuses on using explanation to improve Q-value estimation but does not retrieve any reward function.

3 The REVEALE Framework

Consider an agent trained to operate in an environment modeled as an MDP, $M = (S, A, T, \mathbf{R}, S_0, \gamma)$, where the reward function $\mathbf{R}$ is initially *unknown* to the agent. The agent aims to learn $\mathbf{R}$ using trajectory data $\mathcal{D}$. We consider a factored state representation and use $\phi(s)$ to donate the set of factors of a state s.

Definition 1. *Given an MDP M with an unknown reward function $\mathbf{R}$, a REVEALE instance is defined by a tuple $\langle M, \mathcal{D}, \mathcal{R}, S_V, \mathcal{E}, \mathcal{X}, \mathcal{F} \rangle$ where*

- *M is the underlying MDP with $\mathbf{R}$ initially unknown to the agent;*
- *$\mathcal{D}$ denotes the trajectory dataset;*
- *$\mathcal{R}$ denotes the space of possible reward functions for M;*
- *$S_V \subseteq S$ denotes the set of verification test states used by the human to verify the agent's learned reward;*
- *$\mathcal{E}$ is the space of explanations, corresponding to $\mathcal{R}$;*
- *$\mathcal{X} : S \times \mathcal{R} \rightarrow \mathcal{E}$ is the agent's explanation generation function that generates an explanation $e \in \mathcal{E}$, given a verification test state $s_V \in S_V$ and the reward function $R \in \mathcal{R}$; and*
- *The set $\mathcal{F}$ consists of feedback mapping functions, where each function $\mathcal{F}_i \in \mathcal{F}$ is responsible for mapping a particular type of feedback signals provided by a human in response to the agent's explanation.*

Problem setting. We consider the setting where the reward is learned offline, using a limited number of pairwise rankings of sub-optimal trajectories, $\mathcal{D} = \{(\tau_1^1 \succ \tau_1^2), \ldots, (\tau_n^1 \succ \tau_n^2)\}$, similar to [Brown *et al.*, 2019b]. The learned reward is later used to solve different instances of the domain. The quality of the retrieved reward depends on the composition of the dataset, which, in turn, depends on the source of the dataset. Increasing the size of the training data does not guarantee to learn the intended reward, as the additional trajectories generated from the same source may not provide novel information critical for learning an aligned reward. Furthermore, it is practically infeasible to foresee and construct a dataset that contains information pertaining to novel states that the agent will encounter when deployed.

REVEALE address this without making any assumptions about the composition or the source of the trajectory dataset and utilizes explanations and a human tester. The tester is assumed to be an entity capable of reasoning about reward estimation (e.g., identifying a feature that makes a state good or bad). The tester is not required to be aware of all possible novel scenarios missing from the dataset a priori. Instead, REVEALE exposes wrong reward estimation in novel states using explanations of states that appear in the dataset. An alternative to REVEALE would be to train and deploy the agent using the unverified reward function and collect additional data about undesirable behavior. However, this approach can be unsafe, costly, and time-consuming in many scenarios.

Based on [Brown *et al.*, 2019a], the space of reward functions consistent with the ranked dataset is defined below.

Definition 2. *Given an MDP M and ranked dataset $\mathcal{D}$, a **data-consistent reward set**, denoted $\Delta(\mathcal{D})$, is a set of reward functions under which the higher ranked trajectories have a higher reward than the lower ranked trajectories, $\Delta(\mathcal{D}) = \{R \in \mathcal{R} \mid R(\tau_i^1) > R(\tau_i^2), \forall (\tau_i^1 \succ \tau_i^2) \in \mathcal{D}\}$.*

Explanation generation function ($\mathcal{X}$). The agent's explanations involve two components: (1) the *reward* at the verification test state s_V, $R(s_V)$, following the most likely reward function; and (2) *feature attribution* indicating the influence of each state features of s_V, $\phi(s_V)$, on the reward function.

Feature attribution-based explanations are widely used in interpretable machine learning [Molnar, 2022]. A feature attribution is a scoring function of form $\mathcal{X} : S \times \mathcal{R} \rightarrow \mathbb{R}^{|\phi(s)|}$, evaluating the contribution of each state feature to the output. It is a form of *local* explanation as it only explains the reward function at each state in isolation. We use established local explanation techniques mentioned earlier: gradient as explanation (GaE), LIME, and saliency maps.

Verification and feedback. We use verification tests of the form "Explain the reward at state s_V" with $s_V \in S_V$ selected by the tester. The set S_V is selected from the trajectory dataset $\mathcal{D}$. The agent responds by automatically generating explanations, consisting of the reward value at s_V, $R(s_V)$, and the feature attribution $\mathcal{X}(s_V, R) = \mathcal{X}_R(s_V)$ describing the reasoning behind the reward estimation. Different types of feedback are provided for the agent's response in the following two situations:

- **Approval**: A binary signal indicating the tester's approval $\mathcal{F}_A(\mathcal{X}_R(s_V)) = 1$ or disapproval $\mathcal{F}_A(\mathcal{X}_R(s_V)) = 0$ of the

explanation, denoting the outcome of the verification test.

