High-Confidence Policy Improvement from Human Feedback

Hon Tik Tse, Philip S. Thomas, Scott Niekum

Keywords: Reinforcement Learning from Human Feedback, High-Confidence Policy Improvement, Imitation Learning and Inverse Reinforcement Learning, Reinforcement Learning

Summary

Reinforcement learning from human feedback (RLHF) aims to learn or fine-tune policies via human preference data when a ground-truth reward function is not known. However, many conventional RLHF methods provide no performance guarantees and have an unacceptably high probability of returning poorly performing policies. We propose Policy Optimization and Safety Test for Policy Improvement (POSTPI), an algorithm that provides high-confidence policy performance guarantees without direct knowledge of the ground-truth reward function, given only a preference dataset. The user of the algorithm may select any initial policy π_{init} and confidence level $1-\delta$, and POSTPI will ensure that the probability it returns a policy with performance worse than π_{init} under the unobserved ground-truth reward function is at most δ. We show theory as well as empirical results in the Safety Gymnasium suite that demonstrate that POSTPI reliably provides the desired guarantee.

Contribution(s)

1. We formalize the problem of high-confidence policy improvement from human feedback (HCPI-HF).
   Context: Reinforcement learning from human feedback has been popular in recent years. However, the problem of performing high-confidence policy improvement from human preference data has not been formalized.
2. To address the HCPI-HF problem, we propose a novel algorithm Policy Optimization and Safety Test for Policy Improvement (POSTPI), and demonstrate both theoretically and empirically that POSTPI reliably provides the desired high-confidence policy improvement guarantee.
   Context: Many prior works in RLHF (Brown et al., 2019b; 2020; Javed et al., 2021; Hejna et al., 2024) provide no performance guarantees on the returned policy. While there exist some works that provide performance guarantees (Zhu et al., 2023; Chen et al., 2022; Xu et al., 2020; Pacchiano et al., 2023; Novoseller et al., 2020; Wang et al., 2023), different from these works, we focus specifically on the setting of improving with respect to a user-provided policy with high probability.
3. We propose a novel policy optimization objective that allows POSTPI to return a policy with high probability when the initial policy is sub-optimal, and derive the gradient of this objective.
   Context: Unlike PG-BROIL (Javed et al., 2021), which optimizes the conditional value-at-risk, we optimize the value-at-risk, and explicitly allow the objective to depend on the user-provided initial policy.
4. We propose a novel method for computing high-confidence policy performance bounds in the RLHF setting.
   Context: Unlike a prior approach (Brown et al., 2020), which only considers the uncertainty in the ground-truth reward function, our approach further considers the uncertainty in using a finite number of rollouts to estimate the expected value of a policy.

High-Confidence Policy Improvement from Human Feedback

Hon Tik Tse[1,2], **Philip S. Thomas**[3], **Scott Niekum**[3]
hontik@ualberta.ca, {pthomas,sniekum}@cs.umass.edu

[1]**Department of Computing Science, University of Alberta**
[2]**Alberta Machine Intelligence Institute (Amii)**
[3]**Manning College of Information and Computer Sciences, University of Massachusetts**

Abstract

Reinforcement learning from human feedback (RLHF) aims to learn or fine-tune policies via human preference data when a ground-truth reward function is not known. However, many conventional RLHF methods provide no performance guarantees and have an unacceptably high probability of returning poorly performing policies. We propose Policy Optimization and Safety Test for Policy Improvement (POSTPI), an algorithm that provides high-confidence policy performance guarantees without direct knowledge of the ground-truth reward function, given only a preference dataset. The user of the algorithm may select any initial policy π_{init} and confidence level $1-\delta$, and POSTPI will ensure that the probability it returns a policy with performance worse than π_{init} under the unobserved ground-truth reward function is at most δ. We show theory as well as empirical results in the Safety Gymnasium suite that demonstrate that POSTPI reliably provides the desired guarantee.

1 Introduction

In recent years, reinforcement learning (RL) has found success in many areas, including video games (Mnih et al., 2015; Vinyals et al., 2019), board games (Silver et al., 2016), and healthcare (Yu et al., 2023). These successes rely on the specification of an appropriate reward function that allows an agent to learn the desirable behavior. However, the translation of desirable behavior to an actual reward function can be difficult, especially for complex problems, and misspecified reward functions can lead to undesirable behavior such as reward hacking (Skalse et al., 2022; Pan et al., 2022).

Reinforcement learning from human feedback (RLHF) is one popular technique to address this problem. Instead of relying on a pre-specified reward function, RLHF aims to learn or fine-tune a policy under a reward function inferred from human preference data. In light of its power to learn human-desired behavior from only human preferences, RLHF has found applications in many areas, such as improving RL policies (Christiano et al., 2017), fine-tuning large language models (Ouyang et al., 2022) and improving text-to-image models (Lee et al., 2023; Wu et al., 2023).

However, RLHF still suffers from the following problems. First, optimizing with respect to a learned reward function for too long can hinder performance under the ground-truth reward function of the preference provider, known as *over-optimization*, since the learned reward function is often an imperfect proxy (Gao et al., 2023). Second, even minor misalignment between human intent and the learned reward function can lead to severe performance loss (Zhuang & Hadfield-Menell, 2020). Unfortunately, misalignment can easily arise due to human errors, finite data, or biases in data collection (Casper et al.). In light of these issues, it is in general not guaranteed that policies learned using RLHF will perform well under the ground-truth reward function. As we will demonstrate

in experiments, state-of-the-art RLHF methods often return poorly performing policies with non-negligible probabilities. Furthermore, without access to the ground-truth reward function, we cannot evaluate the performance of the learned policies under it to decide whether to employ these policies. In safety-critical applications where a poorly performing policy can be dangerous, simply employing policies learned from conventional RLHF methods can lead to undesirable outcomes. Therefore, an algorithm that returns a policy with high-confidence performance guarantees in the absence of the ground-truth reward function is especially important.

To solve this problem, we propose Policy Optimization and Safety Test for Policy Improvement (POSTPI). POSTPI consists of two components: 1) candidate proposal, which proposes a candidate policy for the algorithm to return, and 2) the safety test, which evaluates whether the proposed policy is safe to return. The key idea is that the safety test acts as a gatekeeper that only accepts candidate policies that are deemed better than the initial policy under the ground-truth reward function with high confidence. When a candidate policy is not deemed better with sufficient confidence, the algorithm returns *No Solution Found (NSF)*. Assuming a correct model of the preference data, given any initial policy and confidence level $1 - \delta$, POSTPI guarantees that the probability it returns a policy worse than the initial policy under the ground-truth reward function is at most δ. Given the form of this guarantee, POSTPI can be considered a type of *Seldonian* algorithm (Thomas et al., 2019).

In the safety test, we propose a novel method of computing high-confidence performance bounds, which explicitly considers both the uncertainty in the ground-truth reward function, and the uncertainty associated with using rollouts to estimate the expected value of a policy. On the other hand, despite the high-confidence guarantee provided by POSTPI, if candidate proposal proposes policies that are likely to be rejected by the safety test, POSTPI returns NSF frequently and has little practical use. To address this issue, we propose to optimize a novel objective for candidate proposal, which allows POSTPI to return a policy with high probability when the user-provided initial policy is sub-optimal. We provide a derivation of the gradient of this objective, which can be optimized with any policy gradient algorithms.

We prove that POSTPI ensures policy improvement with high confidence and also compare our algorithm with several state-of-the-art RLHF algorithms on two domains from the Safety Gymnasium suite (Ji et al., 2023). We demonstrate empirically that out of the algorithms we tested, POSTPI is the only one that performs policy improvement at a user-specified probability level. Furthermore, we find empirically that most policies accepted by the safety test are improvements over the initial policy under the ground-truth reward function. To the best of our knowledge, our work is the first to ensure high-confidence policy improvement in the RLHF setting.

2 Related Work

2.1 Reinforcement Learning from Human Feedback

Reinforcement learning from human feedback (RLHF) aims to learn or fine-tune a policy using only preferences over trajectories and has received much attention recently.

RLHF typically involves two steps: 1) learning a reward function from preferences, which are usually pairwise comparisons of possibly partial and sub-optimal trajectories, and 2) policy optimization using the learned reward function. Some work requires active query for human preferences (Christiano et al., 2017; Lee et al., 2021a; Ibarz et al., 2018; Palan et al., 2019; Hejna & Sadigh, 2022; Shin et al., 2021; Lee et al., 2021b). Some consider learning a reward function from an offline dataset of preferences before using an RL algorithm for policy optimization. T-REX (Brown et al., 2019a) treats learning the reward function from an offline preference dataset as a supervised learning problem, while B-REX infers a Bayesian posterior distribution over reward functions (Brown et al., 2020). D-REX and SSRR automatically generate preferences by injecting noise into trajectories generated from a learned policy when preferences are not available (Brown et al., 2019b; Chen et al., 2020). Sikchi et al. (2023) utilize both an offline preference dataset and automatically generated preferences. PG-BROIL builds on B-REX and optimizes a policy while taking the epistemic

uncertainty in the reward into consideration (Javed et al., 2021). Recently, Hejna et al. (2024) and Rafailov et al. (2024) convert RLHF into a supervised learning task, circumventing the need to learn a reward function. However, none of these provide probabilistic guarantees on the performance of policies learned from only an offline and finite preference dataset. While there exist other works that provide performance guarantees (Zhu et al., 2023; Chen et al., 2022; Xu et al., 2020; Pacchiano et al., 2023; Novoseller et al., 2020; Wang et al., 2023), different from these works, we focus specifically on the setting of improving with respect to a user-provided policy with high probability.

2.2 Safety in Reinforcement Learning

In this subsection, we review the most closely related work in RL. The Seldonian framework focuses on providing safety guarantees with high confidence, where the definition of safety is chosen by the user of the algorithm (Thomas et al., 2019). Note that Seldonian algorithms are a class of algorithms, and there is no single algorithm that is referred to as the Seldonian algorithm. However, Seldonian algorithms typically involve candidate selection and safety test mechanisms. Candidate selection proposes a solution to be returned by the algorithm, while the safety test evaluates whether the proposed solution is safe to return. Seldonian algorithms allow users to specify the definition of safety, and provide high-confidence guarantees that a solution returned will not produce unsafe behavior. In this paper, we present an algorithm having a similar structure. However, unlike a typical Seldonian algorithm where the user provides the safety definition, we define safety as improvement with respect to a user-specified initial policy under the ground-truth reward function.

