LEVERAGING SUB-OPTIMAL DATA FOR HUMAN-IN-THE-LOOP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

To create useful reinforcement learning (RL) agents, step zero is to design a suitable reward function that captures the nuances of the task. However, reward engineering can be a difficult and time-consuming process. Instead, human-in-the-loop (HitL) RL methods hold the promise of learning reward functions from human feedback. Despite recent successes, many of the HitL RL methods still require numerous human interactions to learn successful reward functions. To improve the feedback efficiency of HitL RL methods (i.e., require less human interaction), this paper introduces Sub-optimal Data Pre-training, SDP, an approach that leverages reward-free, sub-optimal data to improve scalar- and preference-based HitL RL algorithms. In SDP, we start by pseudo-labeling all low-quality data with the minimum environment reward. Through this process, we obtain reward labels to pre-train our reward model *without* requiring human labeling or preferences. This pre-training phase provides the reward model a head start in learning, enabling it to recognize that low-quality transitions should be assigned low rewards. Extensive experiments with both simulated and human teachers reveal that SDP can at least meet, but often significantly improve, state-of-the-art HitL RL performance across a variety of simulated robotic tasks.

1 INTRODUCTION

In reinforcement learning (RL), an agent's objective is to interact with an environment and maximize its total (discounted) expected reward. The reward hypothesis further maintains that a well-specified reward function is sufficient for an agent to learn to solve a task (Sutton & Barto, 2018). However, defining a reward function that precisely captures all task complexities is often tedious and non-trivial (Booth et al., 2023). There have been notable examples of reward misspecification, in which RL agents discovered and exploited unintended shortcuts in the reward function (Skalse et al., 2022). One notorious example is the CoastRunners game, in which the goal should be to finish a boat race as fast as possible — an RL agent instead gained the most reward by spinning its boat in a circle despite concurrently catching on fire and crashing into other boats (Clark & Amodei, 2016).

A promising alternative is to learn reward functions directly from human feedback. In this paradigm, humans can provide feedback in the form of preferences or scalar signals, which can then be used to learn a reward function that is consistent with human desires (Daniel et al., 2014; Christiano et al., 2017). Despite recent progress, existing preference- and scalar-based RL methods still suffer from high human labeling costs that can require thousands of human queries to learn an adequate reward function (Christiano et al., 2017). Prior work attempts to mitigate this issue through several mechanisms, including active learning (Lee et al., 2021a), data augmentation (Park et al., 2022), semi-supervised learning (Park et al., 2022), and meta-learning (Hejna III & Sadigh, 2023).

Alternatively, our work draws on recent advances in offline RL that have demonstrated the value of low-quality data (Yu et al., 2021). However, its potential in HitL RL remains unexplored. As low-quality data is often readily accessible or easy to obtain, this work addresses this gap by asking the question:

> Can we leverage sub-optimal, unlabeled data to improve learning in HitL RL methods?

To that end, we present Sub-optimal Data Pre-training, *SDP*, a tool for HitL RL algorithms to increase human feedback efficiency. SDP leverages sub-optimal trajectories by pseudo-labeling all

transitions with the minimum environment reward. The now pseudo-labeled sub-optimal data serves two purposes. First, we pre-train a regression-based reward model by applying standard supervised learning to minimize the mean squared loss. Intuitively, this pre-training provides the reward model a head start, biasing it towards assigning lower reward values to these low-quality transitions. Second, we initialize the RL agent's replay buffer with the sub-optimal data and make learning updates to the RL agent. This process changes the RL agent's policy and provides different behaviors for the human to provide feedback on (relative to learning with no initial sub-optimal data). This ensures that when the human teacher provides feedback, their time is used efficiently, avoiding redundant feedback on the existing sub-optimal data. Afterward, we follow the standard preference- or scalar-based RL protocol.

This paper's core contribution is showing that we can harness the availability of low-quality, reward-free data for HitL RL approaches by pseudo-labeling it with minimum rewards and treating it as a prior for learning reward models. We first validity the utility of SDP in extensive simulated teacher experiments, combining it with four scalar- and preference-based RL algorithms. These experiments show that SDP significantly improves the efficiency of feedback in complex tasks from both the DeepMind Control (Tassa et al., 2018) and Meta-World (Yu et al., 2020) suites. Crucially, we further highlight the real-world applicability of SDP by demonstrating its success with human teachers in a 16-person user study. Overall, this work takes an important step toward considering how HitL RL approaches can take advantage of readily-available sub-optimal data.

2 Related Work

Human-in-the-Loop RL Several approaches in HitL RL allow agents to leverage human feedback to adapt or learn new behavior. Learning from demonstration is one such methodology that allows a human to provide examples of desired agent behavior (Argall et al., 2009). Human demonstration data has been used to shape the environment's reward function (Brys et al., 2015), develop a reward function from scratch (Abbeel & Ng, 2004), and bias the agent's policy towards certain actions (Taylor et al., 2011). Although demonstrations can be a rich source of feedback, they are often expensive to obtain and may require domain experts (Dragan & Srinivasa, 2012).

Another approach is learning from preference-based feedback where a teacher provides preferences between two or more sets of agent behavior (Christiano et al., 2017). Preference learning has been popularized in recent years as it can require less effort and expertise compared to providing demonstrations. To further reduce the amount of human interaction required, several strategies have been introduced. This has included combining preferences with demonstrations (Ibarz et al., 2018; Bıyık et al., 2022), unsupervised pre-training (Lee et al., 2021a), bi-level optimization (Liu et al., 2022), semi-supervised learning (Park et al., 2022), data augmentation (Park et al., 2022), uncertainty-based exploration (Liang et al., 2022), meta-learning (Hejna III & Sadigh, 2023), and active learning approaches (Hu et al., 2024). Despite its popularity, some argue that comparison feedback might not capture the full intricacies of human preferences, as oftentimes the human is limited to choosing between two options (Daniel et al., 2014; White et al., 2024).

As a result, another body of work focuses on learning from scalar feedback where human teachers can provide scalar signals to evaluate an agent's behavior (Knox & Stone, 2009; Griffith et al., 2013; Loftin et al., 2016; MacGlashan et al., 2017; White et al., 2024). Several works use scalar feedback to either learn a reward model (Daniel et al., 2014; Cabi et al., 2020) or an action-value function (Knox & Stone, 2009; 2013; Warnell et al., 2018) via regression.

Learning from Sub-Optimal Data SDP aims to leverage sub-optimal data for scalar- and preference-based RL algorithms. However, learning from low-quality data or negative examples has been applied in other areas of RL and imitation learning (Chen et al., 2021; Tangkaratt et al., 2021). In standard RL, several works use sub-optimal demonstrations to initialize a policy (Taylor et al., 2011; Hester et al., 2018; Gao et al., 2019). In goal-conditioned RL, Hindsight-Experience-Replay uses failed episodes by treating them as a success with respect to a different goal (Andrychowicz et al., 2017). In inverse reinforcement learning (IRL), Shiarlis et al. (2016) proposed a constrained optimization formulation that can accommodate both successful and failed demonstrations. Brown et al. (2019) makes use of ranked demonstrations to learn a reward function in IRL. Later work in IRL automatically generates ranked trajectories by adding increasing amounts of noise to a learned

policy (Brown et al., 2020). Lastly, in offline RL, Singh et al. (2020) leverages sub-optimal transitions from multiple tasks and assigns reward labels according to the reward function. Our work is most closely related to offline RL work by Yu et al. (2022). This prior work leverages reward-free, sub-optimal data by pseudo-labeling all transitions with zero and adding them to the RL agent's replay buffer. However, we found that directly applying this approach to the HitL RL setting was ineffective (See Appendix D.5).

3 BACKGROUND

In the RL paradigm, agents interact with an environment to maximize the total (discounted) expected reward it can achieve. This interaction process is modeled as a Markov Decision Process (MDP) which consists of $\langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$. At every time-step t, the agent receives a state $s_t \in \mathcal{S}$ from the environment and chooses an action $a_t \in \mathcal{A}$. The environmental transition function, T, determines the probability of transitioning to state s_{t+1} and receiving reward r_{t+1}, given the agent was in state s_t and executed action a_t. The environment then provides the agent r_{t+1}. The agent attempts to learn a policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes the expected return $\mathbb{E}[G] = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, which is defined as the expected sum of discounted future rewards with discount factor $\gamma \in [0, 1)$.

3.1 REWARD LEARNING FROM HUMAN FEEDBACK

This paper assumes that we are in a reward-free paradigm, an MDP/R setting, where our goal is to (1) learn a reward function, $\hat{r}$, from human feedback and (2) learn a policy that maximizes the total expected $\hat{r}$. We follow the standard reward learning framework that uses supervised learning to learn a parameterized reward function, $\hat{r}_\theta$, with parameters θ (Christiano et al., 2017). In both scalar- and preference-based feedback settings, we consider trajectory segments σ, where σ consists of a sequence of states and actions: $\{s_t, a_t, s_{t+1}, a_{t+1}, ..., s_{t+k}, a_{t+k}\}$, with k as the segment size.

Preference-based Reward Learning In preference-based learning, two segments, σ^0 and σ^1, are compared by a teacher, yielding $y \in \{\ 0, 0.5, 1\}$. Specifically, if the teacher preferred segment σ^1 over segment σ^0, then y is set to 1, and if the converse is true y is set to 0. If both segments are equally preferred, then y is set to 0.5. As feedback is collected, it is stored as tuples (σ^0, σ^1, y) in the reward model data set D_{RM}. In general, if $\sigma^i > \sigma^j$, then the segment σ^i is preferred by the teacher over segment σ^j. We follow the Bradley-Terry model (Bradley & Terry, 1952) to define a preference predictor using the reward function $\hat{r}_\theta$:

$$P_\theta(\sigma^1 > \sigma^0) = \frac{\exp(\sum_t \hat{r}_\theta(s_t^1, a_t^1))}{\sum_{i \in \{0,1\}} \exp(\sum_t \hat{r}_\theta(s_t^i, a_t^i))} \tag{1}$$

Intuitively, this model assumes that the probability of the teacher preferring a segment depends exponentially on the total sum of predicted rewards along the segment. To train the reward function, we can use supervised learning where the teacher provides the labels y. More specifically, we update $\hat{r}_\theta$ by minimizing the standard binary cross-entropy objective:

$$L^{CE}(\theta, D) \ = \ - \ E_{(\sigma^0, \sigma^1, y) \sim D}\big[(1 \ - \ y) \ \log P_\theta(\sigma^0 \ > \ \sigma^1) + \ \ y \ \log P_\theta(\sigma^1 \ > \ \sigma^0)\big] \tag{2}$$

Scalar-based Reward Learning The primary difference between scalar and preference-based reward learning is that in scalar-based learning, the human teacher assigns numerical ratings to trajectory segments. In this setting, the comparisons between segments are implicit. More concretely, a teacher assigns a scalar value y to a segment σ^i, and as feedback is collected, it is stored as tuples (σ^i, y) in the reward model data set D_{RM}. We then apply standard regression and update $\hat{r}_\theta$ by minimizing the mean squared error:

$$L^{MSE}(\theta, D) = E_{(\sigma^i, y) \sim D}\big[(y - \sum_t \hat{r}_\theta(s_t^i, a_t^i))^2\big] \tag{3}$$

Human Studies in Human in the Loop RL To evaluate HitL RL algorithms, a common protocol is the use of simulated teachers, where feedback is provided according to a ground truth reward

function. This can be useful, as it offers an efficient and controlled evaluation setting. However, we argue that it is essential to evaluate HitL RL algorithms with *real human feedback*. This is especially important in light of recent work finding discrepancies between preference learning algorithms when using simulated teacher feedback versus human feedback (Metcalf et al., 2024).

