**MORAL: Aligning AI with Human Norms through Multi-Objective Reinforced Active Learning**

Markus Peschl  
Delft University of Technology  
Delft, The Netherlands  
peschl@protonmail.com

Frans A. Oliehoek  
Delft University of Technology  
Delft, The Netherlands  
F.A.Oliehoek@tudelft.nl

Arkady Zgonnikov  
Delft University of Technology  
Delft, The Netherlands  
A.Zgonnikov@tudelft.nl

Luciano C. Siebert  
Delft University of Technology  
Delft, The Netherlands  
L.CavalcanteSiebert@tudelft.nl

**ABSTRACT**

Inferring reward functions from demonstrations and pairwise preferences are auspicious approaches for aligning Reinforcement Learning (RL) agents with human intentions. However, state-of-the-art methods typically focus on learning a single reward model, thus rendering it difficult to trade off different reward functions from multiple experts. We propose Multi-Objective Reinforced Active Learning (MORAL), a novel method for combining diverse demonstrations of social norms into a Pareto-optimal policy. Through maintaining a distribution over scalarization weights, our approach is able to interactively tune a deep RL agent towards a variety of preferences, while eliminating the need for computing multiple policies. We empirically demonstrate the effectiveness of MORAL in two scenarios, which model a delivery and an emergency task, respectively. We show that our approach directly approximates desired behavior. We demonstrate that MORAL efficiently captures specifications from observational data instead of manually specifying them [22, 30, 40]. However, such technical approaches cannot on their own solve the normative alignment problem of deciding which values should ultimately be encoded into an agent [15]. Nonetheless, building methods that allow for learning and trading off different conflicting values could potentially alleviate this issue, thus making such methods an important avenue of research for beneficial artificial intelligence (AI) [35].

**KEYWORDS**

Active Learning; Inverse Reinforcement Learning; Multi-Objective Decision-Making; Value Alignment

**1 INTRODUCTION**

The design of adequate reward functions poses a tremendous challenge for building reinforcement learning (RL) agents that ought to act in accordance with human intentions [4, 13]. Besides complicating the deployment of RL in the real world [11], this can lead to major unforeseen societal impacts, which need to be accounted for when building autonomous systems [6, 45]. To tackle this, the field of value alignment has largely focused on reward learning, which aims to adopt a bottom-up approach of finding goal specifications from observational data instead of manually specifying them [22, 30, 40]. However, such technical approaches cannot on their own solve the normative alignment problem of deciding which values should ultimately be encoded into an agent [15]. Nonetheless, building methods that allow for learning and trading off different conflicting values could potentially alleviate this issue, thus making such methods an important avenue of research for beneficial artificial intelligence (AI) [35].

Jointly optimizing for different rewards can be cast into multi-objective RL (MORL) [32], which constitutes a promising framework for building human-aligned AI [42]. Using game-theoretic notions of optimality, MORL typically aims to find a solution, or a set thereof, that can represent a variety of preferences over the components of a vector-valued reward function. While this can theoretically tackle the overoptimization of narrowly defined tasks, the designer still needs to manually specify multiple reward functions.

Inverse RL (IRL) [16, 51] and preference-based RL [10, 46] offer techniques for avoiding the reward design problem altogether by learning a parametric reward model from demonstrations and pairwise preferences, respectively. In this paper, we combine these approaches in a multi-objective setting with a focus on learning human norms. The motivation for this is twofold: Firstly, previous research on learning multiple reward functions has mostly employed latent-variable IRL models for finding multiagent [19], hierarchical [41, 43] and multitask [17, 50] rewards, whereas finding aggregated rewards from conflicting sequential data has yet to be addressed. Secondly, a major challenge of value alignment is ensuring that an agent can predict its adherence to norms in the environment. Such adherence is implicitly embedded in human goal specifications but missing in manually engineered reward functions [20]. Our goal therefore is to find a policy that acts on a common set of social norms while allowing for fine-tuning the agent with respect to inherent disagreements that may arise.

**Contributions**. We propose Multi-Objective Reinforced Active Learning (MORAL), a method that combines active preference learning and IRL to interactively learn a policy of social norms from expert demonstrations. MORAL first finds a vector-valued reward function through adversarial IRL, which is subsequently used in an interactive MORL loop. By requesting pairwise preferences over trajectories of on-policy experience from an expert, MORAL learns a probability distribution over linear combinations of reward functions under which the optimal policy most closely matches the desired behavior. We show that our approach directly approximates a Pareto-optimal solution in the space of expert reward functions, without the need of enumerating through a multitude of preference weights. Finally, we demonstrate that MORAL efficiently captures norms in two gridworld scenarios, while being able to adapt the agent’s behavior to a variety of preferences.1

---

1Source code is available at https://github.com/mlpeschl/moral_rl.
2 PRELIMINARIES

Multi-Objective RL. We employ a multi-objective Markov decision process (MOMDP) for framing the problem of aligning an agent with multiple experts. Formally, a MOMDP is given by the tuple \((S, A, r, \rho, \mu_0, \gamma)\), with state space \(S\), the set of actions \(A\), a transition distribution \(p'(s|a)\), a vector-valued reward function \(r(s, a) \in \mathbb{R}^m\), a starting state distribution \(\mu_0\) and the discount factor \(\gamma \in [0, 1]\). We consider optimal solutions to be given by a Pareto frontier \(F = \{\pi \in \mathbb{R}^m \mid J_f(\pi') \geq J_f(\pi), \forall \pi' \in F\}\), where \(J_f(\pi) = \mathbb{E}_r(\sum_{t=0}^{T} \gamma^t r(s_t, a_t))\) is the vector-valued return of a policy \(\pi : S \rightarrow \Delta_A\) that maps states to a distribution over actions. Furthermore, we define the convex coverage set (CCS) \(F^* = \{\pi \in F \mid \exists \pi' \in F : \pi'(s_t, a_t) \geq \pi(s_t, a_t), \forall \pi' \in F\}\) to be the subset of Pareto-optimal solutions that can be obtained through optimizing for linear combinations of the different reward components.