Much work has considered computing high-confidence bounds in RL when a reward function is available. Thomas et al. (2015a) focus on providing high-confidence guarantees that the learned policy is not worse than a user-selected threshold. Other work focuses on high-confidence off-policy evaluation (Thomas et al., 2015b; Hanna et al., 2017). Chandak et al. (2021) consider the high-confidence off-policy estimation of the variance of returns. In the absence of a reward function, Brown & Niekum (2018) provide high-confidence bounds of performance, but require solving an MDP in the inner loop. B-REX (Brown et al., 2020) provides a way to compute high-confidence performance bounds efficiently from high-dimensional visual trajectories in the RLHF setting. However, B-REX has not considered using the computed bounds to guide a policy search. Furthermore, such bounds do not take the uncertainty of estimating the performance of a policy from rollouts into consideration. As we will demonstrate in experiments, these bounds can in fact be overly optimistic.

3 Preliminaries

3.1 Markov Decision Process

We model the environment as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, R, T, d_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \to \mathbb{R}$ is the reward function, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, d_0 is the initial state distribution, and γ is the discount factor. At every time step t, the agent observes the state S_t and selects an action A_t. After executing the action A_t, the environment transitions to S_{t+1} and the agent receives a reward $R(S_t)$. We consider a stochastic policy π mapping from states to a probability distribution over actions. We denote the expected value of a policy π under the reward function R by $J(\pi, R) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(S_t)]$.

3.2 RLHF

In the RLHF setting, we do not have access to the ground-truth reward function, denoted as R^*, and are left with an MDP\R $(\mathcal{S}, \mathcal{A}, T, d_0, \gamma)$. We assume access to a labeled dataset of preferences over pairwise trajectories $\mathcal{P} = \{\tau_i \prec \tau_j\}_{(i,j)}$, where τ_j is preferred over τ_i. In order to provide our high-confidence guarantee, we need to reason about the epistemic uncertainty in the ground-truth reward function R^* given the preference data $\mathcal{P}$. To do this, we apply B-REX (Brown et al., 2020). B-REX, similar to Christiano et al. (2017), assumes that the preferences follow the Bradley-Terry

model (Bradley & Terry, 1952), and infers the posterior distribution over reward functions $P(R|\mathcal{P})$ given the preferences $\mathcal{P}$. B-REX then uses Markov chain Monte Carlo (MCMC) to generate reward samples from the Bayesian posterior distribution $P(R|\mathcal{P})$ in an efficient manner. These reward samples are central to the ability of our algorithm to provide our probabilistic guarantee. We denote the set of reward samples by $\mathcal{R} = \{r_i\}$, where each r_i is a distinct reward sample.

3.3 Value-at-Risk

Risk measures have been used as optimization criteria in RL (García & Fernández, 2015). Given a risk-aversion parameter $\alpha \in [0, 1]$, the VaR_α of a random variable X is the largest value that X exceeds with probability at least α. Mathematically, it is the $(1-\alpha)$-quantile of X:

$$\mathrm{VaR}_\alpha[X] = \sup\{x : \Pr(X \geq x) \geq \alpha\}. \tag{1}$$

4 High-Confidence Policy Improvement from Human Feedback

Before detailing our approach, we first formalize the problem of high-confidence policy improvement from human feedback. We consider the RLHF setting and model the problem as an MDP\R, where the ground-truth reward function R^* is not available. We assume access to a labeled dataset of preferences over pairwise trajectories $\mathcal{P} = \{\tau_i \prec \tau_j\}_{(i,j)}$, where τ is a possibly partial trajectory comprised of either states or state-action pairs, and the trajectory τ_j is preferred over τ_i. We assume access to an initial policy π_{init} and a confidence level $1-\delta$. The *High-Confidence Policy Improvement from Human Feedback* (HCPI-HF) problem is to return a solution π_{return} such that

$$\Pr(J(\pi_{\text{return}}, R^*) \geq J(\pi_{\text{init}}, R^*)) \geq 1-\delta. \tag{2}$$

Note that we allow an algorithm to indicate that it has not been able to find an improved policy by returning *No Solution Found* (NSF). We define $J(\text{NSF}, R^*) := J(\pi_{\text{init}}, R^*)$ since NSF is considered safe and not worse than the initial policy π_{init}.

In Equation 2, π_{return} is the only term that is random. π_{return} is determined using an algorithm taking the set of preferences $\mathcal{P}$, the initial policy π_{init}, and the confidence level $1-\delta$ as input. We assume that we are given a set of trajectories, which are fixed and have no randomness. However, we assume that the preferences over the trajectories $\mathcal{P}$ are random. For example, in a practical scenario, such preferences will be provided by a human, and if a human were to assign preferences multiple times, the preferences may slightly vary every time. On the other hand, π_{init} and the confidence level $1-\delta$ are provided by the user of the algorithm and have no randomness. Apart from the randomness in the preferences $\mathcal{P}$, the algorithm also causes randomness in π_{return}. Common sources of randomness in an algorithm involving policy optimization include the on-policy rollouts collected, and the randomly initialized policy and value networks.

5 Policy Optimization and Safety Test for Policy Improvement

In this section, we describe our approach *Policy Optimization and Safety Test for Policy Improvement* (POSTPI) for addressing the HCPI-HF problem. Our approach consists of two main components: 1) candidate proposal, and 2) the safety test. In candidate proposal, we perform policy optimization and return a candidate policy π_C. The candidate policy π_C is then subject to the safety test, where we determine whether π_C is an improvement over π_{init} with high confidence. A trivial design of the safety test is to simply return NSF with probability $1-\delta$, and return the candidate policy with probability δ, regardless of the performance of the candidate policy. While this is a valid solution to the HCPI-HF problem, this solution still returns NSF with high probability, even in scenarios where a candidate policy better than the initial policy can be easily learned. We now describe our design of the safety test, which considers the performance of the candidate policy. As we demonstrate later in our experiments, our safety test does not ensure safety simply by returning NSF with high probability. The

frequency that our safety test returns a solution other than NSF, when provided with a safe policy, increases as the amount of data increases, and tends towards one with reasonable amounts of data.

5.1 Safety Test

The safety test in POSTPI is the mechanism by which the high-confidence guarantee of our proposed approach is provided. It determines whether the candidate policy π_C returned by candidate proposal is an improvement over the initial policy π_{init} under the ground-truth reward function R^* with high confidence. If the safety test is sufficiently confident that π_C is an improvement, the algorithm returns π_C. Otherwise, the algorithm returns *No Solution Found (NSF)*.

To determine whether π_C passes the safety test, we are interested in the following inequality:

$$J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*) \geq 0. \tag{3}$$

Ideally, we want to return π_C as the output of the algorithm if this inequality holds, and return NSF otherwise. However, we cannot directly compute $J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*)$, for two reasons. First, we do not have access to the ground-truth reward function R^*. Second, the expected value $J(\pi, R)$ usually cannot be computed exactly for arbitrary π and R in practice, for example, due to a lack of access to the transition function T. Often, $J(\pi, R)$ can only be estimated using rollouts.

Instead of computing $J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*)$ directly, we compute a high-confidence lower bound on this quantity. We first define a high-confidence lower bound formally:

Definition 5.1 $\text{HCLB}(\theta, 1-\delta)$ *denotes the high-confidence lower bound on a parameter θ with confidence level $1-\delta$, i.e.,* $\Pr(\text{HCLB}(\theta, 1-\delta) \leq \theta) \geq 1-\delta$.

Then, we are interested in computing the high-confidence lower bound $\text{HCLB}(J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*), 1-\delta)$. If this lower bound is greater than or equal to 0, we return π_C. Otherwise, we return NSF. An algorithm following this approach of using this high-confidence lower bound to determine whether to return π_C or NSF always satisfies Equation 2, and solves the HCPI-HF problem. When π_C is actually better than π_{init}, regardless of whether we return π_C or NSF, the returned solution is not worse than π_{init}. When π_C is worse than π_{init}, i.e., $J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*) < 0$, the high-confidence lower bound computed is less than 0 with probability at least $1-\delta$, so we return π_C with probability at most δ. Note that the guarantee that this approach provides is independent of the candidate policy π_C. For example, we can return random candidate policies in candidate proposal, and the safety test will still ensure that the entire algorithm provides the desired guarantee.

We now describe how to compute $\text{HCLB}(J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*), 1-\delta)$. To do so, we need to reason probabilistically about the uncertainty associated with $J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*)$, which includes 1) the uncertainty in the ground-truth reward function R^*, and 2) the uncertainty in estimating $J(\pi, r)$ using a finite number of rollouts.

To address the first source of uncertainty, we apply B-REX (Brown et al., 2020), which infers the posterior distribution over reward functions $P(R|\mathcal{P})$ given the preferences $\mathcal{P}$. By sampling from this posterior distribution, we can utilize the reward samples to reason about the uncertainty in R^* probabilistically. Note that this source of uncertainty is not perfectly accounted for, since we do not have the analytical form of the posterior distribution, and are only drawing a finite number of samples from it. However, since sampling from the posterior distribution using B-REX is computationally cheap, we can simply draw a large number of reward samples.