4 SUB-OPTIMAL DATA PRE-TRAINING

In this section, we present SDP, a tool that leverages sub-optimal trajectories to improve the feedback efficiency for HitL RL. We refer to sub-optimal trajectories as sequences of (s, a) pairs such that:

$$r(s_t, a_t) - r_{\text{min}} < \epsilon \quad \forall t \in [t, t+k] \quad \text{for some small} \quad \epsilon > 0, \tag{4}$$

where r_{min} is the minimum possible environment reward. Equation 4 essentially expresses that the rewards achieved along a sub-optimal trajectory should be close to r_{min}. However, it is important to note that in practice we do not have access to the true reward. This prevents us from directly identifying sub-optimal trajectories using equation 4. Instead, we rely on selecting trajectories that we estimate will align with this criterion, such as gathering trajectories via a random policy.

Once sub-optimal trajectories are collected, we take the approach of pseudo-labeling all transitions with r_{min}. The goal of SDP is then to use this pseudo-labeled data to create a prior for rewards models in HitL RL methods (see Figure 1). Its simplicity enables SDP to be used in conjunction with any off-the-shelf HitL RL algorithm that learns a reward function from feedback.

SDP comprises two phases: (1) the reward model pre-training phase and (2) the agent update phase. In the reward model pre-training phase, we first gather a data set, D_{sub}, of N sub-optimal state, action transitions. We then pseudo-label all transitions in D_{sub} with rewards of r_{min}, resulting in $D_{\text{sub}} = \{s_i, a_i, r_{\text{min}}\}_{i=1}^{N}$. D_{sub} is then used to optimize the reward model $\hat{r}_\theta$ with the mean squared loss in Equation 3. As a result, the reward model $\hat{r}_\theta$ learns to associate all sub-optimal transitions with a low reward. Without such a prior, the reward model would initially have random estimates for these transitions; while the only way to improve such estimates is to obtain feedback from a teacher. Therefore, the reward model pre-training phase provides a valuable reward initialization without requiring any feedback.

Next, in the agent update phase, we initialize the RL agent's replay buffer D_{agent} with D_{sub}. The RL agent then briefly interacts with its environment and performs gradient updates according to its loss functions. The agent update process changes the RL agent's policy and generates new transitions, which are then stored in both the agent's replay buffer D_{agent} and the reward model's data set D_{RM}. It is important to note that in standard scalar- and preference-based reward learning, we query the teacher for feedback on trajectory segments sampled from D_{RM}. Therefore, adding new transitions into D_{RM} during the agent update phase is necessary to ensure that the teacher does not provide redundant feedback to the original sub-optimal transitions (as D_{RM} was empty prior to the agent update phase). When it is time for the teacher to provide their first set of feedback, the feedback can cover a different region of the state and action space, relative to the original sub-optimal data. In Appendix D.1, Figure 8 we empirically show that the agent update phase changes the RL agent's policy by performing policy rollouts and analyzing the differences in state distributions. See Algorithm 1 for the complete pseudocode.

At first glance, labeling sub-optimal transitions with an incorrect reward may seem problematic, as incorrect labels does introduce statistical bias into the reward model and the RL agent's value network. However, as the transitions are sub-optimal, we observed that the bias for using an incorrect reward is low (see Figure 9 in Appendix D.2). Moreover, by using the sub-optimal transitions, we increase the overall amount of data used by both models, which can *decrease* the models' variance, as shown in the offline RL setting (Yu et al., 2022).

5 EXPERIMENTS

This section considers the following four research questions (RQ's):

RQ 1: Can SDP improve upon existing scalar- and preference-based RL methods?

RQ 2: Can SDP effectively leverage sub-optimal trajectories from different tasks to improve performance on a target task?

RQ 3: Can SDP be used with real human feedback?

RQ 4: How sensitive is SDP to various hyperparameters?

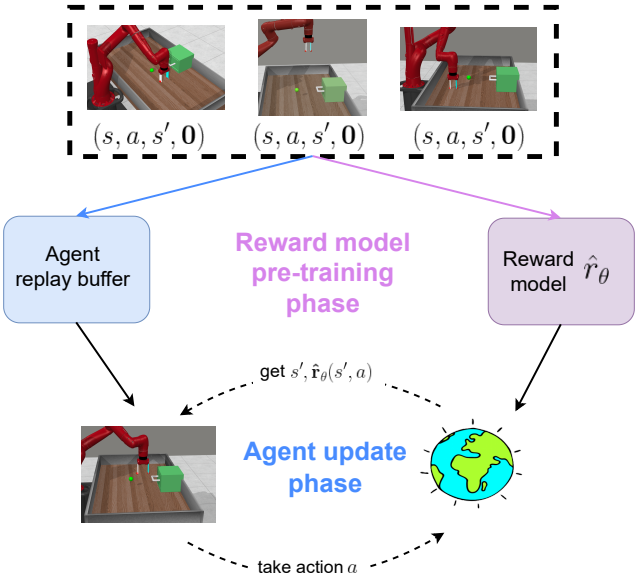


Figure 1: Overview of SDP: After obtaining sub-optimal trajectories, we pseudo-label this data with rewards of $r_{\text{min}} = 0$. We then pre-train the reward model $\hat{r}_\theta$ using this data set. During the agent update phase, we initialize the RL agent's replay buffer with the same pseudo-labeled data set. The agent then interacts in the environment and makes learning updates to obtain new behaviors for a teacher to give feedback.

Algorithm 1 SDP

Require: Reward model $\hat{r}_\theta \leftarrow \theta$ randomly initialized, Reward model data set $D_{\text{RM}} \leftarrow \emptyset$, RL agent with replay buffer $D_{\text{agent}} \leftarrow \emptyset$, Sub-optimal data set D_{sub} with reward labels r_{min}

// REWARD MODEL PRE-TRAIN PHASE
for each gradient step **do**
    Optimize $\hat{r}_\theta$ on D_{sub} with L^{MSE} (equ. 3)
end for
// AGENT UPDATE PHASE
$D_{\text{agent}} \leftarrow D_{\text{sub}}$
for each time-step t **do**
    Collect s_{t+1} by taking action $a_t \sim \pi(s_t)$
    Store $(s_t, a_t, \hat{r}_\theta, s_{t+1})$ in D_{agent}
    Store (s_t, a_t) in D_{RM}
    Update RL agent with D_{agent}
end for
Begin scalar- or preference-based RL using pre-trained $\hat{r}_\theta$, RL agent, and D_{agent}, D_{RM}

5.1 Experimental Design

To demonstrate the versatility and effectiveness of SDP, we apply SDP to both preference and scalar-based RL approaches. However, as preference feedback can be less time-consuming than scalar feedback, we primarily concentrate on preference-based RL in our experiments, exploring scalar

feedback in a smaller capacity. For the preference-based experiments, we combine SDP with four contemporary preference-based algorithms: PEBBLE (Lee et al., 2021a), RUNE (Liang et al., 2022), SURF (Park et al., 2022), and MRN (Liu et al., 2022). We benchmark the performance of the four algorithms augmented with SDP against their original versions without SDP, as well as against SAC. We treat SAC (Haarnoja et al., 2018) as an oracle (i.e., upper bound) because it learns while accessing the true reward function, which is unavailable to the other algorithms. For the scalar-based experiments, we combine SDP with R-PEBBLE (a regression variant of PEBBLE (Lee et al., 2021a)). We compare SDP + R-PEBBLE against R-PEBBLE, Deep TAMER (Warnell et al., 2018) (a scalar feedback RL algorithm), and the (oracle) SAC. We note that SAC is the core RL algorithm used across all baselines.

Implementation Details For SDP, we collected sub-optimal trajectories via a random policy (50,000 state, action transitions for all Section 5.2 experiments). Note that we do not require explicit access to a sub-optimal policy; we only require state, action transitions from said policy. Moreover, all reward model hyperparameters remained the same during both SDP phases (i.e., the reward model pre-train and agent update phases) as well as during the standard scalar- or preference-based RL that followed. As for feedback budgets, we maintained equal budgets for all algorithms within each environment. However, the budget itself was adjusted across environments to reflect their difficulty levels. Refer to Appendix A for a complete overview of the implementation process and specific hyperparameters for all algorithms.

Evaluation We show average offline performance (i.e., freeze the policy and evaluate it with no exploration) over ten episodes using either the ground truth reward function (DMControl experiments) or the success rate (Meta-World experiments). It is important to note that only SAC has access to the true reward function. We perform this evaluation every 10,000 training steps. To systemically evaluate performance, we use a simulated teacher that provides either a scalar rating of a single trajectory segment or preferences between two trajectory segments according to the true reward function. To thoroughly test the effectiveness of SDP, we perform evaluations on four robotic locomotion tasks from the DMControl Suite: Walker-walk, Cheetah-run, Quadruped-walk, and Cartpole-swingup, and five robotic manipulation tasks from Meta-World: Hammer, Door-unlock, Door-lock, Drawer-open, and Window-open. In our experiments, the results are averaged over five seeds with shaded regions or error bars indicating 95% confidence intervals. To test for significant differences in final performance (i.e., the undiscounted return) and learning efficiency (i.e., the total area under the return curve, AUC), we perform Welch t-tests (equal variances not assumed) with a p-value of 0.05. See Appendix D, Tables 9-14 for a summary of final performance and AUC across all experiments.

5.2 Locomotion and Manipulation Results

Preference Feedback Experiments We first address RQ 1 by evaluating the utility of SDP in the preference-based RL setting. Considering all four preference-based algorithms in the nine environments, SDP significantly ($p < 0.05$) improved learning (i.e., either final performance or AUC) in 23 out of the 36 experiments (see Figure 2). In the remaining experiments, there were no significant differences in the performance between SDP and the baseline algorithms. The addition of SDP (i.e., SDP + base algorithm) *never* statistically hurt performance.

Scalar Feedback Experiments Continuing our investigation into RQ 1, we now evaluate the performance of SDP in the scalar-based RL setting. We performed evaluations in Walker-walk, Cheetah-run, and Quadruped-walk. In Figure 3, we found that SDP (purple curve) significantly improves either the final performance or AUC compared to R-PEBBLE (navy curve) and Deep TAMER (yellow curve). More impressively, we found that SDP achieves comparable final performance to SAC (red curve), which uses the ground truth reward function, using as little as 60 feedback queries.

Leveraging Different Task Data in SDP Our previous experiments showed that SDP can leverage sub-optimal data from the target task to improve HitL RL methods. Now, addressing RQ 2, we investigate whether SDP can similarly use sub-optimal data from *related tasks* to improve performance on the target task. We perform three preference learning experiments, comparing PEBBLE with SDP + PEBBLE, using sub-optimal data from a different prior task that has the same virtual robot: (1) Walker-stand for Walker-walk, (2) Quadruped-walk for Quadruped-run, and (3) Drawer-

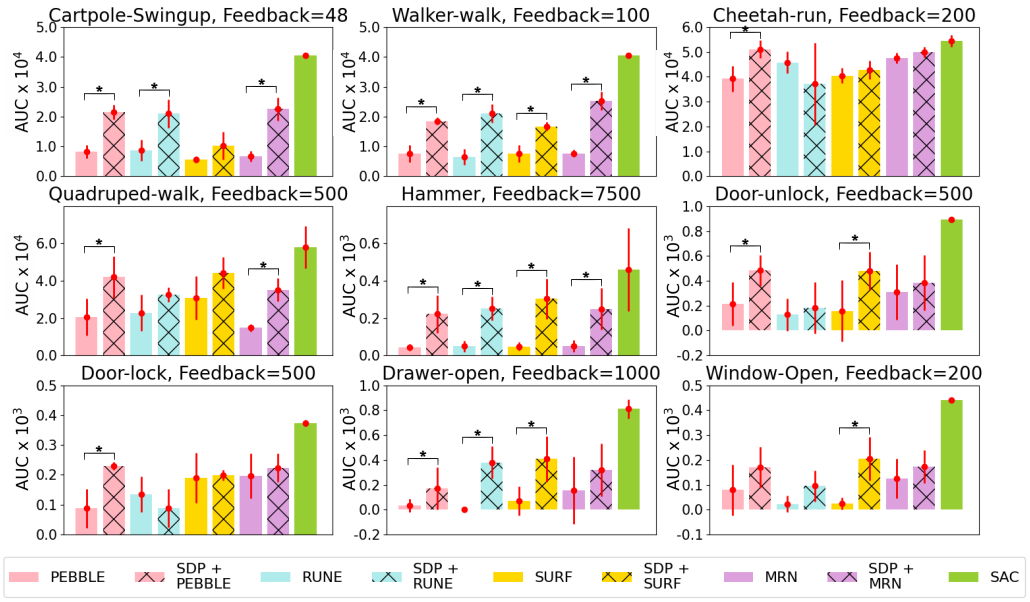


Figure 2: Preference feedback experiments in DMControl and Meta-World suites. The bar plots show AUC +/- 95% confidence intervals. * indicates that SDP + the base preference learning algorithm achieves a statistically greater score ($p < 0.05$) than the base preference learning algorithm.