- **Explanation feedback**: When the verification test fails, the tester provides accurate feedback on explanations generated by the agent in one of the following two forms:

1. *Oracle explanations*, typically provided by the human, in the form of exact feature attribution corresponding to their intended reward, $\mathcal{F}_O(\mathcal{X}_R(s_V)) = \mathcal{X}_O(s_V), \forall s_V \in S_V$, where $\mathcal{X}_O(s_V)$ denotes the exact feature attribution generated by the Oracle. Though this is an ideal setting as $\mathcal{X}_O(s_V)$ provides features that are critical to learning the intended reward, this type of feedback can be harder to collect in practice, except for simpler domains.
2. *Pairwise ranking* over feature attributions generated by the agent, $\mathcal{F}_P(\mathcal{X}_{R_1}(s_V), \mathcal{X}_{R_2}(s_V)) \in \{(\mathcal{X}_{R_1}(s_V) \succ \mathcal{X}_{R_2}(s_V)), (\mathcal{X}_{R_2}(s_V) \succ \mathcal{X}_{R_1}(s_V))\}, \forall s_V \in S_V$. Here, $\mathcal{X}_{R_1}(s_V)$ and $\mathcal{X}_{R_2}(s_V)$ denote explanations of two reward functions, R_1, and R_2 respectively. This is a more realistic form of feedback that identifies the explanation that better captures the intended reward.

Definition 3. *Given an MDP M and a set of test states S_V, an **explanation-consistent reward set**, denoted $\Delta(\mathcal{X}^{S_V})$, is a set of reward functions whose corresponding explanations are approved by the tester:*
$\Delta(\mathcal{X}^{S_V}) = \{R \in \mathcal{R} \mid \mathcal{F}_A(\mathcal{X}_R(s_V)) = 1, \forall s_V \in S_V\}$.

Definition 4. ***Reward ambiguity** is a measure proportional to the size of the consistent reward set Δ, such as $|\Delta|$ when Δ is finite and discrete, and volume of Δ, $\mathcal{V}(\Delta)$, when Δ is continuous, as in a simplex in $\mathbb{R}^k$.*

REVEALE aims to decrease the reward ambiguity by reducing the size of the data-consistent reward set, $\Delta(\mathcal{D})$. This is accomplished by considering only the subset of reward functions in $\Delta(\mathcal{D})$ that successfully passes all the verification tests.

Definition 5. *A solution of a REVEALE instance is a reward function, $R \in \Delta(\mathcal{D}) \cap \Delta(\mathcal{X}^{S_V})$. An optimal solution is a reward function, $R \in \Delta(\mathcal{D}) \cap \Delta(\mathcal{X}^{S_V})$ that is aligned most closely with the actual human intent.*

While ideally, we would want to completely reduce reward ambiguity, as it would guarantee an optimal solution, it may not always be feasible. In the next section, we will first present an algorithm that can find a solution to the REVEALE instance. Subsequently, we will theoretically analyze a simple scenario in which this algorithm will guarantee an optimal solution and measure the extent of reward ambiguity when it deviates from optimality.

4 Solution Approach

The input to our algorithm, *iterative learning and verification of reward* (ILV), consists of a set of ranked trajectories $\mathcal{D}$ and the verification test states S_V. The test states can be selected randomly from $\mathcal{D}$ or by the tester, who possesses a broader scope of knowledge and can identify critical states that affect performance. The algorithm begins by initializing an empty set of feedback $\mathcal{C}$ and retrieves the initial reward function[2] R_m using Equation 1. Then the algorithm alternates between the verification and improvement phases until a reward function is found that passes all the verification tests.

Verification phase. In this phase, for each verification test state $s_V \in S_V$, the reward value $R_m(s_V)$ and the corresponding explanation $\mathcal{X}_{R_m}(s_V)$ are shown to the tester for approval. If approved, then $\mathcal{X}_{R_m}(s_V)$ is added to $\mathcal{C}$ as an Oracle explanation. If disapproved, additional feedback is requested from the tester. When possible, the tester can provide the correct explanation, i.e., $\mathcal{X}_O(s_V)$. Otherwise, the agent generates an alternative explanation $\mathcal{X}_{R'}(s_V)$ and collects the tester's ranking over $\mathcal{X}_{R_m}(s_V)$ and $\mathcal{X}_{R'}(s_V)$. Here, R' is a different reward function sampled from the posterior. Finally, all the feedback is added to $\mathcal{C}$. Note that if the tester fails to distinguish between the two explanations, additional alternative explanations can be generated to help the tester. If the agent does not pass all the tests, the algorithm goes to the improvement phase.

Improvement phase. In the improvement phase, new posterior distribution over the reward function is calculated by combining $\mathcal{D}$ and $\mathcal{C}$ using Equation 2,

$$P(R|\mathcal{D}, \mathcal{C}) \propto P(\mathcal{C}|\mathcal{D}, R)P(\mathcal{D}|R)P(R). \tag{2}$$

Since explanations only depend on the reward function, $P(\mathcal{C}|\mathcal{D}, R) = P(\mathcal{C}|R)$. Based on this posterior distribution, a new MAP reward R_m is calculated. Then the algorithm goes back to the verification phase.

Thus, our algorithm iteratively guides the reward alignment using explanations. In the rest of this section, we define $P(\mathcal{C}|R)$ and analyze the proposed method for different types of explanation generation methods.

