RIME: Robust Preference-based Reinforcement Learning with Noisy Preferences

Jie Cheng [1 2] **Gang Xiong** [1 2] **Xingyuan Dai** [1 2] **Qinghai Miao** [2] **Yisheng Lv** [1 2] **Fei-Yue Wang** [1 2]

Abstract

Preference-based Reinforcement Learning (PbRL) circumvents the need for reward engineering by harnessing human preferences as the reward signal. However, current PbRL methods excessively depend on high-quality feedback from domain experts, which results in a lack of robustness. In this paper, we present RIME, a robust PbRL algorithm for effective reward learning from noisy preferences. Our method utilizes a sample selection-based discriminator to dynamically filter out noise and ensure robust training. To counteract the cumulative error stemming from incorrect selection, we suggest a warm start for the reward model, which additionally bridges the performance gap during the transition from pre-training to online training in PbRL. Our experiments on robotic manipulation and locomotion tasks demonstrate that RIME significantly enhances the robustness of the state-of-the-art PbRL method. Code is available at `https://github.com/CJReinforce/RIME_ICML2024`.

1. Introduction

Reinforcement Learning (RL) has demonstrated remarkable performance in various domains, including gameplay (Perolat et al., 2022; Kaufmann et al., 2023), robotics (Chen et al., 2022), autonomous systems (Bellemare et al., 2020; Zhou et al., 2020), multimodal (Yue et al., 2024), etc. The success of RL frequently relies on the meticulous crafting of reward functions, a process that can be both time-consuming and susceptible to errors. In this context, Preference-Based RL (PbRL) (Akrour et al., 2011; Cheng et al., 2011; Christiano et al., 2017) emerges as a valuable alternative, eliminating the requirement for manually designed reward functions. PbRL adopts a human-in-the-loop paradigm, where human teachers provide preferences over distinct agent behaviors as the reward signal.

Nevertheless, existing works in PbRL have primarily focused on enhancing feedback-efficiency, aiming to maximize the expected return with few feedback queries. This focus induces a substantial reliance on high-quality feedback, typically assuming expertise on the teachers (Liu et al., 2022; Kim et al., 2022). However, humans are prone to errors (Christiano et al., 2017). In broader applications, feedback is often sourced from non-expert users or crowd-sourcing platforms, where the quality can be inconsistent and noisy. Further complicating the matter, Lee et al. (2021a) showed that even a mere 10% corruption rate in preference labels can dramatically degrade the performance. Therefore, the lack of robustness to noisy preference labels hinders the wide applicability of PbRL.

Meanwhile, learning from noisy labels, also known as robust training, is a rising concern in deep learning, since such labels severely degrade the generalization performance of deep neural networks. Song et al. (2022) classifies robust training methods into four key categories: robust architecture (Cheng et al., 2020), robust regularization (Xia et al., 2020), robust loss design (Lyu & Tsang, 2019), and sample selection (Li et al., 2020; Song et al., 2021). However, it is challenging to effectively incorporate these advanced methods for robust training in PbRL. This complexity arises from the pursuit of feedback-efficiency and cost reduction, necessitating access to a limited amount of feedback. Simultaneously, the distribution shift problem during RL training undermines the assumption of i.i.d input data, a core principle that supports robust training methods in deep learning.

In this work, we aim to improve the robustness of preference-based RL methods on noisy and quantitatively limited preferences. To this end, we present RIME: **R**obust preference-based re**I**nforcement learning via war**M**-start d**E**noising discriminator. RIME modifies the training paradigm of the reward model in the widely-adopted two-phase (*i.e.,* pre-training and online training phases) pipeline of PbRL. Figure 1 shows an overview of RIME. In particular, to empower robust learning from noisy preferences, we introduce a denoising discriminator. It utilizes dynamic lower and predefined upper bounds on the Kullback–Leibler (KL) di-

[1]State Key Laboratory of Multimodal Artificial Intelligence Systems, CASIA [2]School of Artificial Intelligence, the University of Chinese Academy of Sciences. Correspondence to: Yisheng Lv <yisheng.lv@ia.ac.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

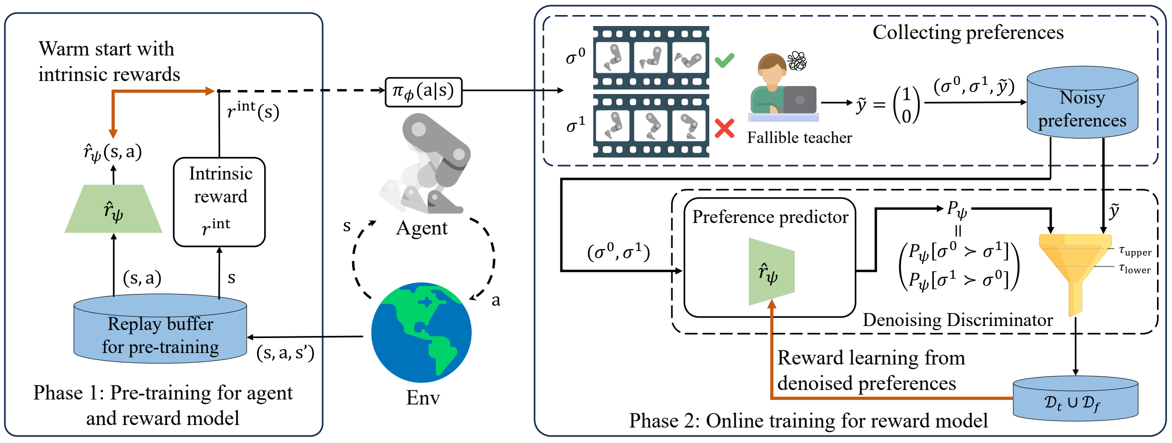

Warm start with intrinsic rewards
$r^{\text{int}}(s)$
$\hat{r}_\psi(s,a)$
$\hat{r}_\psi$
Intrinsic reward r^{int}
(s, a)
s
Replay buffer for pre-training
(s, a, s')
Phase 1: Pre-training for agent and reward model
$\pi_\phi(a|s)$
Agent
s
a
Env
Collecting preferences
σ^0
σ^1
Fallible teacher
$\tilde{y} = \binom{1}{0}$
$(\sigma^0, \sigma^1, \tilde{y})$
Noisy preferences
(σ^0, σ^1)
Preference predictor
$\hat{r}_\psi$
P_ψ
$\begin{pmatrix} P_\psi[\sigma^0 \succ \sigma^1] \\ P_\psi[\sigma^1 \succ \sigma^0] \end{pmatrix}$
$\tilde{y}$
τ_{upper}
τ_{lower}
Denoising Discriminator
Reward learning from denoised preferences
$\mathcal{D}_t \cup \mathcal{D}_f$
Phase 2: Online training for reward model

Figure 1. Overview of RIME. In the pre-training phase, we warm start the reward model $\hat{r}_\psi$ with intrinsic rewards r^{int} to facilitate a smooth transition to the online training phase. Post pre-training, the policy, Q-network, and reward model $\hat{r}_\psi$ are all inherited as initial configurations for online training. During online training, we utilize a denoising discriminator to screen denoised preferences for robust reward learning. This discriminator employs a dynamic lower bound τ_{lower} on the KL divergence between predicted preferences P_ψ and annotated preference labels $\tilde{y}$ to filter trustworthy samples $\mathcal{D}_t$, and an upper bound τ_{upper} to flip highly unreliable labels $\mathcal{D}_f$.

vergence between predicted and annotated preference labels to filter samples. Further, to mitigate the accumulated error caused by incorrect filtration, we propose to warm start the reward model during the pre-training phase for a good initialization. Moreover, we find that the warm start also bridges the performance gap that occurs during the transition from pre-training to online training. Our experimental results indicate that RIME significantly outperforms existing baselines under noisy preference conditions, thereby substantially enhancing robustness for PbRL.

In summary, our work has three main contributions. First, we present RIME, a robust reward learning algorithm for PbRL, designed to effectively train reward models from noisy feedback—an important and realistic topic that has not been studied extensively. Second, we observe a dramatic performance gap during the transition from pre-training to online training in PbRL and propose to warm start the reward model for a seamless transition, which proves to be crucial for both robustness and feedback-efficiency in limited-feedback cases. Last, we show that RIME outperforms existing PbRL baselines under noisy feedback settings, across a diverse set of robotic manipulation tasks from Meta-World (Yu et al., 2020) and locomotion tasks from the DeepMind Control Suite (Tassa et al., 2018; 2020), and further is more suitable for non-expert humans.

2. Related work

Preference-based Reinforcement Learning. Incorporating human feedback into the training of reward models has proven effective in various domains, including natural language processing (Ouyang et al., 2022), multimodal (Lee et al., 2023), and reinforcement learning (Christiano et al., 2017; Ibarz et al., 2018; Hejna III & Sadigh, 2023). In the context of RL, Christiano et al. (2017) proposed a comprehensive framework for PbRL. To improve feedback-efficiency, PEBBLE (Lee et al., 2021b) used unsupervised exploration for policy pre-training. SURF (Park et al., 2021) employed data augmentation and semi-supervised learning to enrich the preference dataset. RUNE (Liang et al., 2021) encouraged exploration by modulating reward uncertainty. MRN (Liu et al., 2022) introduced a bi-level optimization to optimize the Q-function's performance. PT (Kim et al., 2022) utilized Transformer architecture to model non-Markovian rewards, proving effective in complex tasks.

Despite these advancements, the focus on feedback efficiency should not overshadow the equally critical issue of robustness in PbRL. Lee et al. (2021a) indicated that a mere 10% rate of corrupted preferences can significantly impair algorithmic performance. Furthermore, in more extensive application contexts, the collection of preferences from non-experts increases the likelihood of incorporating incorrect labels. Therefore, enhancing the robustness of PbRL remains a vital research direction. In this work, we address robust PbRL via a warm-start denoising discriminator, which dynamically filters denoised preferences and is more adaptable to cases of distribution shift during RL training.

Learning from Noisy Labels. Learning from noisy labels has gained more attention in supervised learning, particularly in light of the wide presence of noisy or imprecise labels in real-world applications. A variety of approaches have been proposed for robust training (Song et al., 2022), including architectural modifications (Goldberger & Ben-Reuven, 2016), regularization (Lukasik et al., 2020), loss function designs (Zhang & Sabuncu, 2018), and sample selection methods (Wang et al., 2021). Despite these advancements, the direct application of these methods to reward learning in PbRL has presented challenges, mainly due to the limited sample sizes and the absence of i.i.d. of samples. In the context of PbRL, Xue et al. (2023) proposed an encoder-decoder architecture to model diverse human preferences, which required approximately 100 times the amount of preference labels used in our experiments. Our approach can be situated within the sample selection category and improves robustness while preserving feedback-efficiency.

Policy-to-Value Reincarnating RL. Policy-to-value reincarnating RL (PVRL) transfers a sub-optimal teacher policy to a value-based RL student agent (Agarwal et al., 2022). Uchendu et al. (2023) found that a randomly initialized Q-network in PVRL leads to the teacher policy being forgotten quickly. Within the widely-adopted pipeline of PbRL, the challenge intrinsic to PVRL also arises during the transition from pre-training to online training, but has been neglected in previous research (Lee et al., 2021b; Park et al., 2021; Liang et al., 2021; Liu et al., 2022). The issue of forgetting the pre-training policy becomes more critical under noisy feedback conditions, as detailed in Section 4.2. Based on the observation, we propose to warm start the reward model for a seamless transition. Our ablation study demonstrates that the warm start is crucial for both robustness and feedback-efficiency.