On the other hand, it is not always possible to generate a large number of rollouts to address the uncertainty in using a finite number of rollouts to approximate policy values $J(\pi, r)$, for example, in safety-critical applications where generating rollouts using an unsafe policy can be dangerous, or in applications where generating rollouts is very expensive. In scenarios where only a small number of rollouts can be generated, not explicitly accounting for this source of uncertainty can lead to unreliable performance bounds. As we demonstrate in Supplementary Materials D.1, when the number of rollouts is small, a prior approach (Brown et al., 2020), which only accounts for the uncertainty

Algorithm 1 Safety Test

Input: candidate policy π_C, initial policy π_{init}, a set of reward samples $\mathcal{R}$, confidence level parameter δ
Compute $\text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1 - \delta/2)$ using rollouts for all $r \in \mathcal{R}$.
Compute L as the $(\delta/2)$-quantile of $\text{HCLB}(J(\pi_C, R) - J(\pi_{\text{init}}, R), 1 - \delta/2)$.
if $L \geq 0$ **then**
  return π_C
else
  return NSF
end if

in R^* and does not consider the uncertainty in using a finite number of rollouts, can compute overly optimistic lower bounds on $J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*)$. A safety test using this approach to compute high-confidence lower bounds fails to ensure the desired high-confidence guarantee.

To address this issue, we propose a novel approach to compute an estimate of the desired high-confidence lower bound, which explicitly accounts for both sources of uncertainty. First, we sample from the posterior distribution $P(R|\mathcal{P})$ to obtain a set of reward samples $\mathcal{R}$. Then, we directly compute *a single quantity*, denoted by L, that, with probability $1 - \delta/2$, is *simultaneously* a lower bound on $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ for $1 - \delta/2$ portion of the reward samples in $\mathcal{R}$.

We now show that L is an estimate of the desired high-confidence lower bound $\text{HCLB}(J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*), 1 - \delta)$. Assuming that we have an infinite number of reward samples, we know the following two facts: First, a quantity that lower bounds $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ for $1 - \delta/2$ portion of the reward samples is a lower bound on $J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*)$ with probability $1 - \delta/2$. Second, the *single quantity* L that we compute is such a lower bound with probability $1 - \delta/2$. Therefore, by Boole's inequality, this *single quantity* L is $\text{HCLB}(J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*), 1 - \delta)$. Note that, in practice, we only use a finite number of reward samples, so the L we compute is only an estimate of $\text{HCLB}(J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*), 1 - \delta)$. To compute L, we prove the following theorem:

Theorem 5.2 *Consider a set of reward samples $\mathcal{R}'$. $\min_{r \in \mathcal{R}'} \text{HCLB}(J(\pi_C, r) - J(\pi_{init}, r), 1 - \delta)$ is a lower bound on $J(\pi_C, r) - J(\pi_{init}, r)$ simultaneously for all $r \in \mathcal{R}'$ with probability $1 - \delta$.*

We now present a proof sketch for Theorem 5.2, and defer the full proof to Supplementary Materials A. Consider the reward sample $r' = \arg\min_{r \in \mathcal{R}'} J(\pi_C, r) - J(\pi_{\text{init}}, r)$. The high-confidence lower bound $\text{HCLB}(J(\pi_C, r') - J(\pi_{\text{init}}, r'), 1 - \delta)$ is at the same time a high-confidence lower bound of $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ for all $r \in \mathcal{R}'$. However, we cannot identify r' as we cannot compute $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ exactly. To address this, we compute $\min_{r \in \mathcal{R}'} \text{HCLB}(J(\pi_C, r - J(\pi_{\text{init}}, r)), 1 - \delta)$, which lower bounds $\text{HCLB}(J(\pi_C, r') - J(\pi_{\text{init}}, r'), 1 - \delta)$.

Using Theorem 5.2, we can pick any subset of reward samples $\mathcal{R}'$ that contains at least $1 - \delta/2$ portion of the reward samples, and compute L as $\min_{r \in \mathcal{R}'} \text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1 - \delta/2)$. One obvious choice of $\mathcal{R}'$ is simply the $1 - \delta/2$ portion of reward samples that correspond to the highest values of $\text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1 - \delta/2)$. If we choose this $\mathcal{R}'$, L is simply the $(\delta/2)$-quantile of $\text{HCLB}(J(\pi_C, R) - J(\pi_{\text{init}}, R), 1 - \delta)$. We can then use L, which is an estimate of $\text{HCLB}(J(\pi_C, R^*) - J(\pi_{\text{init}}, R^*), 1 - \delta)$, to determine whether to accept or reject the candidate policy. The full algorithm of the safety test is shown in Algorithm 1.

In practice, we compute $\text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1 - \delta/2)$ using Student's t-test. Note that we can replace Student's t-test with other statistical tests. We use Student's t-test in all of our experiments as it is easy to compute and works well in practice.

Algorithm 2 Candidate Proposal

Input: the initial policy π_{init}, a set of reward samples $\mathcal{R}$, confidence level parameter δ
Initialize policy π_θ.
repeat
  Estimate $J(\pi_\theta, r) - J(\pi_{\text{init}}, r)$ for all $r \in \mathcal{R}$ using rollouts.
  Estimate $\frac{\partial}{\partial\theta} \text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]$ using Equation 5.
  Perform one step of gradient ascent.
until convergence
return π_θ

5.2 Candidate Proposal

In candidate proposal, we perform policy optimization to propose a candidate policy π_C. In fact, we can employ any algorithm to optimize a policy. This is because the high-confidence guarantee provided by our algorithm only relies on the safety test, and is not contingent on candidate proposal. However, policy optimization methods not taking the knowledge of the safety test into account will likely produce candidate policies π_C that do not pass the safety test. This will lead to the algorithm outputting NSF frequently, reducing its practical use. In this subsection, we present a candidate proposal mechanism specifically designed to return policies that will likely pass the safety test.

Recall that in the safety test, we accept a candidate policy π_C when the $(\delta/2)$-quantile of $\text{HCLB}(J(\pi_C, R) - J(\pi_{\text{init}}, R), 1-\delta/2)$ is greater than or equal to 0. To increase the probability of a candidate policy being accepted, we propose to directly maximize an estimate of this $(\delta/2)$-quantile, leading to the following objective:

$$\text{VaR-EVD} = \text{VaR}_{1-\delta/2}[J(\pi_C, R) - J(\pi_{\text{init}}, R)], \tag{4}$$

where $\text{VaR}_{1-\delta}$ of a random variable is equivalent to the δ-quantile of the random variable. Note that we do not compute the high-confidence lower bounds of $J(\pi_C, R) - J(\pi_{\text{init}}, R)$ in the objective for efficiency. The VaR-EVD objective (**VaR** of the **E**xpected **V**alue **D**ifference) allows plugging in any initial policy π_{init} in the form of $J(\pi_{\text{init}}, R)$. By allowing the specification of an initial policy in the objective, maximizing our objective increases the probability of obtaining a policy that is an improvement over the initial policy.

We now present the gradient of the VaR-EVD objective. Consider a policy π_θ parameterized by θ. Let $r \in \mathcal{R}$ be the reward sample that satisfies the condition $J(\pi_\theta, r) - J(\pi_{\text{init}}, r) = \text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]$. The gradient is:

$$\frac{\partial}{\partial\theta} \text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)] = \frac{\partial}{\partial\theta} J(\pi_\theta, r). \tag{5}$$

We now provide a high-level description of the derivation, and defer the full derivation to Supplementary Materials B. The gradient measures the change in $\text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]$ when the policy parameters θ change. When the change in θ is small enough, the changes induced in $J(\pi_\theta, R) - J(\pi_{\text{init}}, R)$ do not change which reward sample r satisfies $J(\pi_\theta, r) - J(\pi_{\text{init}}, r) = \text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]$. We can therefore simply compute the policy gradient under r.

We present a general algorithm for candidate proposal in Algorithm 2. Note that the gradient in Equation 5 can be optimized with any policy gradient algorithm. For example, it can be optimized with PPO by just using the advantage of the reward sample r in the clipped surrogate objective (Schulman et al., 2017). In our experiments, we use the version of Algorithm 2 using PPO.

We now describe some practical considerations. In both the safety test and candidate proposal, we use the initial policy π_{init} for generating rollouts for computing estimates of $J(\pi_{\text{init}}, r)$ for all $r \in \mathcal{R}$. In practice, these estimates can be computed in advance to any desired level of accuracy. One common misconception is that, when estimating $J(\pi, r)$ for different r's, we need to generate

distinct rollouts for each r. In fact, we only need to use π to generate one set of rollouts, and then evaluate the same set of rollouts under different r's. This is because the choice of the reward sample r only changes the rewards received, and has no impact on the trajectories. If the rewards can be computed efficiently, increasing the number of reward samples induces little computational overhead. Another practical consideration is related to the set of reward samples $\mathcal{R}$. In the case of drawing an infinite number of reward samples, using the same set for both candidate proposal and the safety test does not introduce any bias. However, when using a finite sample, using the same set of reward samples for both policy optimization and the safety test could induce bias in the safety test. To reduce bias, we generate two sets of reward samples of the same size, one for candidate proposal and one for the safety test.

6 Experiments

In this section, we want to answer the following questions: (1) Do empirical studies support our theoretical claims that our algorithm achieves policy improvement with high confidence? (2) How does the probability of returning a policy vary for different initial policies π_{init}? To answer these questions, we perform experiments in two domains, *Circle* and *Goal*, from the Safety Gymnasium suite (Ji et al., 2023). In Circle, the agent has to travel as fast as possible along the circumference of a large circle, while avoiding a forbidden region that overlaps with the circle. In Goal, the agent has to navigate to goals randomly generated on the map while avoiding dangerous regions called hazards. Details of the two domains can be found in Supplementary Materials C. We further justify the design of our candidate proposal and safety test by comparisons with state-of-the-art alternatives. We also apply POSTPI to high-dimensional image inputs. Details and results of these additional experiments can be found in Supplementary Materials D.

6.1 High-Confidence Policy Improvement

In this subsection, we aim to find out whether experiments support our claims that our algorithm achieves policy improvement with high confidence. Note that the claim that our algorithm performs high-confidence policy improvement is supported primarily by theory, and the experiments merely serve to provide empirical support for established theory.

