Query-Policy Misalignment in Preference-Based Reinforcement Learning

Xiao Hu [*1] **Jianxiong Li** [*1] **Xianyuan Zhan** [1] **Qing-Shan Jia** [1] **Ya-Qin Zhang** [1]

Abstract

Preference-based reinforcement learning (PbRL) provides a natural way to align RL agents' behavior with human desired outcomes, but is often restrained by costly human feedback. To improve feedback efficiency, most existing PbRL methods focus on selecting queries to maximally improve the overall quality of the reward model, but counter-intuitively, we find that this may not necessarily lead to improved performance. To unravel this mystery, we identify a long-neglected issue in the query selection schemes of existing PbRL studies: *Query-Policy Misalignment*. We show that the seemingly informative queries selected to improve the overall quality of reward model actually may not align with RL agents' interests, thus offering little help on policy learning and eventually resulting in poor feedback efficiency. We show that this issue can be effectively addressed via *near on-policy query* and a specially designed *hybrid experience replay*, which together enforce the bidirectional query-policy alignment. Simple yet elegant, our method can be easily incorporated into existing approaches by changing only a few lines of code. We showcase in comprehensive experiments that our method achieves substantial gains in both human feedback and RL sample efficiency, demonstrating the importance of addressing *query-policy misalignment* in PbRL tasks.

1. Introduction

Reward plays an imperative role in every reinforcement learning (RL) problem. It specifies the learning objective and incentivizes agents to acquire correct behaviors. With well-designed rewards, RL has achieved remarkable success in solving many complex tasks (Mnih et al., 2015; Silver et al., 2017; Degrave et al., 2022). However, designing a suitable reward function remains a longstanding challenge (Abel et al., 2021; Li et al., 2023; Singh et al., 2009; Sorg, 2011). Due to human cognitive bias and system complexity (Hadfield-Menell et al., 2017), it is difficult to accurately convey complex behaviors through numerical rewards, resulting in unsatisfactory or even hazardous agent behaviors (Li et al., 2023).

Preference-based RL (PbRL), also known as RL from human feedback (RLHF), promises learning reward functions autonomously without the need for tedious hand-engineered reward design (Christiano et al., 2017; Lee et al., 2021a;b; Park et al., 2022; Liang et al., 2022; Shin et al., 2023; Tien et al., 2023). Instead of using provided rewards, PbRL queries a (human) overseer to provide preferences between a pair of agent's behaviors, and the RL agent seeks to maximize a reward function that is trained to be consistent with human preferences. This approach provides a more natural way for humans to communicate their desired outcomes to RL agents, enabling more desirable behaviors (Christiano et al., 2017). However, labeling a large number of preference queries requires tremendous human effort, inhibiting its wide application in real-world scenarios (Lee et al., 2021b; Park et al., 2022; Liang et al., 2022).

To enable feedback-efficient PbRL, it is crucial to carefully select which behaviors to query the overseer's preference and which ones not to, in order to extract as much information as possible from each preference labeling process (Christiano et al., 2017; Lee et al., 2021b; Biyik & Sadigh, 2018; Biyik et al., 2020). Motivated by this, existing works focus on querying the most "informative" behaviors for preferences that are likely to maximally rectify the overall reward model, such as sampling according to ensemble disagreements, mutual information, or behavior entropy (Lee et al., 2021b; Shin et al., 2023). However, it is also observed that these carefully designed query schemes often only marginally outperform the simplest scheme that randomly selects behaviors to query human preferences (Ibarz et al., 2018; Lee et al., 2021b). This counter-intuitive phenomenon brings about a puzzling question: *Why are these seemingly informative queries actually not effective in PbRL training?*

[*]Equal contribution [1]Tsinghua University, Beijing, China. Correspondence to: Xiao Hu <hu-x21@mails.tsinghua.edu.cn>, Xianyuan Zhan <zhanxianyuan@air.tsinghua.edu.cn>, Qing-Shan Jia <jiaqs@tsinghua.edu.cn>.

The Many Facets of Preference Learning Workshop at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023.

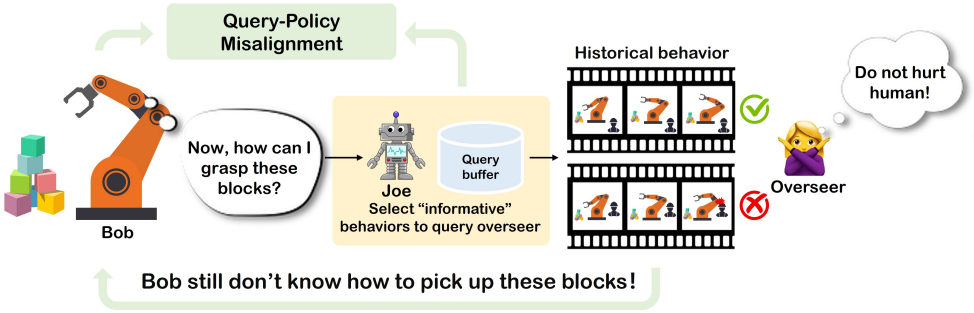


Figure 1: Illustration of *query-policy misalignment*.

In this paper, we identify a long-neglected issue in the query schemes of existing approaches that is responsible for their ineffectiveness: *Query-Policy Misalignment*. Take Figure 1 as an illustration, Bob is a robot currently attempting to pick up some blocks, while Joe is a querier selecting the behaviors that he considers the most informative for the reward model to query the overseer. However, the chosen behaviors may not align with Bob's current interests. So, even after the query, Bob is still clueless about how to pick up blocks, which means that a valuable query opportunity might be wasted. In Section 4, we provide concrete experiments and find that such misalignment is prevalent in previous query schemes. Specifically, we observe that the queried behaviors often fall outside the scope of the current policy's visitation distribution, indicating that they are less likely to be encountered by the current RL agent, and thus are not what the current agent is really interested in. Therefore, the queried behaviors bring little impact on the current policy learning and result in poor feedback efficiency.

Interestingly, we find that the *query-policy misalignment* issue can be easily addressed by *near on-policy query selection*, which can be implemented by making a simple modification to existing query schemes. This technique ensures that the query scheme only selects the recent behaviors of RL agents, which in turn enables the overseer to provide timely feedback on relevant behaviors for current policy learning rather than some irrelevant experiences. By making this minimalist modification to the query schemes of existing methods, we showcase substantial improvements in terms of both feedback and sample efficiency as compared to the base schemes. Further leveraging the insight from *query-policy misalignment*, we introduce a simple technique, called *hybrid experience replay*, that simply updates RL agents using experiences uniformly sampled from not only the entire replay buffer but also some recent experiences. Intuitively, this technique ensures that the RL agent updates more frequently on the region where human preferences have been recently labeled, thereby further aligning the policy learning with the recent human preferences.

In summary, the combination of *near on-policy query selection* and *hybrid experience replay* establishes bidirectional query-policy alignment, making every query accountable for policy learning. Notably, these techniques can be easily incorporated into existing PbRL approaches (Lee et al., 2021b; Park et al., 2022) with minimal modifications. We evaluate our proposed method on benchmark environments in DeepMind Control Suite (DMControl) (Tassa et al., 2018) and MetaWorld (Yu et al., 2020). Simple yet elegant, experimental results demonstrate significant feedback and sample efficiency gains, highlighting the effectiveness of our proposed method and the importance of addressing *query-policy misalignment* in PbRL.

2. Related Work

PbRL provides a natural approach for humans (oracle overseer) to communicate desired behaviors with RL agents by making relative judgments between a pair of behaviors (Akrour et al., 2011; Pilarski et al., 2011; Christiano et al., 2017; Stiennon et al., 2020; Wu et al., 2021). However, acquiring preferences is typically costly, imposing high demands on feedback efficiency (Lee et al., 2021b; Park et al., 2022; Liang et al., 2022).