3. Preliminaries

Preference-based Reinforcement Learning. In standard RL, an agent interacts with an environment in discrete time steps (Sutton & Barto, 2018). At each time step t, the agent observes the current state $\mathbf{s}_t$ and selects an action $\mathbf{a}_t$ according to its policy $\pi(\cdot|\mathbf{s}_t)$. The environment responds by emitting a reward $r(\mathbf{s}_t, \mathbf{a}_t)$ and transitioning to the next state $\mathbf{s}_{t+1}$. The agent's objective is to learn a policy that maximizes the expected return, $\mathcal{R}_0 = \sum_{t=0}^{\infty} \gamma^t r_t$, which is defined as a discounted cumulative sum of the reward with the discount factor γ.

In Preference-based RL, there is no predefined reward function. Instead, a teacher offers preferences between the agent's behaviors, and an estimated reward function $\hat{r}_\psi$ is trained to align with collected preferences. Following previous works (Lee et al., 2021b; Liu et al., 2022; Kim et al., 2022), we consider preferences over two trajectory segments of length H, where segment $\sigma = \{(\mathbf{s}_1, \mathbf{a}_1), ..., (\mathbf{s}_H, \mathbf{a}_H)\}$. Given a pair of segments (σ^0, σ^1), a teacher provides a preference label $\tilde{y}$ from the set $\{(1,0), (0,1), (0.5, 0.5)\}$. The label $\tilde{y} = (1,0)$ signifies $\sigma^0 \succ \sigma^1$, $\tilde{y} = (0,1)$ signifies $\sigma^1 \succ \sigma^0$, and $\tilde{y} = (0.5, 0.5)$ represents an equally preferable case, where $\sigma^i \succ \sigma^j$ denotes that segment i is preferred over segment j. Each feedback is stored in a dataset $\mathcal{D}$ as a triple $(\sigma^0, \sigma^1, \tilde{y})$. Following the Bradley-Terry model (Bradley & Terry, 1952), the preference predicted by the estimated reward function $\hat{r}_\psi$ is formulated as:

$$P_\psi[\sigma^i \succ \sigma^j] = \frac{\exp\left(\sum_t \hat{r}_\psi(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)}{\sum_{k=i,j} \exp\left(\sum_t \hat{r}_\psi(\mathbf{s}_t^k, \mathbf{a}_t^k)\right)} \tag{1}$$

The estimated reward function $\hat{r}_\psi$ is updated by minimizing the cross-entropy loss between the predicted preferences P_ψ and the annotated labels $\tilde{y}$:

$$\mathcal{L}^{\text{CE}}(\psi) = \mathbb{E}\left[\mathcal{L}^{\texttt{Reward}}\right] = -\,\mathbb{E}\Big[\tilde{y}(0)\ln P_\psi[\sigma^0 \succ \sigma^1] + \tilde{y}(1)\ln P_\psi[\sigma^1 \succ \sigma^0]\Big] \tag{2}$$

The policy π can subsequently be updated using any RL algorithm to maximize the expected return with respect to the estimated reward function $\hat{r}_\psi$.

Unsupervised Pre-training in PbRL. Pre-training the agent is important in PbRL because the initial random policy often results in uninstructive preference queries, requiring many queries for even elementary learning progress. Recent studies addressed this issue through unsupervised exploration for policy pre-training (Lee et al., 2021b). Specifically, agents are encouraged to traverse a more expansive state space by using an intrinsic reward derived from particle-based state entropy (Singh et al., 2003). Formally, the intrinsic reward is defined as (Liu & Abbeel, 2021):

$$r^{\text{int}}(\mathbf{s}_t) = \log(\|\mathbf{s}_t - \mathbf{s}_t^k\|) \tag{3}$$

where $\mathbf{s}_t^k$ is the k-th nearest neighbor of $\mathbf{s}_t$. This reward motivates the agent to explore a broader diversity of states. This exploration, in turn, leads to a varied set of agent behaviors, facilitating more informative preference queries.

Noisy Preferences in PbRL. We denote the annotated preference labels as $\tilde{y}$ and the ground-truth preference labels, typically sourced from expert human teachers or scripted teachers, as y. To simulate the noise in human annotations, Lee et al. (2021a) introduced four noisy 0-1 labeling models: Equal, Skip, Myopic, and Mistake. The "Mistake" model, in particular, proved to be significantly detrimental to performance across various environments. It hypothesizes that the preference dataset is contaminated with corrupted preferences whose annotated labels $\tilde{y}$ are $1-y$. Drawing on previous insights, our work starts from addressing robust reward learning under the "Mistake" model settings. This approach is guided by empirical evidence suggesting that solutions developed to overcome complex challenges could be efficiently adapted to simpler cases.

4. RIME

In this section, we formally introduce RIME: **R**obust preference-based re**I**nforcement learning via war**M**-start d**E**noising discriminator. RIME consists of two main components: 1) a denoising discriminator designed to filter out corrupted preferences while accounting for training instability and distribution shifts, and 2) a warm start method to effectively initialize the reward model and enable a seamless transition from pre-training to online training. See Figure 1 for the overview of RIME. The full procedure of RIME is detailed in Appendix A.

4.1. Denoising Discriminator

In the presence of noisy labels, it is well-motivated to distinguish between clean and corrupted samples for robust training. Existing research indicates that deep neural networks first learn generalizable patterns before overfitting to the noise in the data (Arpit et al., 2017; Li et al., 2020). Therefore, prioritizing samples associated with smaller losses as clean ones is a well-founded approach to improve robustness. Inspired by this, a theoretical lower bound on the KL divergence between the predicted preference P_ψ and the annotated preference $\tilde{y}$ for corrupted samples could be established to filter out large-loss corrupted samples.

Theorem 4.1 (KL Divergence Lower Bound for Corrupted Samples)**.** *Consider a preference dataset $\{(\sigma_i^0, \sigma_i^1, \tilde{y}_i)\}_{i=1}^n$, where $\tilde{y}_i$ is the annotated label for the segment pair (σ_i^0, σ_i^1) with the ground truth label y_i. Let x_i denote the tuple (σ_i^0, σ_i^1). Assume the cross-entropy loss $\mathcal{L}^{\texttt{CE}}$ for clean data (whose $\tilde{y}_i = y_i$) is bounded by ρ. Then, the KL divergence between the predicted preference $P_\psi(x)$ and the annotated label $\tilde{y}(x)$ for a corrupted sample x is lower-bounded as:*

$$D_{\mathrm{KL}}\left(\tilde{y}(x) \| P_\psi(x)\right) \geq -\ln \rho + \frac{\rho}{2} + \mathcal{O}(\rho^2) \tag{4}$$

The proof of Theorem 4.1 is presented in Appendix C. Based on Theorem 4.1, the lower bound on KL divergence threshold could be formulated to filter out untrustworthy samples as $\tau_{\text{base}} = -\ln \rho + \alpha\rho$ in practice, where ρ denotes the maximum cross-entropy loss on trustworthy samples observed during the last update, and α is a tunable hyperparameter with a theoretically-determined value range in $(0, 0.5]$.

However, compared to deep learning, the shifting state distribution makes the robust training problem in RL more complicated. To add tolerance for clean samples in cases of distribution shift, we introduce an auxiliary term characterizing the uncertainty for filtration, defined as $\tau_{\text{unc}} = \beta_t \cdot s_{\text{KL}}$, where β_t is a time-dependent parameter, and s_{KL} is the standard deviation of the KL divergence. Our intuition is that the inclusion of out-of-distribution data for training is likely to induce fluctuations in training loss. Therefore, the complete threshold equation is formulated as:

$$\tau_{\text{lower}} = \tau_{\text{base}} + \tau_{\text{unc}} = -\ln \rho + \alpha\rho + \beta_t \cdot s_{\text{KL}} \tag{5}$$

We utilize a linear decay schedule for β_t to initially allow greater tolerance for samples while becoming increasingly conservative over time, *i.e.*,$\beta_t = \max(\beta_{\min}, \beta_{\max} - kt)$. At each training step for the reward model, we apply the threshold in Equation (5) to identify trustworthy sample dataset $\mathcal{D}_t$, as described below:

$$\mathcal{D}_t = \{(\sigma^0, \sigma^1, \tilde{y}) \mid D_{\mathrm{KL}}(\tilde{y} \| P_\psi(\sigma^0, \sigma^1)) < \tau_{\text{lower}}\} \tag{6}$$

To ensure efficient usage of samples, we introduce a label-flipping method for the reintegration of untrustworthy samples. Specifically, we pre-define an upper bound τ_{upper} and reverse the labels for samples exceeding this threshold:

$$\mathcal{D}_f = \{(\sigma^0, \sigma^1, 1 - \tilde{y}) \mid D_{\mathrm{KL}}(\tilde{y} \| P_\psi(\sigma^0, \sigma^1)) > \tau_{\text{upper}}\} \tag{7}$$

Beyond improving sample utilization, the label-flipping method also bolsters the model's predictive confidence and reduces output entropy (Grandvalet & Bengio, 2004). Following two filtering steps, the reward model is trained on the unified datasets $\mathcal{D}_t \cup \mathcal{D}_f$, using the loss function as follows:

$$\mathcal{L}^{\texttt{CE}} = \underset{(\sigma^0, \sigma^1, \tilde{y}) \sim \mathcal{D}_t}{\mathbb{E}} \left[\mathcal{L}^{\texttt{Reward}}(\sigma^0, \sigma^1, \tilde{y})\right] + \underset{(\sigma^0, \sigma^1, 1-\tilde{y}) \sim \mathcal{D}_f}{\mathbb{E}} \left[\mathcal{L}^{\texttt{Reward}}(\sigma^0, \sigma^1, 1 - \tilde{y})\right] \tag{8}$$

Our denoising discriminator belongs to the category of sample selection methods for robust training. It stands out due to its use of a dynamically adjusted threshold enhanced by a term that accounts for instability and distributional shifts, thereby making it more suitable for the RL training process.

4.2. Warm Start

Sample selection methods usually suffer from accumulated errors due to incorrect selection, which highlights the need for good initialization. Meanwhile, we observe a significant degradation in performance during the transition from pre-training to online training (see Figure 2). This gap is clearly observable under noisy feedback settings and is fatal to robustness. It is exacerbated when following the most widely-adopted backbone, PEBBLE, which resets the Q-network and only retains the pre-trained policy after the pre-training phase. Because the Q-network is optimized with a biased reward model trained on noisy preferences to minimize the Bellman residual, this biased Q-function leads to a poor learning signal for the policy, erasing gains made during pre-training.