4.1 Linear Reward Alignment

In this section, we define and analyze $P(\mathcal{C}|R)$ for linear reward models[3], which are described by a linear weighted combination of features describing the state, $R(s) = \mathbf{w}^T\phi(s)$, $\mathbf{w} \in \mathbb{R}^n$. For a linear reward, explanations generated using LIME (LM) will produce the same output as GaE. Therefore, we only present results using GaE and saliency maps (SM). Note that we defined GaE as the gradient of reward function w.r.t. input state features, i.e., $\frac{\mathrm{d}R(s)}{\mathrm{d}\phi(s)}$ and SM as $abs(\frac{\mathrm{d}R(s)}{\mathrm{d}\phi(s)})$. Here, $abs(.)$ is used to indicate absolute value. Given an oracle feedback explanation $\mathcal{X}_O(s_i)$ for state s_i and a distance measurement $D(.)$ (e.g., L2, cosine similarity), the learned reward function R must satisfy the following constraint:

$$D(\mathcal{X}_O(s_i), \mathcal{X}_R(s_i)) = 0. \tag{3}$$

This is because we want the learned reward function to produce the same explanation as the Oracle. Similarly, given the ranking over two explanations $\mathcal{X}_{R_1}(s_i) \succ \mathcal{X}_{R_2}(s_i)$, the learned reward function R must satisfy the following constraint:

$$D(\mathcal{X}_{R_1}(s_i), \mathcal{X}_R(s_i)) < D(\mathcal{X}_{R_2}(s_i), \mathcal{X}_R(s_i)). \tag{4}$$

[2] R_m can be initialized using the existing IRL method used in conjunction with REVEALE, BREX in our case.

[3] Our implicit assumption is that the intended reward function can be represented with a vector in $\mathbb{R}^n$.

This is because we want the explanation generated by the leaned reward function to look more similar to the higher-ranked explanation. We henceforth use $\mathcal{C}$ to denote the set of all such constraints constructed from Equations 3-4 using the feedback. Since an aligned reward function will satisfy all the constraints, we can define $P(\mathcal{C}|R)$ as:

$$P(\mathcal{C}|R) = \frac{1}{Z}\mathbb{I}(\mathcal{C}, R), \tag{5}$$

with, $\mathbb{I}(\mathcal{C}, R) = 1$ when R satisfies all the constraints in $\mathcal{C}$ and 0 otherwise. When using $P(\mathcal{C}|R)$ from Equation 5 in Equation 2, the posterior probability of all the reward functions that do not produce the correct explanations are evaluated to 0 in the improvement phase. The posterior probability of all the reward functions that do produce correct explanations remains proportional to the original probability derived by BREX [Brown *et al.*, 2020]. Intuitively, the key role of REVEALE is to eliminate deceptive reward functions—reward functions that accurately estimate rewards for the training dataset and therefore are selected by BREX—but would fail to produce correct estimates in novel situations.

We first show[4] that ILV returns a correct solution, i.e., a reward function $R \in \Delta(\mathcal{D}) \cap \Delta(\mathcal{X}^{S_V})$.

Proposition 1. *As $lim_{\beta\to\infty}$ in Equation 1, $P(R|\mathcal{D}, \mathcal{C}) = 0,\ \forall R \notin \Delta(\mathcal{D}) \cap \Delta(\mathcal{X}^{S_V})$.*

Proof Sketch. This comes directly from the fact that $P(\mathcal{C}|R) = 0,\ \forall R \notin \Delta(\mathcal{X}^{S_V})$ by definition and $P(\mathcal{D}|R) = 0,\ \forall R \notin \Delta(\mathcal{D})$ as $lim_{\beta\to\infty}$. □

Based on Propositon 1 we know that ILV is guaranteed to produce a correct solution. We now focus on showing that feedback on explanation reduces reward ambiguity using Definition 4. Remember that showing a complete reduction in ambiguity is sufficient to guarantee an optimal solution. We assume that the agent and the tester share the same similarity measure, and discuss results with cosine similarity. We also assume that all the ranking in the trajectory dataset $\mathcal{D}$ and explanation feedback constraint in $\mathcal{C}$ are correct.

Proposition 2. *ILV can remove reward ambiguity completely with **1** Oracle-generated GaE explanation feedback.*

Proof Sketch. Let the intended reward be $R^*(s) = \mathbf{w}^{*T}\phi(s)$. Now, $\forall s \in S$, the explanation given by the Oracle is $\mathcal{F}_O(\mathcal{X}_R(s)) = \mathbf{w}^*$. Hence, a single GaE explanation is sufficient to reveal all the parameters of the intended reward, and completely reduce the reward ambiguity. □

Proposition 2 considers the most ideal scenario and establishes a direct bridge between the feature attribution methods and reward alignment. Intuitively, it tells us that in the most ideal case, ILV is guaranteed to produce an optimal solution using only 1 sample feedback. The next proposition provides a sample complexity for pairwise ranking feedback.

[4]Note that this also holds for non-linear reward models when using Equation 5.

Proposition 3. *To reduce reward function ambiguity by $x\%$ in expectation, it suffices to have ranked feedback over a set of $k = \log_2(1/(1 - x/100))$ randomly generated GaE explanation pairs.*

Proof Sketch. Let $\mathcal{R}$ denote the set of all reward functions, where $\mathcal{R} = \mathbb{R}^n$ and the intended reward be $R^*(s) = \mathbf{w}^{*T}\phi(s)$. We also make the common assumption that $\forall \mathbf{w} \in \mathcal{R}, \|\mathbf{w}\|_1 \leq 1$; so that $\mathcal{R}$ is bounded [Abbeel and Ng, 2004; Brown and Niekum, 2018].