Proximal Policy Optimization (PPO). Given a weight \(w \in \mathbb{R}^m\), we can optimize for policies on the CCS using PPO [39] on the scalarized reward \(r(s, a) = \exp(w^T r(s, a))\). Using on-policy experience, PPO maximizes the return of a parameterized policy \(\pi_\theta\) by performing gradient descent on the clipped objective

\[
\mathcal{L}^{\text{clip}}(\phi) = \mathbb{E}_t \left[ \min(r_t(\phi) \tilde{A}_t, \text{clip}(r_t(\phi), 1 - \epsilon, 1 + \epsilon) \tilde{A}_t) \right],
\]

where \(\mathbb{E}_t\) is the expectation at time \(t\), \(\tilde{A}_t\) is a ratio of the new versus the current policy, \(\tilde{A}_t\) is an estimated advantage at time \(t\) and \(\text{clip}(x, a, b)\) limits the value of \(x\) to the interval \([a, b]\).

Adversarial IRL (AIRL). The maximum entropy IRL [51] goal is to derive a reward function \(r_\phi\) from a demonstration dataset \(\mathcal{D} = \{(s_t, a_t)\}_{t=0}^T\) of expert trajectories \(\tau = (s_t, a_t)\) by solving a maximum likelihood problem \(\max_\phi \mathbb{E}_{\tau \in \mathcal{D}} [-\log p_\phi(\tau)]\),

\[
p_\phi(\tau) \propto \mu_0(s_0) \prod_{t=0}^{T} p(s_{t+1}|s_t, a_t) \exp(r_\phi(s_t, a_t)) = \mathcal{P}_\phi(\tau).
\]

AIRL [14] approximately solves the IRL problem using generative adversarial networks [18]. It jointly trains a policy (generator) \(\pi_\phi\) alongside a discriminator of the form

\[
D_\theta(s, a) = \frac{\exp(f_\theta(s, a))}{\exp(f_\theta(s, a)) + \pi_\phi(a|s)}.
\]

While \(D_\theta\) is trained using a binary cross-entropy loss to distinguish trajectories in \(\mathcal{D}\) from \(\pi_\phi\), the agent maximizes its returns using the reward \(r(s, a) = \log(D_\theta(s, a)) - \log(1 - D_\theta(s, a))\).

3 MULTI-OBJECTIVE REINFORCED ACTIVE LEARNING

MORAL uses a two-step procedure that separates reward and policy training to learn from multiple experts (figure 1). In the first step (IRL), we use a set of trajectories \(\mathcal{D}_E = \bigcup_{i=1}^k \mathcal{D}_{E_i}\) demonstrated by \(k\) distinct experts and run AIRL to obtain a vector of reward functions \(r = (f_{\theta_1}, \ldots, f_{\theta_k})\) and imitation policies \((\pi_{E_1}, \ldots, \pi_{E_k})\) for each subset of trajectories \(\mathcal{D}_{E_i}\). In step two (active MRL), we run an interactive MRL algorithm for learning a distribution over weights that determine a linear combination of the different components in \(r\). We will now explain how this distribution is learned alongside training a deep RL agent in a single loop. Firstly, active MRL takes a prior \(p(w)\) over scalarization weights and initializes a reward function \(r(s, a) = \mathbb{E}_w[w^T r(s, a)]\). Subsequently, we repeatedly (i) optimize for the scalar reward \(r\) by running PPO for a given number of steps, (ii) query an expert for a pairwise comparison \(q_n\) of two trajectories and (iii) update the posterior \(p(w|q_1, \ldots, q_n)\). Finally, the reward function is reinitialized to the posterior scalarization and is used by PPO in the next iteration.

To query and update, we specify a probabilistic model over expert preferences. We employ a Bradley-Terry model [7]

\[
p(w|q_1, \ldots, q_n) \propto p(w) \prod_{i=1}^n p(q_i|w).
\]

In our experiments, we choose the prior \(p(w)\) to be uniform over all weights \(w\) with \(||w|| \leq 1\) and \(w \geq 0\).

This Bayesian model allows us to maintain a posterior that determines which reward components should be prioritized at each iteration. By providing pairwise preferences, an expert is then able to fine-tune the agent to any specific behavior that can theoretically result from the linear combination of reward components. Furthermore, we can leverage the uncertainty of the posterior to enable the agent to form queries the answers to which are highly informative. Namely, we use the active learning procedure which forms queries based on the amount of removed posterior volume [36]

\[
\max_{(\tau_i, \tau_j)} \min \left(\mathbb{E}_w[1 - p(\tau_i > \tau_j|w)], \mathbb{E}_w[1 - p(\tau_j > \tau_i|w)]\right),
\]

where the expectation over \(w\) is approximated using Markov Chain Monte Carlo (MCMC) [9].

Unfortunately, for sufficiently complex MOMDPs, maximizing this expression over all pairs of feasible trajectories proves to be computationally intractable. Instead, we do a discrete search over randomly sampled pairs of trajectories that arise during on-policy RL experience (algorithm 1). Before each policy improvement step, we sample pairs \((\tau_i, \tau_j)\) and evaluate the corresponding minimum in expression (4). If \((\tau_i, \tau_j)\) scores highest among all previous pairs
obtained since the last posterior update, it is saved in a buffer and queued for the next query, unless a better pair is found later on.