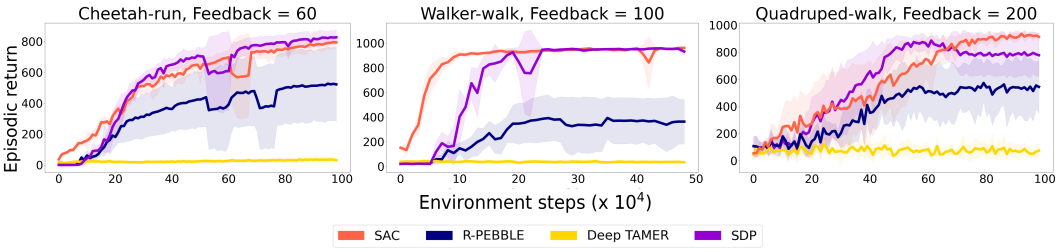


Figure 3: Scalar feedback experiments. In all DMControl experiments, SDP significantly outperforms R-PEBBLE and Deep TAMER ($p < 0.05$) and achieves comparable performance to SAC.

open for Door-open. To obtain the sub-optimal data for the prior tasks, we gathered transitions from partially trained policies as opposed to using random policies. Each partially trained policy achieved a final score of approximately 15-20% of that achieved by a fully trained policy. This ensured that the distribution of sub-optimal data differed between the prior and target tasks. See Appendix A.2 for further details on the experiment setup. Figure 4 demonstrates that in all three tested environments, SDP can successfully leverage sub-optimal data from related tasks (green curve) as it achieved similar performance to SDP when leveraging target task data (purple curve).

5.3 Preference Learning with Human Feedback

To evaluate HitL RL algorithms, we argue that it is critical to understand their efficacy with human teachers. However, human user studies do not appear to be widely adopted in the current literature. To empirically investigate this, we conducted a survey of 45 preference learning studies from 2012 to 2024[1] to understand the prevalence of human user studies in the preference learning literature. We found that *fewer than 50%* tested their proposed algorithms with human participants who were not also the authors. Moreover, of those studies that did involve human participants, only 41% included non-expert individuals, while none provided sufficient demographic information about their partic-

[1]See Appendix B.2 for more details.

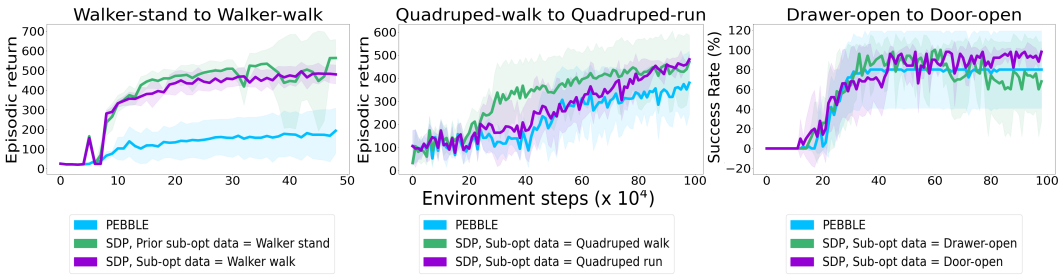


Figure 4: These figures highlight that SDP can leverage sub-optimal data from different prior tasks as it performed comparable to SDP when using target task data.

ipants (e.g., gender, race/ethnicity, level of education). These limitations raise significant concerns about whether and how current preference learning algorithms generalize to different populations.

To that end, we now address RQ 3 and evaluate the efficacy of SDP with real human feedback. We conduct an ethics-approved human-subject study of 16 participants (9 male, 7 female). Age ranges were collected: 18-24 (6), 25-30 (7), and 50-70 (3). Participants self-identified their membership in racial groups: South Asian (6), East Asian (3), White (4), Middle Eastern or North African (2), and multi-racial (1). The participants' highest educational attainments were: high school diploma (3), Bachelor's degrees (8), and Master's degrees (5), and their expertise varied across AI/ML (12), other computer science topics (1), and non-computer science fields (3). This highlights the diversity of our participants in terms of demographics and expertise.

This user study compares the performance of SDP and PEBBLE in two DMControl environments: Pendulum-swingup (with 7 participants) and Cartpole-swingup (with 12 participants). We focused our comparison to PEBBLE as it performed comparably to the other preference learning baselines in Section 5.2. We use a between-subjects experimental design — each participant provides preferences for a single seed of both SDP and PEBBLE algorithms. The preference budgets for Pendulum-swingup and Cartpole-swingup were 40 and 48, respectively. We selected these environments specifically because they could be solved with fewer preferences, aiming to reduce the overall time commitment required from participants. Each trial for a single environment lasted approximately 1-1.5 hours. See Appendix B for more details including user instructions and interface.

We visualize both algorithms' final performance and AUC in Figure 5. We found that in both environments SDP (purple plots) maintains significant ($p < 0.05$) performance gains over PEBBLE (blue plots) in terms of either final performance or AUC. In Cartpole-swingup, we also observed consistent performance from SDP regardless of whether the teacher was human or simulated. This suggests that our prior results with simulated teachers can generalize to settings where human feedback is provided. Moreover, we observed no significant differences in SDP's effectiveness across demographic factors, including gender, age, educational, and computer science background (see Appendix C, Tables 7 and 8). This finding is particularly encouraging for the potential real-world deployment of SDP, highlighting its usability for non-expert users across diverse demographics.

5.4 Ablation and Sensitivity Studies

To further understand the effectiveness of SDP, we perform further analysis of SDP across three dimensions: (1) the phases of SDP, (2) the number of feedback queries, and (3) the amount of sub-optimal data. This analysis aims to address RQ 4 and provide a deeper understanding of the factors influencing SDP's performance. For these experiments, we focus on SDP + R-PEBBLE in the Walker-walk environment. Supplementary results for Cheetah-run can be found in Appendix D.

SDP Component Analysis First, we evaluate the effect of each phase of SDP individually, the reward model pre-train phase and the agent update phase. Figure 6–leftmost demonstrates the importance of using both phases in SDP for scalar-based RL approaches. We found that the SDP variants that only use one of the phases (green and gray curves) result in worse performance than the full SDP (purple curve).

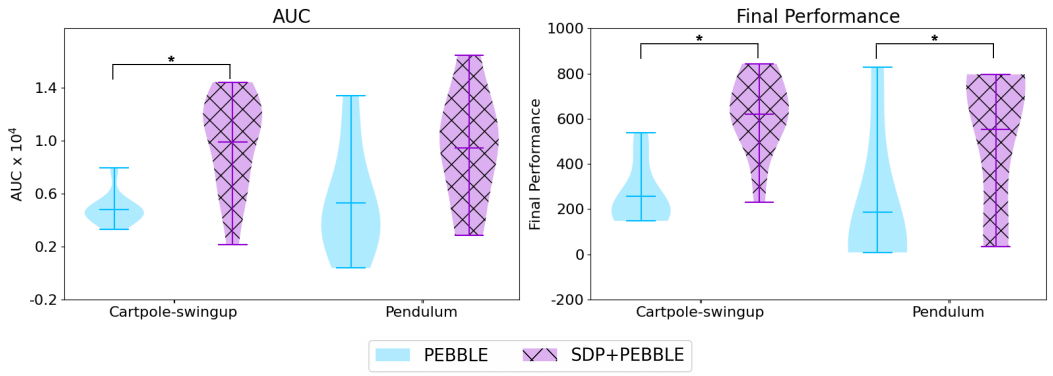


Figure 5: This figures demonstrate that SDP can significantly outperform PEBBLE in terms of both AUC (left) and final performance (right) even when human teachers are providing preferences.

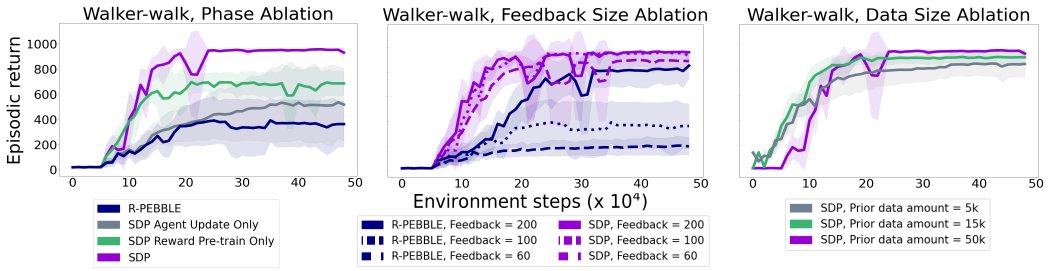


Figure 6: These figures show ablation and sensitivity studies of SDP in Walker-walk .

Effect of Feedback Amount We evaluate SDP and R-PEBBLE with feedback budgets of 60, 100, and 200 to analyze the impact of feedback quantity on performance. As shown in Figure 6 (middle), SDP (purple curves) consistently outperforms R-PEBBLE (navy curves), further demonstrating its effectiveness across varying feedback levels.

Effect of Sub-Optimal Data Amount We evaluate the performance of SDP using varying amounts of sub-optimal transitions: 5000, 15000, and 50000. Figure 6 (rightmost) indicates that while 5000 transitions (gray curve) led to the poorest performance, increasing this amount to 15000 or 50000 (green and purple curves) yielded comparable or improved results, suggesting that more sub-optimal data can benefit SDP.

6 Conclusion

In this work, we present SDP, an approach that improves the feedback efficiency for HitL RL algorithms. SDP is specifically designed to leverage reward-free, sub-optimal data for scalar- and preference-based HitL RL approaches. By pseudo-labeling low-quality data with the minimum environment rewards, we can pre-train the reward model without the need for human labeling. This provides the reward model with a head start in learning. This head start allows the reward model to learn to associate low-quality transitions with low reward values, even before receiving any actual human feedback. Our simulated teacher experiments in DMControl and Meta-World suites demonstrate that SDP can significantly improve a variety of preference- and scalar-based reward learning algorithms. Importantly, we further validite the real-world applicability of SDP by demonstrating its success in a 16-person user study. This work takes an important step towards considering how sub-optimal data can be leveraged for HitL RL.

Ethics Statement

The goal of preference and scalar-based RL is to learn reward functions that encode human preferences. However, this necessitates interaction with human users, raising concerns about potential biases in the collected preferences. If these preferences are primarily drawn from a non-representative group, the resulting RL system may unfairly prioritize the desires or needs of that group over others. To that end, we performed a human subject study that was reviewed and approved by an external ethics committee. This ensured the responsible and ethical collection of human preferences. In addition, we took care to gather participants from a wide range of both demographic and educational backgrounds, which we detail in Section 5.3.

Reproducibility Statement

To ensure the reproducibility of our work, we provide a link to our anonymous downloadable source code: `https://anonymous.4open.science/r/SDP_ICLR-C23F/`. We also provide pseudocode in Section 4. Moreover, we outline extensive training and evaluation details in Section A and Appendix A.2. We also outline all hyperparameters used for all algorithms and environments in Appendix A.2, Tables $1-5$. Lastly, for the human subject study, we provide details on the instructions provided to the participants in Appendix B as well as a description of the preference interface used. In our released codebase, we also provide the specific code files used to run the experiments in the user study. In addition, an anonymized version of the human subject data will be released, as stated in our ethics approval.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Twenty-First International Conference on Machine Learning*, 2004.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Thirty-First Conference on Neural Information Processing Systems*, 2017.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 2009.

