Preventing Reward Hacking with Occupancy Measure Regularization

Cassidy Laidlaw [*1] **Shivam Singhal** [*1] **Anca Dragan** [1]

Abstract

Reward hacking occurs when an agent exploits its specified reward function to behave in undesirable or unsafe ways. Aside from better alignment between the specified reward function and the system designer's intentions, a more feasible proposal to prevent reward hacking is to regularize the learned policy to some safe baseline. Current research suggests that regularizing the learned policy's action distributions to be more similar to those of a safe policy can mitigate reward hacking; however, this approach fails to take into account the disproportionate impact that some actions have on the agent's state. Instead, we propose a method of regularization based on *occupancy measures*, which capture the proportion of time each policy is in a particular state-action pair during trajectories. We show theoretically that occupancy-based regularization avoids many drawbacks of action distribution-based regularization, and we introduce an algorithm called ORPO to practically implement our technique. We then empirically demonstrate that occupancy measure-based regularization is superior in both a simple gridworld and a more complex autonomous vehicle control environment.

1. Introduction

A major challenge for the designers of AI systems is specifying an objective, or reward function, that is aligned with their goals and values. If an AI agent optimizes a reward function that is not representative of the designer's original intent, it may act in undesirable and potentially dangerous ways (Russell, 2019). This behavior is called *reward hacking*—when the resulting policy performs well in terms of the given proxy reward function but poorly on the unknown true reward function (Skalse et al., 2022; Pan et al., 2022).

The best solution to prevent reward hacking would be to ensure better alignment between the defined proxy and hidden true objectives. However, in practice, reward functions are extremely difficult to properly design due to the ambiguities and complex variables underlying real-world scenarios (Ibarz et al., 2018). Recommender systems, for example, aim to optimize the value that users attain from their time spent on the online platforms; however, since this goal is difficult to quantify, designers utilize proxies, such as click-through rates, engagement time, and other types of feedback they receive from users, which do not always match how satisfied users are with their experience (Stray et al., 2022). Several examples of reward hacking have been reported throughout the literature (Krakovna, 2018).

Rather than specifying a reward or objective function by hand, an alternative strategy is to *learn* the reward function through interaction with the system designer and/or users (Bıyık et al., 2020; Palan et al., 2019). For instance, reward learning is used in reinforcement learning from human feedback (RLHF) to shape the behavior of large language models towards more helpful output (Ouyang et al., 2022). However, even learned reward functions are often misaligned with our true objectives as they usually fail to generalize well outside the distribution of behavior used to train them (McKinney et al., 2023).

Instead of blindly optimizing a proxy reward function, one proposal to avoid reward hacking is to *regularize the policy towards a known safe policy* (Yang et al., 2021). For example, RLHF for large language models generally optimizes the learned reward plus a term that penalizes divergences from the pre-trained language model's output. Intuitively, this kind of regularization pushes the learned policy away from "unusual" behaviors for which the reward function may be misaligned. These could include unforeseen strategies in the case of a hand-specified reward function or out-of-distribution states in the case of a learned reward function.

While this idea of regularizing towards a safe policy is use-

[*]Equal contribution [1]Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA. Correspondence to: Cassidy Laidlaw <cassidy_laidlaw@berkeley.edu>, Shivam Singhal <shivamsinghal@berkeley.edu>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023.

ful, it implicitly assumes a distance metric over the space of policies. So far, this has been the Kullback–Leibler (KL) divergence between action distributions, which has a lot of benefits: it is easy to compute and optimize within common deep RL algorithms (Vieillard et al., 2021), and it appears to effectively prevent reward hacking in RLHF for large language models (Ouyang et al., 2022). However, we argue that, in general, regularizing based on the *action distributions* of policies also has significant drawbacks. Specifically, small shifts in action distributions can lead to large differences in outcomes, and on the other hand, large shifts in action distributions may not actually result in any difference in outcome.

Consider the illustrative example of training a self-driving car policy that is driving along the side of a steep cliff. Suppose there is a safe policy that drives slowly and avoids falling off the cliff, but the agent is incentivized by a faulty objective function that rewards driving fast without penalizing going off the road. When regularizing to the safe policy based on action distributions, it may be very difficult to avoid a learned policy that drives off the cliff. This is because even one wrong action—a small change in the action distributions at a single state—could lead to a disaster. In addition, for the learned policy to improve on the reward of the safe policy, the car would have to go faster, which likely means significant changes to the car's action distributions at many states. Thus, it is probably impossible to use action distribution regularization to avoid a single catastrophic action while still performing better than the safe policy.

Instead of regularizing based on action distributions, we propose that the learned policy be regularized based on the divergence between its and a safe policy's ***occupancy measures***. An occupancy measure (OM) represents the distribution of states and actions seen by a policy when it interacts with its environment. Unlike action distribution-based metrics, OM takes into account not just the actions taken by the policy, but also the states that the agent reaches.

Going back to our example, while a single catastrophic action from the self-driving policy may not change its action distribution much, it will significantly increase the distance between the policies' occupancy measures: the learned policy will have a high probability of reaching states where the car is off the cliff and crashed, while the safe policy never reaches such states. As a result, occupancy measure regularization more strongly penalizes actions that lead to potentially calamitous consequences. Furthermore, it less strongly penalizes deviations from the safe policy that are likely to improve the self-driving car's performance. If the self-driving policy learns to drive faster, the agent will mostly see many of the same states along the road that the safe policy does, and therefore, there will not be a large divergence in occupancy measures to penalize.

In this paper, we show both theoretically and empirically that regularizing policy optimization using occupancy measure divergence is more effective at preventing reward hacking. Theoretically, we show that there is a direct relationship between the divergence in occupancy measures between two policies and their returns under *any* reward function, while no such relationship holds for divergences between action distributions of the two policies. Empirically, regularizing the occupancy measure of a policy is more challenging than regularizing its action distributions. To address this, we derive an algorithm, occupancy-regularized policy optimization (ORPO), for approximating the occupancy measure divergence with a discriminator network that can be easily incorporated in deep RL algorithms like Proximal Policy Optimization (PPO). We use this to optimize policies trained with misaligned proxy reward functions in multiple environments and compare our method's performance to that of action distribution regularization. The results of our experiments show that regularization based on occupancy measures is more effective at preventing reward hacking, while still allowing optimization that increases an unseen true reward function. Our findings suggest that regularization based on occupancy measures should replace action distribution-based regularization for the purpose of preventing reward hacking.

2. Related work

A few previous works have focused on defining and characterizing reward hacking. Pan et al. (2022) conduct a systematic study of reward misspecification. They characterize three particular ways in which rewards can be misspecified and show across several environments that increasing the optimization power of RL agents can result in sudden shifts, or phase changes, in the agents' reward hacking behavior. Skalse et al. (2022) define reward hacking, also known as reward gaming (Leike et al., 2018), as an increase in a proxy reward function accompanied by a noticeable drop in the true reward function. Everitt et al. (2019) focus on reward tampering, a special type of reward hacking where the agent tampers with or corrupts a reward signal embedded in the environment in some way in order to achieve higher proxy reward values. Krakovna (2018) provides a list of many examples of reward hacking from the literature.