We compare our algorithm to the following state-of-the-art RLHF baselines: T-REX (Brown et al., 2019a), B-REX (Brown et al., 2020), PG-BROIL (Javed et al., 2021) and CPL (Hejna et al., 2024). For B-REX, we consider optimizing both the mean and MAP rewards. Apart from PG-BROIL, these baselines do not explicitly consider the uncertainty in the ground-truth reward function when performing policy optimization. While PG-BROIL reasons about the ground-truth reward probabilistically, it does not provide performance guarantees on the learned policy. Unlike POSTPI, these baselines were not designed to address the HCPI-HF problem, so poor performance at high-confidence policy improvement should not be misconstrued as experimental evidence that these baselines are not effective for the settings that they were designed for. Nevertheless, these are the most relevant baselines to help us understand 1) the consequences of a lack of performance guarantees, and 2) the benefits of POSTPI. All of these methods, including POSTPI, require a preference dataset. B-REX, PG-BROIL, and POSTPI further involve sampling from the posterior distribution over reward functions given preferences $P(R|\mathcal{P})$. Details of preference label and reward sample generation, and hyperparameter settings can be found in Supplementary Materials C and E respectively.

Our algorithm POSTPI requires specifying the confidence level $1-\delta$ and the initial policy π_{init}. In all of our experiments, we use a confidence level of 0.95, i.e., $\delta = 0.05$. To examine the performance of our algorithm under different initial policies, we generate a range of initial policies with varying performance. We consider a set of initial policies, denoted as $\pi^{\epsilon}_{\text{init}}$, where $\epsilon \in [0, 1]$ is a parameter determining the level of performance of the initial policy. The larger the ϵ, the better the initial policy, with π^{1}_{init} corresponding to a policy trained under the ground-truth reward till convergence, and π^{0}_{init} corresponding to a policy that always receives 0 reward. In our experiments, we consider $\epsilon \in \{0, 0.25, 0.5, 0.75, 1\}$. Details on the initial policies can be found in Supplementary Materials C.

For all algorithms, we evaluate the probability of returning a policy worse than the initial policy under the ground-truth reward function R^* over 20 trials. We approximate the expected value of the returned and initial policies under the ground-truth reward function with 200 rollouts. Since our algorithm ensures policy improvement with high confidence, we hypothesize that this probability for POSTPI is at most $\delta = 0.05$, while the probabilities for baselines may exceed δ. Recall that our algorithm requires generating rollouts to compute high-confidence lower bounds of $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ for all $r \in \mathcal{R}$ in the safety test. We vary the number of rollouts we generate from 2 to 1000 and report the maximum probability of returning a policy worse than $\pi^{\epsilon}_{\text{init}}$. This is to demonstrate that, since our algorithm accounts for the uncertainty in expected value estimates, our algorithm provides the desired guarantee regardless of the number of rollouts used for the safety test.

Table 1 shows the probability of returning a policy worse than $\pi^{\epsilon}_{\text{init}}$ for different ϵ for the Circle and Goal domains. It can be seen that our algorithm, as predicted by our theoretical work, returns a policy worse than $\pi^{\epsilon}_{\text{init}}$ with probability not more than the selected $\delta = 0.05$. We also demonstrate that there are in general no guarantees on the performance of policies returned by contemporary RLHF algorithms. T-REX and B-REX return poorly performing policies with high probability regardless of ϵ. PG-BROIL and CPL, on the other hand, are able to more frequently return policies performing better than $\pi^{\epsilon}_{\text{init}}$ when ϵ is small. However, they still have non-negligible probabilities of returning policies worse than the initial policies. Out of the algorithms tested, only POSTPI returns a policy not worse than any chosen initial policy at a user-specified probability.

Table 1: Probability of returning a policy worse than $\pi^{\epsilon}_{\text{init}}$ over 20 trials in Circle and Goal.

DOMAIN	**CIRCLE**					**GOAL**				
ϵ	0	0.25	0.5	0.75	1	0	0.25	0.5	0.75	1
T-REX	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B-REX (Mean)	1.0	1.0	1.0	1.0	1.0	0.8	0.85	0.9	0.95	1.0
B-REX (MAP)	0.7	0.7	0.7	0.75	0.95	0.8	0.85	0.9	1.0	1.0
PG-BROIL	**0.0**	**0.0**	**0.0**	0.1	0.9	0.1	0.2	0.35	0.7	0.9
CPL	0.6	0.65	0.7	0.9	1.0	0.35	0.45	0.7	0.85	1.0
POSTPI (Ours)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

6.2 Probability of Returning a Policy

Our algorithm returns either a policy or NSF. Although our algorithm provides a high-confidence guarantee on policy improvement, if our algorithm returns NSF most of the time, it has little practical use. We now present the probability of POSTPI returning a policy, computed over 20 trials.

Figure 1 shows the probability of POSTPI returning a policy over 20 trials in the Circle (left) and Goal (right) domains for different initial policies $\pi^{\epsilon}_{\text{init}}$. For the Circle domain, when $\epsilon < 1$, i.e., when the initial policies are sub-optimal and there is room for improvement, POSTPI returns a policy with high probabilities (≥ 0.75). For the Goal domain, POSTPI returns a policy with probability 0.55 when $\epsilon = 0.75$, and returns a policy with high probabilities (≥ 0.9) when $\epsilon \leq 0.5$. For both domains, POSTPI returns a policy with very low probability when $\epsilon = 1$, which is reasonable since improving over π^{1}_{init}, which is trained under the ground-truth reward function till convergence, is difficult. To conclude, POSTPI, by optimizing the useful VaR-EVD objective in candidate proposal, is capable of proposing policies that are likely to pass the safety test when the initial policy is sub-optimal.

Our algorithm provides the high-confidence guarantee that the solution returned (either the candidate policy π_C or NSF) is not worse than the initial policy. In the NSF case, it is trivial that this holds. In the other case, it is informative to look at whether the accepted π_C is actually an improvement over the initial policy with high probability. Note that our algorithm does not provide any guarantees

on this probability. For example, if an algorithm always returns a policy worse than π_{init} as π_C in candidate proposal, this probability will always be 0. We observe that in our experiments, a policy accepted by the safety test is always an improvement over the initial policy in both domains. While not guaranteed by our algorithm, a returned policy has a very high probability of being an actual improvement over the initial policy.

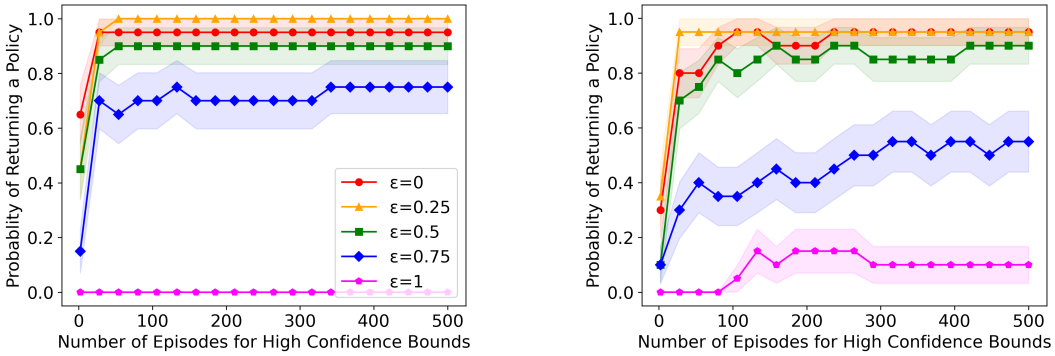

Figure 1: The probability of returning a policy in the Circle (left) and Goal (right) domains over 20 trials for different initial policies $\pi^{\epsilon}_{\text{init}}$. The shaded areas indicate ± 1 standard error.

7 Discussion

In this work, we formalize the problem of high-confidence policy improvement from human feedback (HCPI-HF). We introduce a novel algorithm POSTPI to address this problem. We propose a novel VaR-EVD objective for policy optimization and provide a derivation of its gradient that can be optimized with any standard policy gradient algorithms. We propose a safety test that takes both the uncertainty in the expected value estimates of the evaluated policies and the uncertainty in the ground-truth reward function into consideration. We provide both theoretical and empirical evidence that POSTPI provides the high-confidence guarantee that the solution returned is not worse than a user-specified initial policy. Furthermore, we find that empirically, policies returned by POSTPI are very frequently better than the initial policy.

Limitations. The guarantee provided by our algorithm relies on a few assumptions: 1) The Bradley-Terry model (Bradley & Terry, 1952) is an accurate model of human preferences. However, as pointed out by Laidlaw & Dragan (2022), this is likely not the case. Nevertheless, we expect that our work can be generalized to support other models of human preferences. Note, also, that as long as the model of preferences matches the model used in preference annotation, our theoretical guarantees hold, and can handle noise and potentially cyclic preferences. 2) The set of reward samples $\mathcal{R}$ accurately represents the posterior distribution over reward functions given preferences $P(R|\mathcal{P})$. In the case of the Bradley-Terry model, this requires a matching inverse temperature parameter β during preference label generation and Bayesian inference. This is easy to ensure when generating preference labels ourselves for experiments, but can be difficult when using preference labels annotated by humans. This assumption also requires enough samples from the posterior, which can be relatively easier to overcome by simply generating a large number of reward samples using B-REX (Brown et al., 2020), though it necessitates a trade-off between the efficiency of the algorithm and the soundness of the provided guarantee. 3) The ground-truth reward function lies in the space of reward samples. In our experiments, similar to Javed et al. (2021), we hand-craft state features that allow the ground-truth reward function to be expressed linearly in the state features. In many real-world applications, such construction of state features is often not feasible. While B-REX (Brown et al., 2020) uses a neural network to automatically learn the state features, the size of the neural network required for the learned state features to be expressive enough can be hard to determine in practice. Future work can focus on addressing these issues.

Acknowledgments

We thank reviewers for pointing out related works and providing suggestions to improve the paper. This work has taken place in part in the Safe, Correct, and Aligned Learning and Robotics Lab (SCALAR) at The University of Massachusetts Amherst. SCALAR research is supported in part by the NSF (IIS-2323384) and the Long-Term Future Fund.

References

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: The method of paired comparisons. *Biometrika*, 39(3-4):324–345, 12 1952.

Daniel Brown and Scott Niekum. Efficient probabilistic performance bounds for inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, 2019a.