Overall, this active learning scheme minimizes the number of queries needed during training to converge to a desired reward scalarization. Nonetheless, forming queries based on on-policy experience can only lead to locally optimal solutions. Therefore, assuming fixed weights \( \mathbf{w} \), we optimize for an entropy-regularized objective

\[
\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[ \sum_{t=0}^{T} \mathbf{w}^T \mathbf{r}(s_t, a_t) - \log \pi(s_t, a_t) \right].
\]

This way, MORAL can be interpreted as finding an average of Kullback-Leibler (KL) divergences [27] between the policy distribution over trajectories \( \pi(r) = \mu_0(\mathbf{s}_0) \prod_{t=0}^{T-1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) \pi(a_t|\mathbf{s}_t) \) and the marginal maximum entropy IRL distributions (1). Theorem 3.1. Given \( \mathbf{w} \in \mathbb{R}^k \) with \( \mathbf{w} \geq 0 \), \( \sum_{i=1}^{k} w_i = 1 \), we have that

\[
\pi^* = \arg\min_{\pi} \sum_{i=1}^{k} w_i D_{KL}(\pi(r)||p_{\theta_i}(r)).
\]

Proof: We provide a proof in the appendix.

Theorem 3.1 assumes that all components in \( r \) arise from maximum entropy IRL. However, in practical applications one might want to encode additional prior knowledge into the agent’s behavior through a manually engineered primary reward function \( r_p \). Nonetheless, by applying analogous reasoning, expression (5) can be interpreted as maximizing the returns \( \mathbb{E}_{\pi} \left[ \sum_{t=0}^{T} r_p(s_t, a_t) \right] \) with a KL regularizer in the form of expression (6). Under this interpretation, MORAL interactively finds regularization hyperparameters that determine which expert’s behavior should be prioritized at runtime.

### 3.1 Reward Normalization

Finding scalarization weights in the presence of reward functions with highly different scales is a challenging task for many MORL algorithms. MORAL, on the other hand, learns its weights from preferences, thus making it less susceptible to reward functions that are difficult to compare. Nevertheless, the scale of the reward indirectly impacts the sensitivity of the posterior, since the magnitude of the likelihood (2) depends on the scalar products of the form \( \mathbf{w}^T \mathbf{r}(\tau) \). Although \( \mathbf{w} \) is bounded by the prior, its product with the vector-valued reward \( \mathbf{r}(\tau) \) can become arbitrarily large, which introduces a risk of removing significant parts of the posterior support based on a single query. To tackle this, we utilize the policies obtained from AIRL to normalize each reward component by setting

\[
\bar{f}_{\theta_i}(s, a) = \frac{f_{\theta_i}(s, a)}{J(\pi^*_E)},
\]

where \( J(\pi^*_E) = \mathbb{E}_{\pi^*_E} \left[ \sum_{t=0}^{T} \mathbf{w}^T \mathbf{r}(s_t, a_t) \right] \) is the scalar return of \( \pi^*_E \). This does not introduce any computational overhead, and we simply estimate \( J(\pi^*_E) \) by taking the average obtained return with respect to \( f_{\theta_i} \) after running AIRL.

### 4 EXPERIMENTS

In the following, we will demonstrate the ability of MORAL in simulation studies of two gridworld environments. To enable a qualitative analysis of the method, we assume that in both environments, a ground truth reward function exists, which is used to generate demonstrations and responses to the agent’s queries. Furthermore, by following the experimental setup of related research [30, 47], we consider environments with a primary reward function \( r_p \), encoding a generic task that can easily be solved through deep RL. In this case, we can apply MORAL as before, but add \( r_p \) as an additional reward component to the AIRL reward functions for the active learning step. To form the gridworld state, we make a binary array \( I \in \{0, 1\}^{W \times H} \) of width \( W \) and height \( H \), as well as channels that encode grid occupancy for all \( C \) different object types on the grid. Finally, we employ a convolutional neural network architecture for PPO, consisting of two base convolutional layers with kernel size 3, and 64 as well as 256 output channels respectively. Its activations are then fed into two separate convolutional layers with kernel size 2 and 32 output channels each, followed by a linear layer for the critic and actor heads. The details of our implementation are provided in the appendix.

#### 4.1 Emergency

We start by illustrating how MORAL can be applied to incorporating social norms from a single expert alongside a primary goal. We define the Emergency gridworld as follows: An agent, as well as 6 humans are randomly positioned onto a 6 \( \times \) 6 grid. The humans are lost and need to be escorted before a time limit of \( T = 75 \). Furthermore, the bottom right corner contains a goal state (e.g. a fire extinguisher in a burning room), which the agent uses when standing on its cell. At each step, the agent can either move in one of the four directions, interact with an adjacent cell or do nothing.

We define the agent’s primary goal \( r_p \) to give a reward of \(+1\) for each time step spent in the fire extinguisher cell. On the other hand, the social norm of helping people is not considered in \( r_p \). In order to learn the latter, we find a reward \( f_0 \) by running AIRL on 50 synthetic demonstrations coming from a PPO agent that
maximizes the number of people saved. Subsequently, we form a reward vector \( r = (r_\text{p}, f_j) \) and run interactive MORL using a total of 25 queries. Since we would like to incorporate the goal of saving all people into the primary task of extinguishing fire, we provide preferences in the following way: Given two trajectories \((\tau_i, \tau_j)\), we return \( i > j \) if the number of people saved in \( \tau_i \) exceeds that of \( \tau_j \). If both trajectories save the same number of people, we opt for the trajectory that spent more time in the extinguisher cell. Finally, queries are spread out evenly over \( 6 \times 10^6 \) environment steps.

Figure 2 shows the set of policies obtained during training of MORAL and compares it with a CCS found from a manual scalarization \( \lambda r_p + (1 - \lambda) f_0 \) for different choices of \( \lambda \in [0, 1] \). Also, we do not show solutions corresponding to higher values of \( \lambda \), since these collapse to a single trivial solution. To illustrate the evolution of solutions, we estimate average returns and plot a corresponding point before each update of the weight posterior. MORAL directly approximates a Pareto-optimal point that opts for saving all people present in the world, while maximizing the agent’s performance with respect to the primary goal. Furthermore, MORAL first learns to only save people, which correctly corresponds to the way preferences are provided. Thus, MORAL demonstrates to be successful at directly finding a normative policy while incorporating reward information from multiple sources. To ensure consistency across multiple runs, we also plot the average returns for different numbers of overall queries in figure 3. We see that although 25 queries are necessary to converge to a solution that closely matches the given preferences, MORAL learns a reasonable trade-off after 10 queries, which consistently saves all people at the cost of spending less time on the primary goal.