Query selection schemes in PbRL. It is widely acknowledged that the query selection scheme plays a crucial role in PbRL for improving feedback efficiency (Christiano et al., 2017; Biyik & Sadigh, 2018; Biyik et al., 2020; Ibarz et al., 2018; Lee et al., 2021b). Motivated by this idea, prior works commonly assess the information quality of queries using metrics such as entropy (Biyik & Sadigh, 2018; Ibarz et al., 2018; Lee et al., 2021b), the L2 distance in feature space (Biyik et al., 2020) or ensemble disagreement of the reward model (Christiano et al., 2017; Ibarz et al., 2018; Lee et al., 2021b; Park et al., 2022; Liang et al., 2022). Based on these metrics, researchers often employ complex sampling approaches such as greedy sampling (Biyik & Sadigh, 2018), K-medoids algorithm (Biyik & Sadigh, 2018; Rdusseeun & Kaufman, 1987), or Poisson disk sampling (Bridson, 2007; Biyik et al., 2020), *etc.*, to sample the most "informative" queries. Despite adding extra computational costs, it is observed that these complex schemes often benefit little to policy learning, and sometimes perform similarly to the simplest scheme that directly queries humans with randomly selected queries (Lee et al., 2021b; Ibarz et al., 2018). In this paper, we identify a common issue with the existing

schemes, *query-policy misalignment*, which shows that the selected seemingly "informative" queries may not align well with the current interests of RL agents, providing an explanation of why existing schemes often lead to less improved feedback efficiency.

Other techniques for improving feedback efficiency. In addition to designing effective query selection schemes, there are other efforts to improve the feedback efficiency of PbRL from various perspectives. For instance, some works focus on ensuring that the initial queries are feasible for humans to provide high-quality preferences by initializing RL agents with imitation learning (Ibarz et al., 2018) or unsupervised-pretraining (Lee et al., 2021b). (Liang et al., 2022) shows that adequate exploration can improve both sample and feedback efficiency. Recently, (Park et al., 2022) applies pseudo-labeling (Lee et al., 2013) in semi-supervised learning along with temporal cropping data augmentation to remedy the limited human preferences and achieves SOTA performances. Note that our proposed method is orthogonal to these techniques and can be used in conjunction with any of them with minimal code modifications.

3. Preliminary

The RL problem is typically specified as a Markov Decision Process (MDP) (Puterman, 2014), which is defined by a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, r, T, \gamma)$. $\mathcal{S}, \mathcal{A}$ represent the state and action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the dynamics and $\gamma \in (0, 1)$ is the discount factor. The goal of RL is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward.

Off-policy actor-critic RL. To tackle the high-dimensional state-action space, off-policy actor-critic RL algorithms typically maintain a parametric Q-function $Q_\theta(s, a)$ and a parametric policy $\pi_\phi(a|s)$, which are optimized via alternating between *policy evaluation* (Eq. (1)) and *policy improvement* (Eq. (2)) steps. The *policy evaluation* step seeks to enforce $Q_\theta(s, a)$ to be consistent with the empirical Bellman operator that backs up samples (s, a, s') stored in replay buffer $\mathcal{D}$, while the *policy improvement* step improves π_ϕ via maximizing the learned Q-value:

$$\hat{Q}^{k+1} \leftarrow \arg\min_Q \mathbb{E}_{s,a,s' \sim \mathcal{D}}[(r(s,a) + \gamma \mathbb{E}_{a' \sim \hat{\pi}^k(a'|s')} \left[\hat{Q}^k\left(s', a'\right)\right] - Q(s,a))^2] \tag{1}$$

$$\hat{\pi}^{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi(a|s)} \left[\hat{Q}^{k+1}(s,a)\right] \tag{2}$$

Preference-based RL. Different from the standard RL setting, the reward signal is not available in PbRL. Instead, a (human) overseer provides preferences between pairs of *trajectory segments*, and the agent leverages these feedbacks to learn a reward function $\widehat{r}_\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to be consistent with the provided preferences. A trajectory segment σ is a sequence of observations and actions $\{s_k, a_k, \cdots, s_{k+L-1}, a_{k+L-1}\} \in (\mathcal{S} \times \mathcal{A})^L$, which is typically shorter than the whole trajectories. Given a pair of segments (σ^0, σ^1), the overseer provides a feedback signal y indicating which segment the overseer prefer, i.e., $y \in \{0, 1\}$, where 0 indicates the overseer prefers segment σ^0 over σ^1, 1 otherwise. Following the Bradley-Terry model (Bradley & Terry, 1952) , we can model a preference predictor P_ψ using the reward function $\widehat{r}_\psi(s, a)$:

$$P_\psi\left[\sigma^1 \succ \sigma^0\right] = \frac{\exp \sum_t \widehat{r}_\psi\left(s_t^1, a_t^1\right)}{\sum_{i \in \{0,1\}} \exp \sum_t \widehat{r}_\psi\left(s_t^i, a_t^i\right)} \tag{3}$$

where $\sigma^1 \succ \sigma^0$ denotes the overseer prefers σ^1 than σ^0. Typically, $\widehat{r}_\psi$ is optimized by minimizing the cross-entropy loss of preference predictor P_ψ and the true preference label y.

$$\mathcal{L}^{\text{reward}} = -\mathop{\mathbb{E}}_{(\sigma^0, \sigma^1, y) \sim \mathcal{D}^\sigma} [(1-y) \log P_\psi\left[\sigma^0 \succ \sigma^1\right] + y \log P_\psi\left[\sigma^1 \succ \sigma^0\right]] \tag{4}$$

where $\mathcal{D}^\sigma$ denotes the preference buffer which stores the history overseer's preferences $\{(\sigma^0, \sigma^1, y)\}$. Most recent PbRL methods (Lee et al., 2021b; Park et al., 2022; Liang et al., 2022) are built upon off-policy actor-critic RL algorithms to enhance sample and feedback efficiency. In these methods, pairs of segments (σ^0, σ^1) are selected from trajectories in the off-policy RL replay buffer $\mathcal{D}$. These selected pairs are then sent to the overseer for preference query, yielding feedback (σ^0, σ^1, y). These feedback instances are subsequently stored in the separate preference buffer $\mathcal{D}^\sigma$ for reward learning. The *query selection scheme* refers to the strategy that decides which pair of segments (σ^0, σ^1) should be selected from $\mathcal{D}$ for preference query prior to the reward learning. An effective query selection scheme is of paramount importance to achieve high feedback efficiency in PbRL.

4. Query-Policy Misalignment

A motivating example. In this section, we conduct an intuitive experiment to demonstrate a prevalent but long-neglected issue: *query-policy misalignment*, which accounts for the poor feedback efficiency of existing query selection schemes. Specifically, as illustrated in Figure 2(a), we consider a 2D continuous space with (x, y) coordinates defined on $[-10, 10]^2$. For each step, the RL agent can move Δx and Δy ranging from $[-1, 1]$. We want the agent to navigate from the start to the goal as quickly as possible. We run PEBBLE (Lee et al., 2021b), a popular PbRL method, with two widely used query selection schemes in previous studies: *uniform query selection* that randomly selects segments to query preferences and *disagreement query selection* that selects the segments with the largest ensemble disagreement of preference predictors (Lee et al., 2021b; Christiano et al.,