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, 2019b.

Daniel S. Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. Safe imitation learning via fast Bayesian reward inference from preferences. In *International Conference on Machine Learning*, 2020.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomek Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research*.

Yash Chandak, Shiv Shankar, and Philip S Thomas. High-confidence off-policy (or counterfactual) variance estimation. In *AAAI Conference on Artificial Intelligence*, 2021.

Letian Chen, Rohan R. Paleja, and Matthew C. Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on Robot Learning*, 2020.

Xiaoyu Chen, Han Zhong, Zhuoran Yang, Zhaoran Wang, and Liwei Wang. Human-in-the-loop: Provably efficient preference-based reinforcement learning with general function approximation. In *International Conference on Machine Learning*, 2022.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2023.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015.

Josiah Hanna, Peter Stone, and Scott Niekum. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *AAAI Conference on Artificial Intelligence*, 2017.

Donald Joseph Hejna and Dorsa Sadigh. Few-shot preference learning for human-in-the-loop RL. In *Conference on Robot Learning*, 2022.

Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W Bradley Knox, and Dorsa Sadigh. Contrastive preference learning: Learning from human feedback without reinforcement learning. In *International Conference on Learning Representations*, 2024.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*, 2018.

Zaynah Javed, Daniel S. Brown, Satvik Sharma, Jerry Zhu, Ashwin Balakrishna, Marek Petrik, Anca D. Dragan, and Ken Goldberg. Policy gradient Bayesian robust optimization for imitation learning. In *International Conference on Machine Learning*, 2021.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety-Gymnasium: A unified safe reinforcement learning benchmark. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Cassidy Laidlaw and Anca Dragan. The Boltzmann policy distribution: Accounting for systematic suboptimality in human models. In *International Conference on Learning Representations*, 2022.

Kimin Lee, Laura M. Smith, and Pieter Abbeel. PEBBLE: feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, 2021a.

Kimin Lee, Laura M. Smith, Anca D. Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021b.

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Ellen R. Novoseller, Yibing Wei, Yanan Sui, Yisong Yue, and Joel Burdick. Dueling posterior sampling for preference-based reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Aldo Pacchiano, Aadirupa Saha, and Jonathan Lee. Dueling RL: reinforcement learning with trajectory preferences. In *International Conference on Artificial Intelligence and Statistics*, 2023.

Malayandi Palan, Gleb Shevchuk, Nicholas Charles Landolfi, and Dorsa Sadigh. Learning reward functions by integrating human demonstrations and preferences. In *Robotics: Science and Systems XV*, 2019.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2024.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019. URL https://api.semanticscholar.org/CorpusID:208283920.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Daniel Shin, Daniel S Brown, and Anca D Dragan. Offline preference-based apprenticeship learning. *arXiv preprint arXiv:2107.09251*, 2021.

Harshit Sikchi, Akanksha Saran, Wonjoon Goo, and Scott Niekum. A ranking game for imitation learning. *Trans. Mach. Learn. Res.*, 2023.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems*, 2022.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, 2015a.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *AAAI Conference on Artificial Intelligence*, 2015b.

Philip S. Thomas, Bruno Castro da Silva, Andrew G. Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468): 999–1004, 2019.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in Starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354, 2019.

Yuanhao Wang, Qinghua Liu, and Chi Jin. Is RLHF more difficult than standard RL? A theoretical perspective. In *Advances in Neural Information Processing Systems*, 2023.

Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Better aligning text-to-image models with human preference. *arXiv preprint arXiv:2303.14420*, 2023.

Yichong Xu, Ruosong Wang, Lin F. Yang, Aarti Singh, and Artur Dubrawski. Preference-based reinforcement learning with finite-time guarantees. In *Advances in Neural Information Processing Systems*, 2020.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Comput. Surv.*, 55(2):5:1–5:36, 2023.

Banghua Zhu, Michael I. Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or K-wise comparisons. In *International Conference on Machine Learning*, 2023.

Simon Zhuang and Dylan Hadfield-Menell. Consequences of misaligned AI. In *Advances in Neural Information Processing Systems*, 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Proof of Theorem 5.2

Consider a set of reward samples $\mathcal{R}'$, we now show that with probability $1-\delta$, $\min_{r\in\mathcal{R}'} \text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1-\delta)$ is a lower bound on $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ simultaneously for all $r \in \mathcal{R}'$. Note that for the proof, we use a confidence level of $1-\delta$, while in the main text, the application of this theorem uses a confidence level of $1-\delta/2$. Let $L = \min_{r\in\mathcal{R}'} \text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1-\delta)$. Mathematically, we want to show that

$$\Pr(\forall r \in \mathcal{R}' : L \leq J(\pi_C, r) - J(\pi_{\text{init}}, r)) \geq 1-\delta. \tag{6}$$

Let $r' = \arg\min_{r\in\mathcal{R}'} J(\pi_C, r) - J(\pi_{\text{init}}, r)$. We have

$$\forall r \in \mathcal{R}' : J(\pi_C, r') - J(\pi_{\text{init}}, r') \leq J(\pi_C, r) - J(\pi_{\text{init}}, r). \tag{7}$$

Now consider the high-confidence lower bound on $J(\pi_C, r') - J(\pi_{\text{init}}, r')$, we know that

$$\Pr(\text{HCLB}(J(\pi_C, r') - J(\pi_{\text{init}}, r'), 1-\delta) \leq J(\pi_C, r') - J(\pi_{\text{init}}, r')) \geq 1-\delta, \tag{8}$$

which, using Equation 7, implies that

$$\Pr(\forall r \in \mathcal{R}' : \text{HCLB}(J(\pi_C, r') - J(\pi_{\text{init}}, r'), 1-\delta) \leq J(\pi_C, r) - J(\pi_{\text{init}}, r)) \geq 1-\delta. \tag{9}$$

In other words, $\text{HCLB}(J(\pi_C, r') - J(\pi_{\text{init}}, r'), 1-\delta)$ is a lower bound on $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ simultaneously for all $r \in \mathcal{R}'$ with probability $1-\delta$. However, we cannot identify r', since we cannot compute $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ exactly. Instead of trying to compute $\text{HCLB}(J(\pi_C, r') - J(\pi_{\text{init}}, r'), 1-\delta)$ directly, we observe the following fact:

$$L = \min_{r\in\mathcal{R}'} \text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1-\delta) \leq \text{HCLB}(J(\pi_C, r') - J(\pi_{\text{init}}, r'), 1-\delta), \tag{10}$$

since $r' \in \mathcal{R}'$.

Now, we know that

$$\Pr(\forall r \in \mathcal{R}' : L \leq J(\pi_C, r) - J(\pi_{\text{init}}, r)) \geq 1-\delta. \tag{11}$$

That is, with probability $1-\delta$, $\min_{r\in\mathcal{R}'} \text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1-\delta)$ is a lower bound on $J(\pi_C, r) - J(\pi_{\text{init}}, r)$ simultaneously for all $r \in \mathcal{R}'$.

B Gradient of VaR-EVD

We present the full derivation of the gradient of the VaR-EVD objective (see Equation 4) under a mild assumption. Let $\mathcal{R}$ be the set of reward samples drawn from $P(R|\mathcal{P})$, π_θ be the policy with parameters θ being optimized, and π_{init} be the initial policy. We first start with the assumption:

Assumption B.1 $\forall r_i \neq r_j \in \mathcal{R}, J(\pi_\theta, r_i) - J(\pi_{init}, r_i) \neq J(\pi_\theta, r_j) - J(\pi_{init}, r_j)$.

Assumption B.1 states that the expected value difference of the policy being optimized π_θ and the initial policy π_{init} under any two distinct reward samples r_i and r_j are different. When π_θ is not equal to π_{init}, this is likely to be true. When $\pi_\theta = \pi_{\text{init}}$, the assumption does not hold. However, recall that the only information required from the initial policy π_{init} is the (estimates of) expected

values under all reward samples, i.e., $J(\pi_{\text{init}}, r)$ for all $r \in \mathcal{R}$, which can be computed in advance to any desired level of accuracy. We can simply add a small positive noise to $J(\pi_{\text{init}}, r)$ for all $r \in \mathcal{R}$ while ensuring that the values of the noise added for different r's are different. This simple injection of noise induces minimal change to the values $J(\pi_{\text{init}}, r)$, at the same time making the assumption hold. In fact, unless we initialize π_θ using the parameters of π_{init}, the case when $\pi_\theta = \pi_{\text{init}}$ is extremely rare. Finally, note that even when this assumption does not hold, it does not affect the high-confidence guarantee that our algorithm provides, since our guarantee solely depends on the safety test, and is unaffected by candidate proposal.

We now derive the gradient by first deriving each dimension of the gradient. Let $\mathbf{e}_i^h$ denote the vector with all zero entries except for the i-th dimension, where the entry takes the value h. Simply put, it is the i-th standard basis vector multiplied by h. We have

$$\frac{\partial}{\partial \theta_i} \text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)] \tag{12}$$

$$= \lim_{h \to 0} \frac{1}{h} (\text{VaR}_{1-\delta/2}[J(\pi_{\theta+\mathbf{e}_i^h}, R) - J(\pi_{\text{init}}, R)] - \text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]). \tag{13}$$

Applying Assumption B.1, we know that the expected value difference is different for different reward samples r. This means that we can compute the minimum distance between pairs of expected value differences:

$$d_{\text{min}} = \min_{r_i, r_j \in \mathcal{R}, r_i \neq r_j} |(J(\pi_\theta, r_i) - J(\pi_{\text{init}}, r_i)) - (J(\pi_\theta, r_j) - J(\pi_{\text{init}}, r_j))| > 0. \tag{14}$$

We know that when h is small enough, the change induced to the expected value differences is not enough to overcome the minimum distance between any pairs of expected value differences. Mathematically, this can be written as:

$$\exists h : \forall r \in \mathcal{R}, |(J(\pi_{\theta+\mathbf{e}_i^h}, r) - J(\pi_{\text{init}}, r)) - (J(\pi_\theta, r) - J(\pi_{\text{init}}, r))| < d_{\text{min}}/2. \tag{15}$$

Note that we need $d_{\text{min}}/2$ instead of d_{min} here since there is a pair of reward samples involved in d_{min}, and the expected value difference of each of the two reward samples can change. Then, when h is small enough, for the two VaR terms in Equation 13, if we sort all reward samples twice, once using the expected value differences corresponding to the first VaR term and another time using that corresponding to the second VaR term, the two orders of the reward samples obtained from the two sorts will be the same. Therefore, the reward samples that correspond to expected value differences equal to the two VaR terms are the same. Mathematically, let r_1 be the reward sample such that

$$J(\pi_{\theta+\mathbf{e}_i^h}, r_1) - J(\pi_{\text{init}}, r_1) = \text{VaR}_{1-\delta/2}[J(\pi_{\theta+\mathbf{e}_i^h}, R) - J(\pi_{\text{init}}, R)], \tag{16}$$

and r_2 be the reward sample such that

$$J(\pi_\theta, r_2) - J(\pi_{\text{init}}, r_2) = \text{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)], \tag{17}$$

we then know that $r_1 = r_2$. Note that r_2 does not depend on the dimension of the gradient i. Also note that here we assume that only one reward sample satisfies each of Equations 16 and 17, since we almost never generate identical reward samples from MCMC and we have assumed (by Assumption B.1) that different reward samples correspond to different expected value differences.