Peter Barnett, Rachel Freedman, Justin Svegliato, and Stuart Russell. Active reward learning from multiple teachers. In *SafeAI 2023, The AAAI Workshop on Artificial Intelligence Safety*, 2023.

Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *2nd Conference on Robot Learning*, 2018. URL `http://arxiv.org/abs/1810.04303`.

Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 2022.

Yaniv Blumenfeld, Dar Gilboa, and Daniel Soudry. Beyond signal propagation: Is feature diversity necessary in deep neural network initialization? In *Thirty-Seventh International Conference on Machine Learning*, 2020.

Serena Booth, W. Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The perils of trial-and-error reward design: Misdesign through overfitting and invalid task specifications. In *Thirty-Seventh AAAI Conference on Artificial Intelligence*, 2023.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 1952.

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

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Third Proceedings of the Conference on Robot Learning*, 2020.

Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Erdem Bıyık, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. Active preference-based gaussian process regression for reward learning. In *Robotics: Science and Systems*, 2020. URL https://arxiv.org/abs/2005.02575.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *Robotics: Science and Systems*, 2020.

Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Forth Conference on Robot learning*, 2021.

Jie Cheng, Gang Xiong, Xingyuan Dai, Qinghai Miao, Yisheng Lv, and Fei-Yue Wang. RIME: Robust preference-based reinforcement learning with noisy preferences. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8229–8247. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cheng24k.html.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Thirty-First Conference on Neural Information Processing Systems*, 2017.

Jack Clark and Dario Amodei. Faulty reward functions in the wild, 2016. URL https://openai.com/research/faulty-reward-functions.

Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active reward learning. In *Robotics: Science and Systems*, 2014.

Oliver Daniels-Koch and Rachel Freedman. The expertise problem: Learning from specialized feedback. In *ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022. URL https://arxiv.org/abs/2211.06519.

Anca Dragan and Siddhartha Srinivasa. Formalizing assistive teleoperation. *Robotics: Science and Systems*, 2012.

Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. In *Thirty-Fith International Conference on Machine Learning*, 2019.

Joseph Giovanelli, Alexander Tornede, Tanja Tornede, and Marius Lindauer. Interactive hyperparameter optimization in multi-objective problems via preference learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Twenty-Sixth Conference on Neural Information Processing Systems*, 2013.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Thirty-Fifth International Conference on Machine Learning*, 2018.

Donald Joseph Hejna III and Dorsa Sadigh. Few-shot preference learning for human-in-the-loop rl. In *Sixth Conference on Robot Learning*, 2023.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Simon Holk, Daniel Marta, and Iolanda Leite. Predilect: Preferences delineated with zero-shot language-based reasoning in reinforcement learning. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, volume 32 of *HRI '24*, pp. 259–268. ACM, March 2024. doi: 10.1145/3610977.3634970. URL http://dx.doi.org/10.1145/3610977.3634970.

Xiao Hu, Jianxiong Li, Xianyuan Zhan, Qing-Shan Jia, and Ya-Qin Zhang. Query-policy misalignment in preference-based reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=UoBymIwPJR.

Minyoung Hwang, Gunmin Lee, Hogun Kee, Chan Woo Kim, Kyungjae Lee, and Songhwai Oh. Sequential preference ranking for efficient reinforcement learning from human feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=MIYBTjCVjR.

Minyoung Hwang, Luca Weihs, Chanwoo Park, Kimin Lee, Aniruddha Kembhavi, and Kiana Ehsani. Promptable behaviors: Personalizing multi-objective rewards from human preferences. In *Conference on Computer Vision and Pattern Recognition*, 2023b. URL https://openaccess.thecvf.com/content/CVPR2024/papers/Hwang_Promptable_Behaviors_Personalizing_Multi-Objective_Rewards_from_Human_Preferences_CVPR_2024_paper.pdf.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. In *Thirty-Second Conference on Neural Information Processing Systems*, 2018.

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

W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement. In *Fifth International Conference on Knowledge Capture*, 2009.

W Bradley Knox and Peter Stone. Learning non-myopically from human-generated reward. In *International Conference on Intelligent User Interfaces*, 2013.

W. Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, and Alessandro Allievi. Models of human preference for learning reward functions. *Transactions on Machine Learning Research (TMLR)*, 2023.

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

Kimin Lee, Laura Smith, Anca 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. URL https://openreview.net/forum?id=ps95-mkHF_.

Xinran Liang, Katherine Shu, Kimin Lee, and Pieter Abbeel. Reward uncertainty for exploration in preference-based reinforcement learning. In *Tenth International Conference on Learning Representations*, 2022.

Runze Liu, Fengshuo Bai, Yali Du, and Yaodong Yang. Meta-reward-net: Implicitly differentiable reward learning for preference-based reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=OZKBReUF-wX.

Runze Liu, Yali Du, Fengshuo Bai, Jiafei Lyu, and Xiu Li. PEARL: Zero-shot cross-task preference alignment and robust reward learning for robotic manipulation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 30946–30964. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24o.html.

Yi Liu, Gaurav Datta, Ellen Novoseller, and Daniel S. Brown. Efficient preference-based reinforcement learning using learned dynamics models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL https://arxiv.org/abs/2301.04741.

Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, and David L Roberts. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 2016.

James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, David Roberts, Matthew E Taylor, and Michael L Littman. Interactive learning from policy-dependent human feedback. In *Thirty-Forth International Conference on Machine Learning*, 2017.

Daniel Marta, Simon Holk, Christian Pek, Jana Tumova, and Iolanda Leite. Variquery: Vae segment-based active learning for query selection in preference-based reinforcement learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7878–7885, 2023a. doi: 10.1109/IROS55552.2023.10341795.

Daniel Marta, Simon Holk, Christian Pek, Jana Tumova, and Iolanda Leite. Aligning human preferences with baseline objectives in reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7562–7568. IEEE, 2023b.

Daniel Marta, Simon Holk, Christian Pek, and Iolanda Leite. Sequel: Semi-supervised preference-based rl with query synthesis via latent interpolation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9585–9592. IEEE, 2024.

Thommen George Karimpanal Maxence Hussonnois and Santu Rana. Controlled diversity with preference: Towards learning a diverse set of desired skills. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023.

Shaunak A Mehta and Dylan P Losey. Unified learning from demonstrations, corrections, and preferences during physical human–robot interaction. *ACM Transactions on Human-Robot Interaction*, 13(3):1–25, 2024.

Katherine Metcalf, Miguel Sarabia, Natalie Mackraz, and Barry-John Theobald. Sample-efficient preference-based reinforcement learning with dynamics aware rewards. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=i84V7i6KEMd.

Katherine Metcalf, Miguel Sarabia, Masha Fedzechkina, and Barry-John Theobald. Can you rely on synthetic labellers in preference-based reinforcement learning? it's complicated. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. doi: 10.1609/aaai.v38i9.28877. URL https://ojs.aaai.org/index.php/AAAI/article/view/28877.

Vivek Myers, Erdem Bıyık, Nima Anari, and Dorsa Sadigh. Learning multimodal rewards from rankings. In *Fifth Conference on Robot learning*, 2022.

Vivek Myers, Erdem Biyik, and Dorsa Sadigh. Active reward learning from online preferences. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7511–7518, 2023. URL https://api.semanticscholar.org/CorpusID:257219724.

Jongjin Park, Younggyo Seo, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. Surf: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. In *Tenth International Conference on Learning Representations*, 2022.

Zhizhou Ren, Anji Liu, Yitao Liang, Jian Peng, and Jianzhu Ma. Efficient meta reinforcement learning for preference-based fast adaptation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=61UwgeIotn.

Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. Inverse reinforcement learning from failure. In *Fifteenth International Conference on Autonomous Agents and Multiagent Systems*, 2016.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. In *Conference on Robot Learning*, 2020.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal. A minimaximalist approach to reinforcement learning from human feedback. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2401.04056.

Voot Tangkaratt, Nontawat Charoenphakdee, and Masashi Sugiyama. Robust imitation learning from noisy demonstrations. In *Twenty-Forth International Conference on Artificial Intelligence and Statistics*, 2021.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *Tenth International Conference on Autonomous Agents and Multiagent Systems*, 2011.

Mudit Verma and Katherine Metcalf. Hindsight PRIORs for reward learning from human preferences. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NLevOah0CJ.

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. Exploiting unlabeled data for feedback efficient human preference based reinforcement learning. In *The AAAI 2023 Workshop on Representation Learning for Responsible Human-Centric AI*, 2023. URL https://arxiv.org/abs/2302.08738.

Ngo Anh Vien, Wolfgang Ertel, and Tae Choong Chung. Learning via human feedback in continuous state and action spaces. *Applied intelligence*, 2013.

Ren-Jian Wang, Ke Xue, Yutong Wang, Peng Yang, Haobo Fu, Qiang Fu, and Chao Qian. Diversity from human feedback. In *2nd Workshop on Agent Learning in Open-Endedness (ALOE) at NeurIPS 2023*, 2023. URL https://arxiv.org/abs/2310.06648.

Ruiqi Wang, Weizheng Wang, and Byung-Cheol Min. Feedback-efficient active preference learning for socially aware robot navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11336–11343. IEEE, 2022.

Xiaofei Wang, Kimin Lee, Kourosh Hakhamaneshi, Pieter Abbeel, and Michael Laskin. Skill preferences: Learning to extract and execute robotic skills from human feedback. In *Fifth Conference on Robot Learning*, 2021.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Devin White, Mingkang Wu, Ellen Novoseller, Vernon J Lawhern, Nicholas Waytowich, and Yongcan Cao. Rating-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Nils Wilde, Erdem Bıyık, Dorsa Sadigh, and Stephen L Smith. Learning reward functions from scale feedback. In *Conference on Robot Learning*, 2021.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/16c222aa19898e5058938167c8ab6c57-Paper.pdf.

Wanqi Xue, Bo An, Shuicheng Yan, and Zhongwen Xu. Reinforcement learning from diverse human preferences. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*, 2024.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Third Conference on Robot Learning*, 2020.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2021.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. In *Thiry-Ninth International Conference on Machine Learning*, 2022.

Guoxi Zhang and Hisashi Kashima. Learning state importance for preference-based reinforcement learning. *Machine Learning*, 113(4):1885–1901, 2024.

Jiawei Zhao, Florian Tobias Schaefer, and Anima Anandkumar. Zero initialization: Initializing neural networks with only zeros and ones. *Transactions on Machine Learning Research*, 2022.

Tianchen Zhu, Yue Qiu, Haoyi Zhou, and Jianxin Li. Decoding global preferences: Temporal and cooperative dependency modeling in multi-agent preference-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. doi: 10.1609/aaai.v38i15.29666. URL https://ojs.aaai.org/index.php/AAAI/article/view/29666.

APPENDIX

A SIMULATED EXPERIMENT DETAILS

A.1 BENCHMARKS

PEBBLE PEBBLE has two primary components: unsupervised exploration and off-policy learning with relabeling. The purpose of the unsupervised exploration phase is to collect diverse experiences for a human teacher to provide feedback on. More specifically, PEBBLE optimizes state entropy to explore the environment. Furthermore, PEBBLE uses off-policy reinforcement learning to learn a policy. PEBBLE specifically uses an off-policy RL algorithm as they are more sample efficient compared to their on-policy counterparts. Then as the reward model changes, PEBBLE relabels all transitions in the RL agent's replay buffer with the latest reward model. This is integral as the reward model is non-stationary, and relabeling the transitions stabilizes the learning process.

To adapt PEBBLE to the scalar feedback setting, we make one minor change to the reward model. In the scalar feedback setting, we use a scripted teacher that provides a scalar rating of a single trajectory segment. Therefore, the only update to PEBBLE is with respect to the loss function. Instead of using the cross-entropy loss in Equation 2, we use the mean-squared error loss in Equation 3.

RUNE RUNE is a preference learning algorithm (built on top of PEBBLE) that uses an uncertainty-based exploration strategy to improve feedback efficiency. To encourage exploration for the SAC agent, RUNE adds an intrinsic reward component based on the standard deviation in the reward model.