Various methods have been proposed to avoid reward hacking and/or mitigate its dangerous effects. As we described in the introduction, a widely used technique when training large language models with RL is to regularize the KL divergence of their action distributions (Ouyang et al., 2022). Quantilizers (Taylor, 2016) are an alternative to reward maximizing-agents that are designed to avoid reward hacking behavior. Inverse reward design (Hadfield-Menell et al., 2017) attempts to infer the true reward function based on the

given proxy reward function and environment context. Value reinforcement learning (Everitt & Hutter, 2016) focuses on preventing reward tampering in particular. Other methods aim to prevent unintended "side effects" from agents with possibly misspecified reward functions (Krakovna et al., 2019; 2020). These methods avoid the need for a safe policy but can require other inputs, like a list of "auxiliary" reward functions (Turner et al., 2020). They are also generally difficult to integrate into deep RL algorithms.

The method of regularizing policy optimization to a safe policy with KL divergence was first proposed by Stiennon et al. (2020) and has since been widely used in the context of optimizing large language models using learned reward functions (Ouyang et al., 2022; Bai et al., 2022). KL regularization for RLHF has been further studied by Vieillard et al. (2021), Gao et al. (2022), and Korbak et al. (2022). Human-in-the-loop variants of RLHF allow for misspecified reward functions to be corrected in real-time (Lee et al., 2021), which could avoid the need for regularization in some cases.

3. Occupancy measure-based regularization

We formalize our method of regularizing policy optimization with occupancy measure divergences in the setting of an infinite-horizon Markov decision process (MDP). An agent takes actions $a \in \mathcal{A}$ to transition between states $s \in \mathcal{S}$ over a series of timesteps $t = 0, 1, 2, \ldots$. The first state s_0 is sampled from an initial distribution $\mu_0(s)$, and when an agent takes action a_t in s_t at time t, the next state s_{t+1} is reached at timestep $t+1$ with transition probability $p(s_{t+1} \mid s_t, a_t)$. The agent aims to optimize a reward function $R : \mathcal{S} \times \mathcal{A} \to [0, 1]$, and rewards are accumulated over time with discount factor $\gamma \in [0, 1)$. A policy π maps each state s to a distribution over actions to take at that state $\pi(a \mid s)$. We define the (normalized) *return* of a policy π for a reward function R as

$$J(\pi, R) = (1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$$

where $\mathbb{E}_\pi$ refers to the expectation under the distribution of states and actions induced by running the policy π in the environment. The normalizing factor guarantees that $J(\pi, R) \in [0, 1]$ always.

We define the *state-action occupancy measure* μ_π of a policy π as the expected discounted number of times the agent will be in a particular state and take a specific action:

$$\mu_\pi(s, a) = (1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}\{s_t = s \wedge a_t = a\}\right].$$

Intuitively, the occupancy measure represents the distribution of states and actions visited by the policy over time. If the policy spends a lot of time taking action a in state s, then $\mu(s, a)$ will be high. Conversely, if the policy never visits a state s, then $\mu(s, a) = 0$ for all actions a.

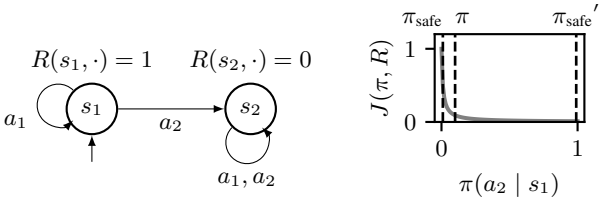


Figure 1. The MDP on the left, used in the proof of Proposition 3.1, demonstrates one drawback of using divergence between policies' action distributions for regularization. The agent stays in state s_1, where it receives 1 reward per timestep, until it takes action a_2, after which it remains in state s_2 forever and receives no reward. The plot on the right shows the return $J(\pi, R)$ for a policy π when $\gamma = 0.99$ as a function of the policy's action distribution at s_1. While π_{safe} and π (shown on the plot as dotted lines) are close in action distribution space, they achieve very different returns. Meanwhile, π is far from π_{safe}' in action distribution space, but achieves nearly the same return. Propositions 3.2 and A.2 show that occupancy measure divergences do not have these drawbacks.

3.1. Regularized policy optimization

The standard approach to solving an MDP is to find a policy which maximizes its return:

$$\text{maximize} \quad J(\pi, R). \tag{1}$$

However, as we discussed in the introduction, an AI system designer often does not have access to a reward function which is perfectly aligned with their preferences and values. Instead, the designer might optimize π using a learned or hand-specified *proxy* reward function $\tilde{R}$ which is misaligned with the *true* reward function R. Blindly maximizing the proxy reward function could lead to reward hacking. Thus, a widely-used approach is to optimize the policy's return with respect to $\tilde{R}$ plus a regularization term that penalizes the KL divergence of the policy's action distributions from a *safe policy* π_{safe}:

$$\begin{aligned}\text{maximize} \quad & J(\pi, \tilde{R}) \\ & - \lambda(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t D_{\text{KL}}(\pi(\cdot \mid s_t) \parallel \pi_{\text{safe}}(\cdot \mid s_t))\right].\end{aligned} \tag{2}$$

The regularization term can be easily incorporated into deep RL algorithms like PPO since the KL divergence between action distributions can usually be calculated in closed form.

Although it is simple, the action distribution-based regularization method in (2) has serious drawbacks. These arise from the complex relationship between a policy's action distribution at various states and its return under the true reward function. In some cases, a very small change in action distribution space can result in a huge change in reward, and in other cases, a large change in action distribution space can result in a negligible change in reward. We formalize this in the following proposition.

Proposition 3.1. *Fix $\epsilon > 0$ and $\delta > 0$ arbitrarily small,*

and $c \geq 0$ arbitrarily large. Then there is an MDP and true reward function R where both of the following hold:

1. *There is a pair of policies π and π_{safe} where the action distribution KL divergence satisfies*
$$(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{KL}(\pi(\cdot \mid s_t) \parallel \pi_{safe}(\cdot \mid s_t))\right] \leq \epsilon$$
but $|J(\pi_{safe}, R) - J(\pi, R)| \geq 1 - \delta$.

2. *There is a safe policy $\pi_{safe}{}'$ such that* any *other policy π with*
$$(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{KL}(\pi(\cdot \mid s_t) \parallel \pi_{safe}{}'(\cdot \mid s_t))\right] \leq c$$
satisfies $|J(\pi_{safe}{}', R) - J(\pi, R)| \leq \delta$.