4.2 Delivery

While the Emergency domain illustrated the effectiveness of MORAL in a simplified setting, we yet have to analyze how MORAL performs in larger environments, as well as regarding increased diversity of norms and goals we would like an agent to learn. To better evaluate MORAL, we therefore define the Delivery environment, a randomly initialized 16 \times 16 grid world (figure 4). As before, the agent has access to the moving, interaction and null actions, but can now encounter a variety of objects. Its primary goal consists of delivering packages to 12 locations, which is available to the agent via a reward \( r_P \) of +1 whenever it interacts with a delivery cell. However, there also exist a multitude of people in need of support that the agent can help and “dirty” tiles that the agent can clean. The agent chooses to do so by interacting with each of the respective cells, after which they turn empty. Finally, we randomly place vases throughout the grid, which break whenever the agent steps on their position and can not be interacted with.

Overall, we limit the episode length to \( T = 50 \) time steps and place 12 of the help, clean objectives as well as 8 vases on the grid. We view this environment as a multi-objective problem including three norms, where help and clean are active behaviors, but the ability to avoid vases is passive i.e., an inaction. Besides forcing the agent to make trade-offs, this choice allows us to effectively study the quality of solutions found through MORAL, by introducing a symmetry with regard to the three explicit objectives. We assume that preferences are automatically provided by a subjective distribution \( m \in \Delta \{1, 2, 3\} \) encoding desired priorities for deliver, help and clean respectively. Given a pair of trajectories \((\tau_1, \tau_2)\), we then calculate two vectors \( s_i = (s_i^1, \ldots, s_i^6) \), where \( s_i^k \) denotes the obtained returns in terms of the \( k \)-th objective in trajectory \( i \). For example,
Aside from providing preferences in a principled way, we use this divergence measure to evaluate the overlap between a policy and the provided preferences throughout training.

We test MORAL using two conflicting demonstration data sets generated by a PPO agent optimizing for (i) helping people and (ii) cleaning tiles, while both try to avoid stepping on vases. As before, we subsequently use MORAL as a regularizer and form \( r = (rp, f_{\theta_1}, f_{\theta_2}) \), where \( \theta_1 \) and \( \theta_2 \) denote the trained AIRL parameters. As opposed to the experiment in the Emergency domain, there now exists an inherent normative conflict in the demonstrations. Thus, instead of tuning the agent to respect a specific policy that incorporates the normative component into the primary goal, we aim to test whether MORAL is able to retrieve solutions that match a variety of preferences. To achieve this, we vary the supplied preference vector \( m \) to match all possible ratios in \( \{1, 2, 3\}^3 \) during the active learning stage. Furthermore, we choose to use 25 queries overall, spread evenly throughout \( 8 \cdot 10^6 \) environment steps.

Figure 5 illustrates the found set of policies, where each point represents a separate run of active learning on different preferences. Since the objective space is three-dimensional, we only show two-dimensional projections and add the third objective through color (figure 5, top three panels). Besides this, the number of broken vases is shown by gray circles around each point, where a bigger radius indicates policies that break more vases and a radius of 0 indicates that no vases are broken on average. To test whether the found policies match the given preferences, we evaluate the KL divergence (8) of average returns \( s \) (generated by the learned policy) to the preference \( m \) that was used during training (figure 5, bottom three panels). We found that MORAL is overall able to retrieve a diverse set of policies, which accurately represent the different preferences: Firstly, the top three panels show mostly non-dominated policies that span a wide variety of trade-offs, which suggests that MORAL recovers a large part of the convex coverage set. Secondly, the bottom three panels indicate that most of the points achieve a near zero divergence. This means that the agent accurately matches the supplied ratios over objectives. As expected, we also see that the number of broken vases correlates with the weight put on the primary task, since the manually engineered delivery reward is entirely agnostic regarding the vase object. Nonetheless, for appropriate choices of \( m \), there exist policies which successfully avoid vases despite delivering an adequate number of packages. These results indicate that when choosing scalarization weights appropriately, minimizing a weighted sum of KL divergences to the respective maximum entropy IRL distributions can achieve implicit normative behaviors without the need of an explicit feedback signal.

To investigate the robustness of MORAL against adversarial preferences, we also trained MORAL on \( r = (f_{\theta_1}, f_{\theta_2}) \) by giving 25 preferences such that \( \tau_i \succ \tau_j \), whenever \( \tau_i \) manages to break more vases. We observe that despite this, the number of broken vases in fact decreases as a function of training steps. Since both experts agree on keeping vases intact, the aggregate reward function cannot be fine-tuned to exhibit the opposite behavior (figure 6). However, such a guarantee against adversarial preferences only holds when all marginal reward functions induce safe behavior. Under this assumption, this result provides evidence that automatically adhering to common implicit behaviors ensures safety against adversarial preference givers.
4.3 Comparison to Deep Reinforcement Learning from Human Preferences

Through its two-step procedure, MORAL is able to combine multiple reward functions from diverse expert behaviors. However, in the active learning stage, we require a single expert to determine which Pareto-optimal policy should ultimately be optimized for. Given enough pairwise comparisons, this directly approximates a policy that best matches the preferences (figure 2). For these reasons, among the related RL algorithms, MORAL is most directly comparable to deep reinforcement learning from human preferences (DRLHP) [10], which directly trains a deep reward model from pairwise preferences. To compare the two, we train DRLHP until convergence in Emergency and Delivery by providing a sufficient number of pairwise comparisons to make up for the missing primary reward and demonstrations that MORAL has access to.