SURF SURF is another preference learning algorithm (built on top of PEBBLE) that improves feedback efficiency by using semi-supervised and data augmentation approaches. To incorporate semi-supervised learning, SURF generates pseudo-labels for unlabeled trajectories by querying the learned reward model. If the reward model confidently (e.g., low output standard deviation) predicts the pseudo-label, then the trajectory, label pair is added to the reward model training data set. Further, SURF proposes a new data augmentation technique that crops sub-sequences of trajectories.

MRN MRN is a preference learning algorithm also integrated with PEBBLE. Unlike other PbRL algorithms, MRN is a bi-level optimization algorithm in which the actor and critic are updated in the inner loop, and the reward model is updated in the outer loop. Importantly, the reward model takes into account the performance of the critic on the preference data.

Deep TAMER In our scalar feedback experiments, we also consider the Deep TAMER benchmark. In Deep TAMER, scalar feedback is used to learn a human reward function via regression. Then the agent acts greedily with respect to this reward function. Furthermore, the original implementation of Deep TAMER was built on top of DQN. Therefore, there was no separate actor-network. In addition, Deep TAMER only used discrete feedback $\in [-1, 0, 1]$.

To make Deep TAMER a fair benchmark, we made a few adjustments. To start, we allow Deep TAMER to learn from real-valued feedback as done in the other scalar-based experiments. However, instead of using the ground truth reward function as feedback, we use the state-action values from a fully-trained SAC agent. We do this because, in TAMER (and Deep TAMER), the teacher is intended to provide feedback representative of the return. Secondly, in Deep TAMER, feedback is provided per (state, action) pair. Therefore, to make sure Deep TAMER received the same amount of feedback as the other baselines, we used *trajectory segment size* $\times$ *feedback amount* for Deep TAMER only. The other benchmarks receive a scalar feedback value that is the sum of rewards along a trajectory segment. Third, to learn the reward model we use standard regression as described in Section 3.1. Lastly, as our testing environments are continuous state and action, we learn a separate actor policy, similarly done in (Vien et al., 2013).

A.2 Training Details

In all of our experiments, we use the hyperparameters in Table 1 for the reward models used in all benchmarks. For the agent update phase of SDP, an additional hyperparameter is associated with the number of environment interactions made before the standard preference/scalar feedback learning loop begins. However, for simplicity, we kept the same value as the existing feedback frequency hyperparameter, as feedback frequency also dictates the number of environment interactions made between feedback sessions

Furthermore, we use most of the existing reward model hyperparameters used in PEBBLE, however, we adjusted the following four hyperparameters: feedback frequency, amount of feedback per session, trajectory segment size (only for Meta-world), and activation function for the final NN layer. We adjusted the first two hyperparameters because PEBBLE originally used a significantly larger feedback budget, therefore we wanted the feedback schedule to better reflect a smaller feedback budget. We used a different trajectory segment size for Meta-world because we wanted to keep the segment sizes the same across both the DMControl and Meta-world environments. Moreover, we found that the output activation function could significantly affect learning, therefore we tested all benchmarks using both Tanh (original activation used) and Leaky-ReLU and chose the reward model that achieved better final performance. For the RUNE and SURF baselines, we use any hyperparameters associated with their specific algorithm according to the original paper (see Table 2). For a fair comparison with SDP, we provide all HiTL baselines (e.g., PEBBLE, R-PEBBLE, Deep TAMER, RUNE, and SURF) with the sub-optimal data set to be used in both the reward model and by the RL agent.

Furthermore, to select trajectory segments for the teacher to provide feedback on, we use uniform sampling in the DMControl tasks and disagreement sampling in the Meta-world tasks. Disagreement sampling is a popular active learning approach in which trajectories with higher uncertainty (based on an ensemble of neural networks) are more likely to be sampled (Christiano et al., 2017). As for the SAC hyperparameters, we use the values found in Tables 3-4.

For the experiments in which we leveraged sub-optimal data from a different task (i.e., Walker-stand, Quadruped-walk, Drawer-open), we gathered 50,000 transitions from partially trained policies. We note that for these experiments, we purposely did not use transitions gathered from a random policy. In these experiments, the prior and target tasks were environments in which the simulated robot was identical. The only difference is the environmental reward. Therefore, the random policy for both environments would be the same. To truly demonstrate transfer, we wanted to ensure we obtained low-quality transitions of the prior task that were different from the target task.

Each partially trained policy achieved a final score of approximately 15-20% of that achieved by a fully trained policy. More specifically, we used the following procedure to train the SAC policies. First, for Walker-stand, we trained a SAC policy for 5,000 time steps, and the average final performance was approximately 194. Second, for Quadruped-walk, we trained a SAC policy for 100,000 timesteps, and the average final performance was approximately 184. Lastly, for Drawer-open, we trained a SAC policy for 50,000 time steps, and the average final success rate was approximately 14%.

Table 1: Hyperparameters for the reward model used in all experiments (both preference and scalar feedback variants).

Hyperparameter	Value
segment size	50
number of random steps (i.e., sub-optimal data transitions)	50000
number of unsupervised exploration steps	9000
consider unsupervised explorations as sub-optimal	False (Walker-walk, Cartpole-swingup, Quadruped-walk – PbRL) True (others)
frequency of feedback	20000 (DMControl) 10000 (Window-open, Door-unlock, Door-lock) 5000 (Hammer, Drawer-open)
feedback budgets in ablations	2000 (Door-open) 1000 (Cheetah-run, Walker-walk)
number of feedback queries per session	50 (Hammer, Drawer-open) 8 (Cartpole-swingup) 20 (others)
sampling scheme	disagreement sampling (Metaworld) uniform sampling (DMControl)
number of training epochs	200 (Window-open, SURF and SDP + SURF) 50 (others)
learning rate	3×10^{-4}
intermediate neural network activation	Leaky ReLU
batch size	128
number of hidden layers	4
number of neurons per hidden layer	128
loss function	Mean Squared error (Scalar feedback) Cross Entropy loss (Preference feedback)
optimizer	Adam (Kingma & Ba, 2015)
For Reward Model Pre-training Phase in SDP	
last layer neural network activation	Tanh
For Human-in-the-Loop Alg	
last layer neural network activation	Leaky ReLU (PEBBLE, RUNE, SURF) Tanh (SDP – Hammer, Drawer-open, Walker-walk) Leaky ReLU (SDP – others)

Table 2: Baseline Specific Hyperparameters

Hyperparameter	Value
Specific RUNE Hyperparameters	
beta schedule	linear decay
beta init	0.05
beta decay	10^{-5}
Specific SURF Hyperparameters	
threshold λ	0.99
unlabeled batch ratio	4
loss weight	1
min/max length of cropped segment	$[45, 55]$
segment length before cropping	60
Specific MRN Hyperparameters	
num meta steps	1000 (Walker-walk) 3000(Quadruped-walk) 10000 (Door-open) 5000 (others)

Table 3: Hyperparameters for SAC that were shared by all algorithms. SAC is the standard RL algorithm that learns from the environment's true reward signal, not via preference learning.

Hyperparameter	Value
optimizer	Adam (Kingma & Ba, 2015)
discount	0.99
alpha learning rate	10^{-4}
actor betas	0.9, 0.999
critic betas	0.9, 0.999
alpha betas	0.9, 0.999
target smoothing coefficient	0.005
actor update frequency	1
critic target update frequency	2
init temperature	0.1
network type	MLP
nonlinearity	ReLU
number of gradient updates per step	1

Table 4: Specific SAC hyperparameters that were tuned for the DMControl and Metaworld experiments. The majority of these hyperparameters were selected from PEBBLE repo (Lee et al., 2021a).

Hyperparameter	Value
DMControl	
batch size	512 (Cartpole-swingup) 1024 (others)
number of hidden layers	2
number of neurons per hidden layer	256 (Cartpole-swingup) 1024 (others)
actor/critic learning rate	5×10^{-5} (Cheetah-run preference feedback) 10^{-4} (Cheetah-run scalar feedback) 5×10^{-4} (Walker-walk) 10^{-4} (others)
number of training steps	0.5×10^6 (Walker-walk, Cartpole-swingup) 10^6 (others)
Metaworld	
batch size	512
number of hidden layers	3
number of neurons per hidden layer	256
actor/critic learning rate	3×10^{-4}
number of training steps	0.5×10^6 (Door-lock and Window-open) 10^6 (others)

Table 5: Tuned hyperparameters for human feedback experiments.

Hyperparameter	Value
SAC	
number of training steps	0.3×10^6
batch size	1024
number of hidden layers	2
number of neurons per hidden layer	1024
actor/critic learning rate	10^{-4}
number of training steps	2.5×10^5
Reward Model	
consider unsupervised explorations as sub-optimal	False
frequency of feedback	20000
number of feedback queries per session	10 (Pendulum-swingup) 8 (Cartpole-swingup)
sampling scheme	uniform sampling
batch size	128
last layer neural network activation	Leaky ReLU

B HUMAN SUBJECT STUDY DETAILS

Our study was approved by an external ethics committee. In total, 16 users participated in our study, 3 of which participated in both Pendulum-swingup and Cartpole-swingup tasks. As for the amount of participation per task, 7 users provided preferences in Pendulum-swingup and 12 users provided preferences in Cartpole-swingup. During the timeline of our user study, we initially had participants interacting in Pendulum-swingup, however we found that it was very simple, therefore we decided to include Cartpole-swingup, as a more difficult task. Each participant provided preferences for a single run of the environment, as each session took approximately 1 to 1.5 hours. Table 5 shows the specific hyperparameters used.

B.1 PROTOCOL STEPS

The study took place in person, where the participant was first provided the consent from to review and sign. Then, participants and the researcher, together, reviewed the instructions outlining the objective of the study. The instructions had two components. First, they outlined participant's goal in the study. Second, as done in Christiano et al. (2017), we provided a guide on what constitutes good and bad behavior in each domain.

Pendulum-swingup The first set of instructions for the study are as follows:

1. Objective: for the pole to swing up and balance.
2. Agent controls how much torque (or force or twist) to apply
3. It does not matter which way the pole swings, left or right, as long as the pole balances.
4. Your task: You will see two clips of behaviors, and your job is to select the clip you think is better given the above objective. If you think both clips are identical in behavior, you can select equally preferable.

The second set of instructions (i.e., advice) for the study are as follows:

1. The first priority is for the pole to begin swinging back and forth (i.e., picking up momentum). Therefore, the video clip where the pole has swung higher is better. Even if the agent is not behaving well in either clips, if you can tell that the pole is higher in one clip than the other, it is better to prefer that clip.
2. In general, if the pole is swinging rapidly in a circle (e.g., complete 360), then this is usually worse behavior than if the pole is barely moving.

Cartpole-swingup The first set of instructions for the study are as follows:

1. Pole is attached to a cart.
2. Agent can move the cart left and right.
3. Goal is for the agent to move the cart such that the pole swings up and balances
4. Your task: You will see two clips of behaviors, and your job is to select the clip you think is better given the above objective. If you think both clips are identical in behavior, you can select equally preferable.

The second set of instructions (i.e., advice) for the study are as follows:

1. The first priority is for the pole to begin swinging back and forth (i.e., picking up momentum). Therefore, the video clip where the pole has swung higher is better. Even if the agent is not behaving well in either clips, if you can tell that the pole is higher in one clip than the other, it is better to prefer that clip.
2. In general, if the pole is swinging rapidly in a circle (e.g., complete 360), then this is usually worse behavior than if the pole is barely moving.
3. If the pole is balancing then falls over, then is is better behavior than if the pole is barely moving.

During the instruction period, participants also watched video clips of what successful and unsuccessful behavior looks like. In addition, participants were able to ask questions about the environment's objective.

Once the instruction period was complete, participants completed a practice round in providing preferences. This was included so users could gain familiarity with the interface. Figure 7 shows a screenshot of the interface used. Participants would use keyboard input to move from one video clip to the next. Then, participants had the opportunity to rewatch either clips as many times as they liked. Afterwards, participants chose which clip they preferred (or equally preferred) via keyboard input. Each video clip consisted of a 50 step segment, which was approximately 2 seconds long. The total study time was $\sim (1 - 1.5)$ hours. After the study was complete, participants filled out a demographic survey which included questions pertaining to their age, race/ethnicity, education level completed, and area of expertise.