Therefore, Equation 13 can be written as:

$$\lim_{h \to 0} \frac{1}{h} (J(\pi_{\theta+\mathbf{e}_i^h}, r_2) - J(\pi_{\text{init}}, r_2) - J(\pi_\theta, r_2) + J(\pi_{\text{init}}, r_2)) \tag{18}$$

$$= \lim_{h \to 0} \frac{1}{h} (J(\pi_{\theta+\mathbf{e}_i^h}, r_2) - J(\pi_\theta, r_2)) \tag{19}$$

$$= \frac{\partial}{\partial \theta_i} J(\pi_\theta, r_2). \tag{20}$$

Combining all dimensions of the gradient, we have

$$\frac{\partial}{\partial\theta}\,\mathrm{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)] \tag{21}$$

$$= \frac{\partial}{\partial\theta} J(\pi_\theta, r), \tag{22}$$

where r is the reward sample that satisfies $J(\pi_\theta, r) - J(\pi_{\text{init}}, r) = \mathrm{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]$.

Based on our derivation above, it may seem that optimizing a policy using this gradient will never allow the ordering of the reward samples according to the expected value differences to change. An immediate result of this is that the reward sample r that satisfies $J(\pi_\theta, r) - J(\pi_{\text{init}}, r) = \mathrm{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]$ will always be the same, meaning that we will always optimize with respect to the same reward sample. This would mean that our algorithm would be no different from those that optimize a point estimate of the ground-truth reward function. However, note that the argument that the changes induced to the expected value differences are not enough to overcome d_{min} only serves to prove the existence of the gradient, and only holds for small enough h. In practice, with the use of an optimizer, we often use a step size that is much larger than the small h we used above in our arguments, inducing changes to θ substantial enough that the ordering of the reward samples changes. In fact, we empirically observe that the reward sample r that satisfies $J(\pi_\theta, r) - J(\pi_{\text{init}}, r) = \mathrm{VaR}_{1-\delta/2}[J(\pi_\theta, R) - J(\pi_{\text{init}}, R)]$ changes throughout optimization.

C Full Experiment Details

In this section, we provide full experiment details. Section C.1 presents details on the domains used for our experiments. Section C.2 presents details on constructing the preference dataset. Section C.3 presents details on generating reward samples from the posterior distribution $P(R|\mathcal{P})$. Section C.4 presents details on the initial policies used for our experiments. Section C.5 presents details on computing the standard error for our experimental results.

C.1 Domains

We consider two domains from Safety Gymnasium (Version 1.2.0, Apache-2.0 license) (Ji et al., 2023), a suite of environments with safety constraints for safe RL built on top of Safety Gym (Ray et al., 2019). We choose to perform experiments in this suite since unsafe behavior is well-defined. The suite consists of a wide range of tasks. For each task, the suite provides several built-in difficulty levels. The suite also allows the design of custom levels. Our experiments focus on safety navigation tasks. We carry out experiments in two specific tasks: *Goal* and *Circle*. At every time step, apart from the reward, a domain in the Safety Gymnasium suite also indicates whether a safety constraint is violated. We simply subtract the cost of violating a safety constraint, which is by default 1, from the reward at every time step, and treat this quantity as the reward. The Safety Gymnasium provides a set of agents to choose from and we use the Point agent for all experiments.

Circle. As shown in Figure 2 (left), in Circle, the agent has to travel as fast as possible along the circumference of a large circle placed at the center of the environment. The faster the agent and the closer the agent is to the circumference, the higher the reward. The exact reward function can be found in the Safety Gymnasium paper (Ji et al., 2023). Apart from the circle, there are forbidden regions slightly overlapping with the circle, giving the agent -1 reward for every time step the agent is in a forbidden region. Therefore, the agent cannot simply travel in circular motion, and has to learn to avoid these regions. We use the provided level 1 Circle environment.

Goal. As shown in Figure 2 (right), Goal is one of the navigation tasks in the suite, where the agent has to navigate towards a goal. When the goal is reached, the agent receives a reward of $+1$, and a new goal with a random position is generated. The agent has to learn to navigate to successive goals while avoiding dangerous zones called *hazards* in the environment. Hazards are simple circular

zones in the environment that give a reward of -1 for every time step an agent is in a hazard. We noticed that the random placements of goals and hazards of the built-in levels allow some policies to achieve good performance even without learning to avoid a hazard, so we designed our custom Goal environment, where four hazards are placed at the center of the environment, and goals are generated around this grid of hazards.

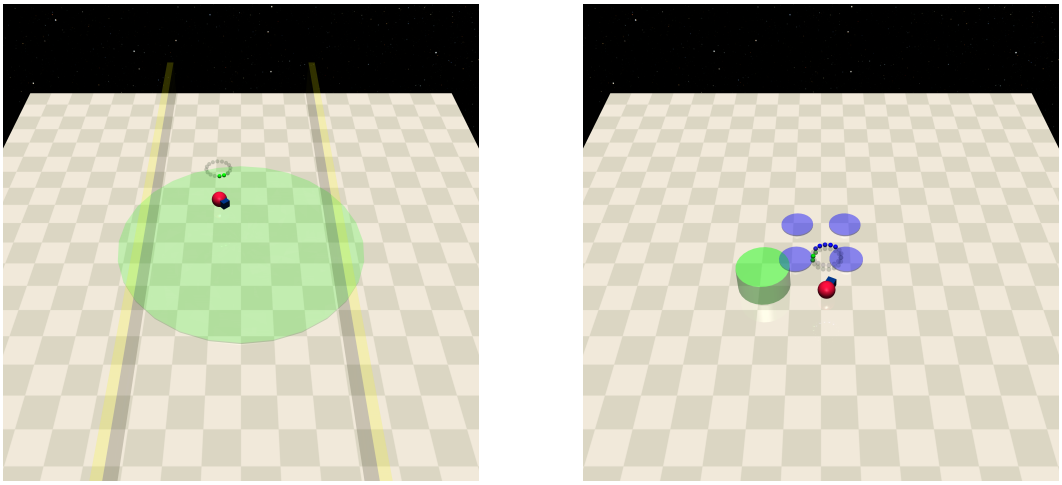

Figure 2: The level 1 Circle domain (left) and the custom Goal domain (right). For the Circle domain, the green circle indicates the circle that the agent should circle around, while the regions beyond the yellow lines are forbidden. For the Goal domain, each blue circle is a hazard that the agent should avoid, while the green cylinder represents the goal.

C.2 Preference Label Generation

In the RLHF setting, we assume access to a preference dataset. Similar to prior work (Brown et al., 2020; Javed et al., 2021), we generate preference data for the purpose of experiments. To generate preferences, we first need a set of trajectories. To generate trajectories, we first use PPO (Schulman et al., 2017) to train a policy under the ground-truth reward function R^* till convergence. We denote the obtained policy as the demonstration policy π_{demo}. Details of training the demonstration policy can be found in Section C.4.

After training the demonstration policy, we use it to generate a set of trajectories. We treat a full episode as a trajectory. It is good to ensure that RLHF learns to avoid unsafe behavior, even when it has not been directly observed and dispreferred. To test whether our algorithm POSTPI can achieve this, we generate preferences over trajectories with minimal unsafe behavior. Note that this makes the experimental settings more challenging, instead of the opposite. To do so, we generate trajectories using the trained π_{demo}, but filter out trajectories with unsafe behavior. For the two domains we consider, we find the empirical return of an episode to be a good indicator of whether unsafe behavior has occurred, and filter trajectories using their returns. For the Circle domain, we generate 30 trajectories with returns above 30. For the Goal domain, we generate 30 trajectories with returns above 15.

After generating trajectories, we compute preference labels following the Bradley-Terry model (Bradley & Terry, 1952). We randomly sample two distinct trajectories τ_i, τ_j from the set of generated trajectories, and assign preference labels according to the following probability:

$$\Pr(\tau_i \prec \tau_j) = \frac{e^{\beta R(\tau_j)}}{e^{\beta R(\tau_i)} + e^{\beta R(\tau_j)}}, \tag{23}$$

where $\tau_i \prec \tau_j$ indicates that τ_j is preferred over τ_i, β is the inverse temperature parameter, and $R(\tau) = \sum_{s \in \tau} R(s)$ is the return of trajectory τ. We choose $\beta = 5$ in all of our experiments. We generate a total of 50 preferences for both domains.

Unlike other work which assumes that the preferences follow the Bradley-Terry model (Bradley & Terry, 1952), CPL (Hejna et al., 2024) assumes that preferences follow the regret preference model. To obtain preferences based on regret, while maintaining a fair comparison with POSTPI, we reuse the pairs of trajectories generated for POSTPI, but regenerate the preference labels according to the regret model. In particular, we compute the advantage of a state-action pair using the value network trained while training the demonstration policy π_{demo}.