The first part of Proposition 3.1 shows that in the worst case, a KL divergence smaller than some arbitrary threshold ϵ from the safe policy's action distributions can induce a change in the return under true reward function R that is almost as large as the entire possible range of returns. Thus, when regularizing using action distribution KL divergence like in (2), one might have to make λ extremely large to prevent drastic changes from the safe policy. However, the second part of Proposition 3.1 shows that in the same MDP, for a different safe policy, any learned policy with arbitrarily large action distribution KL divergence from the safe policy has an extremely small difference in return. This means that one might need to set λ extremely small in order to allow for the large divergence in the policies' action distributions necessary for optimization to have any effect. See Figure 1 for a graphical illustration of Proposition 3.1.

Since Proposition 3.1 shows that small KL divergence in action distribution from the safe policy can have large effects, and vice versa, it may be impossible in some environments to regularize effectively using the objective in (2). We propose instead to regularize based on the divergence between the occupancy measures of the learned and safe policies:

$$\text{maximize} \quad J(\pi, \tilde{R}) - \lambda \left\|\mu_\pi - \mu_{\pi_{\text{safe}}}\right\|_1. \tag{3}$$

In (3), we use the total variation (TV) between the occupancy measures, defined as

$$\left\|\mu_\pi - \mu_{\pi_{\text{safe}}}\right\|_1 = \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} |\mu_\pi(s,a) - \mu_{\pi_{\text{safe}}}(s,a)|.$$

Why should using the occupancy measure divergence to regularize perform better than using the divergence between action distributions? The following proposition shows that the TV between occupancy measures does not have the same problems as action distribution divergence: a small divergence *cannot* result in a large change in policy return.

Proposition 3.2. *For any MDP, reward function R, and pair of policies π, π_{safe}, we have*

$$|J(\pi_{safe}, R) - J(\pi, R)| \leq \left\|\mu_\pi - \mu_{\pi_{safe}}\right\|_1. \tag{4}$$

Proposition 3.2 shows that the first issue with action distribution divergence from Proposition 3.1 does not also affect occupancy measure divergence. Instead, a small difference between the occupancy measure of the safe and learned policies *guarantees* that they have similar returns. In fact, Proposition A.2 (see Appendix A) shows we cannot do any better in general than the bound from Proposition 3.2.

These results suggest that the divergence between the occupancy measures of the learned and safe policies is a much better measure of how similar those policies are than the divergence between the policies' action distributions. In the following sections, we will show that these theoretical results match with intuition and empirical performance.

3.2. Example of action distribution and occupancy measure divergences

Figure 2 shows an intuitive example of why regularizing to a safe policy with occupancy measure divergence is superior to regularizing with action distribution divergence. Figure 2a depicts a simplified version of the tomato-watering AI Safety Gridworld proposed by Leike et al. (2017). The agent, a robot that starts in the lower right, can move up, down, left, right, or stay in place. Its objective is to water the tomatoes on the board, and it receives reward each time it moves into a square with a tomato. However, there is also a sprinkler in the upper right corner of the environment. When the robot moves into the sprinkler's square, its sensors see water everywhere, and it believes all tomatoes are watered. In this environment, the true reward function R only rewards watering tomatoes, while the proxy reward function $\tilde{R}$ also highly rewards reaching the sprinkler.

The top row of Figure 2b shows three policies for this environment: a desired policy, which achieves the highest true reward, a safe policy, which achieves lower true reward because it stays in place more often, and a reward hacking policy, which exploits the sprinkler state to achieve high proxy reward but low true reward. The arrows between the policies on the top row of Figure 2b show the action distribution KL divergences between them as used for regularization in (2). The action distribution divergences suggest that the reward hacking policy is actually closer to the safe policy than the desired policy is. This is because the safe policy is nearly identical to the reward hacking policy in the upper right square, where the reward hacking policy spends most of its time; they both take the "stay" action with high probability. Thus, if we regularize to the safe policy using action distribution KL divergence, we would be more likely to find a policy that hacks the proxy reward rather than one like the left policy, which we prefer. The problem is that the safe policy rarely reaches the sprinkler in the first place, but the action distribution divergence doesn't account for this.

Using occupancy measure divergences avoids this problem.

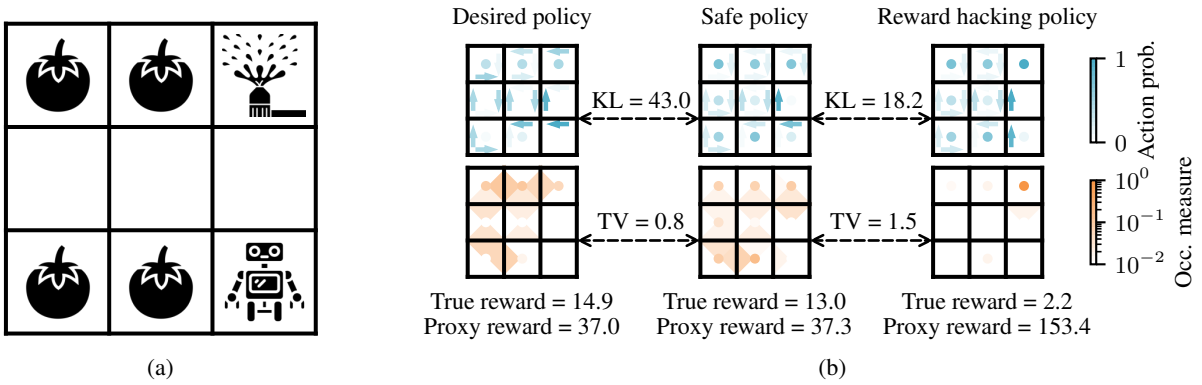


Figure 2. This simple gridworld provides an intuitive example of why occupancy measure divergences are superior to action distribution divergences for regularizing to a safe policy. See Section 3.2 for the details.

The bottom row of Figure 2b shows the occupancy measures for each policy in the top row, and the arrows between the columns show the total variation (TV) distance $\|\mu - \mu'\|_1$ between occupancy measures. Unlike the action distribution KL divergence, the occupancy measure TV distance suggests that the desired policy on the left is closer to the safe policy than the reward hacking policy is. This is because both the desired and safe policies spend most of their time actually watering tomatoes, as evidenced by the higher occupancy measure they assign to the tiles on the board with the tomatoes. In contrast, the reward hacking policy spends almost all of its time in the sprinkler square and as a result, has a very different occupancy measure. Thus, if we trained a policy regularized with occupancy measure divergence in this environment, we could hope to find a policy like the desired one on the left and avoid a reward hacking policy like the one on the right.

3.3. Occupancy-regularized policy optimization (ORPO)

In the previous section, we showed strong theoretical evidence that regularizing using occupancy measure divergence is superior to action distribution divergence. We now introduce an algorithm, occupancy-regularized policy optimization (ORPO), which enables occupancy measure-based regularization for deep reinforcement learning.