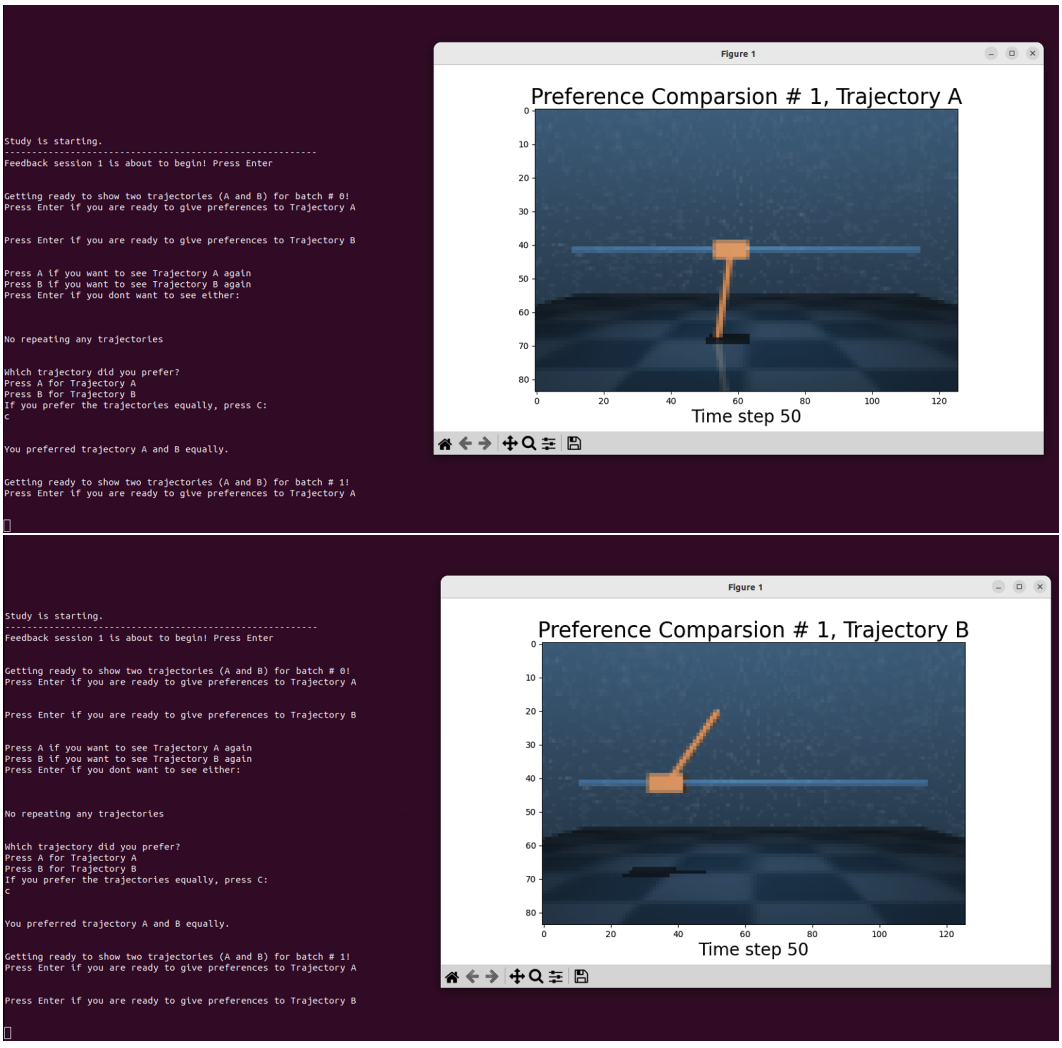


Figure 7: This shows the user interface used for the human subject study. Participants would view each video clip, sequentially, then decide which clip they preferred.

B.2 Preference Learning Survey Details

Table 6 provides a list of all papers included in our preference learning survey. We now describe the criteria for inclusion and exclusion in our survey. To find papers, we used Google Search with key words: preference learning, reinforcement learning from human feedback, and preference based reinforcement learning. We also found papers by reviewing those that cited popular preference learning works. This included Christiano et al. (2017) and Lee et al. (2021a), as these were two of the first papers that popularized preference learning. We included conference, journal and workshop papers in our survey. We did not include papers that were strictly on Arxiv. To keep the survey within scope, we did not include any preference learning works related to large language or foundation models. We also did not include any works that were entirely theoretical (e.g., no empirical results were provided).

Table 6: Articles used in Preference Learning Survey from Section 5.3. NA indicates no information was provided.

Paper Reference	Human Users(Non-Author)	Number of Users
(Myers et al., 2023)	Yes	22
(Metcalf et al., 2024)	Yes	50
(Myers et al., 2022)	Yes	50
(Christiano et al., 2017)	Yes	NA
(Hejna III & Sadigh, 2023)	Yes	4
(Hwang et al., 2023b)	Yes	5
(Biyik & Sadigh, 2018)	Yes	10
(Hwang et al., 2023a)	Yes	5
(Bıyık et al., 2020)	Yes	10
(Wilde et al., 2021)	Yes	18
(Bıyık et al., 2022)	Yes	15
(Ibarz et al., 2018)	Yes	NA
(Knox et al., 2023)	Yes	143
(Holk et al., 2024)	Yes	32
(Metcalf et al., 2023)	Yes	40
(Marta et al., 2024)	Yes	NA
(Marta et al., 2023a)	Yes	70
(Marta et al., 2023b)	Yes	20
(Ren et al., 2022)	Yes	NA
(Mehta & Losey, 2024)	Yes	15
(White et al., 2024)	Yes	20
(Wang et al., 2022)	Yes	10
(Wang et al., 2023)	No	
(Zhang & Kashima, 2024)	No	
(Lee et al., 2021a)	No	
(Liang et al., 2022)	No	
(Park et al., 2022)	No	
(Hu et al., 2024)	No	
(Liu et al., 2022)	No	
(Liu et al., 2023)	No	
(Xue et al., 2024)	No	
(Daniels-Koch & Freedman, 2022)	No	
(Verma et al., 2023)	No	
(Barnett et al., 2023)	No	
(Verma & Metcalf, 2024)	No	
(Metcalf et al., 2023)	No	
(Swamy et al., 2024)	No	
(Lee et al., 2021b)	No	
(Wang et al., 2021)	No	
(Maxence Hussonnois & Rana, 2023)	No	
(Wilson et al., 2012)	No	
(Liu et al., 2024)	No	
(Giovanelli et al., 2024)	No	
(Zhu et al., 2024)	No	
(Cheng et al., 2024)	No	

C Additional Human Teacher Results

Task	Feedback	Background	AUC	P Value	Final Performnce	P Value
Cartpole-swingup	48	Non-CS CS	8258.44 ± 1898.47 10208.11 ± 2311.16	0.183	648.88 ± 35.17 614.14 ± 114.81	0.692
Pendulum-swingup	40	Non-CS CS	7677.84 ± 3540.70 10188.70 ± 3858.94	0.258	327.65 ± 407.80 641.68 ± 237.76	0.23

Table 7: This table shows the AUC and final performance of SDP (mean +/- 95% confidence intervals) for CS and non-CS participants in the human subject study.

Task	Feedback	Demographic	AUC	P Value
Cartpole-swingup	48	Female	8720.03 ± 3185.4	0.085
		Male	11770.22 ± 1669.96	
Pendulum-Swingup	40	Female	8587.13 ± 2772.96	0.331
		Male	10134.67 ± 4822.22	
Cartpole-swingup	48	Age 18-30	10109.54 ± 2370.61	0.313
		Age 50-70	10923.05 ± 1292.83	
Pendulum-Swingup	40	Age 18-30	8619.07 ± 3381.48	0.318
		Age 50-70	10607.93 ± 5249.72	
Cartpole-swingup	48	Education: Bachelors or higher	11044.55 ± 1783.25	0.794
		Education: High school diploma	6247.99 ± 5186.47	
Pendulum-Swingup	40	Education: Bachelors or higher	10188.88 ± 3858.95	0.742
		Education: High school diploma	7677.85 ± 3540.7	

Table 8: This table shows the AUC of SDP (mean +/- 95% confidence intervals) for different demographic conditions. This table highlights that SDP can be effective irrespective of demographic background (gender, age, and educational background).

Tables 7-8 highlight that there is no significant difference in the performance of SDP when human teachers have differing backgrounds. This includes a background in CS, educational level, age, and gender. We note that we did not perform statistical analysis comparing racial groups as some experiments had only one participant of a particular racial background, making comparisons less meaningful.

D Additional Simulated Teacher Results

For simplicity, in all additional experiments in this section, we only compare SDP + PEBBLE with PEBBLE (or R-PEBBLE).

D.1 Study of Agent Update Phase

Figure 8 emphasizes how the agent update phase does result in new transitions, therefore the teacher provides feedback to transitions that are different from the original sub-optimal transitions used for pre-training.

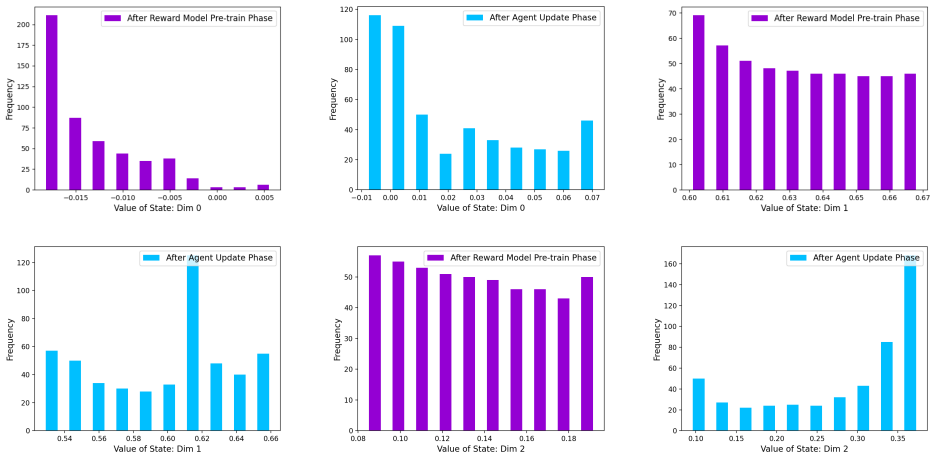


Figure 8: Door open, preference learning exp. These plots show how the agent's policy has changed from the reward model pre-training phase (purple histograms) to the agent update phase (blue histograms), thereby resulting in a different distribution for the state features.

D.2 True Reward Values of Random Policy

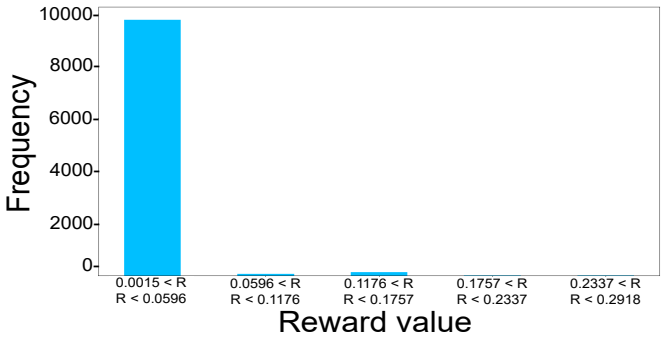
Figure 9 shows the true reward values for sub-optimal data gathered through a random policy in the DMControl suite. This emphasizes that the true reward value is close to the value we pseudo-label the sub-optimal transitions with (i.e., zero), therefore SDP should not yield a large incorrect reward bias.