Table 1 shows the results when providing DRLHP with 1000 preferences in the same way as MORAL for the Emergency domain. As before, trajectories with more people saved are preferred unless equal, in which case extinguishing fire becomes a priority. Although this leads DRLHP to learn a policy that consistently saves most people, it significantly lacks in terms of extinguished fire. This is unsurprising, since DRLHP is not designed to handle multi-objective problems and can not utilize the manually engineered reward signal in any meaningful way. This is because the deep reward model is nonstationary, which poses the combination with the stationary reward \( r_P \) to be challenging. As a result, DRLHP needs to maintain a single model for all competing objectives, which can lead to catastrophic forgetting of extinguishing fire when updating the reward network to save more people.

A similar trend can be observed in the Delivery environment, where we compare mean performance of DRLHP versus MORAL on three preference configurations, each of which prefers one of the objectives most strongly. However, since we assumed that avoidance of vases is encoded in the preferences only implicitly, we cannot supply DRLHP with the same set of feedback. Instead, we train DRLHP to prefer trajectories that have a lower mean squared error to the vector of expected returns achieved by MORAL. Figure 7 shows training curves of both methods, where each row represents a preference ratio of \((3, 1, 1)\), \((1, 3, 1)\) and \((1, 1, 3)\) respectively. Aiming to make the comparison fairer, we offset MORAL by the number of total training steps needed for IRL. As before, DRLHP manages to retrieve solutions that loosely resemble the supplied preferences, but fails to converge to Pareto-optimal policies. Furthermore, we notice that for the latter two preferences, sparse objectives such as minimizing the number of broken vases are not picked up by DRLHP. We suspect this to be an exploration issue, where trajectories that break fewer vases are unlikely to arise in queries. Thus, DRLHP optimizes for the remaining objectives as they lead to a higher increase in correctly predicting an expert’s preferences. Overall, we conclude that MORAL is more suitable than DRLHP in multi-objective settings that require trading off conflicting objectives from expert data. Nonetheless, MORAL has a theoretical advantage in this environment, since it allows for incorporation of prior knowledge as well as conflicting expert demonstrations.

### Table 1: Comparison of MORAL and DRLHP in Emergency.

|                  | People Saved | Extinguished Fire | Nr. of Queries | Steps (IRL)  |
|------------------|--------------|-------------------|----------------|--------------|
| MORAL            | 5.76 (±0.13) | 40.08 (±2.9)      | 25             | 3¢6 (3¢6)    |
| DRLHP            | 5.62 (±0.17) | 12.32 (±3.0)      | 1000           | 12¢6 (-)     |

Figure 6: Average number of broken vases over three training runs. MORAL learns a safe policy, despite being provided with adversarial preferences.

Figure 7: Mean training curves of DRLHP and MORAL on preference ratios \((3, 1, 1)\) (top), \((1, 3, 1)\) (middle) and \((1, 1, 3)\) (bottom).

4.4 Ablation

In this section, we evaluate MORAL with respect to the necessity of active queries, as well as its robustness against noisy preferences. To do so, we contrast active versus randomly chosen queries for the same set of preferences as in figure 5 and plot the average preference deviation (8) in figure 8 (a). In the case of actively generated queries, there is a clear decrease in preference deviation as a function of total queries. Random queries do not benefit from the increase in queries as much as MORAL, with 50 queries in fact scoring worse than 25. We conjecture that this is due to the on-policy sampling of trajectories, which strongly restricts the space of trajectories that the agent can query for. On the other hand, MORAL will, always seek pairs of trajectories that naturally exhibit larger variances between competing goals in order to generate queries with high
information content. When ensuring that the agent maintains adequate levels of exploration throughout training, this suffices to ensure a decrease in deviation even in the large query regime.

To investigate the robustness of MORAL in the presence of contradictory feedback, we train policies on the same set of preferences as before in figure 8 (b), but provide random answers to each of the 50 queries with a certain probability. Unsurprisingly, we see a sharp increase in deviation when injecting a noise level of 0.1, above which the growth in error diminishes. Nonetheless, active queries with a random answer probability of 0.3 still retrieve slightly more accurate representations than random queries without any noise. Such robustness with respect to noise is important, since our experiments only cover synthetic simulation studies, whereas human feedback is unlikely to be as consistent. Even though random noise is only an approximation of human error, we conclude from our results that seeking volume removal in the active learning loop does not make the algorithm more susceptible to converging to locally optimal scalarization weights in this case.

5 RELATED WORK

Machine Ethics Using the notion of uncertainty and partial observability, RL has been suggested as a framework for ethical decision-making [2]. We frame the problem of learning norms in a multi-objective context, which can be interpreted as inducing partial observability over the set of reward scalarizations one would wish to optimize. Overall, the motivation behind our approach is conceptually similar to policy orchestration [30] and ethics shaping [47] (table 2). Policy orchestration [30] also adopts a multi-objective view to incorporate ethical values into reward-driven RL agents, by solving a bandit problem that uses IRL to alternately play ethical and reward maximizing actions. Similarly to policy orchestration, MORAL employs a two-step procedure. However, besides the use of deep RL, MORAL differs since it learns to combine reward functions, whereas policy orchestration learns to combine policies. This allows MORAL to learn Pareto optimal policies at the cost of interpretability. Furthermore, policy orchestration requires a manual specification of the scalarization parameter, which MORAL can automatically infer through the use of active learning.

Ethics shaping [47] learns a reward shaping term from demonstrations, but does not scale beyond manually engineered features. Besides that, their approach is constrained to learning from a single expert. Although AIRL has been previously suggested to alleviate the issue of scalability [31], a method that is able to trade off rewards from multiple sources has not yet been developed in this setting. Finally, Ecoffet et al. [12] suggest a sequential voting scheme for learning conflicting values, but it requires explicit encoding of the different values at stake.

Inverse Reinforcement Learning Similarly to related IRL research, our work builds on AIRL [14] for inferring rewards from a multimodal distribution of demonstrations. Unlike previous research, which has focused on introducing latent variable models [19, 23, 28, 41, 43, 50] in the context of multiagent, multitask and hierarchical reward learning, we instead focus on the combination of labeled demonstration data. As such, our setup is similar to Gleave and Habryka [17] and Xu et al. [48], where a reward function is meta-learned by having explicit access to different task distributions. However, we learn from demonstrations in a multi-objective context, which has, to our knowledge, not yet been studied before.