C.3 Reward Sample Generation

Using the set of preferences, we can sample from the posterior distribution over reward functions given preferences $P(R|\mathcal{P})$ using B-REX (Brown et al., 2020). B-REX involves pre-training a reward model to automatically learn state features. B-REX represents each reward sample as a vector of the same dimensions as the state features, and computes rewards as the dot product between the learned state features and the vector representation of the reward sample. However, it is not guaranteed that following this method, the learned state features are expressive enough to represent the ground-truth reward function linearly. Our algorithm cannot provide the desired guarantee when the ground-truth reward function does not lie in the space of reward samples. In our experiments, we intend to focus on studying our high-confidence guarantee in ideal conditions. Therefore, similar to a prior work (Javed et al., 2021), we hand-craft state features to allow the ground-truth reward function R^* to be expressed linearly in the state features. This ensures that the ground-truth reward function lies within the space of reward samples, which is necessary to provide our high-confidence guarantee.

For the Goal domain, the reward function has the following form:

$$r_t = (D_{t-1} - D_t) + \mathbb{1}_{\{\text{ agent in goal}\}} - \sum_{i=1}^{4} \mathbb{1}_{\{\text{ agent in hazard } i\}}, \tag{24}$$

where D_t is the distance between the agent and the goal at time t, $\mathbb{1}_{\{\text{ condition}\}}$ is the indicator function that evaluates to 1 when the condition is true and 0 otherwise. Simply put, the agent receives a small dense reward $D_{t-1} - D_t$ that guides the agent towards the goal. The agent also receives a reward of $+1$ for reaching the goal and a penalty of -1 for reaching any of the four hazards.

We simply construct the state features as a 6-dimension vector as follows:

$$\phi(s_t) = [D_{t-1} - D_t, \mathbb{1}_{\{\text{ agent in goal}\}}, \mathbb{1}_{\{\text{ agent in hazard 1}\}}, \ldots, \mathbb{1}_{\{\text{ agent in hazard 4}\}}]. \tag{25}$$

Then, the ground-truth reward function R^* can be represented as the vector $\mathbf{w}_{R^*} = [1, 1, -1, -1, -1, -1]$. The ground-truth reward at every time step t can be computed as $R^*(s_t) = \mathbf{w}_{R^*}^T \phi(s_t)$. We apply a similar approach to the Circle domain.

With the state features available, we now draw reward samples from the posterior distribution $P(R|\mathcal{P})$ as vectors of unit L2 norm with the same dimension as the state features. Constraining the reward samples to have unit L2 norm is a standard approach. Note that the ground-truth reward function can still be expressed as a vector lying on the L2 unit norm ball, since scaling the reward function by a positive constant does not affect the set of corresponding optimal policies.

We run MCMC to generate reward samples. We use a uniform prior assigning the same probability density to all reward samples. We follow the likelihood function in B-REX (Brown et al., 2020), using $\beta = 5$ that matches the β used when generating preference labels. We propose reward samples by adding independent Gaussian noise to each dimension of the current reward sample. The standard deviation of the Gaussian noise is chosen so that the probability of accepting a proposed reward sample lies between 0.2 and 0.8. The standard deviation of the Gaussian noise is 1 for the Circle domain and 0.1 for the Goal domain. We run a total of 20K MCMC steps with a burn-in of 4K

steps. We sample from the chain every 20 steps to reduce auto-correlation. For the sampled reward functions, we split them into two sets by taking the ones with odd indices into one set, and taking the remaining ones into another set. We use one set for candidate proposal and the other set for the safety test to reduce bias in the safety test. Each set contains a total of 400 reward samples, which we find to be sufficient empirically.

C.4 Initial Policies

In this subsection, we describe our approach to generate the set of initial policies $\pi^{\epsilon}_{\text{init}}$ used in the experiments in Section 6. We are interested in a set of initial policies with varying performance under the ground-truth reward function.

First, we would like to train a policy with good performance under the ground-truth reward function. To do this, we train a policy using PPO (Schulman et al., 2017) under the ground-truth reward function till convergence. We use the implementation of PPO by Stable-Baselines3 (Version 2.2.0a6, MIT license) (Raffin et al., 2021), and default hyperparameters apart from the ones listed in Table 5. We use the same set of hyperparameters for both the Circle and Goal domains, apart from the total number of time steps to run, which is 3M and 5M for the Circle and Goal domains respectively. For each domain, we trained 5 policies with different random seeds for each of the three learning rates: 3e$-$5, 1e$-$4, and 3e$-$4. We pick the policy with the highest expected return out of the 15 policies. This policy, referred to as the demonstration policy π_{demo}, has good performance under the ground-truth reward function. Note that this policy is the same as the one used to generate trajectories during preference label generation (see Section C.2). We use π_{demo} as one of the initial policies, simulating the case of improving with respect to a policy already performing well under the ground-truth reward function. This policy π_{demo} corresponds to the case when $\epsilon = 1$, i.e., $\pi_{\text{demo}} = \pi^{1}_{\text{init}}$.

We now describe our approach to obtain policies performing worse than π_{demo} to different extents. Recall that the only information of the initial policy π_{init} required by our algorithm is the expected value $J(\pi_{\text{init}}, r)$ for all reward samples $r \in \mathcal{R}$, and these expected values can be computed in advance to any desired level of accuracy (see the final paragraph of Section 5.2). Therefore, instead of training different policies with varying performance, we simply multiply $J(\pi_{\text{demo}}, r)$ for all r by a constant ϵ to simulate varying levels of sub-optimal policies. Mathematically, we provide $\epsilon J(\pi_{\text{demo}}, r)$ for all $r \in \mathcal{R}$ to our algorithm, where $\epsilon \in [0, 1]$ and $J(\pi_{\text{demo}}, r)$ is estimated using 200 rollouts. We refer to these policies as $\pi^{\epsilon}_{\text{init}}$. Note that $\pi^{\epsilon}_{\text{init}}$ corresponds to a policy that always obtains ϵ fraction of the rewards obtained by π_{demo}, for all reward samples. For example, for a particular trajectory, if π_{demo} obtains a reward of 1 at time step t under the reward sample r, $\pi^{\epsilon}_{\text{init}}$ will obtain a reward of ϵ at time step t under the reward sample r. Furthermore, since π_{demo} is trained under the ground-truth reward function till convergence, for the two domains we consider, it obtains positive rewards most of the time. Therefore, $\epsilon < 1$ corresponds to policies worse than π_{demo}, with lower ϵ corresponding to worse policies.

In our experiments, we consider $\epsilon \in \{0, 0.25, 0.5, 0.75, 1\}$. While it is possible to consider $\epsilon > 1$, since π_{demo} is obtained by training a policy under the ground-truth reward function till convergence, we expect that it would be difficult to improve with respect to π^{1}_{init}, let alone an initial policy $\pi^{\epsilon}_{\text{init}}$ with $\epsilon > 1$. Therefore, we focus on $\epsilon \in [0, 1]$ in our experiments.

For evaluation purposes, we also need to compute the expected value of the initial policies under the ground-truth reward function R^*. When $\epsilon = 1$, i.e., the initial policy is π_{demo}, we simply estimate $J(\pi_{\text{demo}}, R^*)$ using 200 rollouts. For $\epsilon < 1$, since we already have the estimate of $J(\pi_{\text{demo}}, R^*)$, we simply estimate $J(\pi^{\epsilon}_{\text{init}}, R^*)$ by multiplying the estimate of $J(\pi_{\text{demo}}, R^*)$ by ϵ.

C.5 Statistical Significance

In Figure 1, which presents the probability of our algorithm returning a policy, the shaded area indicates ± 1 standard error of this probability. Factors of variability include the randomness in generating preference labels, random initialization of the policy and value networks, and the rollouts

generated during training and evaluating the policy. The standard error is computed as $\frac{\hat{\sigma}}{\sqrt{N_{\text{trial}}}}$, where $\hat{\sigma}$ is the sample standard deviation, and $N_{\text{trial}} = 20$ is the number of trials.

D Additional Experiments

In this section, we present additional experiments not covered in the main text. Section D.1 compares our safety test with one that uses a previous approach (Brown et al., 2020) to compute high-confidence bounds. Section D.2 compares our candidate proposal with state-of-the-art alternatives. Section D.3 presents experiments generalizing our algorithm to high-dimensional image inputs.

D.1 Comparison to B-REX's Bound

In this subsection, we empirically demonstrate that our safety test is more reliable than a counterpart that does not account for the uncertainty in using a finite number of rollouts to estimate the expected values of policies. In B-REX, Brown et al. (2020) proposed a method to compute high-confidence lower bounds that reasons only about the uncertainty in the ground-truth reward function R^*, but does not take the number of rollouts used to estimate $J(\pi_C, r) - (\pi_{\text{init}}, r)$ into consideration. This approach can compute overly optimistic bounds, especially when the number of samples is small. We empirically demonstrate that a safety test using this approach of computing high-confidence bounds, which we refer to as B-REX style, fails to provide the high-confidence guarantee as claimed.

Table 2 shows the comparison of using the B-REX style safety test and using our safety test in the Goal domain when using a small number of rollouts (20) for the safety test. It can be seen that our safety test returns policies worse than the initial policy with probability not larger than $\delta = 0.05$, but the B-REX style safety test returns policies worse than the initial policy with probabilities greater than $\delta = 0.05$ for $\epsilon = 0.75$ and 1. This is because when the number of rollouts used is small, the B-REX style safety test is prone to computing overly optimistic bounds. As a result, the B-REX style safety test more frequently accepts policies that are in fact not better than the initial policy, causing it to not provide the high-confidence guarantee. On the other hand, our algorithm duly accounts for the uncertainty in the expected value estimates and provides the high-confidence guarantee as expected regardless of the number of rollouts used in the safety test. In light of this, our safety test is safer to employ in practice.

Table 2: Probability of POSTPI returning a policy worse than $\pi^{\epsilon}_{\text{init}}$ for different ϵ over 20 trials in the Goal domain when using a B-REX style safety test and our safety test, and using a small number of rollouts (20) for the safety test.