Inspired by these observations, we propose to warm start the reward model to facilitate a smoother transition from pre-training to online training. Specifically, we pre-train the reward model to approximate intrinsic rewards during the pre-training phase. Because the output layer of the reward model typically uses the tanh activation function (Lee et al., 2021b), we firstly normalize the intrinsic reward to the range $(-1, 1)$ as follows:

$$r_{\text{norm}}^{\text{int}}(\mathbf{s}_t) = \text{clip}(\frac{r^{\text{int}}(\mathbf{s}_t) - \bar{r}}{3\sigma_r}, -1 + \delta, 1 - \delta) \tag{9}$$

where $0 < \delta \ll 1$. $\bar{r}$ and σ_r are the mean and standard deviation of the intrinsic rewards, respectively. Then the agent receives the reward $r^{\text{int}}_{\text{norm}}$ and stores each tuple $(\mathbf{s}_t, \mathbf{a}_t, r^{\text{int}}_{\text{norm}}, \mathbf{s}_{t+1})$ in a replay buffer, denoted as $\mathcal{D}_{\text{pretrain}}$. During the reward model update, we sample batches of $(\mathbf{s}_t, \mathbf{a}_t)$ along with all encountered states $\mathcal{S} = \{\mathbf{s}|\mathbf{s} \text{ in } \mathcal{D}_{\text{pretrain}}\}$ for nearest neighbor searches. The loss function for updating the reward model $\hat{r}_\psi$ is given by the mean squared error as:

$$\mathcal{L}^{\texttt{MSE}} = \mathop{\mathbb{E}}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}_{\text{pretrain}}} \left[\frac{1}{2} \left(\hat{r}_\psi(\mathbf{s}_t, \mathbf{a}_t) - r^{\text{int}}_{\text{norm}}(\mathbf{s}_t) \right)^2 \right] \quad (10)$$

Thanks to warm start, both the Q-network and reward model are aligned with intrinsic rewards, allowing for the retention of all knowledge gained during pre-training (*i.e.,*policy, Q-network, and reward model) for subsequent online training. Moreover, the warm-started reward model contains more information than random initialization, enhancing the discriminator's ability initially.

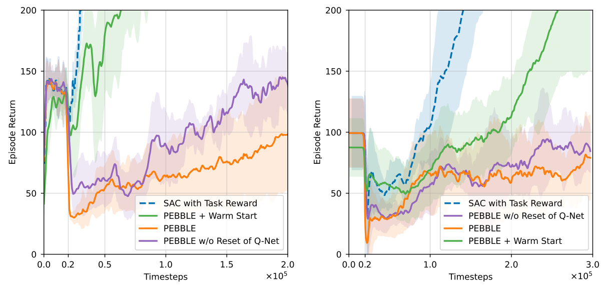


Figure 2. Performance degradation during transition on Walker-walk (left) and Quadruped-walk (right) with 30% noisy preferences. We pre-train an agent using SAC for 20k steps. The warm start method shows a smaller transition gap and faster recovery.

5. Experiments

5.1. Setups

We evaluate RIME on six complex tasks, including robotic manipulation tasks from Meta-world (Yu et al., 2020) and locomotion tasks from DMControl (Tassa et al., 2018; 2020). The details of experimental tasks are shown in Appendix D.1. Similar to prior works (Lee et al., 2021a;b; Park et al., 2021), to ensure a systematic and fair evaluation, we consider a scripted teacher that provides preferences between two trajectory segments based on the sum of ground-truth reward values for each segment. To generate noisy preferences, we follow the procedure of the "mistake" scripted teacher in Lee et al. (2021a), which flips correct preferences with a probability of ϵ. We refer to ϵ as the error rate. We choose PEBBLE (Lee et al., 2021b) as our backbone algorithm to implement RIME. In our experiments, we compare RIME against ground-truth reward-based SAC and four state-of-the-art PbRL algorithms: PEBBLE (Lee et al., 2021b), SURF (Park et al., 2021), RUNE (Liang et al., 2021), and MRN (Liu et al., 2022). Here, SAC is considered as an upper bound for performance, as it utilizes a ground-truth reward function not available in PbRL settings. We include SAC in our comparisons because it is the backbone RL algorithm of PEBBLE.

Implementation Details. For the hyperparameters of RIME, we fix $\alpha = 0.5$, $\beta_{\text{min}} = 1$ and $\beta_{\text{max}} = 3$ in the lower bound τ_{lower}, and fix the upper bound $\tau_{\text{upper}} = 3\ln(10)$ for all experiments. The decay rate k in τ_{upper} is $1/30$ for DM-Control tasks, and $1/300$ for Meta-world tasks, respectively. Other hyperparameters are kept the same as PEBBLE. For the sampling of queries, we use the disagreement sampling scheme for all PbRL algorithms, following the setting in Christiano et al. (2017). For the implementation of baselines, we use their corresponding publicly released repositories (see Table 8 for source codes). The feedback amount in total and per query session in each environment with specified error rate are detailed in Table 1.

For each task, we run all algorithms independently ten times and report the average performance along with the standard deviation. Tasks from Meta-world are measured on success rate, while tasks from DMControl are measured on ground-truth episode return. More details on the algorithm implementation are provided in Appendix D.2.

Table 1. Feedback amount in each environment with specified error rate. The "value" column refers to the feedback amount in total / per session.

Environment	Error rate	Value	Environment	Error rate	Value
Walker	$\epsilon < 0.2$	500/50	Button Press	$\epsilon < 0.2$	10000/50
Walker	$\epsilon \geq 0.2$	1000/100	Button Press	$\epsilon \geq 0.2$	20000/100
Cheetah	$\epsilon < 0.2$	500/50	Sweep Into	$\epsilon < 0.2$	10000/50
Cheetah	$\epsilon \geq 0.2$	1000/100	Sweep Into	$\epsilon \geq 0.2$	20000/100
Quadruped	$\epsilon < 0.2$	2000/200	Hammer	$\epsilon < 0.2$	20000/100
Quadruped	$\epsilon \geq 0.2$	4000/400	Hammer	$\epsilon = 0.2, 0.25$	40000/200
			Hammer	$\epsilon = 0.3$	80000/400

5.2. Results

For robotic manipulation tasks, we consider three tasks from Meta-world: Button-press, Sweep-into, and Hammer. For locomotion tasks, we choose three environments from DMControl: Walker-walk, Cheetah-run, and Quadruped-walk. Figure 3 and Figure 4 show the learning curves of RIME and baselines on Meta-world and DMControl tasks with five error rates, respectively. Table 2 shows the mean and standard deviation of metrics across the five error rates.

Since some preferences are corrupted, we observe that there is a gap between all PbRL methods and the best performance (*i.e.,*SAC with task reward), but RIME exceeds the PbRL baselines by a large margin in almost all environments. Especially, RIME remains effective in cases where all baselines struggle, such as Button-press with $\epsilon = 0.2$,

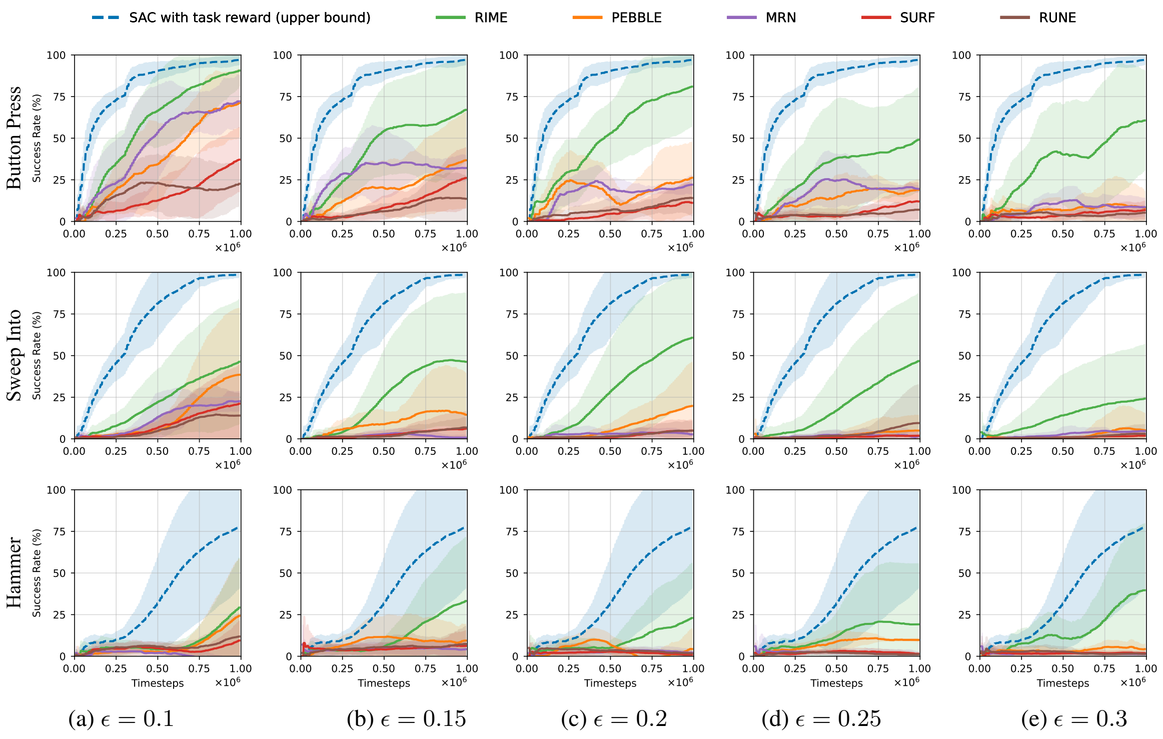


Figure 3. Learning curves for robotic manipulation tasks from Meta-world, where each row represents a specific task and each column corresponds to a different error rate ϵ. SAC serves as a performance upper bound, using a ground-truth reward function unavailable in PbRL settings. The corresponding number of feedback in total and per session are shown in Table 1. The solid line and shaded regions respectively denote the mean and standard deviation of the success rate, across ten runs.

Hammer with $\epsilon = 0.3$, and Walker with $\epsilon = 0.3$, etc. These results demonstrate that RIME significantly improves robustness against noisy preferences. We also observe that although some feedback-efficient baselines based on PEBBLE perform comparable to or even exceed PEBBLE in low-level noise, they become ineffective as the error rate rises. Additionally, Table 2 shows that PEBBLE is a robust algorithm second only to RIME. These results reveal that the pursuit of feedback efficiency leads to over-reliance on feedback quality.

5.3. Ablation Study

Performance with more types of (noisy) teachers. To investigate whether our method can generalize to more situations, we evaluate RIME, PEBBLE, and MRN with the other four types of teachers proposed by Lee et al. (2021a): Oracle, Skip, Equal, and Myopic. “Oracle” teacher provides ground-truth preferences. “Skip” teacher will skip the query if the cumulative rewards of segments are small. “Equal” teacher will give equal preference $\tilde{y} = (0.5, 0.5)$ if the difference between the cumulative rewards of two segments is small. “Myopic” teacher focuses more on the behavior at the end of segments. More details of these four teachers are shown in Appendix D.3. We report mean and standard deviation across five runs in Table 3. We found that RIME not only performs the best when teachers can provide ambiguous or wrong labels (Equal and Myopic), but it is also comparable with baselines on correct labels (Oracle and Skip). Based on the superior performance of RIME with multiple teachers, it has better chances of performing well with real teachers as well (Lee et al., 2021a).