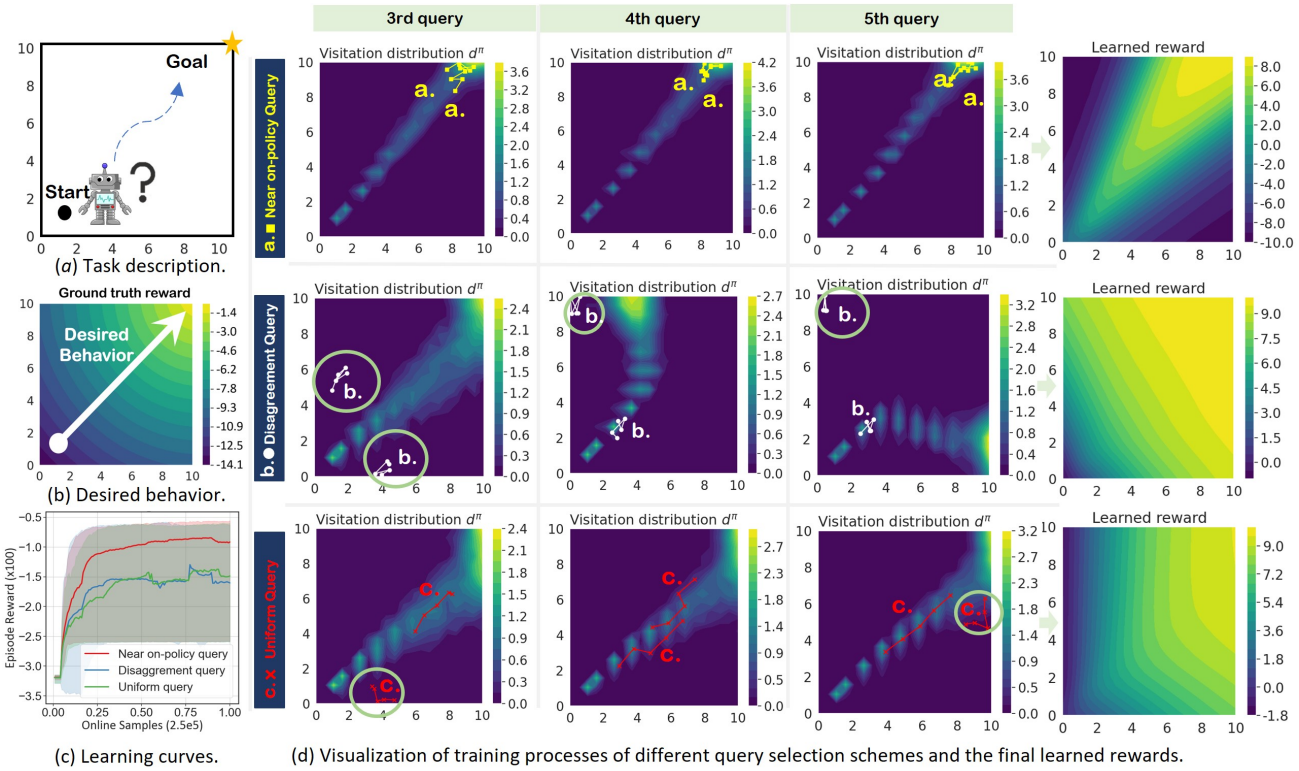


Figure 2: Impacts of *query-policy misalignment* in PbRL training.

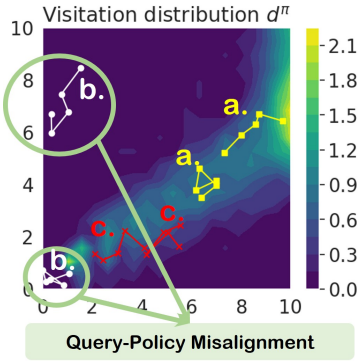


Figure 3: *Query-policy misalignment*

2017; Ibarz et al., 2018; Park et al., 2022). We also run PEBBLE with *near on-policy query selection* that selects the recently collected segments from the policy-environment interactions and will be further investigated in later content. We track the visitation distribution d^π of current agent's policy π throughout the training process and plot the selected segments of different query selection schemes in Figure 2(d). Please refer to the Appendix B for detailed experimental setups.

We observe that the selected segments of existing query selection schemes typically fall outside the scope of the visitation distribution d^π (marked with green circles). We refer to this phenomenon as *query-policy misalignment*, as illustrated in Figure 3. Such misalignment wastes valuable feedback because the overseer provides preferences on experiences that are less likely encountered by, or in other words, irrelevant to the current RL agent's learning. Therefore, although these selected segments may improve the overall quality of the reward model in the full state-action space, they contribute little to RL training and potentially cause feedback inefficiency. By contrast, *near-on-policy selection* selects fresh segments that are recently visited by the current RL policy, enabling timely feedback on the current status of the policy and leading to significant performance gain, as shown in Figure 2(c). Furthermore, while the learned reward of *near on-policy query selection* may differ from the ground truth reward, it provides the most useful information to guide the agent to navigate toward the right direction and more strictly discourages detours as compared to the vague per-step ground truth reward. This suggests that the learned reward captures more targeted information in solving the task while also blocking out less useful information in the ground truth reward, thus enabling more effective policy learning.

Insights from the theoretical perspective. As mentioned earlier, existing PbRL methods struggle with feedback inefficiency caused by *query-policy misalignment*. Surprisingly, we find that this issue can be easily resolved by two pivotal techniques: *near on-policy query selection* and *hybrid experience replay*. To provide a comprehensive understanding of these two techniques, we begin by outlining a theoretical

analysis that encapsulates the underlying intuition behind them. Given the learned reward function $\widehat{r}_\psi$ and the current stochastic policy π, we denote $Q^\pi_{\widehat{r}_\psi}$ as the Q-function of π associated with $\widehat{r}_\psi$ and $\hat{Q}^\pi_{\widehat{r}_\psi}$ as the *estimated* Q-function obtained from the policy evaluation step in Eq. (1), which serves as an approximation of $Q^\pi_{\widehat{r}_\psi}$. Q^π_r denotes the Q-function of π with true reward r. Define the distribution-dependent norms $\|f(x)\|_\mu := \mathbb{E}_{x\sim\mu}\left[|f(x)|\right]$. We present the following theorem:

Theorem 4.1. *Given the two conditions* $\|\widehat{r}_\psi - r\|_{d^\pi} \leq \epsilon$ *and* $\|Q^\pi_{\widehat{r}_\psi} - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \leq \alpha$*, the value approximation error* $\|Q^\pi_r - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi}$ *is upper bounded as:*

$$\|Q^\pi_r - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \leq \frac{\epsilon}{1-\gamma} + \alpha \tag{5}$$

The proof is presented in Appendix A. Note that both the conditions are dependent on d^π, when they are jointly satisfied, the Q-function of the current policy π with learned reward can be adequately estimated with bounded deviation from the Q-function of π with true reward. If we look more closely, the condition $\|\widehat{r}_\psi - r\|_{d^\pi} \leq \epsilon$ implies that the reward (or preference) prediction should exhibit higher accuracy within the on-policy distribution d^π, which reveals why *near on-policy selection* can boost the performance. On the other hand, the condition $\|Q^\pi_{\widehat{r}_\psi} - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \leq \alpha$ implies that the policy evaluation, based on empirical Bellman iteration, should also be approximately accurate within d^π.

5. Query-Policy Alignment for Preference-based RL (QPA)

Inspired by the insights from Theorem 4.1, we introduce an elegant solution: QPA, which can effectively address *query-policy misalignment* and is also compatible with existing off-policy PbRL methods with **only 20 lines of code** modifications. Please see Algorithm 1 for the outline of our method.

5.1. Near On-policy Query Selection

In contrast to existing query selection schemes, we highlight that the *segment query selection should be aligned with the on-policy distribution*. In particular, it is crucial to ensure that the pairs of segments (σ^0, σ^1) selected for preference queries are within the scope of the current policy's visitation distribution d^π. By assigning more overseer's feedback to segments obtained from on-policy trajectories of the current policy π, we aim to enhance the accuracy of the preference (reward) predictor specifically within the on-policy distribution d^π. This aligns with the intuition from the condition $\|\widehat{r}_\psi - r\|_{d^\pi} \leq \epsilon$. We refer to this query selection scheme as *on-policy query selection*.

In practice, a natural approach to implement on-policy query selection is to utilize the current policy π to interact with the environment and generate a set of trajectories. Then, pairs of segments (σ^0, σ^1) can be selected from these trajectories to obtain overseer's preference query. While such "absolute" on-policy query selection ensures that all selected segments conform to the on-policy distribution d^π, it may have a negative impact on the sample efficiency of off-policy RL due to the additional on-policy rollout. Instead of performing the "absolute" on-policy query selection, an alternative is to select segments that are within or "near" on-policy distribution. As we mentioned in Section 3, in typical off-policy PbRL, (σ^0, σ^1) are selected in trajectories sampled from the RL replay buffer $\mathcal{D}$. A simple yet effective way to perform near on-policy query selection is to choose (σ^0, σ^1) from the most recent trajectories stored in $\mathcal{D}$. For the sake of clarity, we refer to the buffer that stores the most recent trajectories as the near on-policy buffer $\mathcal{D}^{\text{on}}$. It's worth noting that this simple approach strikes a balance between on-policy query and sample efficiency in RL. Furthermore, it is particularly easy to implement and enables a minimalist modification to existing PbRL methods. Take the well-known and publicly-available B-pref (Lee et al., 2021a) PbRL implementation framework as an example, all that's required is to reduce the size of the first-in-first-out query selection buffer, which can be thought of as a knockoff of $\mathcal{D}^{\text{on}}$.