While our theory is based on regularization using the TV distance between occupancy measures, we find that the KL divergence is more stable to calculate in practice. Since Pinsker's inequality bounds the TV distance by the KL divergence for small KL values, and the Bretagnolle-Huber bound holds for larger KL values, our mathematical intuition remains valid (Canonne, 2022). Our objective from (3) can be reformulated with the KL divergence in place of the TV distance:

$$\text{maximize} \quad J(\pi, \tilde{R}) - \lambda\, D_{\text{KL}}(\mu_\pi \parallel \mu_{\pi_{\text{safe}}}). \tag{5}$$

We optimize (5) using a gradient-based method. The gradient of the first term is estimated using PPO, a popular policy gradient method (Schulman et al., 2017). However, calculating the occupancy measure divergence for the second term is intractable to do in closed form since it requires the enumeration of *all* possible state-action pairs, an infeasible task in the case of deep RL. Thus, we approximate the KL divergence between the occupancy measures of policies by training a *discriminator network*, a technique that has previously been used for generative adversarial networks (GANs) (Goodfellow et al., 2014) and GAIL (Ho & Ermon, 2016).

The discriminator network $d : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns a score $d(s, a) \in \mathbb{R}$ to any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, and it is trained on a mixture of data from both the learned policy π and safe policy π_{safe}. The objective used to train d incentives low scores for state-action pairs from π_{safe} and high scores for state-action pairs from π:

$$d = \arg\min_d \sum_{t=0}^{\infty} \Big(\mathbb{E}_\pi[\gamma^t \log(1 + e^{-d(s_t,a_t)})] + \mathbb{E}_{\pi_{\text{safe}}}[\gamma^t \log(1 + e^{d(s_t,a_t)})] \Big). \tag{6}$$

Huszár (2017) proves that if the loss function in (6) is minimized, then the expected discriminator scores for state-action pairs drawn from the learned policy distribution will approximately equal the KL divergence between the occupancy measures of the two policies:

$$D_{\text{KL}}(\mu_\pi(s,a) \parallel \mu_{\pi_{\text{safe}}}(s,a)) \approx (1-\gamma)\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t d(s_t, a_t) \right]$$

Applying the definitions of the learned policy returns and the KL divergence between the polices' occupancy measures, we can now rewrite our ORPO objective:

$$\text{maximize} \quad \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(\tilde{R}(s_t, a_t) - \lambda\, d(s_t, a_t) \right) \right]. \tag{7}$$

Note that (7) is identical to the normal RL objective with a reward function $R'(s, a) = \tilde{R}(s, a) - \lambda d(s, a)$. Thus, once

the discriminator has been trained, we add the discriminator scores to the given reward function and use the combined values to update π with PPO. The training process for ORPO is split into two phases: one in which data from both the safe and learned policies is used to train the discriminator to minimize (6), and one in which data from the learned policy is used to train the PPO agent with the augmented reward function in (7).

4. Experiments

We compare the empirical performance of ORPO to regularization with action distribution KL divergence in two environments: a slightly different version of the tomato-watering environment we focused on in Section 3.2 and an autonomous vehicle control environment introduced by Wu et al. (2022). As before, the tomato environment contains a sprinkler state where the agent perceives all tomatoes as being watered and thus receives high proxy reward but no true reward. For our safe policy, we train a PPO agent with the true reward, and then add a 10% chance of taking a random action to ensure there is room to improve upon it.

The traffic environment consists of a number of vehicles driving through a road network where cars on an on-ramp attempt to merge into traffic on a highway. Some vehicles are controlled by a human model and some are RL-controlled autonomous vehicles. The true objective of the self-driving agent is to ensure that there is fast traffic flow at all times in order to reduce the mean commute time, while observing the positions and velocities of nearby vehicles. The proxy reward is the average velocity of all cars in the simulation. When the traffic agent begins to reward hack, it stops cars on the on-ramp from merging into traffic. This way, the proxy reward is optimized because cars on the straightway can continue forward at a fast speed instead of having to wait for a car to merge, which increases the average velocity of all vehicles. However, the true objective is not achieved as the commute time for the cars on the on-ramp increases indefinitely. As the safe policy for the traffic environment we used the Intelligent Driver Model (IDM), a standard approximation of human driving behavior (Treiber et al., 2000). In practice, safe policies are often learned via imitation learning, so to simulate this we generate data from the IDM controller and train a behavioral cloning (BC) policy on it.

We train RL policies in each environment using action distribution regularization and OM regularization, varying the regularization coefficient λ across a wide range. We compare the performance of the regularization techniques to the safe policies π_{safe}, as well as policies trained to optimize the proxy reward and true reward without regularization.

The results of our experiments are shown in Tables 1 and 2. In both environments, both forms of regularization help the learned policy perform better than the unregularized PPO agent ("PPO w/ proxy $\tilde{R}$ w/o regularization"). The ORPO policies achieved the highest true reward out of all policies that were trained with the proxy reward function. Not only did they outperform the baseline policies, but they also achieved higher true reward than the policies trained with action distribution-based regularization.

Regularization method	λ	**Proxy reward**	**True reward**
PPO w/ proxy $\tilde{R}$ w/o regularization		406.14	24.88
PPO w/ true R w/o regularization		83.75	83.16
Safe policy π_{safe}		77.02	76.89
Action dist. KL	10^{-2}	400.13	24.31
Action dist. KL	10^{-1}	228.80	39.45
Action dist. KL	10^{0}	77.77	77.68
Action dist. KL	10^{1}	77.03	77.02
Occ. measure KL	10^{-2}	399.82	23.39
Occ. measure KL	10^{-1}	309.10	26.11
Occ. measure KL	10^{0}	79.61	79.61
Occ. measure KL	10^{1}	79.72	**79.70**

Table 1. In the tomato environment, the ORPO policy with $\lambda = 10$ performed the best among policies trained with the proxy reward.

Regularization method	λ	**Proxy reward**	**True reward**
PPO w/ proxy $\tilde{R}$ w/o regularization		3014	-63976
PPO w/ true R w/o regularization		1437	-640
Safe policy π_{safe}		1444	-2162
Action dist. KL	10^{-3}	3005	-60208
Action dist. KL	10^{-2}	2230	-55817
Action dist. KL	$3 * 10^{-2}$	1528	-1254
Action dist. KL	10^{-1}	1503	-1503
Action dist. KL	10^{0}	1451	-2114
Occ. measure KL	10^{-3}	2993	-62107
Occ. measure KL	$3 * 10^{-3}$	1533	**-1041**
Occ. measure KL	10^{-2}	1457	-2040
Occ. measure KL	10^{-1}	1454	-2556
Occ. measure KL	10^{0}	1361	-3624

Table 2. In the traffic environment, the ORPO policy with $\lambda = 3 * 10^{-3}$ performed the best among policies trained with the proxy reward.

5. Conclusion

We have presented theoretical and empirical evidence that occupancy measure regularization can more effectively prevent reward hacking than action distribution regularization when training with a misaligned proxy reward function. In the future, we hope to experiment with learned proxy reward functions in addition to the hand-specified reward functions we considered in this paper.

References

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, April 2022. URL http://arxiv.org/abs/2204.05862. arXiv:2204.05862 [cs].