Comparison with other robust training methods. Since the reward learning in PbRL is posed as a classification problem, the robust training methods in Machine Learning could be migrated to compare with RIME. Consider a sample selection method: adaptive denoising training (ADT) (Wang et al., 2021), two robust loss functions: Mean Absolute Error (MAE) (Ghosh et al., 2017) and t-CE (Feng et al., 2021), and a robust regularization method: label smoothing (LS) (Wei et al., 2021), as our baselines. ADT drops a-$\tau(t)$ proportion of samples with the largest cross-entropy loss at each training iteration, where $\tau(t) = \min(\gamma t, \tau_{\max})$. We set $\tau_{\max} = 0.3$, $\gamma = 0.003$, and 0.0003 for tasks from DMControl and Meta-world, respectively. MAE loss if formulated as $\mathcal{L}^{\texttt{MAE}}(\psi) = \mathbb{E}[|\tilde{y} - P_\psi|]$, while t-CE loss is formulated as $\mathcal{L}^{\texttt{t-CE}}(\psi) = \mathbb{E}[\sum_{i=1}^{t} \frac{(1-\tilde{y}^\top \cdot P_\psi)^i}{i}]$. Label smoothing method replace $\tilde{y}$ in Equation (2) with $(1-r)\cdot\tilde{y} + \frac{r}{2}\cdot[1,1]^\top$. We adopt $t = 4$ for t-CE loss and $r = 0.1$ for label smoothing

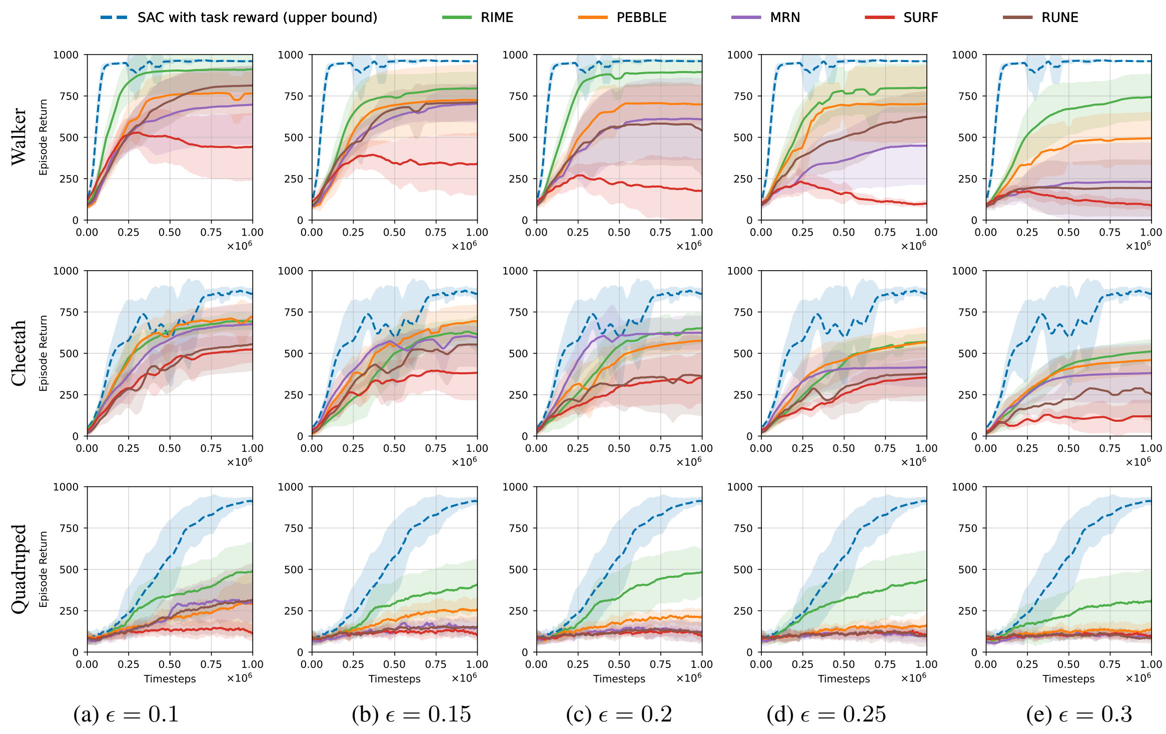
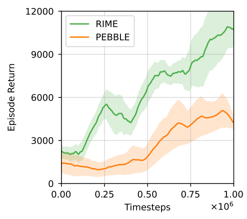


Figure 4. Learning curves on locomotion tasks from DMControl, where each row represents a specific task and each column corresponds to a different error rate ϵ setting. SAC serves as a performance upper bound, using a ground-truth reward function unavailable in PbRL settings. The corresponding number of feedback in total and per session are shown in Table 1. The solid line and shaded regions respectively denote the mean and standard deviation of episode return, across ten runs.

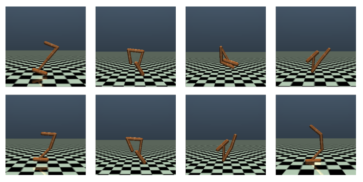
respectively. All baselines are implemented based on PEBBLE. Table 4 shows the result on four tasks with "Mistake" teacher and error rate as $\epsilon = 0.3$. Additional experiments with fixed lower bound τ_{lower} are provided in Appendix E. We observe that label smoothing almost fails to handle corrupted labels in our experiments. Sample selection methods (RIME and ADT) work better compared to other types of methods, and RIME still outperforms baselines. The reason is that the dynamic threshold with tolerance for out-of-distribution data is particularly well-suited to the RL training process.

Performance with real non-expert human teachers. The ultimate goal of improving robustness in PbRL is to better align with human users. To investigate how RIME performs with non-expert humans, we conduct experiments on Hopper utilizing non-expert human instructors, following the approach of Christiano et al. (2017); Lee et al. (2021b). Specifically, we selected five students with no prior knowledge on robotics from unrelated majors to provide annotations in an online setting. These students were instructed solely on the objective: to train an agent to perform backflips and received no additional information or guidance on the task. Their annotations were later used to train the algorithms RIME and PEBBLE. The feedback amount in total and per session are 100 and 10 respectively. For further details on the annotation protocol, refer to Appendix D.4.

We employ a hand-crafted reward function designed by experts (Christiano et al., 2017) as the ground-truth scripted teacher. We notice that compared to ground-truth preferences, the error rate of our non-expert annotations reached nearly 40%. The learning curves are shown in Fig. 5a. We find that RIME significantly outperforms PEBBLE when learning from actual non-expert human teachers and successfully performs consecutive backflips using 100 non-expert feedback, as shown in Figure 5b.



(a) Learning curves of RIME and PEBBLE on Hopper



(b) Frames of consecutive backflips from agents trained by RIME.

Figure 5. Ablation study on real non-expert human teachers

Trade-off between sample efficiency and robustness. To quantify the trade-off between sample efficiency and robust-

Table 2. Results on tasks from Meta-world and DMControl with "mistake" teacher. The result shows the mean and standard deviation of metric (*i.e.,*episode return for DMControl tasks and success rate for Meta-world tasks) across all five error rates within 10 runs.

Algorithm	DMControl			Meta-world		
	Walker	Cheetah	Quadruped	Button Press	Sweep Into	Hammer
PEBBLE	692.05 ± 192.67	**604.77** ± 126.63	208.66 ± 106.81	50.07 ± 29.53	19.09 ± 29.82	26.22 ± 32.08
SURF	211.28 ± 195.25	341.43 ± 178.10	125.51 ± 40.15	42.60 ± 27.55	16.04 ± 22.86	11.43 ± 22.76
RUNE	584.06 ± 271.84	424.17 ± 205.16	152.66 ± 131.43	27.04 ± 18.89	15.02 ± 19.18	12.14 ± 19.30
MRN	537.40 ± 281.36	538.74 ± 169.63	139.65 ± 88.24	43.48 ± 30.58	14.74 ± 22.89	6.35 ± 9.55
RIME	**837.79** ± 133.49	**602.18** ± 96.10	**415.52** ± 180.74	**85.70** ± 22.92	**51.96** ± 42.90	**42.28** ± 42.31

Table 3. Results on tasks from Meta-world and DMControl with 4 other types of (noisy) teacher. The result shows the mean and standard deviation of metric (*i.e.,*episode return for DMControl tasks and success rate for Meta-world tasks) averaged over 5 runs.

Domain	Environment	Algorithm	Oracle	Equal	Skip	Myopic	Average
DMControl	Walker	PEBBLE	877.44 ± 44.06	930.90 ± 17.77	904.31 ± 26.59	762.53 ± 165.98	868.80
		MRN	913.66 ± 51.84	942.80 ± 14.14	919.61 ± 48.87	882.34 ± 19.68	914.60
		RIME	958.87 ± 3.08	954.89 ± 1.43	950.83 ± 16.44	952.16 ± 1.80	**954.19**
	Quadruped	PEBBLE	620.35 ± 193.74	743.04 ± 107.30	776.01 ± 65.86	622.78 ± 200.04	690.55
		MRN	682.98 ± 182.25	666.56 ± 298.40	653.28 ± 150.78	525.91 ± 233.77	633.18
		RIME	678.36 ± 33.02	784.05 ± 56.96	755.58 ± 116.24	688.44 ± 130.59	**726.61**
Meta-world	Button-Press	PEBBLE	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	99.8 ± 0.4	99.95
		MRN	100.0 ± 0.0	100.0 ± 0.0	99.6 ± 0.5	100.0 ± 0.0	99.90
		RIME	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	**100.00**
	Hammer	PEBBLE	37.46 ± 44.95	53.20 ± 34.20	55.40 ± 33.97	48.40 ± 40.39	48.62
		MRN	67.20 ± 39.92	44.13 ± 34.86	52.20 ± 23.22	41.60 ± 33.97	51.28
		RIME	56.00 ± 27.28	53.80 ± 36.17	54.80 ± 34.25	70.60 ± 38.95	**58.80**

Table 4. Results of different robust training methods on tasks from Meta-world and DMControl with "mistake" teacher and error rate as $\epsilon = 0.3$. The result shows the mean and standard deviation of the metric averaged over 5 runs.

Algorithm	DMControl		Meta-world	
	Walker	Quadruped	Button Press	Hammer
PEBBLE	431 ± 157	125 ± 38	22.0 ± 13.8	8.6 ± 4.8
+ ADT	572 ± 247	**295** ± 194	**74.1** ± 20.9	37.6 ± 26.1
+ MAE	453 ± 295	246 ± 22	71.2 ± 31.0	17.8 ± 26.9
+ t-CE	548 ± 240	234 ± 47	36.0 ± 35.2	20.2 ± 31.4
+ LS	425 ± 172	117 ± 32	27.8 ± 21.0	4.2 ± 2.3
RIME	**741** ± 139	**301** ± 184	**80.0** ± 27.7	**58.5** ± 42.0

ness, we conduct experiments with RIME, where we either held the feedback volume constant and increase the error rate, or maintain the error rate while increasing the feedback volume, as shown in Tables 5 and 6, respectively. Table 5 indicates a progressive decline in performance with rising error rates, with a notable deterioration at an error rate of 0.3 across all four environments. Table 6 demonstrates that increasing the feedback volume improves performance, particularly at an error rate of 0.3, where doubling the feedback approximately doubles the performance gains. The same analysis repeated for PEBBLE is shown in Appendix E.