Now, consider a set of randomly generated pairwise ranked GaE explanations, $\mathcal{C}_{GaE}^{\mathcal{X}_P}$:

$$\{(\mathcal{X}_{R_1}(s_1) \succ \mathcal{X}_{R_2}(s_1)), \ldots, (\mathcal{X}_{R_1}(s_k) \succ \mathcal{X}_{R_2}(s_k))\}$$

By Definition 4, all w in the explanation-consistent reward set, $\Delta(\mathcal{X}^{\mathcal{C}_{GaE}^{\mathcal{X}_P}})$, must satisfy the following half-space constraints $\forall(\mathcal{X}_{R_1}(s_i) \succ \mathcal{X}_{R_2}(s_i)) \in \mathcal{C}_{GaE}^{\mathcal{X}_P}$:

$$\mathbf{w}^T(\hat{\mathcal{X}}_{R_1}(s_i) - \hat{\mathcal{X}}_{R_2}(s_i)) > 0$$

where $\hat{\mathcal{X}}_{R_j}(s_i)$ is normalized vector of $\mathcal{X}_{R_j}(s_i)$. We want to find the bound k on the size of $\mathcal{C}_{GaE}^{\mathcal{X}_P}$ that is sufficient for reducing the size of $\mathcal{R}$ by $x\%$. According to [Brown *et al.*, 2019a], $|\Delta(\mathcal{C}_{GaE}^{\mathcal{X}_P})| = \frac{|\mathcal{R}|}{2^k}$ in expectation where $|\mathcal{C}_{GaE}^{\mathcal{X}_P}| = k$. Therefore, $x = (1 - \frac{1}{2^k}) * 100\%$ volume of $\mathcal{R}$ is removed in expectation using feedback over a set of $k = \log_2(1/(1 - x/100))$ randomly generated GaE explanation pairs. □

Remark: that the proof of Proposition 3 relies on GaE explanations being random vectors in $\mathbb{R}^n$, which may not hold in practice.

We now consider the efficiency of ILV when using SM-based explanations.

Proposition 4. *It is sufficient to have 1 Oracle-generated SM Feedback to reduce $|\Delta(\mathcal{X}^{\mathcal{C}_{SM}^{\mathcal{X}_P}})| \leq 2^d$, $d = \dim(\mathbf{w})$, $\mathcal{C}_{SM}^{\mathcal{X}_O}$ is the set of Oracle-generated SM feedback for S_V.*

Proof Sketch. Let's consider the same definition of $\mathcal{R}$ and R^* as Proposition 3. Now, $\forall s \in S$, the explanation given by the Oracle is $\mathcal{F}_O(\mathcal{X}_R(s)) = abs(\mathbf{w}^*)$. By Definition 4, the explanation-consistent reward set, $\Delta(\mathcal{X}^{\mathcal{C}_{SM}^{\mathcal{X}_P}}) = \{\mathbf{w} \in \mathcal{R} : abs(\mathbf{w}) = abs(\mathbf{w}^*)\}$. Then we can construct a $\Delta(\mathcal{X}^{\mathcal{C}_{SM}^{\mathcal{X}_P}})$ of size at most 2^d, by taking $+, -$ sign combination of each element of $\mathbf{w}^*$. Therefore, $|\Delta(\mathcal{X}^{\mathcal{C}_{SM}^{\mathcal{X}_P}})| \leq 2^d$. □

An implication of Proposition 4 is that ILV might fail to find an optimal solution when using SM-based explanations. Therefore, based on Propositions 2 and 4 we can infer that GaE can be more effective than SM in reducing reward ambiguity under certain conditions. Our empirical results show a similar trend for both linear and non-linear rewards.

4.2 Non-Linear Reward Alignment

We now define $P(\mathcal{C}|R)$ for non-linear rewards that are represented by a neural network parameterized by θ, denoted by R_θ. While we can still use the posterior distribution we derived in the previous section and use MCMC optimization to retrieve the MAP reward function; however, getting a good

estimation can be computationally expensive. Therefore, we focus on retrieving the maximum likelihood reward function by minimizing the following loss:

$$\mathcal{L}(\mathcal{D}, \mathcal{C}, \theta) = -\log(P(\mathcal{D}, \mathcal{C}|R_\theta)) \tag{6}$$

We decompose the loss from Equation 6 into two separate components: data likelihood loss $\mathcal{L}_\mathcal{D}(\mathcal{D}, \theta)$ and explanation loss $\mathcal{L}_E(\mathcal{C}, \theta)$, corresponding to $-\log(P(\mathcal{D}|R_\theta))$ and $-\log(P(\mathcal{C}|R_\theta))$ respectively. The data likelihood loss is defined by TREX [Brown *et al.*, 2019b] as follows:

$$\mathcal{L}_\mathcal{D}(\mathcal{D}, \theta) = \sum_{(\tau_i^1 \succ \tau_i^2) \in \mathcal{D}} -\log \frac{e^{\beta R_\theta(\tau_i^1)}}{e^{\beta R_\theta(\tau_i^1)} + e^{\beta R_\theta(\tau_i^2)}} \tag{7}$$

For oracle feedback, a natural choice is minimizing the distance function, analogous to Equation 3. The explanation loss is defined as follows:

$$\mathcal{L}_E(\mathcal{C}, \theta) = \sum_{\mathcal{X}_O(s_i) \in \mathcal{C}} D(\mathcal{X}_O(s_i), \mathcal{X}_{R_\theta}(s_i)). \tag{8}$$