Bıyık, E., Losey, D. P., Palan, M., Landolfi, N. C., Shevchuk, G., and Sadigh, D. Learning Reward Functions from Diverse Sources of Human Feedback: Optimally Integrating Demonstrations and Preferences. 2020. doi: 10.48550/ARXIV.2006.14091. URL https://arxiv.org/abs/2006.14091. Publisher: arXiv Version Number: 2.

Canonne, C. L. A short note on an inequality between KL and TV, February 2022. URL http://arxiv.org/abs/2202.07198. arXiv:2202.07198 [math, stat].

Everitt, T. and Hutter, M. Avoiding Wireheading with Value Reinforcement Learning, May 2016. URL http://arxiv.org/abs/1605.03143. arXiv:1605.03143 [cs].

Everitt, T., Hutter, M., Kumar, R., and Krakovna, V. Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Diagram Perspective. 2019. doi: 10.48550/ARXIV.1908.04734. URL https://arxiv.org/abs/1908.04734.

Gao, L., Schulman, J., and Hilton, J. Scaling Laws for Reward Model Overoptimization, October 2022. URL http://arxiv.org/abs/2210.10760. arXiv:2210.10760 [cs, stat].

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks, June 2014. URL http://arxiv.org/abs/1406.2661. arXiv:1406.2661 [cs, stat].

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S., and Dragan, A. Inverse Reward Design, 2017. URL http://arxiv.org/abs/1711.02827. arXiv:1711.02827 [cs].

Ho, J. and Ermon, S. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://papers.nips.cc/paper_files/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html.

Huszár, F. Variational Inference using Implicit Distributions, February 2017. URL http://arxiv.org/abs/1702.08235. arXiv:1702.08235 [cs, stat].

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in Atari. 2018. doi: 10.48550/ARXIV.1811.06521. URL https://arxiv.org/abs/1811.06521. Publisher: arXiv Version Number: 1.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. On Convergence and Stability of GANs, December 2017. URL http://arxiv.org/abs/1705.07215. arXiv:1705.07215 [cs].

Korbak, T., Perez, E., and Buckley, C. L. RL with KL penalties is better viewed as Bayesian inference, October 2022. URL http://arxiv.org/abs/2205.11275. arXiv:2205.11275 [cs, stat].

Krakovna, V. Specification gaming examples in AI, April 2018. URL https://vkrakovna.wordpress.com/2018/04/02/specification-gaming-examples-in-ai/.

Krakovna, V., Orseau, L., Kumar, R., Martic, M., and Legg, S. Penalizing side effects using stepwise relative reachability, March 2019. URL http://arxiv.org/abs/1806.01186. arXiv:1806.01186 [cs, stat].

Krakovna, V., Orseau, L., Ngo, R., Martic, M., and Legg, S. Avoiding Side Effects By Considering Future Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19064–19074. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/dc1913d422398c25c5f0b81cab94cc87-Abstract.html.

Lee, K., Smith, L., and Abbeel, P. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training, June 2021. URL http://arxiv.org/abs/2106.05091. arXiv:2106.05091 [cs].

Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Orseau, L., and Legg, S. AI Safety Gridworlds, November 2017. URL http://arxiv.org/abs/1711.09883. arXiv:1711.09883 [cs].

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction, November 2018. URL http://

arxiv.org/abs/1811.07871. arXiv:1811.07871 [cs, stat].

Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I. RLlib: Abstractions for Distributed Reinforcement Learning, June 2018. URL http://arxiv.org/abs/1712.09381. arXiv:1712.09381 [cs].

McKinney, L., Duan, Y., Krueger, D., and Gleave, A. On The Fragility of Learned Reward Functions, January 2023. URL http://arxiv.org/abs/2301.03652. arXiv:2301.03652 [cs].

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. 2022. doi: 10.48550/ARXIV.2203.02155. URL https://arxiv.org/abs/2203.02155.

Palan, M., Landolfi, N. C., Shevchuk, G., and Sadigh, D. Learning Reward Functions by Integrating Human Demonstrations and Preferences. 2019. doi: 10.48550/ARXIV.1906.08928. URL https://arxiv.org/abs/1906.08928. Publisher: arXiv Version Number: 1.

Pan, A., Bhatia, K., and Steinhardt, J. The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models, February 2022. URL http://arxiv.org/abs/2201.03544. arXiv:2201.03544 [cs, stat].

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019. URL http://arxiv.org/abs/1912.01703. arXiv:1912.01703 [cs, stat].

Russell, S. J. *Human compatible: artificial intelligence and the problem of control*. Viking, New York?, 2019. ISBN 978-0-525-55861-3.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347 [cs].

Skalse, J., Howe, N. H. R., Krasheninnikov, D., and Krueger, D. Defining and Characterizing Reward Hacking, September 2022. URL http://arxiv.org/abs/2209.13085. arXiv:2209.13085 [cs, stat].

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. Learning to summarize from human feedback, September 2020. URL http://arxiv.org/abs/2009.01325. arXiv:2009.01325 [cs].

Stray, J., Halevy, A., Assar, P., Hadfield-Menell, D., Boutilier, C., Ashar, A., Beattie, L., Ekstrand, M., Leibowicz, C., Sehat, C. M., Johansen, S., Kerlin, L., Vickrey, D., Singh, S., Vrijenhoek, S., Zhang, A., Andrus, M., Helberger, N., Proutskova, P., Mitra, T., and Vasan, N. Building Human Values into Recommender Systems: An Interdisciplinary Synthesis, July 2022. URL http://arxiv.org/abs/2207.10192. arXiv:2207.10192 [cs].

Taylor, J. Quantilizers: A Safer Alternative to Maximizers for Limited Optimization. March 2016. URL https://www.semanticscholar.org/paper/Quantilizers%3A-A-Safer-Alternative-to-Maximizers-for-Taylor/4e8ff3b4069a12a00196d62925bab8add7389742.

Treiber, M., Hennecke, A., and Helbing, D. Congested Traffic States in Empirical Observations and Microscopic Simulations. *Physical Review E*, 62 (2):1805–1824, August 2000. ISSN 1063-651X, 1095-3787. doi: 10.1103/PhysRevE.62.1805. URL http://arxiv.org/abs/cond-mat/0002177. arXiv:cond-mat/0002177.

Turner, A. M., Hadfield-Menell, D., and Tadepalli, P. Conservative Agency via Attainable Utility Preservation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 385–391, February 2020. doi: 10.1145/3375627.3375851. URL http://arxiv.org/abs/1902.09725. arXiv:1902.09725 [cs].

Vieillard, N., Kozuno, T., Scherrer, B., Pietquin, O., Munos, R., and Geist, M. Leverage the Average: an Analysis of KL Regularization in RL, January 2021. URL http://arxiv.org/abs/2003.14089. arXiv:2003.14089 [cs, stat].