Besides this, IRL has been applied in the context of value alignment [22], where inverse reward design (IRD) [21] has been proposed to learn a distribution of reward functions through IRL that leverages uncertainty to avoid unintended behavior. Although we also learn a distribution over reward functions, IRD focuses on finding safe goal specifications from a single reward function, whereas we study the extraction of value-aligned policies from a multitude of demonstrations. As a result, our research is more similar to multitask IRD [26], which studies formal criteria for combining reward functions from multiple sources. We, on the other hand, drop the formal assumptions and propose a practical method for combining reward functions learned through deep neural networks.

Learning from Expert Feedback Besides IRL, there exist a variety of approaches for training RL agents from expert data, including scalar-valued input [25, 44], natural language [3], intervention [38] and pairwise preferences [10, 46]. Similarly to Christiano et al. [10], we employ a Bradley-Terry model for training a nonstationary reward function by comparing trajectories from on-policy RL experience. However, our model of pairwise preferences operates on a set of abstract high-level reward functions, whereas [10] learn a single end-to-end reward model. Furthermore, our approach combines demonstration and preference data, which is more similar to Ibarz et al. [24]. Nonetheless, [24] uses demonstration data for retraining a preference-based reward model, which does not account for conflicting demonstrations. MORAL, on the other hand, allows for the inclusion of multiple experts as well as prior knowledge, thus making it suitable for resolving normative conflicts.

Preference-based reward learning with multiple experts has been recently studied by Myers et al. [29]. They propose an active learning algorithm for efficiently learning multimodal reward functions
that can represent different preferences. This differs from our work in two key points: Firstly, aside from the incorporation of demonstrations, we show how to use active learning to find a trade-off between conflicting reward functions. Secondly, unlike their work, we study how to learn a policy alongside the reward function.

Finally, by combining different expert reward functions, we have shown that MORAL interpolates between maximum-entropy IRL distributions. From this point of view, we consider our work a counterpart to Brown et al. [8], which ranks demonstration data in order to extrapolate beyond the behavior of a single expert.

**Multi-Objective Decision-Making** Typically, MORL algorithms trade off multiple objectives by learning a policy, or a set thereof, that can represent a range of Pareto-optimal solutions [32, 49]. On the other hand, our model learns a distribution over reward functions, which interactively guides the search to produce a single Pareto-optimal policy. Aside from sample efficiency, this mitigates the problem of varying reward scale, which has previously been addressed by multi-objective maximum a posteriori policy optimization (MO-MPO) [1]. However, MO-MPO requires explicit preferences over objectives, which is not always feasible when combining learned rewards that are inherently difficult to compare.

By using on-policy RL experience to learn scalarization weights, MORAL can be viewed as an interactive MORL algorithm. To date, interactive MORL has mainly been applied to bandits for linear [34] and nonlinear [33] transformations of the reward components, but has not yet been studied in the full RL setting. We believe that this is the case because MORL research usually assumes environments with manually engineered reward functions, in which big parts of the Pareto boundary exhibit interesting solutions. In the case of trading off learned reward functions, however, we suggest that our interactive approach poses a more adequate option.

**6 DISCUSSION**

In our work, we propose MORAL, a method for combining learned reward functions from multiple experts. We have shown MORAL to be a technical approach for aligning deep RL agents with human norms, which uses active learning to resolve value conflicts within expert demonstrations. We consider our research a step towards multi-objective RL with learned rewards, which has not yet been addressed before. Previous approaches such as ethics-shaping [47] and policy orchestration [30] have highlighted the strength of combining reward functions with expert demonstrations for value alignment, whereas DRLHP [10] has demonstrated the scalability of deep preference-based RL (table 2). MORAL unifies these ideas into a single method, which allows it to be applied in the presence of deep function approximation and multiple experts. This theoretical advantage is reflected in our experiments, which show that, unlike DRLHP, MORAL succeeds in retrieving Pareto-optimal solutions. Furthermore, we have shown MORAL to automatically learn implicit social norms if expert demonstrations agree on them. This informativeness about desirable behavior has been previously identified as a desideratum for combining reward information by Krasheninnikov et al. [26] and we have shown that it allows the active queries to focus on higher-level normative conflicts.

Nonetheless, several avenues for future research remain to be addressed. Firstly, generating queries from on-policy experience puts MORAL at risk of local optimality. In sparse environments, we therefore consider the introduction of a separate exploration policy for active learning to be useful. Secondly, combining multiple forms of expert supervision is challenging, due to a risk of accumulating errors and modelling assumptions for each type of input. We suppose future research in AIRL will be necessary to prevent overfitting of the reward network. Similarly to Gleave and Habryka [17], we found the reoptimization of AIRL reward functions to decrease performance, indicating that the learned rewards are entangled with the state distribution of the generator policy. Although this will require significant progress in deep IRL, we expect future methods to be easily integrated into MORAL by replacing AIRL. Furthermore, one could pursue unsupervised techniques to extend MORAL to unlabeled demonstration datasets. When learning social norms from large scale real-world demonstration data, it might be infeasible to learn separate reward functions for each expert. Unsupervised learning of reward functions that correspond to the different modes of behavior instead could alleviate this issue.

Overall, our research highlights the importance of multi-objective sequential decision-making without explicitly provided reward functions. Aside from value alignment [42], the ability to detect and respond to a divergence in values has been recognized as a central trait for building human-like AI [5]. Further, following the principle of meaningful human control [37], MORAL can contribute to increase an agent’s responsiveness to conflicting human norms, while maintaining human autonomy in determining desired trade-offs. This research contributes to the broader goal of designing and developing safe AI systems that can align to human values and norms.

**REFERENCES**

[1] Abbas Abdolmaleki, Sandy Huang, Leonard Hasenclever, Michael Neunert, Francis Song, Martina Zambelli, Murilo Martins, Nicolas Heess, Raia Hadsell, and Martin Riedmiller. 2020. A distributional view on multi-objective policy optimization. In *International Conference on Machine Learning*, PMLR, 11–22.