5.2. Hybrid Experience Replay

Following the near on-policy query selection and reward learning procedure, it's also important to ensure that *value learning is aligned with the on-policy distribution*. Specifically, more attention should be paid to improving the veracity of Q-function within on-policy distribution d^π, where the preference (reward) predictor performs well in the preceding step, in accordance with the insight from the condition $\|Q^\pi_{\widehat{r}_\psi} - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \leq \alpha$ in Theorem 4.1.

To update the Q-function, existing off-policy PbRL algorithms perform the empirical Bellman iteration Eq.(1) by simply sampling transitions (s, a, s') uniformly from the replay buffer $\mathcal{D}$. Although the aforementioned near on-policy query selection can effectively facilitate the learning of the reward function within d^π, the Q-function may not be accurately approximated on d^π due to inadequate empirical Bellman iterations using (s, a, s') transitions drawn from d^π. Taking inspiration from combined Q-learning (Zhang & Sutton, 2017) and some prioritized experience replay methods that consider on-policyness (Liu et al., 2021; Sinha et al., 2022), we devise a *hybrid experience replay* mechanism. To elaborate, we still sample transitions (s, a, s') uniformly, but from two different sources. Specifically, half of the uniformly sampled transitions are drawn from $\mathcal{D}$, while the other half are drawn from the near on-policy buffer $\mathcal{D}^{\text{on}}$. The proposed mechanism can provide assurance that the Q-function is updated adequately near d^π.

Algorithm 1: QPA

Input : Frequency of overseer feedback K, number of queries per feedback session M, near on-policy buffer size N, data augmentation ratio τ

Initialize : Initialize replay buffer $\mathcal{D}$, query buffer $\mathcal{D}^\sigma$, near on-policy buffer $\mathcal{D}^{\text{on}}$ with size N

(Option) Unsupervised pretraining (Lee et al., 2021b)
for *each iteration* **do**
  Collect and store new experience $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$, $\mathcal{D}^{\text{on}} \leftarrow \mathcal{D}^{\text{on}} \cup \{(s, a, r, s')\}$
  if *iteration* $\%K == 0$ **then**
    /* Near on-policy query selection (see Section 5.1) */
    $\{(\sigma^0, \sigma^1)\}_{i=1}^{M} \sim \mathcal{D}^{\text{on}}$
    Query for preferences $\{y\}_{i=1}^{M}$, and store preference $\mathcal{D}^\sigma \leftarrow \mathcal{D}^\sigma \cup \{(\sigma^0, \sigma^1, y)\}_{i=1}^{M}$
    for *each gradient step* **do**
      Sample a minibatch preferences $\mathcal{B} \leftarrow \{(\sigma^0, \sigma^1, y)\}_{i=1}^{h} \sim \mathcal{D}^\sigma$
      /* Data augmentation for reward learning (see Section 5.3) */
      Generate augmented preferences $\hat{\mathcal{B}} \leftarrow \{(\hat{\sigma}^0, \hat{\sigma}^1, y)\}_{i=1}^{h \times \tau}$ based on $\mathcal{B}$
      Optimize $\mathcal{L}^{\text{reward}}$ in Eq. (4) *w.r.t.* $\widehat{r}_\psi$ using $\hat{\mathcal{B}}$
  for *each gradient step* **do**
    /* Hybrid experience replay (see Section 5.2) */
    Sample minibatch $\mathcal{D}_{\text{mini}} \leftarrow \{(s, a, r, s')\}_{i=1}^{\frac{n}{2}} \sim \mathcal{D}$, $\mathcal{D}^{\text{on}}_{\text{mini}} \leftarrow \{(s, a, r, s')\}_{i=1}^{\frac{n}{2}} \sim \mathcal{D}^{\text{on}}$
    Optimize SAC agent using $\mathcal{D}_{\text{mini}} \cup \mathcal{D}^{\text{on}}_{\text{mini}}$

5.3. Data Augmentation for Reward Learning

Besides the above two key designs, we also adopt the temporal data augmentation technique for reward learning (Park et al., 2022). To further clarify, we randomly subsample several shorter pairs of snippets $(\hat{\sigma}^0, \hat{\sigma}^1)$ from the queried segments (σ^0, σ^1, y), and put these $(\hat{\sigma}^0, \hat{\sigma}^1, y)$ into the preference buffer $\mathcal{D}^\sigma$ for optimizing the cross-entropy loss in Eq.(4). Data augmentation has been widely used in many deep RL (Kostrikov et al., 2020; Laskin et al., 2020b;a) algorithms, and also has been seamlessly integrated into previous PbRL algorithms for consistency regularization (Park et al., 2022). Diverging from the practical implementation in (Park et al., 2022), we generates multiple $(\hat{\sigma}^0, \hat{\sigma}^1, y)$ instances from a single (σ^0, σ^1, y), as opposed to generating only one $(\hat{\sigma}^0, \hat{\sigma}^1, y)$ instance from each (σ^0, σ^1, y), which effectively expands the preference dataset. An in-depth ablation analysis of the data augmentation technique is provided in Section 6.3.

6. Experiment

In this section, we present extensive evaluations on 6 locomotion tasks in DMControl (Tassa et al., 2018) and 3 robotic manipulation tasks in MetaWorld (Yu et al., 2020). Similar to prior works (Christiano et al., 2017; Lee et al., 2021b;a; Park et al., 2022; Liang et al., 2022), we assume the existence of an oracle scripted overseer who provides preferences y on (σ^0, σ^1) based on the cumulative ground truth rewards of each segment defined in the benchmark environments. We evaluate the performance of PbRL algorithms based on these ground truth reward functions.

6.1. Implementation and Experiment Setups

QPA can be incorporated into any off-policy PbRL algorithms. We implement QPA on top of the widely-adopted PbRL backbone framework B-Pref (Lee et al., 2021a).

In our experiments, we demonstrate the efficacy of QPA in comparison to PEBBLE (Lee et al., 2021b) and the SOTA method, SURF (Park et al., 2022). As QPA, PEBBLE, and SURF all employ SAC (Haarnoja et al., 2018) for policy learning, we utilize SAC with ground truth reward as a reference performance upper bound for these approaches. For PEBBLE and SURF, we employ the disagreement query selection scheme in their papers that selects segments with the largest ensemble disagreement of reward models (Lee et al., 2021b; Park et al., 2022; Ibarz et al., 2018; Christiano et al., 2017). To be specific, we train an ensemble of three reward networks $\widehat{r}_\psi$ with varying random initializations and select (σ^0, σ^1) based on the variance of the preference predictor P_ψ. While leveraging an ensemble of reward models for query selection may offer improved robustness and efficacy in complex tasks as observed in (Lee et al., 2021b; Ibarz et al., 2018), the additional computational cost incurred by multiple reward models can be unacceptable in scenarios where the reward model is particularly large, *e.g.,* large language models (LLM). Hence in QPA, we opt to use a **single** reward model and employ near on-policy query selection (simply randomly selects segments from the on-policy buffer $\mathcal{D}^{\text{on}}$). After all feedback is provided and the reward learning phase is complete, we switch from the hybrid experience replay to the commonly used uniform experience replay for policy evaluation in QPA.

In each task, QPA, SURF, and PEBBLE utilize an equal amount of total preference queries and the same feedback frequency for a fair comparison. We perform 10 evaluations on locomotion tasks and 100 evaluations on robotic manipulation tasks across 5 runs every 10^4 environment steps and report the mean (solid line) and 95% confidence interval (shaded regions) of the results, unless otherwise specified. Please see Appendix B for more details.