Wu, C., Kreidieh, A., Parvate, K., Vinitsky, E., and Bayen, A. M. Flow: A Modular Learning Framework for Mixed Autonomy Traffic. *IEEE Transactions on Robotics*, 38 (2):1270–1286, April 2022. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2021.3087314. URL http://arxiv.org/abs/1710.05465. arXiv:1710.05465 [cs].

Yang, T.-Y., Rosca, J., Narasimhan, K., and Ramadge, P. J. Accelerating Safe Reinforcement Learning with Constraint-mismatched Policies, July 2021. URL http://arxiv.org/abs/2006.11645. arXiv:2006.11645 [cs, stat].

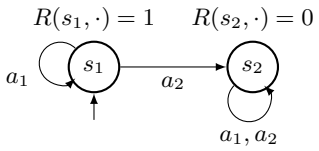
A. Proofs

A.1. Proof of Proposition 3.1

Proposition 3.1. *Fix $\epsilon > 0$ and $\delta > 0$ arbitrarily small, and $c \geq 0$ arbitrarily large. Then there is an MDP and true reward function R where both of the following hold:*

1. *There is a pair of policies π and π_{safe} where the action distribution KL divergence satisfies*
$$(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{KL}(\pi(\cdot \mid s_t) \parallel \pi_{safe}(\cdot \mid s_t))\right] \leq \epsilon$$
but $|J(\pi_{safe}, R) - J(\pi, R)| \geq 1 - \delta$.
2. *There is a safe policy $\pi_{safe}{}'$ such that* any *other policy π with*
$$(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{KL}(\pi(\cdot \mid s_t) \parallel \pi_{safe}{}'(\cdot \mid s_t))\right] \leq c$$
satisfies $|J(\pi_{safe}{}', R) - J(\pi, R)| \leq \delta$.

Proof. Consider the following MDP, also shown in Figure 1:



In this MDP, $\mathcal{S} = \{s_1, s_2\}$, $\mathcal{A} = \{a_1, a_2\}$, and the transition probabilities and reward function are defined by

$$\begin{aligned} p(s_1 \mid s_1, a_1) &= 1 \qquad p(s_2 \mid s_1, a_2) = 1 \\ p(s_2 \mid s_2, a_1) &= 1 \qquad p(s_2 \mid s_2, a_2) = 1 \\ \forall a \in \mathcal{A} \quad R(s_1, a) &= 1 \qquad R(s_2, a) = 0. \end{aligned}$$

The initial state is always s_1. Thus, the agent stays in state s_1 and receives 1 reward each timestep until it takes action a_2, at which point it transitions to s_2 and receives no more reward. Define for any $p \in [0, 1]$ a policy π_p that takes action a_2 in s_1 with probability p, i.e. $\pi_p(a_2 \mid s_1) = p$; in s_2, suppose π_p chooses uniformly at random between a_1 and a_2. Then

$$\begin{aligned} J(\pi_p, R) &= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}(s_t = s_1) \\ &\overset{\text{(i)}}{=} (1-\gamma)\sum_{t=0}^{\infty}\gamma^t (1-p)^t \\ &\overset{\text{(ii)}}{=} \frac{1-\gamma}{1-\gamma(1-p)} \end{aligned} \tag{8}$$

where (i) is due to the fact that remaining in s_1 after t timesteps requires t independent events of $1 - p$ probability, and (ii) uses the formula for sum of an infinite geometric series.

We will prove the proposition using

$$\begin{aligned} \gamma &= \max\left\{1 - \frac{\epsilon\delta}{2\log(2/\delta)}, 1 - \frac{\delta}{2}\right\} \\ \pi &= \pi_{2(1-\gamma)/\delta} \\ \pi_{\text{safe}} &= \pi_{(1-\gamma)\delta/2} \\ \pi_{\text{safe}}{}' &= \pi_q \quad \text{where} \quad q = \max\left\{1 - \frac{1}{2\exp\{2(1/e + c(\delta+\gamma)/\delta)\}},\ (1-\gamma)/\delta\right\}. \end{aligned}$$

To start, we need to show that

$$(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{\text{KL}}(\pi(\cdot \mid s_t) \parallel \pi_{\text{safe}}(\cdot \mid s_t))\right] \leq \epsilon. \tag{9}$$

Since π and π_{safe} are identical at s_2, we need only consider the KL divergence between the policies' action distributions at s_1. Thus we can rewrite the LHS of (9) as

$$\begin{aligned}(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{\text{KL}}(\pi(\cdot \mid s_t) \parallel \pi_{\text{safe}}(\cdot \mid s_t))\right] &= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t(1-2(1-\gamma)/\delta)^t D_{\text{KL}}(\pi(\cdot \mid s_1) \parallel \pi_{\text{safe}}(\cdot \mid s_1)) \\ &= \frac{\delta}{2\gamma+\delta} D_{\text{KL}}(\pi(\cdot \mid s_1) \parallel \pi_{\text{safe}}(\cdot \mid s_1)) \\ &\overset{\text{(i)}}{\leq} \frac{\delta}{2} D_{\text{KL}}(\pi(\cdot \mid s_1) \parallel \pi_{\text{safe}}(\cdot \mid s_1)).\end{aligned}$$

(i) is due to the fact that $\gamma \geq 1-\delta/2$ by definition. Expanding the KL term gives

$$\frac{\delta}{2}\left(2(1-\gamma)/\delta \log\left(\frac{2(1-\gamma)/\delta}{(1-\gamma)\delta/2}\right) + \left(1-2(1-\gamma)/\delta\right)\log\left(\frac{1-2(1-\gamma)/\delta}{1-(1-\gamma)\delta/2}\right)\right). \tag{10}$$

Assuming $\delta < 1$ (otherwise the result is trivially true), we have

$$\begin{aligned}2(1-\gamma)/\delta &> (1-\gamma)\delta/2 \\ 1-2(1-\gamma)/\delta &< 1-(1-\gamma)\delta/2.\end{aligned}$$

This implies that the right log term in (10) is negative, so we can bound (10) as

$$\begin{aligned}&< (1-\gamma)\log\left(\frac{2(1-\gamma)/\delta}{(1-\gamma)\delta/2}\right) \\ &= 2(1-\gamma)\log\left(\frac{2}{\delta}\right) \\ &\overset{\text{(i)}}{\leq} 2\frac{\epsilon\delta}{2\log(2/\delta)}\log\left(\frac{2}{\delta}\right) \\ &= \epsilon,\end{aligned}$$

which is the desired bound in (9). (i) uses the fact that $\gamma \geq 1 - \frac{\epsilon\delta}{2\log(2/\delta)}$ by definition.