D.3 Zero Weight Study

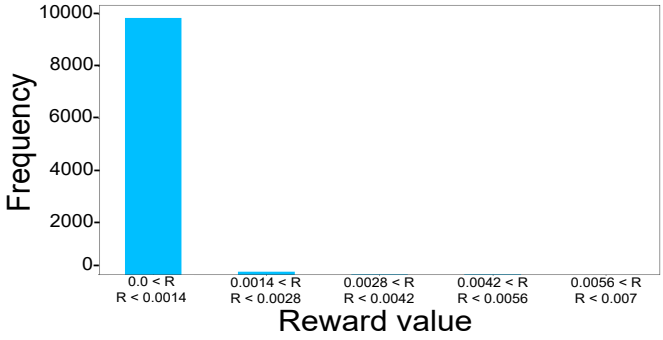
To understand the effect of each component of SDP, we performed two additional ablations. First, in the reward model pre-training phase, the goal is for the reward model to learn to output zero. However, a trivial means to achieve an output of zero is to set all weights and biases in the neural network to zero. Therefore, we compare the full SDP to SDP using a zero-weight initialization as a replacement for the reward model pre-training. We found that using a zero-weight initialization for the reward model in place of the pre-training phase results in significantly degraded performance (see the green curve in Figure 10–right). This is not surprising, as previous works have found that a zero-weight initialization can negatively affect the training of neural networks (Blumenfeld et al., 2020; Zhao et al., 2022). In addition, Figure 11 demonstrates that the reward model pre-training phase does not produce reward model weights of zero, emphasizing why we do not experience the same performance degradation that occurs if we use a zero-weight initialization.

D.4 Fake Input Study

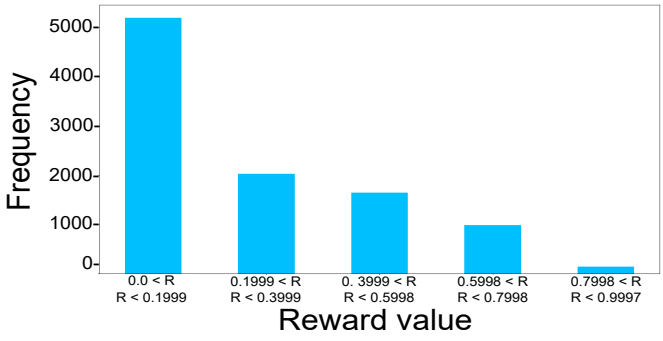
Second, SDP makes use of state, action transitions that are gathered through a sub-optimal policy. Therefore, these are transitions that an agent experiences while interacting with its environment. However, as previously noted, the goal of the reward model pre-training phase is for the reward model to learn to output zero. Therefore, is it necessary for the reward model to pre-train on inputs that are real environment transitions? Instead, can we pre-train the reward model on transitions that did not result from an agent-environment interaction? To test this, we created "fake" inputs of size dim(state) + dim(action), and for each input dimension, we randomly sampled a value from $\mathcal{N}(0,\ 1)$. We obtained 50,000 "fake" transitions and used this data for the reward model pre-training phase. In this experiment, our goal is to understand the effect of the type of inputs on the reward model pre-training phase, therefore we kept the agent update phase as is (i.e., provided the true sub-optimal data for this phase). We then compared SDP using true transitions to SDP using "fake" transitions in Figures 10–left and 12. We found that the full SDP (purple curve) in which we pre-train the reward model on true sub-optimal transitions yields significantly greater final performance than SDP using



(a) Walker-walk.



(b) Cheetah-run.



(c) Quadruped-walk.

Figure 9: Distribution of true reward values for transitions obtained with a random policy

"fake" transitions (green curve). This highlights the importance of using true sub-optimal transitions in SDP.

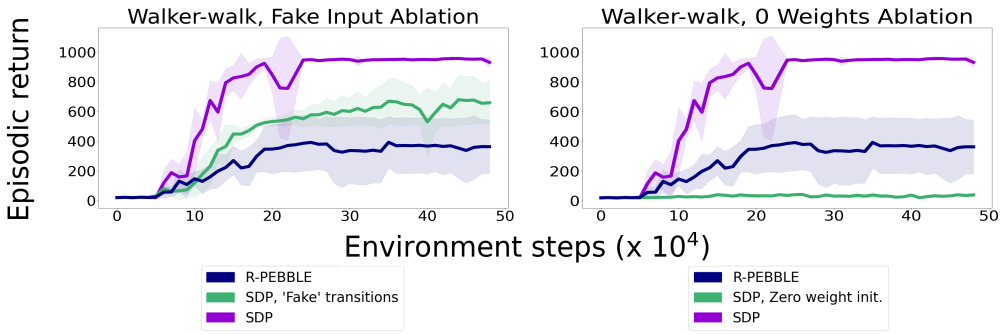
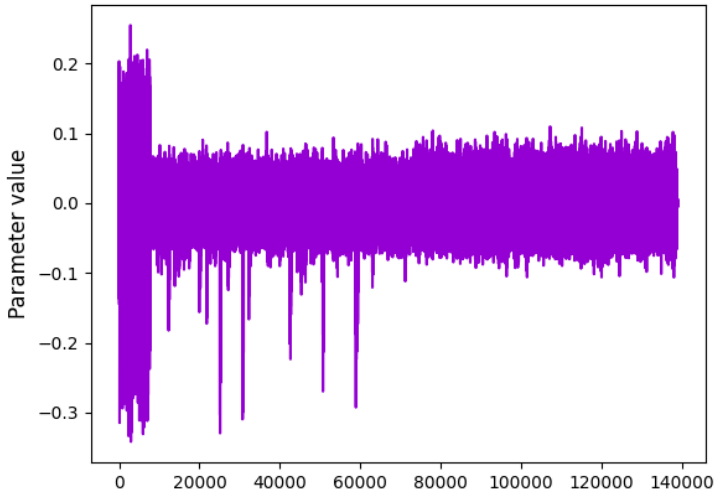


Figure 10: Zero weights and fake input studies in Walker-walk



(a) Scalar feedback experiment in Walker-walk.

Figure 11: This figure shows the reward model weights of SDP after the reward model pre-training phase. This demonstrates that the reward model pre-train phase of SDP does not result in zero neural network weights.

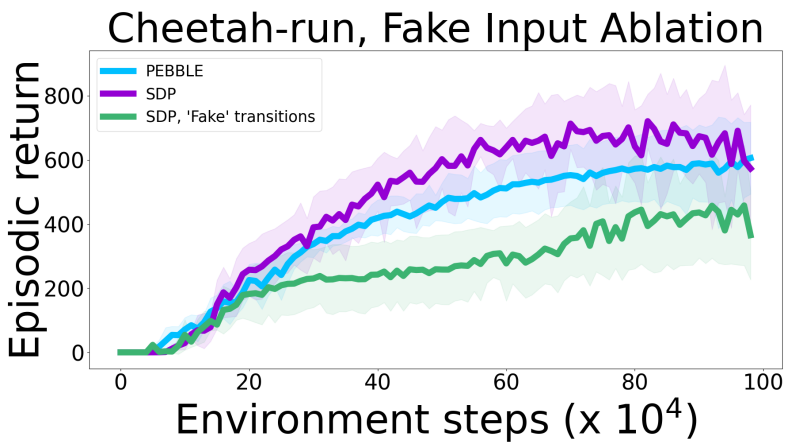


Figure 12: Fake Input Study in Cheetah Run Preference Learning

D.5 SDP Component Studies

Figure 13 shows additional phase ablations in the preference learning experiments done on the Cheetah-run environment. This highlights the importance of using both phases in SDP. In addition, we ran an experiment in Walker-walk where we directly apply the procedure Yu et al. (2022) uses for leveraging sub-optimal data in the offline RL setting. They simply store the sub-optimal transitions in the RL agent's replay buffer with the pseudo reward label of 0 and follow standard offline RL. We repeated this with the other difference of following standard preference learning (PEBBLE) afterwards. We found that the average final performance was 104.32 with a 95% confidence interval of 46.12, which is close to 4 times less than the final performance of SDP.

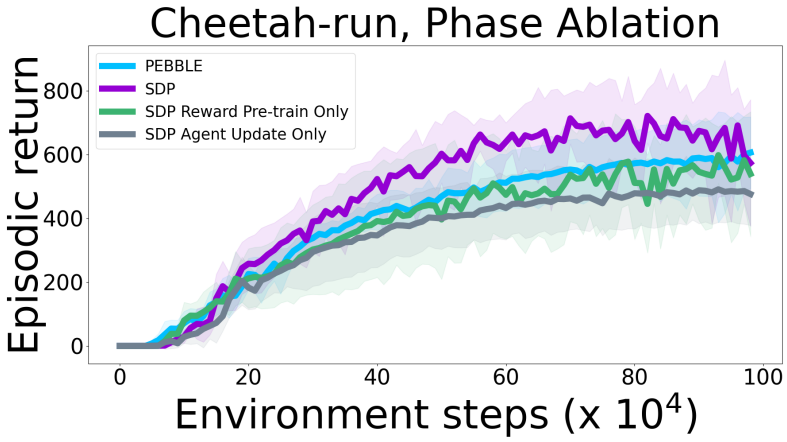


Figure 13: Phase Ablation Study in Cheetah Run Preference Learning

Figure 14 shows an additional ablation over the amount of prior data used in SDP. We observed similar performance gains as described in section 5.4.

D.6 Effect of Relabelling Replay Buffer

SDP is combined with four preference learning algorithms, PEBBLE, RUNE, SURF, and MRN. A core feature of these algorithms is that every time the reward model is updated, all transitions inside the RL agent's replay buffer are updated using the latest reward model. Figure 15 shows the effects of not relabeling the sub-optimal data (i.e., the reward labels) with the latest learned reward model. This means the reward labels remain frozen at zero throughout the entire training process.

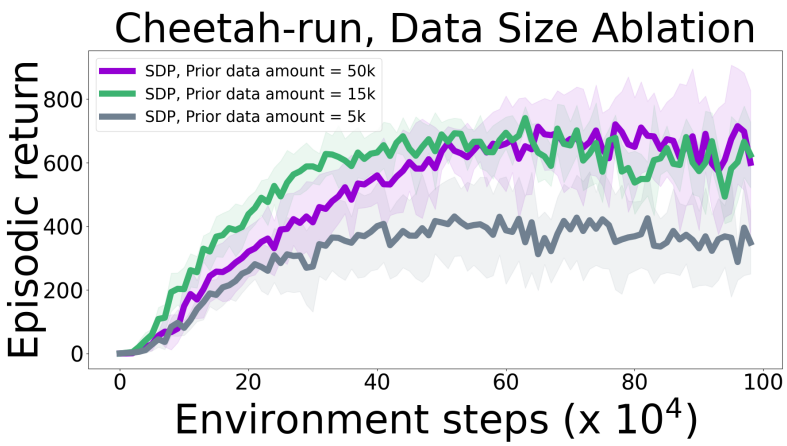


Figure 14: Prior Data Amount Study in Cheetah Run Preference Learning

This ablation was to demonstrate that if the sub-optimal data in the agent's replay buffer is not updated with the latest reward model, then performance will suffer. This is likely the case because the incorrect reward bias from the pseudo-labeling process persists, whereas when we relabel the transitions with the updated reward model, the incorrect reward bias may reduce over time.