6.2. Benchmarks Tasks with Scripted Teachers

Locomotion tasks in DMControl suite. DMControl (Tassa et al., 2018) provides diverse high-dimensional locomotion tasks based on MuJoCo physics (Todorov

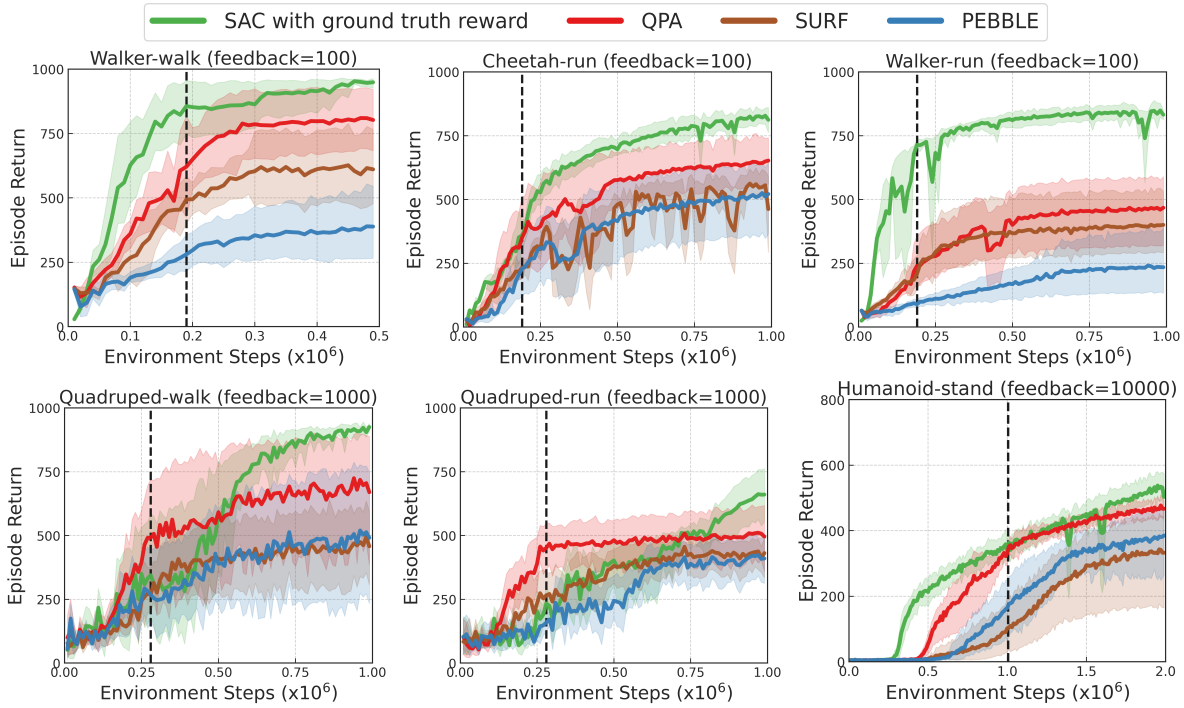


Figure 4: Learning curves on locomotion tasks as measured on the ground truth reward. The dashed black line represents the last feedback collection step.

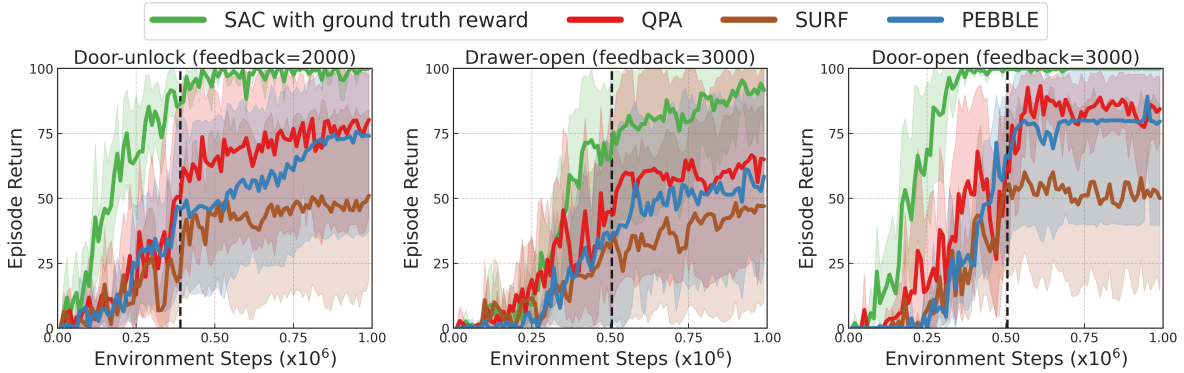


Figure 5: Learning curves on robotic manipulation tasks as measured on the ground truth success rate. The dashed black line represents the last feedback collection step.

et al., 2012). We choose 6 complex tasks in DMControl: *Walker_walk, Walker_run, Cheetah_run, Quadruped_walk, Quadruped_run, Humanoid_stand*. Figure 4 shows the learning curves of SAC (green), QPA (red), SURF (brown), and PEBBLE (blue) on these tasks. As illustrated in the figure, QPA enjoys significantly better feedback efficiency and outperforms SURF and PEBBLE by a substantial margin on all the tasks. To be mentioned, the SOTA PbRL method SURF adopts a pseudo-labeling based semi-supervised learning technique to enhance feedback efficiency. By contrast, QPA removes these complex designs and achieved consistently better performance with a minimalist algorithm. To further demonstrate the feedback efficiency of QPA, we have included additional experimental results in Appendix C, showcasing the performance of these methods under varying total amounts of feedback and different feedback frequencies. Surprisingly, we observe that in some complex tasks (e.g., *Quadruped_walk, Quadruped_run*), QPA can even surpass SAC with ground truth reward during the early training stages, despite experiencing stagnation of performance improvement as feedback provision is halted in the later stages. We provide additional experiment results in Section 6.4 to further elaborate on this phenomenon.

Robotic manipulation tasks in Meta-world. We conduct experiments on 3 complex manipulation tasks in Meta-world (Yu et al., 2020): *Door-unlock, Drawer-open, Door-open*. The learning curves are presented in Figure 5. Similar to prior works (Christiano et al., 2017; Lee et al., 2021b;a; Park et al., 2022; Liang et al., 2022), we employ the ground

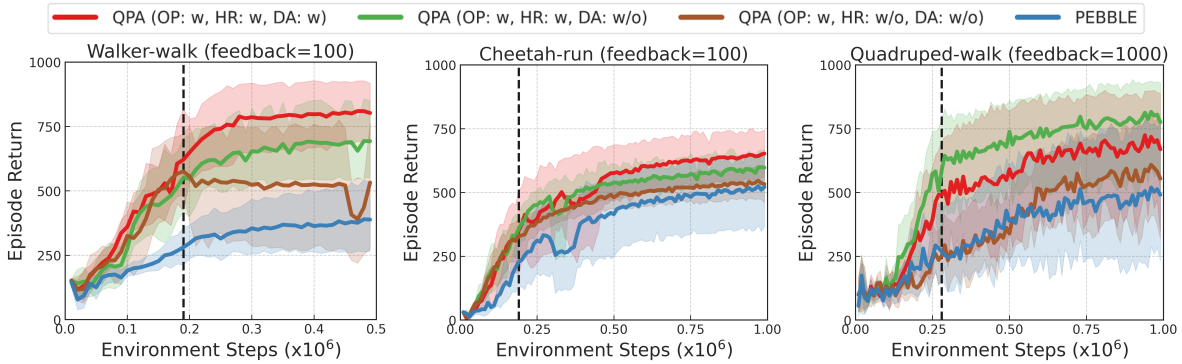
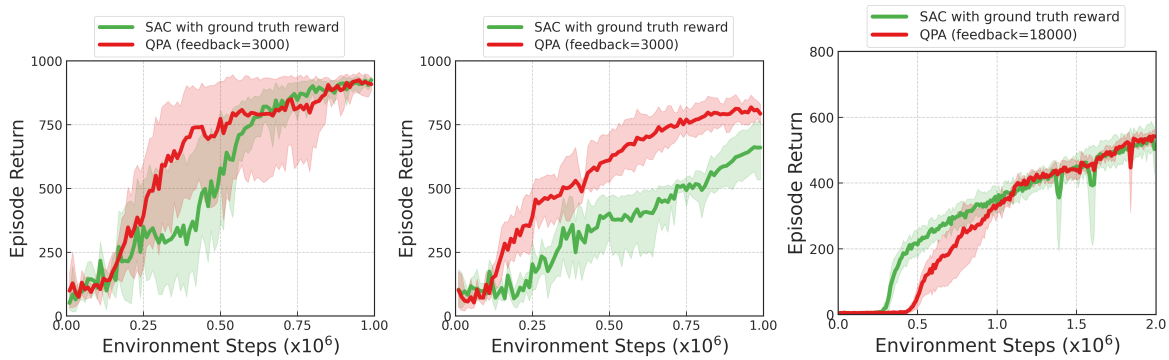


Figure 6: Contribution of each technique in QPA, i.e., near on-policy query (OP), hybrid experience replay (HR), and data augmentation (DA). The dashed black line represents the last feedback collection step.