Component analysis. We perform an ablation study to individually evaluate each technique in RIME: warm start (WS), lower bound τ_{lower}, and upper bound τ_{upper} of KL divergence.

Table 5. Results of RIME as the error rate increases with constant amount of feedback, across 5 runs.

Environment	Feedback volume	Error rate		
		0.1	0.2	0.3
Walker	500	**909** ± 132	806 ± 162	493 ± 91
Quadruped	2000	**484** ± 166	400 ± 166	117 ± 24
Button-press	10000	**99.9** ± 0.3	92.2 ± 15.6	38.8 ± 33.8
Hammer	20000	**46.5** ± 43.9	41.2 ± 48.0	1.8 ± 3.1

Table 6. Results of RIME as the feedback volume increases with constant error rate, across 5 runs. N refers to the minimal feedback volume for each environment shown in Table 1.

Domain	Environment	Error rate	Feedback volume	
			N	$2N$
DMControl	Walker	0.2	806 ± 162	**894** ± 80
		0.3	493 ± 91	**741** ± 139
	Quadruped	0.2	400 ± 166	**477** ± 152
		0.3	117 ± 24	**301** ± 184
Meta-world	Button-press	0.2	92.2 ± 15.6	**93.4** ± 13.6
		0.3	38.8 ± 33.8	**80.0** ± 27.7

We present results in Table 7 in which we compare the performance of removing each component from RIME. We observe that warm start is crucial for robustness when the number of feedback is quite limited (*i.e.,*on Walker-walk). This is because the limited samples restrict the capability of the reward model, leading to more rounds of queries to cross the transition gap. Moreover, identifying whether the sample is corrupted or not is challenging for the discriminator initially due to the limited training data, underscoring the need for well-initialized reward models.

The lower bound τ_{lower} for filtering trustworthy samples is important in high error rate ($\epsilon = 0.3$ on Walker-walk and Button-press) and adequate feedback (on Button-press) situations. The upper bound τ_{upper} for flipping labels always brings some improvements in our ablation experiments. The full algorithm outperforms every other combination in most tasks. Additionally, the results show that although the contribution of warm start and denoising discriminator vary in different environments with low-level noise, they are both effective and their combination proves essential for the overall success of RIME in environments with high-level noise.

Table 7. Ablation study of components in RIME on Walker and Button-press with different error rates, across 5 runs.

Component			Walker		Button Press	
WS	τ_{lower}	τ_{upper}	$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.1$	$\epsilon = 0.3$
✗	✗	✗	749 ± 123	431 ± 157	93.1 ± 10.6	22.0 ± 13.8
✓	✗	✗	821 ± 93	483 ± 144	92.7 ± 11.8	25.8 ± 16.2
✗	✓	✓	688 ± 148	457 ± 190	97.2 ± 4.6	64.7 ± 26.5
✓	✗	✓	886 ± 70	492 ± 188	89.8 ± 11.5	35.1 ± 24.1
✓	✓	✗	842 ± 107	693 ± 167	96.9 ± 4.0	51.4 ± 30.0
✓	✓	✓	**909** ± 132	**741** ± 139	**99.9** ± 0.3	**80.0** ± 27.7

6. Conclusion

In this paper, we present RIME, a robust algorithm for preference-based reinforcement learning (PbRL) designed for effective reward learning from noisy preferences. Unlike previous research which primarily aims to enhance feedback efficiency, RIME focuses on improving robustness by employing a sample selection-based discriminator to dynamically denoise preferences. To reduce accumulated error due to incorrect selection, we utilize a warm-start method for the reward model, enhancing the initial capability of the denoising discriminator. The warm-start approach also facilitates a seamless transition from pre-training to online training. Our experiments show that RIME substantially boosts the robustness of the state-of-the-art PbRL method across a wide range of complex robotic manipulation and locomotion tasks. Ablation studies further demonstrate that the warm-start approach is crucial for both robustness and feedback efficiency. We believe that RIME has the potential to broaden the applicability of PbRL by leveraging preferences from non-expert users or crowd-sourcing platforms.

Limitations. The intrinsic challenge of PbRL with noisy preferences is the trade-off between sample-efficiency and robustness. As shown in Table 1, 5, and 6, RIME still needs to increase the amount of feedback in total to perform reasonably as the error rate increases. Therefore, exploring how to achieve better trade-offs is an interesting future direction. Another limitation is that RIME introduces several hyperparameters in the denoising discriminator, resulting in extra tuning efforts for optimal performance. Moreover, the divergence between the noise in actual human preferences and the noise models proposed by BPref deserves further investigation. While attaining high performance across multiple simulated teachers suggests a likelihood of good performance under real human teachers, it remains imperative to investigate simulated noise models that more closely align with the characteristics of real-world human preference noise.

Impact Statement

Compared to natural language processing, control tasks typically demand higher-quality human feedback (Kim et al., 2022). Our work reduces the difficulty of annotating human preferences for control tasks, allowing for the presence of noise in preferences and thereby alleviating the requirement of domain knowledge for annotators. This enables human preferences for control tasks to be sourced from crowd-sourcing platforms or ordinary users, potentially expanding the application scope of PbRL.

Acknowledgments

We are grateful to the anonymous reviewers and editors for their insightful suggestions. This work was partially supported by the National Science and Technology Major Project (2022ZD0117102), the National Natural Science Foundation of China under Grants 62271485, U1909204 and 62303462, the Provincial Key Research and Development Program of Zhejiang (project number: 2022C01129), Ningbo International Science and Technology Cooperation Project (2023H020), and Beijing Natural Science Foundation under Grant L233005. Lastly, special thanks go to Tongtian Yue, Ruixi Qiao, and Yueyang Yang for their assistance in data annotation, engaging discussions, and meticulous review of draft versions.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. *Advances in Neural Information Processing Systems*, 35: 28955–28971, 2022.

Akrour, R., Schoenauer, M., and Sebag, M. Preference-based policy learning. In *Proceedings of the 2011th European Conference on Machine Learning and Knowledge Discovery in Databases-Volume Part I*, pp. 12–27. Springer, 2011.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Chen, Y., Wu, T., Wang, S., Feng, X., Jiang, J., Lu, Z., McAleer, S., Dong, H., Zhu, S.-C., and Yang, Y. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

Cheng, L., Zhou, X., Zhao, L., Li, D., Shang, H., Zheng, Y., Pan, P., and Xu, Y. Weakly supervised learning with side information for noisy labeled images. In *European Conference on Computer Vision*, pp. 306–321, 2020.

Cheng, W., Fürnkranz, J., Hüllermeier, E., and Park, S.-H. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *Proceedings of the 2011th European Conference on Machine Learning and Knowledge Discovery in Databases-Volume Part I*, pp. 312–327. Springer, 2011.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30, 2017.

Feng, L., Shu, S., Lin, Z., Lv, F., Li, L., and An, B. Can cross entropy loss be robust to label noise? In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 2206–2212, 2021.

Ghosh, A., Kumar, H., and Sastry, P. S. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Goldberger, J. and Ben-Reuven, E. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations*, 2016.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. *Advances in Neural Information Processing Systems*, 17, 2004.

Hejna III, D. J. and Sadigh, D. Few-shot preference learning for human-in-the-loop rl. In *Conference on Robot Learning*, pp. 2014–2025. PMLR, 2023.

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari. *Advances in Neural Information Processing Systems*, 31, 2018.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. Champion-level drone racing using deep reinforcement learning. *Nature*, 620 (7976):982–987, 2023.

Kim, C., Park, J., Shin, J., Lee, H., Abbeel, P., and Lee, K. Preference transformer: Modeling human preferences using transformers for rl. In *International Conference on Learning Representations*, 2022.

Lee, K., Smith, L., Dragan, A., and Abbeel, P. B-pref: Benchmarking preference-based reinforcement learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021a.

Lee, K., Smith, L. M., and Abbeel, P. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, pp. 6152–6163. PMLR, 2021b.

Lee, K., Liu, H., Ryu, M., Watkins, O., Du, Y., Boutilier, C., Abbeel, P., Ghavamzadeh, M., and Gu, S. S. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

Li, M., Soltanolkotabi, M., and Oymak, S. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 4313–4324. PMLR, 2020.

Liang, X., Shu, K., Lee, K., and Abbeel, P. Reward uncertainty for exploration in preference-based reinforcement learning. In *International Conference on Learning Representations*, 2021.

Liu, H. and Abbeel, P. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

Liu, R., Bai, F., Du, Y., and Yang, Y. Meta-reward-net: Implicitly differentiable reward learning for preference-based reinforcement learning. *Advances in Neural Information Processing Systems*, 35:22270–22284, 2022.

Lukasik, M., Bhojanapalli, S., Menon, A., and Kumar, S. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pp. 6448–6458. PMLR, 2020.

Lyu, Y. and Tsang, I. W. Curriculum loss: Robust learning and generalization against label corruption. In *International Conference on Learning Representations*, 2019.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Park, J., Seo, Y., Shin, J., Lee, H., Abbeel, P., and Lee, K. Surf: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. In *International Conference on Learning Representations*, 2021.

Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623): 990–996, 2022.

Singh, H., Misra, N., Hnizdo, V., Fedorowicz, A., and Demchuk, E. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23 (3-4):301–321, 2003.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. Robust learning by self-transition for handling noisy labels. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1490–1500, 2021.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Trochim, P., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., et al. dm_control: Software and tasks for continuous control. *arXiv preprint arXiv:2006.12983*, 2020.

Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., Bennice, M., Fu, C., Ma, C., Jiao, J., et al. Jump-start reinforcement learning. In *International Conference on Machine Learning*, pp. 34556–34583. PMLR, 2023.

Wang, W., Feng, F., He, X., Nie, L., and Chua, T.-S. Denoising implicit feedback for recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 373–381, 2021.

Wei, J., Liu, H., Liu, T., Niu, G., Sugiyama, M., and Liu, Y. To smooth or not? when label smoothing meets noisy labels. *arXiv preprint arXiv:2106.04149*, 2021.

Xia, X., Liu, T., Han, B., Gong, C., Wang, N., Ge, Z., and Chang, Y. Robust early-learning: Hindering the memorization of noisy labels. In *International Conference on Learning Representations*, 2020.

Xue, W., An, B., Yan, S., and Xu, Z. Reinforcement learning from diverse human preferences. *arXiv preprint arXiv:2301.11774*, 2023.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020.

Yue, T., Cheng, J., Guo, L., Dai, X., Zhao, Z., He, X., Xiong, G., Lv, Y., and Liu, J. Sc-tune: Unleashing self-consistent referential comprehension in large vision language models. *arXiv preprint arXiv:2403.13263*, 2024.

Zhang, Z. and Sabuncu, M. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in Neural Information Processing Systems*, 31, 2018.

Zhou, M., Luo, J., Villella, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., Alban, M., Fadakar, I., Chen, Z., et al. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776*, 2020.

A. RIME Algorithm Details

In this section, we provide the full procedure for RIME based on the backbone PbRL algorithm, PEBBLE (Lee et al., 2021b), in Algorithm 1.