[2] David Abel, J. MacGlashan, and M. Littman. 2016. Reinforcement Learning as a Framework for Ethical Decision Making. In *AAAI Workshop: AI, Ethics, and Society*.

[3] Md Sultan Al Nahian, Spencer Frazier, Mark Riedl, and Brent Harrison. 2020. Learning norms from stories: A prior for value aligned agents. In *AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. Association for Computing Machinery, Inc, 124–130. https://doi.org/10.1145/3375627.3375825 arXiv:1912.03553

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

| Ethics-Shaping [47] | Policy-Orchestration [30] | DRLHP [10] | MORAL |
|---------------------|--------------------------|-----------|-------|
| **Deep Learning**   | ✓                        | ✓         | ✓     |
| **Multi-Objective** | ∼                        | ✓         | ×     |
| **Multiple Experts**| ×                        | ∼         | ✓     | ✓     |

Table 2: Comparison of MORAL to previous work in terms of supported capabilities.
A ARCHITECTURES
This section describes all network architectures used throughout the experiments. For ease of exposition, we omit grid world dimensionalities of each environment, and instead only report the amount of output channels and kernel sizes respectively. Furthermore, each convolutional layer uses a stride of 1 and no padding, which we found sufficient due to the relatively small sizes of the grids.

A.1 Proximal Policy Optimization
For PPO, we employ a convolutional actor-critic architecture with shared base layers, as illustrated in figure 9. We use two convolutional layers to form a feature array with 256 channels, which is then passed to the actor and critic in parallel. Finally, the actor employs a linear layer with output dimensions equal to the number of actions $|\mathcal{A}| = 9$ on the flattened feature representations of the final convolutional layer. Similarly, the critic employs a final linear layer with a scalar output to predict the current value. To draw action samples from the actor, a softmax is performed over its last linear layer, and we treat the resulting vector as a categorical distribution. In between layers, we employ standard ReLU activations to facilitate nonlinearity.

A.2 Reward Network - Emergency
Due to the small size of the environment, we employ a dense neural network for the AIRL discriminator architecture in Emergency. In this case, we flatten grid world states into a single vector and pass them through each layer, as illustrated in figure 10. Similarly to [14], we decompose the network output $f_\theta(s, s') = g_\theta(s) + \gamma h_\theta(s') - h_\theta(s)$, where $\gamma \in [0, 1]$ is the discount factor. While [14] propose this decomposition to retrieve a state-only reward function, we instead only use it for matching the AIRL implementation, but use $f_\theta$ as the reward function in subsequent steps.

A.3 Convolutional Reward Network - Delivery
We found the MLP architecture of the AIRL discriminator shown in figure 10 to be insufficient in larger grid world sizes. For this reason, we employ a convolutional reward network in Delivery as shown in figure 11. In principle, the network follows the same structure as in Emergency, but replaces linear layers with convolutional ones. To do so, we employ three convolutional layers followed by a single linear layer that acts on respective flattened feature maps for both $h_\theta$ and $g_\theta$ and form our reward estimate as before. Finally, we use LeakyReLU activations with a slope of $\alpha = 0.01$ on all hidden layers.

A.4 Deep Reinforcement Learning from Human Preferences
For DRLHP we train a PPO agent using the architecture shown in figure 9 in parallel with a deep reward model, which we show in figure 12. The reward model takes a state-action pair at each time step and outputs a predicted reward $r_\phi(st, at)$. We first one-hot encode the action $at$ and then embed it into a vector with the same dimensionality as the input state $st$. To do so, we train a linear embedding layer with output dimensions $C \cdot W \cdot H$, where $(C, W, H)$ denote the amount of channels, width and height of the state $st$ respectively. Embedded actions then get reshaped and concatenated with $st$ along the channel dimension to form an array of dimension $(2C, W, H)$ (batch dimension omitted). This array is fed through three convolutional layers with 128, 64 and 32 output channels respectively. Finally, the resulting flattened feature maps are processed by a linear layer to produce the reward estimate. As in the AIRL discriminator architecture, we employ LeakyReLU activations with a slope parameter $\alpha = 0.01$. 
Figure 12: Reward model architecture for DRLHP with convolutional (yellow) and linear (blue) layers. Actions are embedded through a linear layer and concatenated with the current state before being fed through subsequent layers.

### B PROOF OF THEOREM 2.1

By definition of the Kullback-Leibler divergence, we have

\[
D_{KL}(\pi(\tau) || p_{\theta_j}(\tau)) = \mathbb{E}_\pi \left[ \sum_{t=0}^{T} \log \pi(a_t|s_t) - r_{\theta_j}(s_t, a_t) \right] + \log Z_{\theta_j},
\]

where \( Z_{\theta_j} = \int p_{\theta_j}(\tau) d\tau \) is the partition function. Using \( \sum w_i = 1 \), we can now take the weighted sum of Kullback-Leibler divergences and obtain

\[
\sum_{i=1}^{k} w_i \mathbb{E}_\pi \left[ \sum_{t=0}^{T} \log \pi(a_t|s_t) - r_{\theta_j}(s_t, a_t) \right] + \sum_{i=1}^{k} w_i \log Z_{\theta_j} = \mathbb{E}_\pi \left[ \sum_{t=0}^{T} \log \pi(a_t|s_t) - \left( \sum_{i=1}^{k} w_i r_{\theta_j}(s_t, a_t) \right) \right] + \sum_{i=1}^{k} w_i \log Z_{\theta_j}.\]

Minimizing over \( \pi \) yields the desired expression, since the normalization functions \( Z_{\theta_j} \) as well as the weights \( w_i \) are constants as a function of the policy \( \pi \).