(a) Quadruped_walk (b) Quadruped_run (c) Humanoid_stand

Figure 7: Learning curves of QPA compared to SAC with ground truth reward given more overseer feedback throughout the entire training process .

truth success rate as a metric to quantify the performance of these methods. Once again, these results provide further evidence that QPA effectively enhances feedback efficiency across a diverse range of complex tasks.

6.3. Ablation Study

To evaluate the impact of each design component in QPA, we incrementally apply near on-policy query (OP), hybrid experience replay (HR), and data augmentation (DA) to the backbone algorithm PEBBLE. Figure 6 confirms that near on-policy query has a positive impact on performance improvement. Moreover, the combination of near on-policy query and experience replay proves to be indispensable for the success of our method. While data augmentation does not always guarantee performance improvement, it tends to enhance performance in most cases. We provide more component analysis results in Appendix C.

QPA also exhibits good hyperparameter robustness and achieves consistent performance improvement over SURF and PEBBLE across various tasks with different parameter values of the query buffer size N and data augmentation ratio τ. We provide detailed ablation results on hyperparameters in Appendix C. In all the tasks presented in Figure 4 and Figure 5, we utilize a data augmentation ratio of $\tau = 20$. For locomotion tasks and the majority of manipulation tasks, we set the size of the near on-policy buffer as $N = 10$.

6.4. Additional Benefit of the Reward Learned by QPA

As observed in Section 6.2, although SAC with ground truth reward is expected to attain higher performance as compared to PbRL algorithms built upon SAC, it is found that QPA can surpass it during the early training stages in certain tasks. To further investigate this phenomenon, we conduct experiments that increase the total amount of feedback and allow the overseer to provide feedback throughout the training process. As illustrated in Figure 7, QPA often exhibits faster learning compared to SAC with ground truth reward in this setting, and in *Quadruped_run* task even achieves higher scores. This phenomenon may be attributed to the ability to **encode the long-term horizon information** of the reward function learned by QPA, which can be more beneficial for the **current policy** to learn and successfully solve the task. Such a property is also uncovered in the motivating example in Section 4. This intriguing and noteworthy phenomenon also highlights the possibility of PbRL methods with learned rewards outperforming RL methods with per-step ground truth rewards. We hope that this observation will inspire further investigations into the essence of learned rewards in PbRL and foster the development of more feedback-efficient PbRL methods in the future.

7. Conclusion and Discussion

This paper addresses a long-neglected issue in existing PbRL studies, namely *query-policy misalignment*, which hinders the existing query selection schemes from effectively improving feedback efficiency. To tackle this issue, we propose a bidirectional query-policy alignment (QPA) approach that incorporates *near on-policy query selection* and *hybrid experience replay*. QPA can be implemented with minimal code modifications on existing PbRL algorithms. Simple yet effective, extensive evaluations on DMControl and Meta-World benchmarks demonstrate substantial gains of QPA in terms of feedback and sample efficiency, highlighting the importance of addressing the *query-policy misalignment* issue in PbRL research. However, note that the *query-policy misalignment* issue is inherently not present in on-policy PbRL methods, as they naturally select on-policy segments to query preferences. These methods, however, suffer from severe sample inefficiency compared to off-policy PbRL methods. In contrast, our QPA approach not only enables sample-efficient off-policy learning, but also achieves high feedback efficiency, presenting a superior solution for the practical implementation of PbRL in real-world scenarios.

References

Abel, D., Dabney, W., Harutyunyan, A., Ho, M. K., Littman, M., Precup, D., and Singh, S. On the expressivity of markov reward. *Advances in Neural Information Processing Systems*, 34:7799–7812, 2021.

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, 2011.

Biyik, E. and Sadigh, D. Batch active preference-based learning of reward functions. In *Conference on robot learning*, pp. 519–528. PMLR, 2018.

Biyik, E., Huynh, N., Kochenderfer, M., and Sadigh, D. Active preference-based gaussian process regression for reward learning. In *Robotics: Science and Systems*, 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.

Bridson, R. Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches*, 10(1):1, 2007.

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.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S. J., and Dragan, A. Inverse reward design. *Advances in neural information processing systems*, 30, 2017.

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.

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020a.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020b.

Lee, D.-H. et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896, 2013.

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. URL https://openreview.net/forum?id=ps95-mkHF_.

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.

Li, J., Hu, X., Xu, H., Liu, J., Zhan, X., Jia, Q.-S., and Zhang, Y.-Q. Mind the gap: Offline policy optimization for imperfect rewards. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WumysvcMvV6.

Liang, X., Shu, K., Lee, K., and Abbeel, P. Reward uncertainty for exploration in preference-based reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=OWZVD-l-ZrC.

Liu, X.-H., Xue, Z., Pang, J., Jiang, S., Xu, F., and Yu, Y. Regret minimization experience replay in off-policy reinforcement learning. *Advances in Neural Information Processing Systems*, 34:17604–17615, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

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*, 2022. URL https://openreview.net/forum?id=TfhfZLQ2EJO.

Pilarski, P. M., Dawson, M. R., Degris, T., Fahimi, F., Carey, J. P., and Sutton, R. S. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *2011 IEEE international conference on rehabilitation robotics*, pp. 1–7. IEEE, 2011.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Rdusseeun, L. and Kaufman, P. Clustering by means of medoids. In *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, volume 31, 1987.

Shin, D., Dragan, A., and Brown, D. S. Benchmarks and algorithms for offline preference-based reward learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=TGuXXlbKsn.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Singh, S., Lewis, R. L., and Barto, A. G. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606. Cognitive Science Society, 2009.

Sinha, S., Song, J., Garg, A., and Ermon, S. Experience replay with likelihood-free importance weights. In *Learning for Dynamics and Control Conference*, pp. 110–123. PMLR, 2022.

Sorg, J. D. *The optimal reward problem: Designing effective reward for bounded agents*. PhD thesis, University of Michigan, 2011.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

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.

Tien, J., He, J. Z.-Y., Erickson, Z., Dragan, A., and Brown, D. S. Causal confusion and reward misidentification in preference-based reward learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Wu, J., Ouyang, L., Ziegler, D. M., Stiennon, N., Lowe, R., Leike, J., and Christiano, P. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

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.

Zhang, S. and Sutton, R. S. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.

A. Proofs

In this section, we present the proof of the theoretical interpretation in Theorem 4.1. Note that we do not aim to provide a tighter bound but rather to offer an insightful theoretical interpretation to *query-policy alignment*.

Given the learned reward function $\widehat{r}_\psi$, the current stochastic policy π and its state-action visitation distribution d^π, we denote $Q^\pi_{\widehat{r}_\psi}$ as the Q-function of π associated with $\widehat{r}_\psi$ and $\hat{Q}^\pi_{\widehat{r}_\psi}$ as the *estimated* Q-function obtained from the policy evaluation step in Eq. (1), which serves as an approximation of $Q^\pi_{\widehat{r}_\psi}$. Q^π_r denotes the Q-function of π with true reward r. Define the distribution-dependent norms $\|f(x)\|_\mu := \mathbb{E}_{x\sim\mu}\left[|f(x)|\right]$. We first recall Theorem 4.1 and then present its proofs.