ϵ	0	0.25	0.5	0.75	1
B-REX Style Safety Test	**0.05**	**0.0**	**0.05**	0.15	0.1
Our Safety Test	**0.05**	**0.0**	**0.0**	**0.0**	**0.0**

D.2 Choice of candidate proposal

We now empirically justify the choice of optimizing the VaR-EVD objective in candidate proposal. In candidate proposal, the goal is to return a policy that is likely to pass the safety test. In fact, any policy optimization algorithm can be used in candidate proposal. However, we hypothesize that our candidate proposal that optimizes the novel VaR-EVD objective, which allows the specification of an initial policy and takes into account knowledge of the safety test, should produce candidate policies that are more likely to be accepted in the safety test than other policy optimization methods.

As seen in Section 6.1, T-REX (Brown et al., 2019a) and B-REX (Brown et al., 2020) frequently return policies that are worse than π^{0}_{init} with high probabilities. Therefore, we mainly compare our

candidate proposal with PG-BROIL (Javed et al., 2021) and CPL (Hejna et al., 2024), two state-of-the-art RLHF methods.

We compare the probability of returning a policy for our candidate proposal and using PG-BROIL and CPL as candidate proposal for different initial policies $\pi^{\epsilon}_{\text{init}}$. We use 1000 rollouts for computing high-confidence bounds in the safety test to ensure that the difference in results is not caused by insufficient data in the safety test.

Table 3 shows the probability of returning a policy when using our candidate proposal, PG-BROIL as candidate proposal, and CPL as candidate proposal for the Circle and Goal domains. It can be seen that CPL returns policies that are rejected by the safety test with high probabilities. PG-BROIL and our candidate proposal return policies that are accepted with similar probabilities in the Circle domain, but our candidate proposal largely outperforms PG-BROIL in the Goal domain. By allowing the specification of an initial policy, and taking the knowledge of the safety test into account, maximizing the VaR-EVD objective in candidate proposal returns policies that are accepted by the safety test more often than state-of-the-art alternatives.

Table 3: Probability of returning a policy for the Circle and Goal domains using our candidate proposal and using PG-BROIL and CPL as candidate proposal.

DOMAIN	**CIRCLE**					**GOAL**				
ϵ	0	0.25	0.5	0.75	1	0	0.25	0.5	0.75	1
PG-BROIL	**1.0**	0.95	**0.9**	**0.8**	0.0	0.9	0.8	0.65	0.25	0.05
CPL	0.3	0.25	0.2	0.0	0.0	0.6	0.3	0.2	0.05	0.0
POSTPI (Ours)	0.95	**1.0**	**0.9**	0.75	0.0	**0.95**	**0.95**	**0.95**	**0.6**	**0.15**

D.3 High-Dimensional Image Input

In this subsection, we demonstrate that our algorithm POSTPI scales to high-dimensional image inputs. We use the same two domains as before and adopt the same procedures of generating preference labels and reward samples (see Section C for more details). The only change we make here is the observations used when performing policy optimization in candidate proposal. We replace the original vector observations with pixels captured by a camera placed in front of the agent. We note that the original observations come from lidar sensors detecting all directions of the agent, while the camera only captures information in front of the agent. This reduction in the information contained in observations, coupled with the increased difficulty of learning from pixels, causes a drop in the performance of the candidate policies. As a result, the probability of returning a policy drops, especially for the more difficult Goal domain. We focus on the probability of returning a policy worse than the initial policy in this subsection.

Table 4 shows the probability of returning a policy worse than different initial policies in both domains. It can be seen that even in a setting where learning a well-performing policy is difficult, our algorithm still provides the desired guarantee. As discussed in Section 6.2, we are also interested in seeing whether the accepted π_C is actually an improvement over the initial policy with high probability. Note, again, that our algorithm provides no guarantees on this probability. Nevertheless, we find that a policy accepted by the safety test has a high probability (> 0.85) of being an improvement over the initial policy in both domains.

One thing to note is that for the results presented in Table 4, the candidate policies were trained for the same number of steps as in the case of the original observations (3M and 5M for the Circle and Goal domains respectively). Due to the increased training time for image inputs, we initially trained the policies for only 2M steps for both domains, and we observed that the probability of returning a policy worse than the initial policy was sometimes 0.1, which is slightly higher than δ. This is likely

Table 4: Probability of POSTPI returning a policy worse than $\pi_{\text{init}}^{\epsilon}$ for different ϵ over 20 trials in the Circle and Goal domains with image observations.

ϵ	0	0.25	0.5	0.75	1
Circle	**0.0**	**0.05**	**0.0**	**0.0**	**0.0**
Goal	**0.0**	**0.0**	**0.05**	**0.0**	**0.0**

caused by a failure to fulfill the assumptions made by Student's t-test. In POSTPI's safety test, we use Student's t-test to compute $\text{HCLB}(J(\pi_C, r) - J(\pi_{\text{init}}, r), 1 - \delta/2)$. Student's t-test makes the normality assumption, but we observed that empirically the data distribution was not normal. We experimented with other statistical tests, and some more conservative tests were able to ensure the desired guarantee, at the expense of the probability of returning a policy. We suggest using Student's t-test for users who do not strictly require the high-confidence policy improvement guarantee, and are merely interested in an algorithm that is safer than algorithms without any guarantees. For users who are interested in an algorithm that strictly provides the guarantee, we suggest replacing Student's t-test with more conservative statistical tests or tests with weaker assumptions.

E Hyperparameters

For POSTPI, B-REX (Mean), B-REX (MAP), PG-BROIL, and T-REX, the major difference during policy optimization is the reward functions used. B-REX (Mean) is trained under the mean reward of the posterior distribution $P(R|\mathcal{P})$. B-REX (MAP) is trained under the MAP reward of the posterior distribution. T-REX is trained under a reward function learned in the T-REX manner (Brown et al., 2019a). Our algorithm chooses the reward sample at every iteration using the VaR, while PG-BROIL chooses reward samples using their BROIL objective (Javed et al., 2021).

As the reward functions used by these algorithms have the same magnitude of unit L2 norm, we share most hyperparameters for these algorithms. We adapt the implementation of PPO in Stable-Baselines3 (Version 2.2.0a6, MIT license) (Raffin et al., 2021) to optimize the policy for these algorithms, and use default hyperparameters apart from the ones presented in Table 5. The learning rate is picked by grid search from the following values [1e−5, 3e−5, 1e−4, 3e−4, 1e−3] by training five policies with different random seeds under each learning rate. We find the learning rate 1e−4 to perform best for both domains, and use this value for all of these algorithms. These algorithms are trained for 3M and 5M steps in the Circle and Goal domains respectively. We use the Adam optimizer (Kingma & Ba, 2015) for all experiments. We present details and hyperparameters specific to each algorithm below.

Table 5: Hyperparameters of algorithms built on PPO.

HYPERPARAMETER	VALUE
Number of hidden layers	2
Number of hidden units	128
Activation Function	ReLU
Number of environment steps per update	4000

PG-BROIL. As mentioned in the main text, we choose a confidence level of 0.95, i.e., $\delta = 0.05$, for POSTPI. For PG-BROIL, for a fair comparison with our algorithm POSTPI, we choose $\alpha = 0.95$. We choose $\lambda = 0$ as we observe it to perform best empirically.

B-REX. We compute the mean reward and select the MAP reward using the set of reward samples reserved specifically for candidate proposal (see the final paragraph of Section 5.2). These rewards are then maximized using PPO, leading to the results of B-REX (Mean) and B-REX (MAP) in the main text respectively.

T-REX. We follow the approach detailed in Section 4 of the T-REX paper (Brown et al., 2019a) to learn the reward function. For a fair comparison with POSTPI, we use the same set of preferences used by POSTPI to train the T-REX reward function. We pick the learning rate by grid search from the following values [3e−3, 1e−2, 3e−2, 1e−1, 3e−1, 1]. We find the best learning rate to be 1e−1 for both domains, and observe 200 epochs to be sufficient for convergence. We normalize the trained T-REX reward function to have unit L2 norm.

CPL. Different from POSTPI and other baselines, CPL assumes that the preferences follow the regret preference model (Hejna et al., 2024). For a fair comparison with POSTPI, we use the same set of pairs of trajectories used by POSTPI, but re-generate the preferences according to the regret preference model. When generating preferences according to the regret model, we follow the CPL paper and use $\gamma = 1$. After obtaining the CPL preferences, we train a policy using the CPL variant with regularization presented in Section 3 of the CPL paper (Hejna et al., 2024). We keep the policy architecture the same as POSTPI and other baselines. We pick the learning rate by grid search from the following values [3e−4, 1e−3, 3e−3, 1e−2]. We find the best learning rate to be 1e−3, and find 1000 epochs to be sufficient for convergence. For other hyperparameters, we follow the CPL paper (Hejna et al., 2024), and use $\alpha = 0.1$, $\lambda = 0.5$, and $\beta = 0.0$.

F Compute Resources

The experiments are performed on a compute server, mainly using NVIDIA GeForce GTX TITAN X (12GB) GPUs. POSTPI, PG-BROIL, B-REX, and T-REX have similar runtimes. Each trial of these algorithms takes 5 hours and 7 hours in the Circle and Goal domains respectively. For the experiments in Section D.3 that involve image inputs, each trial takes 26 hours and 48 hours for the Circle and Goal domains respectively. On the other hand, due to the small number of preferences, each trial of CPL can be trained in under 10 minutes. For evaluating the trained policies, all algorithms share similar runtimes. Evaluating the expected value of the policies using 200 rollouts takes 1 hour and 1.5 hours for the Circle and Goal domains respectively. Evaluating the policies for experiments in Section D.3 takes 5 hours and 7 hours for the Circle and Goal domains respectively. Note that the number of hours presented above is approximate. The experiments presented in this paper were run in a highly parallel manner using 60 to 100 GPUs simultaneously, and finished in approximately one week. Running the experiments in a sequential manner is expected to take a much longer time.