### C MARKOV CHAIN MONTE CARLO

In our implementation, we follow the procedure by [36] for simplifying the approximation of the Bradley-Terry posterior using Markov Chain Monte Carlo (MCMC). Namely, instead of using the original likelihood

\[
p(\tau_i > \tau_j|w) = \frac{\exp(w^T r(\tau_i))}{\exp(w^T r(\tau_j)) + \exp(w^T r(\tau_i))},
\]

we instead opt for a proxy likelihood of the form

\[
\hat{p}(\tau_i > \tau_j|w) = \min(1, \exp(w^T \Delta_{ij})),
\]

where \( \Delta_{ij} = r(\tau_i) - r(\tau_j) \). Its mode always evaluates to 0, which allows us to efficiently obtain posterior estimates through the Metropolis-Hastings algorithm [9] with a warm start to the distribution mode.

### D HYPERPARAMETERS - EMERGENCY

In the following, we will list all hyperparameter configurations used for the experiments of the main sections. Aside from algorithm specific hyperparameters, we always employ a learning rate for PPO (lr-PPO) that determines the gradient step size used in the agent’s Adam optimizer, a trust region clip parameter (\( \epsilon \)-clip) that determines how far the updated policy is allowed to diverge from the old, a time discounting parameter \( \gamma \), the amount of gradient steps taken on the policy loss per epoch (Epochs PPO) and the amount of environment episodes used for each epoch in PPO (Batch Size PPO). All policies were trained in a vectorized environment with 12 instances for Environment Steps amount of interactions.

#### D.1 AIRL

In AIRL, we use an Adam optimizer with its own learning rate for the discriminator (lr-Discriminator). Furthermore, Batch Size Discriminator determines the amount of state-action pairs used in a single training batch. Hyperparameters are reported in table 3.

| Hyperparameter            | Value |
|---------------------------|-------|
| lr-Discriminator          | 5e-4  |
| lr-PPO                    | 5e-4  |
| Batch Size Discriminator  | 512   |
| Batch Size PPO            | 12    |
| Environment Steps         | 3e6   |
| \( \epsilon \)-clip       | 0.1   |
| \( \gamma \)              | 0.999 |
| Epochs PPO                | 5     |

Table 3: AIRL hyperparameters in Emergency.

#### D.2 Active Learning

In the active learning step of MORAL, we query at fixed time intervals with a prespecified amount of total queries (\# Queries) that get evenly distributed across the amount of available environment steps. Besides that, no additional hyperparameters are necessary. However, we note that if the posterior converges to a local optimum prematurely, one can employ a normalization parameter \( c > 0 \) to multiply the vector valued reward function \( R(s, a) := c \cdot r \). For small choices of \( c \), one can expect to make the posterior less sensitive to updates at each step. Nonetheless, we found an inclusion of such hyperparameter to be unnecessary in our experiments, since marginal reward functions are normalized by their respective optimal values regardless. We report active learning hyperparameters in table 4.

#### D.3 DRLHP

To make DRLHP conceptually similar to MORAL, we employ queries at constant time intervals using a fixed amount of total queries (\# Queries) across the available environment steps. Besides that, we update the deep reward model after a constant amount of environment steps (Update Reward Model Frequency) with the Adam optimizer and a corresponding learning rate (lr-Reward Model). Overall, higher entropy regularization was necessary to ensure
Table 4: Active learning hyperparameters in Emergency.

| Hyperparameter         | Value   |
|------------------------|---------|
| lr-PPO                 | 3e-4    |
| # Queries              | 25      |
| Batch Size PPO         | 12      |
| Entropy Regularization | 0.25    |
| Environment Steps      | 666     |
| 𝜖-clip                 | 0.1     |
| 𝛾                      | 0.999   |
| Epochs PPO             | 5       |

E.2 Active Learning

The following table shows the typical hyperparameter setup for the active learning step of MORAL. Note, however, that while the amount of total environment steps were held fixed throughout different runs, the total number of queries varied, as described in the respective experiments. See table 7 for the active MORL hyperparameters in Delivery.

Table 7: Active learning hyperparameters in Delivery.

| Hyperparameter         | Value   |
|------------------------|---------|
| lr-PPO                 | 3e-4    |
| # Queries              | 25      |
| Batch Size PPO         | 12      |
| Entropy Regularization | 0.25    |
| Environment Steps      | 866     |
| 𝜖-clip                 | 0.1     |
| 𝛾                      | 0.999   |
| Epochs PPO             | 5       |

E.3 DRLHP

To ensure that the DRLHP has a comparable amount of available information about the expert’s underlying preferences, we provide 5000 overall queries over the course of training. We show DRLHP hyperparameters for Delivery in table 8.

Table 8: Hyperparameter setup for DRLHP in Delivery.

| Hyperparameter         | Value   |
|------------------------|---------|
| lr-PPO                 | 3e-4    |
| lr-Reward Model        | 3e-5    |
| Update Reward Model Frequency | 50    |
| # Queries              | 25000   |
| Batch Size PPO         | 12      |
| Batch Size Reward Model| 12      |
| Entropy Regularization | 1       |
| Environment Steps      | 1266    |
| 𝜖-clip                 | 0.1     |
| 𝛾                      | 0.999   |
| Epochs PPO             | 5       |

Table 5: Hyperparameter setup for DRLHP in Emergency.

Table 6: AIRL hyperparameters in Delivery.

| Hyperparameter         | Value   |
|------------------------|---------|
| lr-Discriminator       | 5e-5    |
| lr-PPO                 | 5e-4    |
| Batch Size Discriminator| 512    |
| Batch Size PPO         | 4       |
| Environment Steps      | 666     |
| 𝜖-clip                 | 0.1     |
| 𝛾                      | 0.999   |
| Epochs PPO             | 5       |

Table 7: Active learning hyperparameters in Delivery.

E.1 AIRL

To avoid overfitting and balance the discriminator and generator performances, we lower the learning rate of the discriminator. We show AIRL hyperparameters for Delivery in table 6.

E HYPERPARAMETERS - DELIVERY

In Delivery, the choice of hyperparameters is similar, besides a consistent increase in environment steps due to a higher task complexity.