Theorem 1. *Given the two conditions* $\|\widehat{r}_\psi - r\|_{d^\pi} \leq \epsilon$ *and* $\|Q^\pi_{\widehat{r}_\psi} - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \leq \alpha$*, the value approximation error* $\|Q^\pi_r - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi}$ *is upper bounded as:*

$$\|Q^\pi_r - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \leq \frac{\epsilon}{1-\gamma} + \alpha \tag{6}$$

Proof. By repeatedly applying triangle inequality, we have

$$\begin{aligned}
\|Q^\pi_r - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} &= \|Q^\pi_r - Q^\pi_{\widehat{r}_\psi} + Q^\pi_{\widehat{r}_\psi} - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \\
&\leq \|Q^\pi_r - Q^\pi_{\widehat{r}_\psi}\|_{d^\pi} + \|Q^\pi_{\widehat{r}_\psi} - \hat{Q}^\pi_{\widehat{r}_\psi}\|_{d^\pi} \\
&= \mathbb{E}_{(s,a)\sim d^\pi}\left|Q^\pi_r(s,a) - Q^\pi_{\widehat{r}_\psi}(s,a)\right| + \alpha \\
&= \mathbb{E}_{(s,a)\sim d^\pi}|r(s,a) + \gamma\mathbb{E}_{s'\sim T, a'\sim\pi}Q^\pi_r(s',a') \\
&\qquad - \widehat{r}_\psi(s,a) - \gamma\mathbb{E}_{s'\sim T,a'\sim\pi}Q^\pi_{\widehat{r}_\psi}(s',a')| + \alpha \\
&\leq \epsilon + \gamma\|Q^\pi_r - Q^\pi_{\widehat{r}_\psi}\|_{d^\pi} + \alpha \\
&\leq \epsilon + \gamma\epsilon + \gamma^2\epsilon + \ldots + \gamma^\infty\epsilon + \alpha \\
&= \frac{\epsilon}{1-\gamma} + \alpha
\end{aligned} \tag{7}$$

□

B. Experimental Details

B.1. 2D Navigation Experiment in Section 4

In this section, we present the detailed task descriptions and implementation setups of the motivating example in Section 4.

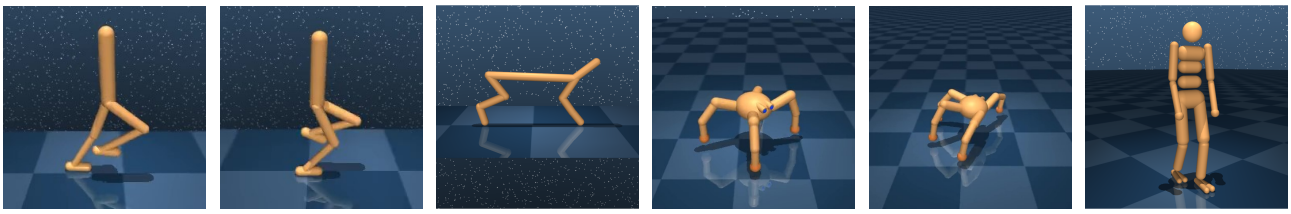
Task description. As illustrated in Figure 2 (a), we consider a 2D continuous space with (x, y) coordinates defined on $[-10, 10]^2$. For each step, the RL agent can move Δx and Δy ranging from $[-1, 1]$. The objective for the agent is to navigate from the starting point $(1, 1)$ to the goal location $(10, 10)$ as quickly as possible. The hand-engineered reward function (ground truth reward in Figure 2 (b)) to provide preferences is defined as the negative distance to the goal, *i.e.*, $r(s,a) = -\sqrt{(x-10)^2 + (y-10)^2}$.

Implementation details. We train PEBBLE (Lee et al., 2021b) using 3 different query selection schemes: *uniform query selection*, *disagreement query selection*, and *near on-policy query selection* (see Section 5.1). In each feedback session, we can obtain one pair of segments to query overseer preferences. The total amount of feedback is set to 8. Each segment contains 5 transition steps. For all schemes, we select the 1st queries using *uniform query selection* according to PEBBLE implementation. After the 1st query selection, we start selecting queries using different selection schemes. The 2nd selected pairs of segments of different schemes are tracked in Figure 3, and the 3rd to 5th selected pairs are tracked in Figure 2 (d).

B.2. DMControl and Meta-world Experiments

B.2.1. Task Descriptions

Locomotion tasks in DMControl suite. DMControl (Tassa et al., 2018) provides diverse high-dimensional locomotion tasks. For our study, we choose 6 complex tasks *Walker_walk, Walker_run, Cheetah_run, Quadruped_walk, Quadruped_run,*

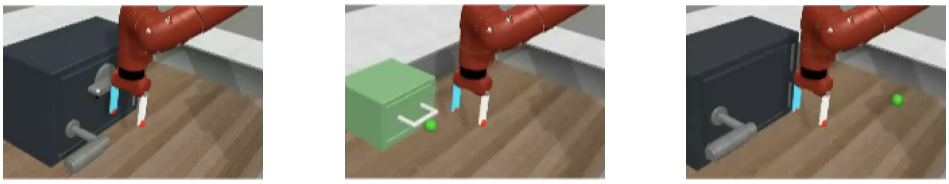


(a) Walker_walk (b) Walker_run (c) Cheetah_run (d) Quadruped_walk (e) Quadruped_run (f) Humanoid_stand

Figure 8: Rendered images of locomotion tasks from DMControl.

Humanoid_stand as depicted in Figure 8. In *Walker_walk* and *Walk_run*, the ground truth reward is a combination of terms encouraging an upright torso and forward velocity. The observation space is 24 dimensional, and the action space is 6 dimensional. In *Cheetah_run*, the ground truth reward is linearly proportional to the forward velocity up to a maximum of 10m/s. The observation space is 17 dimensional, and the action space is 6 dimensional. In *Quadruped_walk* and *Quadruped_run*, the ground truth reward includes the terms encouraging an upright torso and forward velocity. The observation space is 58 dimensional, and the action space is 12 dimensional. In *Humanoid_stand*, the ground truth reward is composed of terms that encourage an upright torso, a high head height, and minimal control. The observation space is 67 dimensional, and the action space is 21 dimensional.

Robotic manipulation tasks in Meta-world. Meta-world (Yu et al., 2020) provides diverse high-dimensional robotic manipulation tasks. For all Meta-world tasks, the observation space is 39 dimensional and the action space is 4 dimensional. For our study, we choose 3 complex tasks *Door_unlock, Drawer_open, Door_open* as depicted in Figure 8. For *Door_unlock*, the goal is to unlock the door by rotating the lock counter-clockwise and the initial door position is randomized. For *Drawer_open*, the goal is to open a drawer and the initial drawer position is randomized. For *Door_open*, the goal is to open a door with a revolving joint and the initial door position is randomized. Please refer to (Yu et al., 2020) for detailed descriptions of the ground truth rewards.



(a) Door_unlock (b) Drawer_open (c) Door_open

Figure 9: Rendered images of robotic manipulation tasks from Meta-world.

B.2.2. Implementation Details

Implementation framework. We implement QPA on top of the widely-adopted PbRL backbone framework B-Pref[1] (Lee et al., 2021a). B-Pref provides a standardized implementation of PEBBLE (Lee et al., 2021b), which can be regarded as the fundamental backbone algorithm for off-policy actor-critic PbRL algorithms. Therefore, similar to prior works (SURF (Park et al., 2022), RUNE (Liang et al., 2022), etc), we opt for PEBBLE as our backbone algorithm as well. B-Pref implements 3 distinct buffers: the RL replay buffer $\mathcal{D}$, the query selection buffer $\mathcal{D}'$ and the preference buffer $\mathcal{D}^\sigma$. $\mathcal{D}$ stores historical trajectories for off-policy RL agent's training. $\mathcal{D}'$ is a copy of $\mathcal{D}$ excluding the predicted reward. It is specifically utilized for segment query selection. $\mathcal{D}^\sigma$ stores the historical feedback (σ^0, σ^1, y) for reward training. Every time the reward model $\widehat{r}_\psi$ is updated, all of the past experience stored in $\mathcal{D}$ is relabeled accordingly. We implement SURF using their officially released code[2], which is also built upon B-Pref.

Query selection scheme. For PEBBLE and SURF, we employ the ensemble disagreement query selection scheme in

[1] https://github.com/rll-research/BPref

[2] https://openreview.net/forum?id=TfhfZLQ2EJO

their papers. To be specific, we train an ensemble of three reward networks $\widehat{r}_\psi$ with varying random initializations. When selecting segments from $\mathcal{D}'$, we firstly uniformly sample a large $\{(\sigma^0, \sigma^1)\}$ batch from $\mathcal{D}'$. Subsequently, we choose (σ^0, σ^1) from this batch based on the highest variance of the preference predictor P_ψ. For QPA, we simply reduce the size of $\mathcal{D}'$. This minimal modification ensures that the first-in-first-out buffer $\mathcal{D}'$ exclusively stores the most recent trajectories, effectively emulating the characteristics of the near on-policy buffer $\mathcal{D}^{\text{on}}$. In QPA, instead of utilizing ensemble disagreement query based on multiple reward models, we use a **single** reward model and employ near on-policy query selection (simply randomly selects segments from the near on-policy buffer $\mathcal{D}^{\text{on}}$). This approach significantly reduces the computational cost compared to ensemble disagreement query selection, particularly in scenarios where the reward model is notably large, such as large language models (LLMs). Although leveraging an ensemble of reward models for query selection may offer improved robustness and efficacy in complex tasks as observed in (Lee et al., 2021b; Ibarz et al., 2018), we showcase that QPA, employing the simple near on-policy query selection, can substantially outperform SURF and PEBBLE with ensemble disagreement query selection.