Algorithm 1 RIME

```
1:  Initialize policy $\pi_\phi$, Q-network $Q_\theta$ and reward model $\hat{r}_\psi$
2:  Initialize replay buffer $\mathcal{B} \leftarrow \emptyset$
3:  for each pre-training step $t$ do                                        ▷ UNSUPERVISED PRE-TRAINING
4:      Collect $\mathbf{s}_{t+1}$ by taking $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
5:      Compute normalized intrinsic reward $r^{\text{int}}_{\text{norm},t} \leftarrow r^{\text{int}}_{\text{norm}}(\mathbf{s}_t)$ as in Equation (9)
6:      Store transitions $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r^{\text{int}}_{\text{norm},t})\right\}$
7:      for each gradient step do
8:          Sample minibatch $\left\{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_{j+1}, r^{\text{int}}_{\text{norm},j})\right\}_{j=1}^{B} \sim \mathcal{B}$
9:          Optimize policy and Q-network with respect to $\phi$ and $\theta$ using SAC
10:         Update reward model $\hat{r}_\psi$ according to Equation (10)                ▷ WARM START
11:     end for
12: end for
13: Initialize the maximum KL divergence value $\rho = \infty$
14: Initialize a dataset of noisy preferences $\mathcal{D}_{\text{noisy}} \leftarrow \emptyset$
15: for each training step $t$ do                                              ▷ ONLINE TRAINING
16:     if step to query preferences then                                      ▷ ROBUST REWARD LEARNING
17:         Generate queries from replay buffer $\{(\sigma_i^0, \sigma_i^1)\}_{i=1}^{N_{\text{query}}} \sim \mathcal{B}$ and corresponding human feedback $\{\tilde{y}_i\}_{i=1}^{N_{\text{query}}}$
18:         Store preferences $\mathcal{D}_{\text{noisy}} \leftarrow \mathcal{D}_{\text{noisy}} \cup \{(\sigma_i^0, \sigma_i^1, \tilde{y}_i)\}_{i=1}^{N_{\text{query}}}$
19:         Compute lower bound $\tau_{\text{lower}}$ according to Equation (5)
20:         Filter trustworthy samples $\mathcal{D}_t$ using lower bound $\tau_{\text{lower}}$ as in Equation (6)
21:         Flip labels using upper bound $\tau_{\text{upper}}$ to obtain dataset $\mathcal{D}_f$ as in Equation (7)
22:         Update reward model $\hat{r}_\psi$ with samples from $\mathcal{D}_t \cup \mathcal{D}_f$ according to Equation (8)
23:         Relabel entire replay buffer $\mathcal{B}$ using $\hat{r}_\psi$
24:         Update parameter $\rho$ with the maximum KL divergence between predicted and annotated labels in dataset $\mathcal{D}_t \cup \mathcal{D}_f$
25:     end if
26:     Collect $\mathbf{s}_{t+1}$ by taking $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
27:     Store transitions $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \widehat{r}_\psi(\mathbf{s}_t, \mathbf{a}_t))\}$
28:     for each gradient step do
29:         Sample minibatch from replay buffer $\{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_{j+1}, \hat{r}_\psi(\mathbf{s}_j, \mathbf{a}_j))\}_{j=1}^{B} \sim \mathcal{B}$
30:         Optimize policy and Q-network with respect to $\phi$ and $\theta$ using SAC
31:     end for
32: end for
```

B. Effects of biased reward model

Previous work empirically showed the detrimental impact of noisy preference on the reward model (Lee et al., 2021a). To further demonstrate the effects of a biased reward model, we introduce the following theorem and give the proof as follows.

Assumption B.1 (Fitting error of reward model). Post the phase of reward learning, the fitting error between the learned reward model $\hat{r}_\psi$ and the ground-truth reward function r^* within the state-action distribution encountered by policy π is upper-bounded by a value δ:

$$\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi} |\hat{r}_\psi(\mathbf{s},\mathbf{a}) - r^*(\mathbf{s},\mathbf{a})| \leq \delta \tag{11}$$

Theorem B.2 (Upper bound of Q-function error). *Consider a Markov Decision Process characterized by the state transition function P, ground-truth reward function r^*, and discount factor γ. Let Q^π denote the Q-function for policy π with respect to the learned reward model $\hat{r}_\psi$. Then the error between Q^π and the optimal Q-function Q^* is upper-bounded by the fitting* *error δ of the reward model:*

$$\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi} |Q^\pi(\mathbf{s},\mathbf{a}) - Q^*(\mathbf{s},\mathbf{a})| \le \frac{\delta}{1-\gamma\delta} \tag{12}$$

Proof.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi} |Q^\pi(\mathbf{s},\mathbf{a}) - Q^*(\mathbf{s},\mathbf{a})| && (13)\\
=&\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi} \left|\hat{r}_\psi(\mathbf{s},\mathbf{a}) + \gamma\sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s},\mathbf{a})\sum_{\mathbf{a}'}\pi(\mathbf{a}'|\mathbf{s}')Q^\pi(\mathbf{s}',\mathbf{a}') - r^*(\mathbf{s},\mathbf{a}) - \gamma\sum_{\mathbf{s}'}P(\mathbf{s}'|\mathbf{s},\mathbf{a})\max_{\mathbf{a}'}Q^*(\mathbf{s}',\mathbf{a}')\right| && (14)\\
\le&\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi} |\hat{r}_\psi(\mathbf{s},\mathbf{a}) - r^*(\mathbf{s},\mathbf{a})| + \gamma\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}\sum_{\mathbf{s}'}P(\mathbf{s}'|\mathbf{s},\mathbf{a})\left|\sum_{\mathbf{a}'}\pi(\mathbf{a}'|\mathbf{s}')Q^\pi(\mathbf{s}',\mathbf{a}') - \max_{\mathbf{a}'}Q^*(\mathbf{s}',\mathbf{a}')\right| && (15)\\
=&\delta + \gamma\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}\sum_{\mathbf{s}'}P(\mathbf{s}'|\mathbf{s},\mathbf{a})\left|\sum_{\mathbf{a}'}\pi(\mathbf{a}'|\mathbf{s}')(Q^\pi(\mathbf{s}',\mathbf{a}') - \max_{\mathbf{a}''}Q^*(\mathbf{s}',\mathbf{a}''))\right| && (16)\\
\le&\delta + \gamma\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}\sum_{\mathbf{s}'}P(\mathbf{s}'|\mathbf{s},\mathbf{a})\sum_{\mathbf{a}'}\pi(\mathbf{a}'|\mathbf{s}')(\max_{\mathbf{a}''}Q^*(\mathbf{s}',\mathbf{a}'') - Q^\pi(\mathbf{s}',\mathbf{a}')) && (17)\\
\le&\delta + \gamma\mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\rho^\pi}\sum_{\mathbf{s}'}P(\mathbf{s}'|\mathbf{s},\mathbf{a})\sum_{\mathbf{a}'}\pi(\mathbf{a}'|\mathbf{s}')(Q^*(\mathbf{s}',\mathbf{a}') - Q^\pi(\mathbf{s}',\mathbf{a}')) && (18)\\
=&\delta + \gamma\mathop{\mathbb{E}}_{(\mathbf{s}',\mathbf{a}')\sim\rho^\pi}|Q^\pi(\mathbf{s}',\mathbf{a}') - Q^*(\mathbf{s}',\mathbf{a}')| && (19)\\
\le&\delta + \gamma\delta^2 + \gamma^2\delta^3 + \ldots && (20)\\
\le&\frac{\delta}{1-\gamma\delta} && (21)
\end{aligned}$$

□

C. Proof for Theorem 4.1

Theorem (4.1). *Consider a preference dataset $\{(\sigma_i^0,\sigma_i^1,\tilde{y}_i)\}_{i=1}^n$, where $\tilde{y}_i$ is the annotated label for the segment pair (σ_i^0,σ_i^1) with the ground truth label y_i. Let x_i denote the tuple (σ_i^0,σ_i^1). Assume the cross-entropy loss $\mathcal{L}^{\text{CE}}$ for clean data (whose $\tilde{y}_i = y_i$) within this distribution is bounded by ρ. Then, the KL divergence between the predicted preference $P_\psi(x)$ and the annotated label $\tilde{y}(x)$ for a corrupted sample x is lower-bounded as follows:*

$$D_{\text{KL}}(\tilde{y}(x)\|P_\psi(x)) \ge -\ln\rho + \frac{\rho}{2} + \mathcal{O}(\rho^2) \tag{22}$$

Proof. For a clean sample (σ^0,σ^1) with annotated label $\tilde{y}$ and ground-truth label y, we have $\tilde{y} = y$. Denote the predicted label as P_ψ. In PbRL, the value of $y(0)$ can take one of three forms: $y(0)\in\{0, 0.5, 1\}$. We categorize and discuss these situations as follows:

1. For $y(0) = 0$:

   Because the cross-entropy loss $\mathcal{L}^{\text{CE}}$ for clean data is bounded by ρ, we can express:

$$\mathcal{L}^{\text{CE}}(P_\psi,\tilde{y}) = -\ln(1 - P_\psi(0)) \le \rho \tag{23}$$

   From the above, we have:

$$P_\psi(0) \le 1 - \exp(-\rho) \tag{24}$$

   Then if the label is corrupted, denoted by $\tilde{y}_c$ (*i.e.*,$\tilde{y}_c = (1,0)$ in this case), the KL divergence between the predicted label and the corrupted label is formulated as follows:

$$D_{\text{KL}}(\tilde{y}_c\|P_\psi) = -\ln P_\psi(0) \ge -\ln(1-\exp(-\rho)) \tag{25}$$

2. For $y(0) = 1$:

   The discussion parallels the $y(0) = 0$ case. Hence, the KL divergence between the predicted label and the corrupted label also maintains a lower bound:

$$D_{\mathrm{KL}}(\tilde{y}_c \| P_\psi) \geq -\ln(1 - \exp(-\rho)) \tag{26}$$

3. For $y(0) = 0.5$:

   Although this case is not under the mistake model settings (Lee et al., 2021a), the lower bound still holds in this case. Due to the bounded cross-entropy loss $\mathcal{L}^{\text{CE}}$ for clean data, we have:

$$\mathcal{L}^{\text{CE}}(P_\psi, \tilde{y}) = -\frac{1}{2}\ln P_\psi(0) - \frac{1}{2}\ln(1 - P_\psi(0)) \leq \rho \tag{27}$$

   Solving the inequality (27), we can get:

$$P_\psi(0)^2 - P_\psi(0) + \exp(-2\rho) \leq 0 \tag{28}$$

   When $\rho \geq \ln 2$, the inequality (28) has a solution:

$$1 - p \leq P_\psi(0) \leq p \tag{29}$$

   where $p = \frac{1+\sqrt{1-4\exp(-2\rho)}}{2}$.

   Then if the label is corrupted, *i.e.*,$\tilde{y}_c \in \{(0,1),(1,0)\}$, the KL divergence between the predicted label and the corrupted label is formulated as follows:

$$D_{\mathrm{KL}}(\tilde{y}_c \| P_\psi) \geq \min(-\ln P_\psi(0), -\ln(1 - P_\psi(0))) = -\ln p \tag{30}$$

   Construct an equation about ρ:

$$f(\rho) = p - 1 + \exp(-\rho) = \frac{1 + \sqrt{1 - 4\exp(-2\rho)}}{2} - 1 + \exp(-\rho) \tag{31}$$

   where $\rho \geq \ln 2$.