For ranked feedback, we want the explanation $\mathcal{X}_{R_\theta}$ to be more similar to the higher-ranked explanation. We achieve this by adopting a constructive representation learning loss. More specifically, using generalized triplet loss [Sohn, 2016]:

$$\mathcal{L}_E(\mathcal{C}, \theta) = \sum_{(\mathcal{X}_{R_1}(s_i) \succ \mathcal{X}_{R_2}(s_i)) \in \mathcal{C}} -\log \frac{e^{-\alpha D_1}}{e^{-\alpha D_1} + e^{-\alpha D_2}}, \tag{9}$$

$D_1 = D(\mathcal{X}_{R_\theta}(s_i), \mathcal{X}_{R_1}(s_i))$ and $D_2 = D(\mathcal{X}_{R_\theta}(s_i), \mathcal{X}_{R_2}(s_i))$ respectively and $\alpha \in [0, \infty)$. Finally, we optimize both loss functions together:

$$\mathcal{L}_\mathcal{D}(\mathcal{D}, \theta) + \lambda \mathcal{L}_E(\mathcal{C}, \theta). \tag{10}$$

5 Experimental Setup

We evaluate the effectiveness of learning aligned linear and non-linear rewards with REVEALE using three explanation generation techniques: gradient as explanations (**GaE**), **LIME**, and saliency map (**SM**). We evaluate the performance in five proof-of-concept domains. Training data is generated by sub-optimally solving a set of training instances, and the learned reward is evaluated on the test instances. Test instances differ from training instances in terms of start state distribution and location of risky regions. Reward learning is challenging due to two factors: (1) the trajectories only cover a subset of the states in the environment, and (2) the presence of *spurious feature correlation*, where two or more state features always co-occur in the trajectory.

Metrics and benchmarks. Reward alignment is measured using (1) the accuracy of predicting the trajectory ranking in test instances, (2) the quality of reward estimation in states unseen during training, and (3) the average reward achieved by executing a policy computed using the learned reward in the test instances. The results of our approach are compared with (1) the policy computed using the true reward function (Optimal), and (2) two recent IRL algorithms: BREX [Brown *et al.*, 2020] (for linear rewards) and TREX [Brown *et al.*, 2019b] (for non-linear rewards). Hence, we use **REX** to refer to either BREX or TREX, depending on the problem and (non-)linearity of the reward.

Implementation details. Verification states S_V are selected from $\mathcal{D}$. We implemented all algorithms in Python and tested them on an Ubuntu machine with 32GB RAM and 12GB GPU. The reported values are averaged over 60 different random seeds. It is important to note that the generated dataset varies for each seed. For non-linear rewards, we utilized a 4-layer neural network with Relu activation. Below, we provide a brief description of the domains used for evaluation.

WaterWorld. This domain, based on [Leike *et al.*, 2017], tests the agent's response to a distribution shift. There are two types of surfaces in the problem: 'water' and 'ground'. We consider a linear reward function, with a negative reward for stepping into the water. The water locations are fixed in the training environment but are scattered in the test environments. Therefore, this scenario exhibits spurious feature correlation, where water locations are correlated with fixed grid positions in the dataset. Reward ambiguity arises as the agent may struggle to differentiate whether the negative reward is associated with the surface type or the grid location.

DogWalk. In this domain, the AV needs to learn to stop for both pedestrians and dogs (Figure 1). Each state is represented by ⟨location, human, dog, human+dog, harmless objects⟩. We adopt a linear reward structure for this environment.

LavaLand. This domain was introduced in [Hadfield-Menell *et al.*, 2017] and consists of a grid with different terrain types, each represented by binary indicators. In the training data, the lava indicator is always false. Therefore, it is possible to construct a reward function that produces the correct reward estimation for all the training states without considering the lava indicator. As a result, the agent may not learn to avoid lava when it navigates to a goal location, potentially resulting in unsafe behavior when deployed.

Navigation (AVNav). This domain, designed by us, represents a safe route planning problem where trajectories are ranked based on different routes. Each state corresponds to a road segment and is characterized by ⟨current goal, average speed, #potholes, mobile network quality, and accident history in the segment⟩. The non-linear reward function incentivizes the AV to select routes that are safe (with low potholes and accident history) and comfortable (with good mobile network quality) while reaching the destination. However, the trajectory data exhibits spurious feature correlation as roads with good mobile network quality also tend to have a bad accident history.

CoinRush. This domain is similar to the CoinRun environment described in [Langosco *et al.*, 2022]. The cells in the grid have different types of coins and enemies. The target is to gather as many coins as possible while avoiding enemies. However, in the trajectory data, the enemies and coins always have fixed colors, resulting in spurious feature correlation. In the test environment, however, their colors can be different.

In WaterWorld and CoinRush, the ambiguity revolves around which features should receive attribution. In the other domains, the ambiguity lies in determining which features to attribute and whether the attribution should be positive or

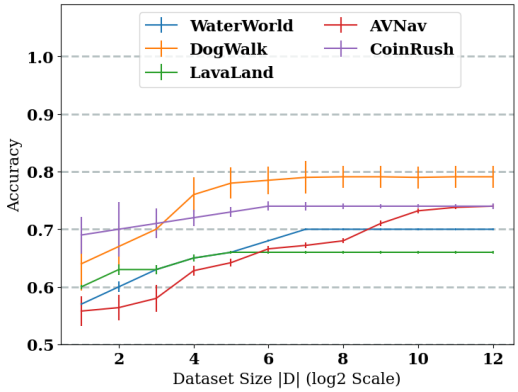


Figure 2: Effect of dataset size $|\mathcal{D}|$ on the accuracy of pairwise trajectory rank prediction by REX.

negative. AVNav and CoinRush feature a non-linear reward structure, while the other problems employ linear rewards.