Next, we will show that $|J(\pi_{\text{safe}}, R) - J(\pi, R)| \geq 1-\delta$. First, we can calculate the return of π_{safe} using (8):

$$\begin{aligned}J(\pi_{\text{safe}}, R) &= \frac{1-\gamma}{1-\gamma(1-(1-\gamma)\delta/2)} \\ &= \frac{1}{1+\gamma\delta/2} \\ &\overset{\text{(i)}}{\geq} 1-\gamma\delta/2 \\ &\geq 1-\delta/2.\end{aligned} \tag{11}$$

(i) uses the fact that $\frac{1}{1+x} \geq 1-x$ for positive x. The return of π can be calculated similarly as

$$\begin{aligned}J(\pi, R) &= \frac{1-\gamma}{1-\gamma(1-2(1-\gamma)/\delta)} \\ &= \frac{\delta}{2\gamma+\delta} \\ &\overset{\text{(i)}}{\leq} \frac{\delta}{2},\end{aligned} \tag{12}$$

where (i) uses the fact that $\gamma \geq 1-\delta/2$. Combining (11) and (12) gives $|J(\pi_{\text{safe}}, R) - J(\pi, R)| \geq 1-\delta$ as desired.

To prove part 2, consider any π satisfying

$$(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{\text{KL}}(\pi(\cdot \mid s_t) \parallel \pi_{\text{safe}}{}'(\cdot \mid s_t))\right] \leq c.$$

Let $p = \pi(a_2 \mid s_1)$. Then clearly by the definition of π_{safe},

$$(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{\text{KL}}(\pi_p(\cdot \mid s_t) \parallel {\pi_{\text{safe}}}'(\cdot \mid s_t))\right] \leq c, \tag{13}$$

i.e. π_p also satisfies the inequality. Furthermore, note that $J(\pi, R) = J(\pi_p, R)$. We will show that $p \geq (1-\gamma)/\delta$. This will imply that

$$\begin{aligned} J(\pi, R) &= J(\pi_p, R) \\ &= \frac{1-\gamma}{1-\gamma(1-p)} \\ &\leq \frac{1-\gamma}{1-\gamma(1-(1-\gamma)/\delta)} \\ &= \frac{\delta}{\gamma+\delta} && \leq \delta. \end{aligned}$$

Since ${\pi_{\text{safe}}}' = \pi_q$ and $q \geq (1-\gamma)/\delta$ by definition, $J({\pi_{\text{safe}}}', R) \leq \delta$ also. Since the return of both policies must also be nonnegative, this implies $|J({\pi_{\text{safe}}}', R) - J(pi, R)| \leq \delta$, which is the desired bound.

Now, we just need to show that $p \geq (1-\gamma)/\delta$. We do so by contradiction, i.e. assume that $p < (1-\gamma)/\delta$. We can rewrite the LHS of (13) as

$$\underbrace{\frac{1-\gamma}{1-\gamma(1-p)}}_{\text{(a)}}\left[\underbrace{p\log\left(\frac{p}{q}\right)}_{\text{(b)}} + \underbrace{(1-p)\log\left(\frac{1-p}{1-q}\right)}_{\text{(c)}}\right]. \tag{14}$$

We will give lower bounds for each part of (14). For (a), we have

$$\frac{1-\gamma}{1-\gamma(1-p)} > \frac{1-\gamma}{1-\gamma(1-(1-\gamma)/\delta)} = \frac{\delta}{\gamma+\delta}.$$

For (b), note that $q \leq 1$, so

$$p\log\left(\frac{p}{q}\right) \geq p\log p \geq -\frac{1}{e},$$

since the function $f(x) = x\log x$ has its minimum at $f(x) = -1/e$. For (c), note that $1-p > 1-(1-\gamma)/\delta \geq 1-(1-(1-\delta/2))/\delta = 1/2$. Thus we can bound

$$\begin{aligned} (1-p)\log\left(\frac{1-p}{1-q}\right) &> \frac{1}{2}\log\left(\frac{1}{2(1-q)}\right) \\ &\geq \frac{1}{2}\log\left(\frac{1}{2\frac{1}{2\exp\{2(1/e+c(\delta+\gamma)/\delta)\}}}\right) \\ &= \frac{1}{e} + c\frac{\delta+\gamma}{\delta}. \end{aligned}$$

Combining the three bounds on the components of (14) gives

$$\begin{aligned} &(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t D_{\text{KL}}(\pi_p(\cdot \mid s_t) \parallel {\pi_{\text{safe}}}'(\cdot \mid s_t))\right] \\ &> \frac{\delta}{\gamma+\delta}\left[-\frac{1}{e} + \frac{1}{e} + c\frac{\delta+\gamma}{\delta}\right] \\ &= c, \end{aligned}$$

which contradicts (13), thus completing the proof. □

A.2. Proof of Proposition 3.2

We first prove another useful proposition:

Proposition A.1. *The return of a policy π under a reward function R is given by*

$$J(\pi, R) = \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mu_\pi(s,a)R(s,a).$$

Proof. Applying the definitions of return and occupancy measure, we have

$$\begin{aligned}
J(\pi, R) &= (1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t)\right] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} R(s,a)\,\mathbb{P}_\pi\left(s_t = s \wedge a_t = a\right) \\
&= (1-\gamma)\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} R(s,a)\sum_{t=0}^{\infty}\gamma^t\,\mathbb{P}_\pi\left(s_t = s \wedge a_t = a\right) \\
&= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} R(s,a)\,(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t\,\mathbb{1}\left\{s_t = s \wedge a_t = a\right\}\right] \\
&= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mu_\pi(s,a) R(s,a).
\end{aligned}$$

□

According to Proposition A.1, the return of a policy is simply a weighted sum of the reward function, where the weights are given by the occupancy measure. We now prove Proposition 3.2.

Proposition 3.2. *For any MDP, reward function R, and pair of policies π, π_{safe}, we have*

$$|J(\pi_{safe}, R) - J(\pi, R)| \leq \left\|\mu_\pi - \mu_{\pi_{safe}}\right\|_1. \tag{4}$$

Proof. Applying Proposition A.1, Hölder's inequality, and the fact that $R(s,a) \in [0,1]$, we have

$$\begin{aligned}
&|J(\pi_{\text{safe}}, R) - J(\pi, R)| \\
&= \left|\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} (\mu_{\pi_{\text{safe}}}(s,a) - \mu_\pi(s,a))R(s,a)\right| \\
&\leq \left(\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |R(s,a)|\right)\left(\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} |\mu_{\pi_{\text{safe}}}(s,a) - \mu_\pi(s,a)|\right) \\
&\leq \left\|\mu_\pi - \mu_{\pi_{\text{safe}}}\right\|_1.
\end{aligned}$$

□

A.3. Additional results

The following proposition demonstrates that there is always some reward function for which the bound in (4) is tight up to a factor of two.