   Denote $z = \exp(-\rho)$, Equation (31) can be simplified as follows:

$$f(z) = z + \frac{\sqrt{1 - 4z^2}}{2} - \frac{1}{2} \tag{32}$$

   where $0 < z \leq \frac{1}{2}$.

   Derivative of function f with respect to z, we have:

$$f^{'}(z) = 1 - 2\sqrt{\frac{1}{\frac{1}{z^2} - 4}} \tag{33}$$

   Function $f^{'}(z)$ decreases monotonically when $z \in (0, 0.5]$, is greater than 0 on the interval $(0, \frac{\sqrt{2}}{4})$, and is less than 0 on the interval $(\frac{\sqrt{2}}{4}, 0.5]$. Therefore, we have:

$$f(z) \leq \max(f(0), f(\frac{1}{2})) = 0 \tag{34}$$

   Thus, $p \leq 1 - \exp(-\rho)$ when $\rho \geq \ln 2$. In turn, we have:

$$D_{\mathrm{KL}}(\tilde{y}_c \| P_\psi) = -\ln p \geq -\ln(1 - \exp(-\rho)) \tag{35}$$

To sum up, inequality (36) holds for the corrupted samples:

$$D_{\mathrm{KL}}(\tilde{y}_c \| P_\psi) \geq -\ln(1 - \exp(-\rho)) \tag{36}$$

Perform Taylor expansion of the lower bound at $\rho = 0$, we can get:

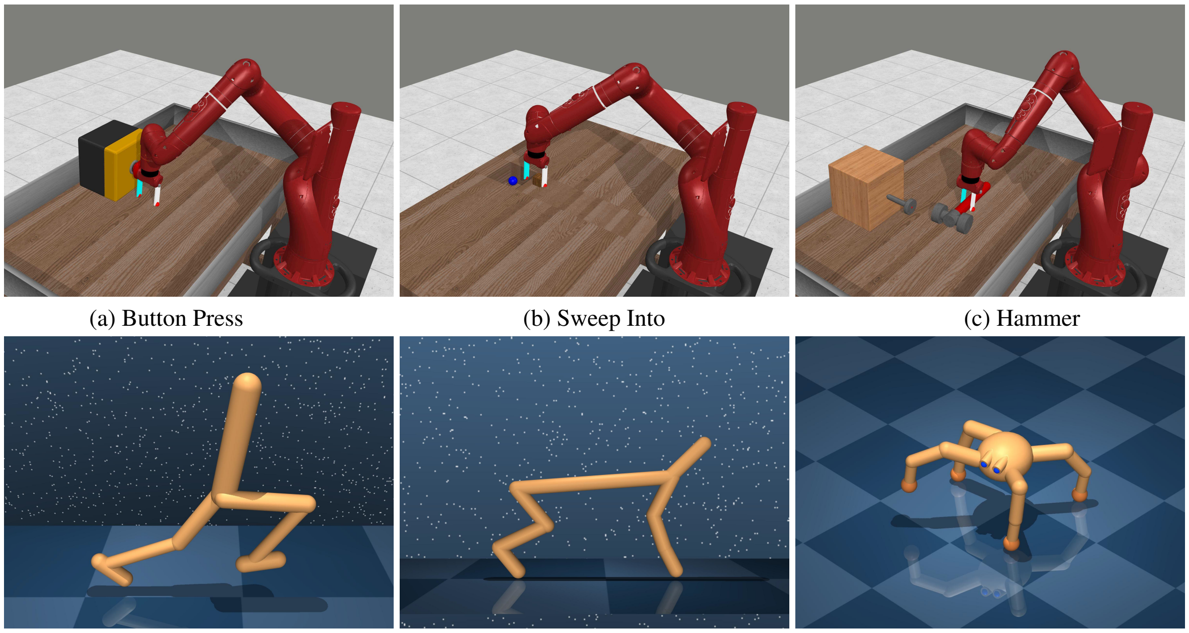
$$D_{\mathrm{KL}}(\tilde{y}_c \| P_\psi) \geq -\ln(1 - \exp(-\rho)) = -\ln \rho + \frac{\rho}{2} + \mathcal{O}(\rho^2) \tag{37}$$

□

D. Experimental Details

D.1. Tasks

The robotic manipulation tasks from Meta-world (Yu et al., 2020) and locomotion tasks from DMControl (Tassa et al., 2018; 2020) used in our experiments are shown in Figure 6.



(a) Button Press (b) Sweep Into (c) Hammer

(d) Walker (e) Cheetah (f) Quadruped

Figure 6. Six tasks from Meta-world (a-c) and DMControl (d-f).

Meta-world Tasks:

- ○ Button Press: An agent controls a robotic arm to press a button. The button's initial position is randomized.
- ○ Sweep Into: An agent controls a robotic arm to sweep a ball into a hole. The ball's starting position is randomized.
- ○ Hammer: An agent controls a robotic arm to hammer a screw into a wall. The initial positions of both the hammer and the screw are randomized.

DMControl Tasks:

- ○ Walker: A planar walker is trained to control its body and walk on the ground.
- ○ Cheetah: A planar biped is trained to control its body and run on the ground.
- ○ Quadruped: A four-legged ant is trained to control its body and limbs, enabling it to crawl on the ground.

D.2. Implementation Details

For the implementation of baselines, we use their corresponding publicly released repositories that are shown in Table 8. SAC serves as a performance upper bound because it uses a ground-truth reward function which is unavailable in PbRL settings for training. The detailed hyperparameters of SAC are shown in Table 9. PEBBLE's settings remain consistent with its original implementation, and the specifics are detailed in Table 10. For SURF, RUNE, MRN, and RIME, most hyperparameters are the same as those of PEBBLE and other hyperparameters are detailed in Table 11, 12, 13, and 14, respectively. The total amount of feedback and feedback amount per session in each experimental condition are detailed in Table 1. The reward model comprises an ensemble of three MLPs. Each MLP consists of three layers with 256 hidden units, and the output of the reward model is constrained using the tanh activation function.

Table 8. Source codes of baselines.

Algorithm	Url
SAC, PEBBLE	`https://github.com/rll-research/BPref`
SURF	`https://github.com/alinlab/SURF`
RUNE	`https://github.com/rll-research/rune`
MRN	`https://github.com/RyanLiu112/MRN`

Table 9. Hyperparameters of SAC.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	2 (DMControl), 3 (Meta-world)	Initial temperature	0.1
Hidden units per each layer	1024 (DMControl), 256 (Meta-world)	Optimizer	Adam
Learning rate	0.0005 (Walker), 0.001 (Cheetah)	Critic target update freq	2
	0.0001 (Quadruped), 0.0003 (Meta-world)	Critic EMA τ	0.005
Batch Size	1024 (DMControl), 512 (Meta-world)	(β_1, β_2)	$(0.9, 0.999)$
Steps of unsupervised pre-training	9000	Discount γ	0.99

Table 10. Hyperparameters of PEBBLE.

Hyperparameter	Value
Segment Length	50
Learning rate	0.0005 (Walker, Cheetah), 0.0001 (Quadruped), 0.0003 (Meta-world)
Frequency of feedback	20000 (Walker, Cheetah), 30000 (Quadruped), 5000 (Meta-world)
Number of reward functions	3

Table 11. Hyperparameters of SURF.

Hyperparameter	Value
Unlabeled batch ratio μ	4
Threshold τ	0.999 (Cheetah, Sweep Into), 0.99 (others)
Loss weight λ	1
Min/Max length of cropped segment	$45/55$
Segment length before cropping	60

Table 12. Hyperparameters of RUNE.

Hyperparameter	Value
Initial weight of intrinsic reward β_0	0.05
Decay rate ρ*	0.001 (Walker), 0.0001 (Cheetah, Quadruped, Button Press)
	0.00001 (Sweep Into, Hammer)

*: Following the instruction of Liang et al. (2021), we carefully tune the hyperparameter ρ in a range of $\rho \in \{0.001, 0.0001, 0.00001\}$ and report the best value for each environment.

Table 13. Hyperparameters of MRN.

Hyperparameter	Value
Bi-level updating frequency N	5000 (Cheetah, Hammer, Button Press), 1000 (Walker)
	3000 (Quadruped), 10000 (Sweep Into)

Table 14. Hyperparameters of RIME.

Hyperparameter	Value
Coefficient α in the lower bound τ_{lower}	0.5
Minimum weight $\beta_{\min}$	1
Maximum weight $\beta_{\max}$	3
Decay rate k	1/30 (DMControl), 1/300 (Meta-world)
Upper bound τ_{upper}	$3\ln(10)$
δ in Equation (9)	1×10^{-8}
Steps of unsupervised pre-training	2000 (Cheetah), 9000 (others)

D.3. Details of scripted teachers

For a pair of trajectory segments (σ^0, σ^1) with length H, where $\sigma^i = \{(\mathbf{s}_1^i, \mathbf{a}_1^i), \ldots, (\mathbf{s}_H^i, \mathbf{a}_H^i)\}, (i = 0, 1)$. The ground-truth reward function from the environment is $r(\mathbf{s}, \mathbf{a})$. Then scripted teachers are defined as follows (Lee et al., 2021a):

Oracle: Oracle teacher provides ground-truth preferences. It prefers the segment with larger cumulative ground-truth rewards. For example, if $\sum_{i=1}^{H} r(\mathbf{s}_i^0, \mathbf{a}_i^0) > \sum_{i=1}^{H} r(\mathbf{s}_i^1, \mathbf{a}_i^1)$, then it returns the label as $(1, 0)$.

Equal: Equal teacher gives equal preference $(0.5, 0.5)$ if the difference between the cumulative rewards of two segments is small. In particular, if $|\sum_{t=1}^{H} r(\mathbf{s}_t^1, \mathbf{a}_t^1) - \sum_{t=1}^{H} r(\mathbf{s}_t^0, \mathbf{a}_t^0)| < \delta$, then it return the label $(0.5, 0.5)$. $\delta = \frac{H}{T} R_{\text{avg}} \epsilon_{\text{adapt}}$, where T is the episode length, R_{avg} is the average returns of current policy, ϵ_{adapt} is a hyperparameter and is set to 0.1 in experiments, following the setting in Lee et al. (2021a).

Skip: Skip teacher skips the query if the cumulative rewards of segments are small. In particular, if $\max_{i \in \{0,1\}} \sum_{t=1}^{H} r(\mathbf{s}_t^i, \mathbf{a}_t^i) < \delta$, then it will skip this query.

Myopic: Myopic teacher focuses more on the behavior at the end of segments. It prefers the segment with larger discounted cumulative ground-truth rewards. For example, if $\sum_{i=1}^{H} \gamma^{H-t} r(\mathbf{s}_i^0, \mathbf{a}_i^0) > \sum_{i=1}^{H} \gamma^{H-t} r(\mathbf{s}_i^1, \mathbf{a}_i^1)$, then it returns the label $(1, 0)$.

Mistake: Mistake teacher flips the ground-truth preference labels with a probability of ϵ.