6 Results and Discussion

Effect of the trajectory dataset size. We examine whether increasing $|\mathcal{D}|$ improves the prediction accuracy of REX by varying $|\mathcal{D}|$ from 2 to 2048 (Figure 2) when using BREX and TREX. We observe no improvement in the accuracy of BREX beyond 128 trajectories and 1024 for TREX. This is because the additional trajectories fail to provide information about novel situations that the agent may encounter during deployment. Thus, increasing the #trajectories *does not* guarantee improved prediction accuracy. This result emphasizes the importance of incorporating *alternative input forms*, such as feedback on explanations, instead of solely increasing $|\mathcal{D}|$, for learning aligned reward functions.

Effect of feedback size on accuracy. We compare the accuracy by varying the feedback size from 2 to 512, with $\mathcal{D}$ size of 128 and 1024 for linear and non-linear reward respectively. We observe that 64 explanations for linear reward and 256 for non-linear reward help reach maximal accuracy. Based on these two results, for all the subsequent experiments, we use 128 trajectories, 64 ranked feedback over pairs of explanations, and cosine similarity for domains with linear rewards. For non-linear rewards we use, 1024 trajectories, 256 ranked feedback over pairs of explanations, and L2 distance.

Prediction accuracy. Figure 3 shows the average accuracy of ranking prediction tested on 2000 pairs of trajectories from test environments. Oracle-generated explanations are denoted by $\mathcal{X}_O$ and pairwise rankings over explanations are denoted by $\mathcal{X}_P$. In every domain, except CoinRush, REVEALE with ranked GaE explanations achieves the highest accuracy and matches the accuracy of prediction based on Oracle-generated explanations. In LavaLand and DogWalk, SM identified 'lava' and 'dogs' as important features, respectively, but could not identify whether they should be positively attributed because it uses the absolute value of the gradient. BREX also suffers from this drawback. In domains where the ambiguity is about which features should be attributed, such as location or surface type in WaterWorld, SM

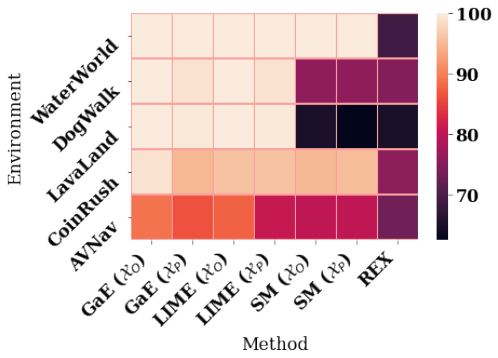


Figure 3: Accuracy of pairwise trajectory rank prediction using different techniques.

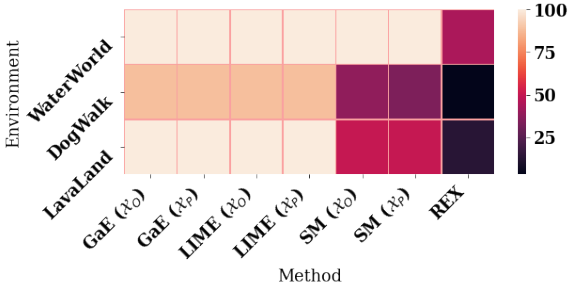


Figure 4: Frequency (%) of avoiding risky regions during execution, using the learned reward.

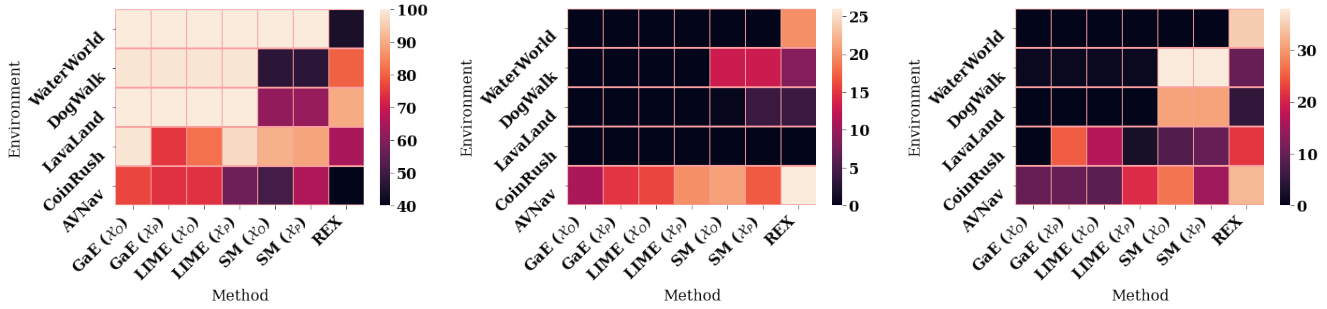
performs comparably to other approaches. However, BREX often associated the reward with location, instead of surface type. Overall, our results indicate that REVEALE with any explanation improves the performance of traditional IRL methods such as REX.