Proposition A.2. *Fix an MDP and pair of policies π, π_{safe}. Then there is some reward function R such that*

$$|J(\pi_{safe}, R) - J(\pi, R)| \geq \frac{1}{2}\left\|\mu_\pi - \mu_{\pi_{safe}}\right\|_1.$$

Proof. Define two reward functions

$$\begin{aligned}
R_1(s,a) &= \mathbb{1}\{\mu_{\pi_{\text{safe}}}(s,a) \geq \mu_\pi(s,a)\} \\
R_2(s,a) &= \mathbb{1}\{\mu_{\pi_{\text{safe}}}(s,a) \leq \mu_\pi(s,a)\}.
\end{aligned}$$

Using Proposition A.1, we have

$$
\begin{aligned}
&|J(\pi_{\text{safe}}, R_1) - J(\pi, R_1)| + |J(\pi, R_2) - J(\pi_{\text{safe}}, R_2)| \\
&\geq J(\pi_{\text{safe}}, R_1) - J(\pi, R_1) + J(\pi, R_2) - J(\pi_{\text{safe}}, R_2) \\
&= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big(\mu_{\pi_{\text{safe}}}(s,a) - \mu_\pi(s,a)\Big)\Big(R_1(s,a) - R_2(s,a)\Big) \\
&= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big(\mu_{\pi_{\text{safe}}}(s,a) - \mu_\pi(s,a)\Big) \begin{cases} 1 & \mu_{\pi_{\text{safe}}}(s,a) > \mu_\pi(s,a) \\ -1 & \mu_{\pi_{\text{safe}}}(s,a) < \mu_\pi(s,a) \\ 0 & \mu_{\pi_{\text{safe}}}(s,a) = \mu_\pi(s,a) \end{cases} \\
&= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big|\mu_{\pi_{\text{safe}}}(s,a) - \mu_\pi(s,a)\Big| \\
&= \|\mu_\pi - \mu_{\pi_{\text{safe}}}\|_1.
\end{aligned}
$$

Since both of the terms on the first line are positive, one must be at least $\frac{1}{2}\|\mu_\pi - \mu_{\pi_{\text{safe}}}\|_1$, which completes the proof. □

B. Environment details

B.1. Tomato environment

In Figure 3, we have the setup of the tomato environment board we used for training.

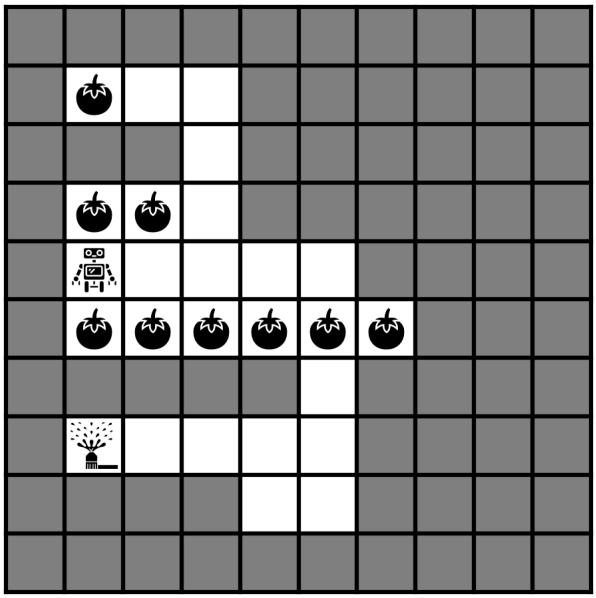


Figure 3. Here, the gray squares represent walls, and the white squares represent open spaces where the agent can travel.

The sprinkler state is down a shallow hallway, and on the other end a tomato is down another shallow hallway. We wanted to try out a scenario where the reward hacking would be relatively difficult for the agent to find to see whether or not our method works for more complex gridworld scenarios.

B.2. Traffic environment

In Figure 4, we have a simplified rendering of the traffic flow environment merge scenario.

Within this particular frame, reward hacking is taking place. As we can see the blue RL vehicle has stopped completely on the on-ramp, resulting in cars to collect behind it. This way, the proxy reward, which is the average velocity of all vehicles

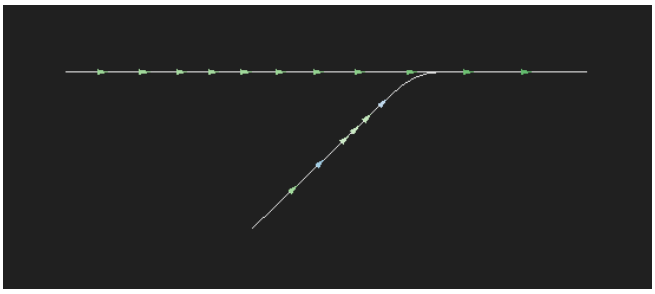


Figure 4. Here, the green cars are controlled by the human driver model IDM controller, and the blue cars are controlled by RL.

in the simulation, is optimized as the cars on the straightway are able to continue speeding along the road without having to wait for merging cars. However, little to no true reward of the average commute time is achieved as the cars on the on-ramp aren't able to continue their commute.

C. Experiment details

Here, we give some extra details about the architectures and hyperparameters we used for training the ORPO agents. We build ORPO using RLLib (Liang et al., 2018) and PyTorch (Paszke et al., 2019). For all RL experiments we train with 5 random seeds and report the median reward.

Network architectures The policy model for both the traffic and tomato environments was a simple fully connected network (FC-net) with a width of 512 and depth of 4. This model size was chosen as it empowered the agents significantly, enough for them to reward hack consistently. The discriminator model for both environments was a simple FC-net with a width of 256 and depth of 4.

Replay buffer We found that our initial implementation of ORPO suffered from instability in the traffic environment. Adversarial training algorithms like GANs are known to be unstable, since they try to use a gradient-based method to find a solution to a saddle point problem (Kodali et al., 2017). To address this instability, we use a replay buffer to train the discriminator. When trajectories are sampled from the learned and safe policies, they are added to a buffer; if the buffer reaches a certain capacity, previously stored trajectories are evicted. Then, the trajectories used to train the discriminator at each iteration are randomly sampled from the replay buffer.

C.1. Hyperparameters

Some hyperparameters for the traffic environment were tuned by Pan et al. (2022).

Hyperparameter	Value (Tomato)	Value (Traffic)
Training iterations	500	250
Batch size	3000	6000
SGD minibatch size	128	6000
SGD epochs per iteration	5	5
Optimizer	Adam	Adam
Learning rate	1e-3	5e-5
Gradient clipping	0.1	0.1
Discount rate (γ)	0.99	0.99
GAE coefficient (λ)	0.98	0.97
Entropy coefficient	0.01	0.01
KL target	0.01	0.02
Value function loss clipping	10	10,000
Value function loss coefficient	0.1	0.5

Table 3. PPO/ORPO hyperparameters.

Hyperparameter	Value (Tomato)	Value (Traffic)
Extra discriminator training batches	2	2
Discriminator reward clipping	1000	10
Replay buffer capacity	100	100
Regularization coefficient (λ)	Varied	Varied

Table 4. ORPO-specific hyperparameters.

For the ORPO-specific parameters, specifying a greater number of extra discriminator training batches means that the discriminator will be trained with that many extra batches sampled from the replay buffer. The replay buffer capacity specifies roughly how many previous iterations of data to keep in the buffer. The coefficient λ that is used for determining how much regularization to apply was varied throughout the experiments and noted in our result tables 1 and 2.