D.4. Details of experiments with human teachers

Human experiments adopt an online paradigm consistent with PEBBLE's pipeline, where agent training alternates with reward model training. When it is the timestep to collect preferences (post-agent training and pre-reward training), the training program generates segment pairs, saving each segment in GIF format. The program then pauses, awaiting the input of human preferences. At this juncture, human labelers engage, reading the paired segments in GIF format and labeling their preferences. Subsequently, this annotated data is fed into the training program for reward model training.

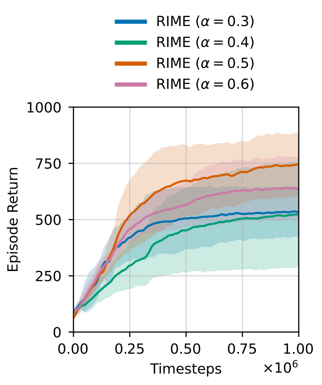
To ensure a fair comparison between RIME and PEBBLE, for one human labeler, we start the training programs for RIME and PEBBLE simultaneously. When both training programs are waiting for inputting preferences, we collate all GIF pairs from both RIME and PEBBLE and shuffle the order. Then the human labeler starts working. Therefore, from the labeler's perspective, he/she does not know which algorithm the currently labeled segment pair comes from and just focuses on labeling according to his/her preference. The labeled data is then automatically directed to the respective training programs. We conduct this experiment parallelly on each of the five labelers, thus preferences from different users do not get mixed.

In the Hopper task, the labeling process itself requires approximately 5 minutes, not accounting for waiting time. However, due to the online annotation, labelers experience downtime while waiting for agent training and GIF pair generation. Consequently, considering all factors, the total time commitment for labeling amounts to about 20 minutes per annotator.

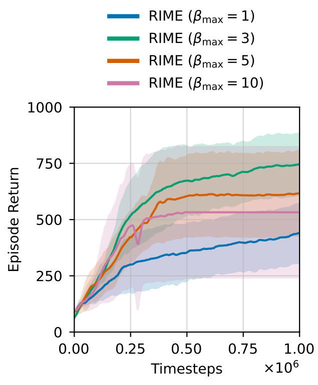
E. Additional Experiment Results

Effects of hyperparameters of RIME. We investigate how the hyperparameters of RIME affect the performance under noisy feedback settings. In Figure 7 we plot the learning curves of RIME with a different set of hyperparameters: (a) coefficient α in the lower bound τ_{lower}: $\alpha \in \{0.3, 0.4, 0.5, 0.6\}$, (b) maximum value of β_t: $\beta_{\max} \in \{1, 3, 5, 10\}$, (c) decay rate $k \in \{0.01, 1/30, 0.06, 0.1\}$, and (d) upper bound of KL divergence $\tau_{\text{upper}} \in \{2\ln(10), 3\ln(10), 4\ln(10)\}$.

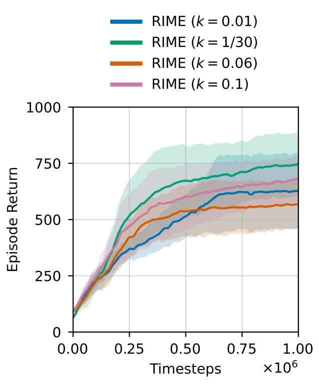
For the coefficient α in the lower bound τ_{lower}, we find the theoretical value $\alpha = 0.5$ performs the best. The maximum weight $\beta_{\max}$ and decay rate k control the weight of uncertainty term in the lower bound τ_{lower}: $\beta_t = \max(\beta_{\min}, \beta_{\max} - kt)$. The combination of $\beta_{\max} = 3$ and $k = 1/30$ also performs optimally. Due to the quite limited feedback amount (1000 feedback) and training epochs for the reward model (around $150 \sim 200$ epochs on Walker-walk), RIME is sensitive to the weight of uncertainty term. If one tries to increase $\beta_{\max}$ to add more tolerance for trustworthy samples in early-stage, we recommend increasing the decay rate k simultaneously so that the value of β_t decays to its minimum within about $1/3$ to $1/2$ of the total epochs. For the upper bound τ_{upper}, although we use $3\ln(10)$ for balanced performance on DMControl tasks, individually fine-tuning τ_{upper} can further improve the performance of RIME on the corresponding task, such as using $\tau_{\text{upper}} = 4\ln(10)$ for Walker-walk.



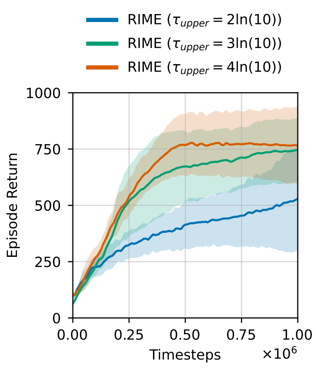
(a) Coefficient α



(b) Maximum weight $\beta_{\max}$



(c) Decay rate k



(d) Upper bound τ_{upper}

Figure 7. Hyperparameter analysis on Walker-walk using 1000 feedback with $\epsilon = 0.3$. The results show the mean and standard deviation averaged over five runs.

Effects of different uncertainty terms in the lower bound. In RIME, we use an auxiliary uncertainty term τ_{unc} in the lower bound τ_{lower} to accommodate tolerance during the early training stages and in cases of distribution shifts. The standard deviation of the KL divergence, denoted as the KL metric in this section, is employed to discern these cases. Here, we compare this with two other metrics: the disagreement metric and a combination of both, termed as KL + disagreement. The disagreement metric uses the standard deviation of $P_\psi[\sigma^0 \succ \sigma^1]$ across the ensemble of reward models (denoted as s_P) to discern cases of distribution shifts: $\tau_{\text{unc}} = \gamma_t \cdot s_P$. Our intuition is that the predictions of the model for OOD data typically vary greatly. Notably, this metric induces sample-level, rather than buffer-level, thresholds, potentially offering more nuanced threshold control. The combined metric, KL + disagreement, integrates both as $\tau_{\text{unc}} = \beta_t \cdot s_{\text{KL}} + \gamma_t \cdot s_P$.

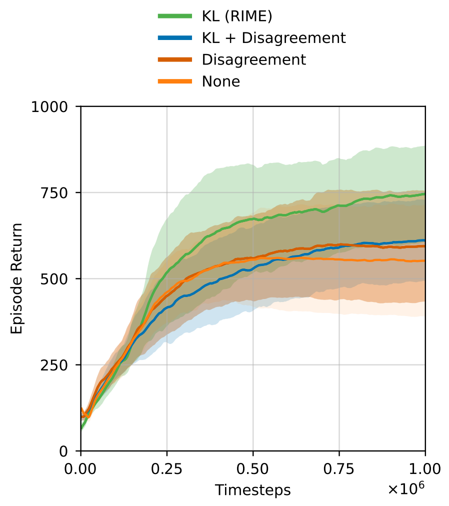


Figure 8. Uncertainty term analysis on Walker-walk using 1000 feedback with error rate $\epsilon = 0.3$, across ten runs.

For reference, we also include a group devoid of any uncertainty term, termed the "None" group. As shown in Figure 8, the KL metric outperforms the other approaches on Walker-walk with an error rate of $\epsilon = 0.3$. This might be because the disagreement metric fluctuates violently at every query time, often leading to excessive trust in new data, which hinders the stabilization of the lower bound.

Comparison with fixed lower bound. We conducted experiments to compare fixed lower bound with dynamic lower bound (RIME) and presented the results in Table 15. Notice that the convergence value of τ_{lower} in RIME are 0.972 and 0.711 for Walker and Button-press, respectively. Table 15 indicates that the dynamic lower bound employed by RIME outperforms the fixed lower bound method substantially. This superiority stems from RIME's ability to adapt its lower bound value according to the situation during training. By contrast, employing a fixed lower bound might exacerbate incorrect selections either in the early or late phase of training, depending on whether the lower bound is small or large respectively. The issue of incorrect selection will in turn lead to cumulative errors and compromise the effectiveness of selection-based robust training methods.

Explore the error rate limits of RIME. To understand the boundaries of RIME, we conduct supplementary experiments in both the Walker-walk and Cheetah-run environments, varying the error rates ϵ within the range $\{35\%, 40\%, 45\%, 50\%\}$, across 5 runs. The results, presented in Table 16, reveal that RIME failed with 45% and 40% noisy data in the Walker-walk and Cheetah-run environments, respectively. Interestingly, even with the feedback amount increased to ten times the

Table 15. Ablation study of the lower bound on Walker and Button-press over 5 runs.

Method	Value of τ_{lower}	Walker ($\epsilon = 0.3$)	Button-press ($\epsilon = 0.3$)
	0.5	98 ± 92	36.8 ± 45.4
Fixed	0.7	179 ± 165	64.8 ± 26.4
lower	0.9	256 ± 88	58.0 ± 41.9
bound	1.1	468 ± 186	49.0 ± 37.6
	1.3	325 ± 79	56.6 ± 34.0
RIME	dynamic	**741** ± 139	**80.0** ± 27.7

Table 16. Performance of RIME with different noise levels over 5 runs.

Environment	Feedback volume	Error rate			
		35%	40%	45%	50%
Walker	1000	646.58	497.63	164.31	/
	5000	/	500.64	210.86	145.05
	10000	/	554.69	312.61	217.34
Cheetah	1000	403.72	246.51	/	/
	5000	440.4	347.4	/	/
	10000	503.57	393.0	/	/

minimum, it has not change the result of failure. The performance gains from increasing the amount of feedback are limited under high error rates ($\epsilon \geq 0.4$).

Trade-off between sample efficiency and robustness. We repeat the same analysis that is detailed in 5.3 for PEBBLE and present the results in Table 17 and Table 18 below. Similarly, we observe a gradual decline in PEBBLE's performance with rising error rates. Doubling the amount of feedback engenders a marginal enhancement to PEBBLE; however, this improvement is negligible on Quadruped and Button-press when $\epsilon \geq 0.2$. Intriguingly, by comparing the 4-th column (N) of Table 6 with the 5-th column ($2N$) of Table 18, we find that RIME even outperforms PEBBLE with only half the number of feedbacks in most cases.

Table 17. Results of PEBBLE as the error rate increases with constant amount of feedback, across 5 runs.

Environment	Feedback volume	Error rate		
		0.1	0.2	0.3
Walker	500	**749** ± 123	490 ± 252	230 ± 172
Quadruped	2000	**292** ± 166	171 ± 26	125 ± 38
Button-press	10000	**93.1** ± 10.6	21.6 ± 15.3	17.8 ± 25.2
Hammer	20000	**36.6** ± 41.4	20.0 ± 17.8	15.7 ± 12.0

Table 18. Results of PEBBLE as the feedback volume increases with constant error rate, across 5 runs. N refers to the minimal feedback volume for each environment shown in Table 1.

Domain	Environment	Error rate	Feedback volume	
			N	$2N$
DMControl	Walker	0.2	490 ± 252	**656** ± 158
		0.3	230 ± 172	**431** ± 157
	Quadruped	0.2	171 ± 26	**212** ± 47
		0.3	125 ± 38	**165** ± 35
Meta-world	Button-press	0.2	21.6 ± 15.3	**26.2** ± 35.7
		0.3	17.8 ± 25.2	**22.0** ± 13.8