While all the approaches, including REX, achieve near-optimal prediction accuracy on the trajectory dataset used to learn the reward, the performance degrades in the test scenarios. This is because (1) the agent encounters novel states and doesn't know how to estimate rewards, or (2) the agent learns spurious correlation and estimates the wrong reward. As evident from Figure 3, REVEALE improves the prediction performance in such cases by a large margin. In the absence of prior knowledge about novel situations, it may be challenging to predict the agent's performance solely by assessing its reward, policy, or value function in states present in the training instances. However, by examining the consistency of the agent's explanations in states that appear in the training data, the tester can infer its behavior in many novel situations and provide valuable feedback. The agent then utilizes this feedback to reduce reward ambiguity.

Reward estimation. We examine how novel states are ranked under the learned reward compared to the intended reward. Based on the ground truth, we divide the state space into states that are ranked *above median reward value* and *below median reward value*. Then we rank them under the learned reward and present ranking differences in terms of overestimation, underestimation, and proportional estimation in Figure 5. We observe significant improvement in reward

Models	WaterWorld	DogWalk	LavaLand	AVNav	CoinRush
REX	-6.33 ± 0.42 [-7.1]	-4.19 ± 0.12 [-6.8]	-7.72 ± 0.08 [-7.8]	-6.20 ± 0.31 [-24.0]	00.00 ± 0.0 [00.0]
GaE ($\mathcal{X}_O$)	2.00 ± 0.00 [2.0]	-2.01 ± 0.01 [-2.1]	-2.50 ± 0.00 [-2.5]	$-4.08 \pm 0.0.2$ [-4.5]	29.34 ± 0.00 [29.3]
GaE ($\mathcal{X}_P$)	2.00 ± 0.00 [2.0]	-2.01 ± 0.01 [-2.1]	-2.50 ± 0.00 [-2.5]	-4.41 ± 0.07 [-4.7]	29.11 ± 0.00 [17.7]
SM ($\mathcal{X}_O$)	2.00 ± 0.00 [2.0]	-15.6 ± 0.68 [-23.0]	-5.50 ± 0.00 [-5.5]	-4.94 ± 0.04 [-5.7]	28.34 ± 0.00 [17.7]
SM ($\mathcal{X}_P$)	2.00 ± 0.00 [2.0]	-15.6 ± 0.68 [-23.0]	-5.50 ± 0.00 [-5.5]	-4.51 ± 0.02 [-4.8]	28.57 ± 0.01 [17.0]
LIME ($\mathcal{X}_O$)	2.00 ± 0.00 [2.0]	-2.01 ± 0.01 [-2.1]	-2.50 ± 0.00 [-2.5]	-4.29 ± 0.03 [-4.8]	29.34 ± 0.00 [29.3]
LIME ($\mathcal{X}_P$)	2.00 ± 0.00 [2.0]	-2.01 ± 0.01 [-2.1]	-2.50 ± 0.00 [-2.5]	-5.07 ± 0.02 [-5.5]	29.34 ± 0.00 [29.3]
Optimal	**2.00**	**−2.00**	**−2.50**	**−3.58**	**29.34**

Table 1: Average reward $\pm$ standard error and worst case reward (shown in brackets [.]) achieved by executing policies with the learned reward functions.



(a) Proportional Estimation (b) Overestimation (c) Underestimation

Figure 5: Quality of reward estimation in novel states.

estimation using REVEALE. While REX often over or underestimates the reward in novel states, explanation-guided alignment of reward results in a more proportional estimation.

Average and worse case reward. Table 1 shows the average and worst reward obtained with different approaches in test environments. We report the worst-case reward since it provides insights into the degree of unsafe behavior that may arise when the reward is not well-aligned. We also report the average reward obtained with the true reward function in each setting, denoted by **Optimal**. Our results show that REVEALE with GaE using $\mathcal{X}_P$ feedback performs better on most domains. SM often identifies the magnitude of correlation but struggles to determine whether it is positive or negative. LIME performs similarly to GaE when feedback is $\mathcal{X}_O$, but performs relatively poorly when feedback is $\mathcal{X}_P$. This is because LIME works with a large set of states in the neighborhood of the input states, unlike GaE and SM, which only work with a single state. Therefore, when exact inputs are given, LIME works very well. With $\mathcal{X}_P$ feedback, the error can propagate to many states, causing worse performance than GaE. Overall, our results show that REVEALE can learn reward functions that are better aligned compared to existing approaches.

Avoiding risky states. Aggregate metrics such as average reward may hide important shortcomings of the techniques. Therefore, we evaluate the efficiency of different techniques in avoiding risky states, when operating based on learned reward. In LavaLand, DogWalk, and WaterWorld domains, the agent gets a large penalty for encountering risks such as a state with lava, humans or dogs, and water, respectively. Figure 4 shows the percentage of times the agent manages to avoid these states when executing the policy computed using the learned reward with each technique in the test environments. We observe that GaE and LIME-based methods produce safer trajectories in WaterWorld and LavaLand domains, and encounter relatively fewer risks (11%) in the DogWalk domain when using ranked feedback. On the other hand, SM produces a safe policy only in the WaterWorld domain. REX fails in all three domains.