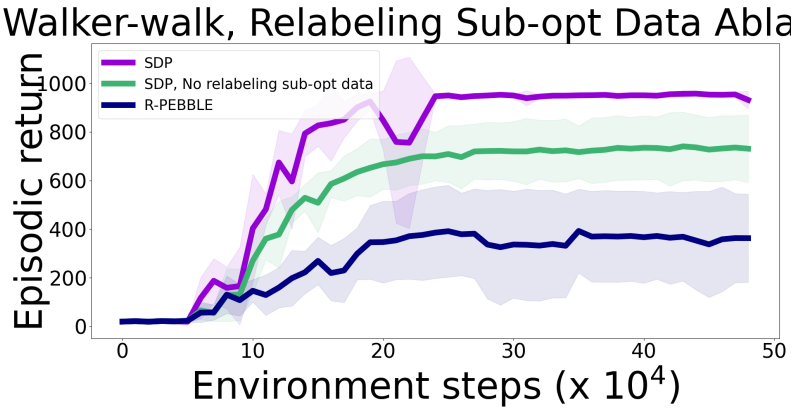


Figure 15: Relabeling sub-opt data study: Walker-walk, scalar feedback

D.7 Effect of Transition Quality in SDP

Moreover, we show that the effectiveness of SDP relies on the use of sub-optimal data transitions. If we use high-quality data transitions (i.e., transitions that came from a fully trained RL agent policy), SDP will fail (see Figure 16). This unsurprising result confirms that pseudo-labeling high-reward transitions with zero can significantly hurt the reward model and the agent's performance

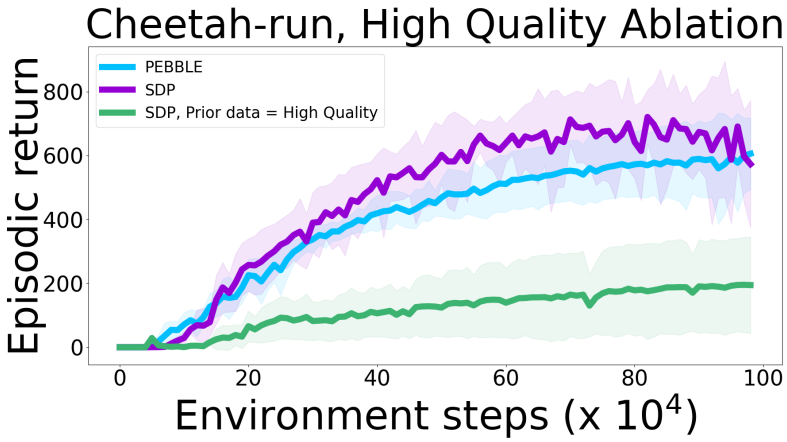


Figure 16: Using high-quality data in SDP study: Cheetah-run, preference feedback

D.8 SCALAR-BASED EXPERIMENT STATISTICS

Tables 9 and 10 provide a summary of the mean final performance and the mean area under the curve for all environments and benchmarks in the scalar feedback setting

Learning from scalar feedback			
	SDP	R-PEBBLE	Deep TAMER
Walker-walk	931.31 ± 36.59	362.99 ± 181.23*	33.10 ± 11.10*
Cheetah-run	862.72 ± 49.13	522.10 ± 238.87*	30.71 ± 16.26*
Quadruped-walk	777.38 ± 156.74	543.92 ± 193.21*	73.74 ± 48.38*

Table 9: This table shows the mean final performance plus and minus the 95% confidence interval. * indicates SDP achieves a significantly greater mean final performance.

Learning from scalar feedback			
	SDP	R-PEBBLE	Deep TAMER
Walker-walk	34981.65 ± 1559.24	13147.1 ± 6102.31*	1823.42 ± 578.48*
Cheetah-run	55144.89 ± 4266.94	33557.58 ± 15185.47*	2459.23 ± 1226.34*
Quadruped-walk	58370.85 ± 10759.	37076.52 ± 11968.48*	7365.68 ± 3556.27*

Table 10: This table shows the mean area under the learning curve (AUC) plus and minus the 95% confidence interval. * indicates SDP achieves a significantly greater AUC.

D.9 Preference Learning Experiment Statistics

Tables 11-14 provide a summary of the mean final performance and the mean area under the curve for all environments and benchmarks in the preference feedback setting

Task	Feedback	Method	Final Return	P Value
Cartpole-swingup	48	PEBBLE	226.4 ± 136.35	0.007*
		SDP + PEBBLE	533.94 ± 57.71	
		RUNE	263.48 ± 176.2	
		SDP + RUNE	561.77 ± 108.17	0.021*
		SURF	83.12 ± 44.53	
		SDP + SURF	324.19 ± 179.8	0.039*
		MRN	151.05 ± 76.31	
		SDP + MRN	644.03 ± 57.99	0.0*
Walker-walk	100	PEBBLE	173.9 ± 100.26	0.001*
		SDP + PEBBLE	490.07 ± 50.07	
		RUNE	160.6 ± 61.85	0.0*
		SDP + RUNE	575.71 ± 104.52	
		SURF	202.1 ± 95.17	0.005*
		SDP + SURF	435.99 ± 39.45	
		MRN	215.14 ± 58.09	0.0*
		SDP + MRN	672.3 ± 85.53	
Cheetah-run	200	PEBBLE	582.51 ± 67.62	0.062
		SDP + PEBBLE	704.63 ± 102.27	
		RUNE	681.59 ± 68.05	0.645
		SDP + RUNE	616.78 ± 279.34	
		SURF	642.52 ± 84.3	0.266
		SDP + SURF	682.34 ± 65.08	
		MRN	728.36 ± 38.69	0.305
		SDP + MRN	748.99 ± 55.59	
Quadruped-walk	500	PEBBLE	350.24 ± 228.91	0.017*
		SDP + PEBBLE	753.18 ± 104.79	
		RUNE	363.65 ± 163.93	0.009*
		SDP + RUNE	704.1 ± 102.16	
		SURF	550.51 ± 211.19	0.089
		SDP + SURF	767.78 ± 140.67	
		MRN	200.12 ± 63.04	0.0*
		SDP + MRN	733.83 ± 62.87	

Table 11: This table shows the final performance (mean +/- 95% confidence intervals) for all DM-Control preference learning experiments. * indicates significant differences.

Task	Feedback	Method	AUC	P Value
Cartpole-swingup	48	PEBBLE	8248.14 ± 2148.59	0.0*
		SDP + PEBBLE	21519.9 ± 2435.02	
		RUNE	8817.21 ± 3534.57	0.004*
		SDP + RUNE	20970.54 ± 4643.96	
		SURF	5593.44 ± 1099.56	0.074
		SDP + SURF	10288.22 ± 4584.16	
		MRN	6774.95 ± 1783.95	0.0*
		SDP + MRN	22564.32 ± 3846.6	
Walker-walk	100	PEBBLE	7571.79 ± 2903.09	0.001*
		SDP + PEBBLE	18458.81 ± 1224.69	
		RUNE	6525.52 ± 2566.88	0.0*
		SDP + RUNE	21059.76 ± 3056.28	
		SURF	7675.24 ± 2857.99	0.001*
		SDP + SURF	16700.51 ± 1449.62	
		MRN	7711.05 ± 1295.46	0.0*
		SDP + MRN	25184.26 ± 3071.65	
Cheetah-run	200	PEBBLE	39245.94 ± 5189.16	0.006*
		SDP + PEBBLE	51121.68 ± 3488.87	
		RUNE	45824.63 ± 4459.04	0.792
		SDP + RUNE	37202.31 ± 16330.83	
		SURF	40522.18 ± 2989.22	0.218
		SDP + SURF	42777.1 ± 3775.07	
		MRN	47569.45 ± 2191.57	0.106
		SDP + MRN	49968.24 ± 2187.97	
Quadruped-walk	500	PEBBLE	20584.95 ± 9814.49	0.018*
		SDP + PEBBLE	42044.34 ± 11183.84	
		RUNE	22861.89 ± 9861.31	0.082
		SDP + RUNE	32664.79 ± 3929.32	
		SURF	30884.8 ± 11660.52	0.074
		SDP + SURF	44176.44 ± 8404.76	
		MRN	14805.07 ± 1860.39	0.002*
		SDP + MRN	35209.32 ± 6225.09	

Table 12: This table shows the AUC (mean +/- 95% confidence intervals) for all DMControl preference learning experiments. * indicates significant differences.

Task	Feedback	Method	Final Return	P Value
Hammer	7500	PEBBLE	10.0 ± 13.58	0.003*
		SDP + PEBBLE	66.0 ± 21.18	
		RUNE	4.0 ± 7.01	0.001*
		SDP + RUNE	68.0 ± 17.0	
		SURF	0.0 ± 0.0	0.033*
		SDP + SURF	36.0 ± 25.16	
		MRN	4.0 ± 7.01	0.027*
		SDP + MRN	46.0 ± 27.5	
Door-unlock	500	PEBBLE	42.0 ± 34.35	0.066
		SDP + PEBBLE	80.0 ± 15.68	
		RUNE	26.0 ± 22.59	0.354
		SDP + RUNE	36.0 ± 38.65	
		SURF	20.0 ± 35.06	0.02*
		SDP + SURF	80.0 ± 22.17	
		MRN	48.0 ± 38.17	0.399
		SDP + MRN	56.0 ± 37.02	
Door-lock	500	PEBBLE	38.0 ± 30.06	0.035*
		SDP + PEBBLE	80.0 ± 7.84	
		RUNE	62.0 ± 32.51	0.793
		SDP + RUNE	40.0 ± 30.87	
		SURF	70.0 ± 31.85	0.684
		SDP + SURF	60.0 ± 13.58	
		MRN	66.0 ± 26.93	0.268
		SDP + MRN	78.0 ± 17.88	
Drawer-open	1000	PEBBLE	4.0 ± 7.01	0.045*
		SDP + PEBBLE	36.0 ± 25.16	
		RUNE	0.0 ± 0.0	0.001*
		SDP + RUNE	66.0 ± 14.24	
		SURF	20.0 ± 35.06	0.018*
		SDP + SURF	80.0 ± 14.67	
		MRN	20.0 ± 35.06	0.136
		SDP + MRN	56.0 ± 40.21	
Window-open	200	PEBBLE	34.0 ± 37.44	0.234
		SDP + PEBBLE	54.0 ± 26.35	
		RUNE	12.0 ± 21.04	0.098
		SDP + RUNE	38.0 ± 24.42	
		SURF	14.0 ± 20.44	0.014*
		SDP + SURF	66.0 ± 26.35	
		MRN	46.0 ± 32.61	0.208
		SDP + MRN	68.0 ± 31.06	

Table 13: This table shows the final performance (mean +/- 95% confidence intervals) for all Meta-world preference learning experiments. * indicates significant differences.

Task	Feedback	Method	AUC	P Value
Hammer	7500	PEBBLE	424.0 ± 175.18	0.017*
		SDP + PEBBLE	2222.0 ± 1009.47	
		RUNE	490.0 ± 300.85	0.001*
		SDP + RUNE	2520.0 ± 627.47	
		SURF	482.0 ± 215.59	0.006*
		SDP + SURF	3042.0 ± 1073.54	
		MRN	494.0 ± 311.76	0.016*
		SDP + MRN	2494.0 ± 1108.52	
Door-unlock	500	PEBBLE	2142.0 ± 1738.06	0.031*
		SDP + PEBBLE	4838.0 ± 1251.13	
		RUNE	1272.0 ± 1275.71	0.348
		SDP + RUNE	1836.0 ± 2063.68	
		SURF	1580.0 ± 2462.83	0.047*
		SDP + SURF	4802.0 ± 1537.81	
		MRN	3082.0 ± 2232.43	0.342
		SDP + MRN	3842.0 ± 2230.38	
Door-lock	500	PEBBLE	876.0 ± 647.87	0.008*
		SDP + PEBBLE	2300.0 ± 135.0	
		RUNE	1342.0 ± 590.05	0.809
		SDP + RUNE	884.0 ± 633.87	
		SURF	1892.0 ± 842.85	0.429
		SDP + SURF	1986.0 ± 184.83	
		MRN	1966.0 ± 751.51	0.306
		SDP + MRN	2236.0 ± 479.9	
Drawer-open	1000	PEBBLE	314.0 ± 533.02	0.117
		SDP + PEBBLE	1702.0 ± 1706.12	
		RUNE	38.0 ± 49.71	0.004*
		SDP + RUNE	3794.0 ± 1306.6	
		SURF	696.0 ± 1167.88	0.015*
		SDP + SURF	4078.0 ± 1839.07	
		MRN	1550.0 ± 2690.98	0.211
		SDP + MRN	3206.0 ± 2114.16	
Window-open	200	PEBBLE	794.0 ± 1015.04	0.131
		SDP + PEBBLE	1698.0 ± 822.8	
		RUNE	230.0 ± 334.14	0.064
		SDP + RUNE	948.0 ± 631.92	
		SURF	242.0 ± 248.44	0.01*
		SDP + SURF	2048.0 ± 867.93	
		MRN	1252.0 ± 796.8	0.223
		SDP + MRN	1728.0 ± 666.48	

Table 14: This table shows the area under the curve (mean +/- 95% confidence intervals) for all Metaworld preference learning experiments. * indicates significant differences.