7 Summary and Future Work

We present the REVEALE framework, an interpretable approach for reward alignment and verification. Additionally, we introduce an algorithm called ILV that can be utilized to solve a REVEALE instance and infer more aligned reward functions. Our empirical results on five domains demonstrate that learning with REVEALE generalizes well and achieves higher prediction accuracy and average reward, often matching optimal performance. These results highlight the benefits of our approach in learning the intended reward function, thereby supporting a safer deployment of autonomous agents in the open world. In the future, we aim to develop techniques to automatically identify critical states for verification and integrate REVEALE with active learning methods [Settles, 2012; Wray and Zilberstein, 2016] in order to minimize the number of required feedback queries. Investigating the efficiency of other explanation methods, particularly causal explanations [Nashed *et al.*, 2022], is another future direction.

Acknowledgments

This work was supported in part by the National Science Foundation grant IIS-2205153.

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine learning*, 2004.

[Alqaraawi *et al.*, 2020] Ahmed Alqaraawi, M. Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Bianchi-Berthouze. Evaluating saliency map explanations for convolutional neural networks: A user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 275–285, 2020.

[Amir *et al.*, 2019] Ofra Amir, Finale Doshi-Velez, and David Sarne. Summarizing agent strategies. *Autonomous Agents and Multi-Agent Systems*, 33:628–644, 2019.

[Argall *et al.*, 2009] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[Basu *et al.*, 2018] Chandrayee Basu, Mukesh Singhal, and Anca D. Dragan. Learning from richer human guidance: Augmenting comparison-based learning with feature queries. In *Proceedings of the 13th International Conference on Human-Robot Interaction*, pages 132–140, 2018.

[Brown and Niekum, 2018] Daniel S. Brown and Scott Niekum. Efficient Probabilistic Performance Bounds for Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[Brown *et al.*, 2019a] Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automaticaly-ranked demonstrations. In *Conference on Robot Learning*, 2019.

[Brown *et al.*, 2019b] Daniel S. Brown, Wonjoon Goo, Nagarajan Prabhat, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the 36th International Conference on Machine Learning*, pages 783–792, 2019.

[Brown *et al.*, 2020] Daniel S. Brown, Scott Niekum, Russell Coleman, and Ravi Srinivasan. Safe imitation learning via fast bayesian reward inference from preferences. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1165–1177, 2020.

[Brown *et al.*, 2021] Daniel S. Brown, Jordan Jack Schneider, and Scott Niekum. Value alignment verification. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1105–1115, 2021.

[Chakraborti *et al.*, 2017] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. *arXiv preprint arXiv:1701.08317*, 2017.

[Dietterich, 2017] Thomas G. Dietterich. Steps toward robust artificial intelligence. *AI Magazine*, 38(3):3–24, 2017.

[Guan *et al.*, 2021] Lin Guan, Mudit Verma, Sihang Guo, Ruohan Zhang, and Subbarao Kambhampati. Widening the pipeline in human-guided reinforcement learning with explanation and context-aware data augmentation. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 21885–21897, 2021.

[Hadfield-Menell *et al.*, 2017] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J. Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, 2017.

[Huang *et al.*, 2018] Sandy H. Huang, Kush Bhatia, Pieter Abbeel, and Anca D. Dragan. Establishing appropriate trust via critical states. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3929–3936, 2018.

[Langosco *et al.*, 2022] Lauro Langosco, Jack Koch, Lee Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 12004–12019, 2022.

[Leike *et al.*, 2017] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds. *ArXiv*, abs/1711.09883, 2017.

[Molnar, 2022] Christoph Molnar. *Interpretable Machine Learning: A Guide For Making Black Box Models Explainable*. Lulu.com, 2022.

[Nashed *et al.*, 2022] Samer B. Nashed, Saaduddin Mahmud, Claudia V. Goldman, and Shlomo Zilberstein. A unifying framework for causal explanation of sequential decision making. *ArXiv*, abs/2205.15462, 2022.

[Ng and Russell, 2000] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 663–670, 2000.

[Palan *et al.*, 2019] Malayandi Palan, Nicholas C. Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions by integrating human demonstrations and preferences. In *Proceedings of Robotics: Science and Systems*, June 2019.

[Ramachandran and Amir, 2007] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, pages 2586–2591, 2007.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[Saisubramanian *et al.*, 2022] Sandhya Saisubramanian, Shlomo Zilberstein, and Ece Kamar. Avoiding negative side effects due to incomplete knowledge of AI systems. *AI Magazine*, 42(4):62–71, 2022.

[Settles, 2012] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[Simonyan *et al.*, 2014] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[Sohn, 2016] Kihyuk Sohn. Improved deep metric learning with multi-class N-pair loss objective. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 1849–1857, 2016.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Tabrez *et al.*, 2020] Aaquib Tabrez, Shivendra Agrawal, and Bradley Hayes. Explanation-based reward coaching to improve human performance via reinforcement learning. In *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction*, pages 249–257, 2020.

[Tayyub *et al.*, 2022] Jawad Tayyub, Muhammad Sarmad, and Nicolas Schönborn. Explaining deep neural networks for point clouds using gradient-based visualisations. *arXiv preprint arXiv:2207.12984*, 2022.

[Wray and Zilberstein, 2016] Kyle Hollins Wray and Shlomo Zilberstein. A POMDP formulation of proactive learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 3202–3208, 2016.

[Ziebart *et al.*, 2008] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

[Zilberstein, 2015] Shlomo Zilberstein. Building strong semi-autonomous systems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 4088–4092, 2015.