Data augmentation. In the officially released code of SURF, they implement the temporal data augmentation, which generates one $(\hat{\sigma}^0, \hat{\sigma}^1, y)$ instance from each (σ^0, σ^1, y) pair. In contrast to their implementation, in QPA, we generate multiple $(\hat{\sigma}^0, \hat{\sigma}^1, y)$ instances from a single (σ^0, σ^1, y) pair, which effectively expands the preference dataset. Specifically, we sample multiple pairs of snippets $(\hat{\sigma}^0, \hat{\sigma}^1)$ from the queried segments (σ^0, σ^1, y). These pairs of snippets $(\hat{\sigma}^0, \hat{\sigma}^1)$ consist of sequences of observations and actions, but they are shorter than the corresponding segments (σ^0, σ^1). In each pair of snippets $(\hat{\sigma}^0, \hat{\sigma}^1)$, $\hat{\sigma}^0$ has the same length as $\hat{\sigma}^1$, but they have different initial states.

Hyperparameter setting. QPA, PEBBLE, and SURF all employ SAC (Soft Actor-Critic) (Haarnoja et al., 2018) for policy learning and share the same hyperparameters of SAC. We provide the full list of hyperparameters of SAC in Table 1. Both QPA and SURF utilize PEBBLE as the off-policy PbRL backbone algorithm and share the same hyperparameters of PEBBLE as listed in Table 2. The additional hyperparameters of SURF based on PEBBLE are set according to their paper and are listed in Table 3. The additional hyperparameters of QPA are presented in Table 4.

Table 1: Hyperparameters of SAC

Hyperparameter	Value	Hyperparameter	Value
Discount	0.99	Critic target update freq	2
Init temperature	0.1	Critic EMA	0.005
Alpha learning rate	1e-4	Actor learning rate	5e-4 (Walker_walk, Cheetah_run, Walker_run) 1e-4 (Other tasks)
Critic learning rate	5e-4 (Walker_walk, Cheetah_run, Walker_run) 1e-4 (Other tasks)	Actor hidden dim	1024
Critic hidden dim	1024	Actor hidden layers	2
Critic hidden layers	2	Actor activation function	ReLU
Critic activation function	ReLU	Optimizer	Adam
Bacth size	1024		

C. More Experimental Results

In this section, we present more experimental results. For each task, we perform 10 evaluations across 5 runs every 10^4 environment steps and report the mean (solid line) and 95% confidence interval (shaded regions) of the results, unless otherwise specified.

C.1. Results under Varying Total Feedback and Feedback Frequencies

To further demonstrate the feedback efficiency of QPA, we compare its performance with that of SURF and PEBBLE under varying total feedback and different feedback frequencies on certain locomotion tasks. As illustrated in Figure 10-13, QPA (red) consistently outperforms SURF (brown) and PEBBLE (blue) across a wide range of total feedback quantities and feedback frequencies.

Table 2: Hyperparameters of PEBBLE

Hyperparameter	Value
Length of segment	50
Unsupervised pre-training steps	9000
Total feedback	100 (Walker_walk, Cheetah_run, Walker_run) 1000 (Quadruped_walk, Quadruped_walk) 2000 (Door_unlock) 3000 (Drawer_open, Door_open) 10000 (Humanoid_stand)
Frequency of feedback	5000 (Humanoid_stand, Drawer_open, Door_open) 20000 (Walker_walk, Cheetah_run, Walker_run, Door_unlock) 30000 (Quadruped_walk, Quadruped_walk)
# of queries per session	10 (Walker_walk, Cheetah_run, Walker_run) 30 (Drawer_open, Door_open) 50 (Humanoid_stand) 100 (Quadruped_walk, Quadruped_walk, Door_unlock)
Size of query selection buffer	100

Table 3: Additional hyperparameters of SURF

Hyperparameter	Value
Unlabeled batch ratio	4
Threshold	0.999 (Cheetah_run), 0.99 (Other tasks)
Loss weight	1
Min/Max length of cropped segment	[45, 55]
Segment length before cropping	60

Table 4: Additional hyperparameters of QPA

Hyperparameter	Value
Size of near on-policy buffer N	30 (Drawer_open), 10 (Other tasks)
Data augmentation ratio τ	20
Min/Max length of subsampled snippets	[35, 45]

C.2. Additional Ablation Study

We emphasize that the outstanding performance of QPA, as demonstrated in Section 6, was **not** achieved by meticulously selecting QPA's hyperparameters. On the contrary, as indicated in Table 4, we consistently use a data augmentation ratio of $\tau = 20$ across all tasks and use a near on-policy buffer size of $N = 10$ in the majority of tasks. To delve deeper into the impact of hyperparameters on QPA's performance, we conduct an extensive ablation study across a range of tasks. The total amount of feedback and feedback frequency remain unchanged from Section 6.

Effect of data augmentation ratio τ. The data augmentation ratio, denoted as τ, represents the number of instances $(\hat{\sigma}^0, \hat{\sigma}^1, y)$ generated from a single pair (σ^0, σ^1, y). To explore the impact of the data augmentation ratio on QPA's performance, we evaluate the performance of QPA under different data augmentation ratios $\tau \in \{0, 10, 20, 100\}$. Figure 14 illustrates that QPA consistently demonstrates superior performance across a diverse range of data augmentation ratios. While a larger data augmentation ratio does not necessarily guarantee improved performance, it is worth noting that $\tau = 20$ is generally a favorable choice in most cases.

Effect of near on-policy buffer size N. The size of the on-policy buffer, denoted as N, signifies the number of the recent trajectories stored in the first-in-first-out buffer $\mathcal{D}^{\text{on}}$. A larger N implies that the near on-policy buffer $\mathcal{D}^{\text{on}}$ contain additional

historical trajectories generated by a past policy π' that may be significantly different from the current policy π, potentially deviating from the main idea of *near on-policy query selection*. Therefore, it is reasonable for QPA to opt for a smaller value of N. To further investigate how the on-policy buffer size affects QPA's performance, we evaluate the performance of QPA under different on-policy buffer sizes $N \in \{5, 10, 50\}$. Figure 15 demonstrates QPA consistently showcases superior performance, particularly when using a smaller value for N.

In *Quadruped_run*, it is observed that QPA with $N = 50$ does not exhibit improved performance. This outcome can be attributed to the fact that the larger value of N compromises the essence of near "on-policy" within $\mathcal{D}^{\text{on}}$. This once again highlights the significance of the *near on-policy query selection* principle. A smaller value of N may not always result in a significant improvement in performance. This could be because the pair of segments selected from a very small near on-policy buffer is more likely to be similar to each other, resulting in less informative queries. Overall, it is generally considered favorable to select $N = 10$ for achieving better performance.

Additional component analysis. To further evaluate the impact of each design component in QPA, we provide more component analysis results in Figure 16. Based on the results presented in Section 6.3, we conduct a supplementary experiment where we solely employ the *hybrid experience replay* (HR) component in QPA. It is worth noting that in certain tasks, using only the *hybrid experience replay* technique can result in slightly improved performance. This could be attributed to the advantage of on-policyness (Liu et al., 2021; Sinha et al., 2022) that is facilitated by *hybrid experience replay*. Figure 16 confirms that the combination of *near on-policy query selection* and *hybrid experience replay* is essential to QPA's success.

D. Human Experiments

In this section, we compare the agent trained with real human preferences (provided by the authors) to the agent trained with hand-engineered preferences (provided by the hand-engineered reward function in Appendix B.2.1) on the *Cheetah_run* task. We report the training results in Figure 17.

Avoiding reward exploitation via human preferences. As shown in Figure 17, the agent trained with real human preferences exhibits more natural behavior, while the agent trained with hand-engineered preferences often behaves more aggressively and may even roll over. This is because the hand-engineered reward function is based solely on the linear proportion of forward velocity, without fully considering the agent's posture. Consequently, it can be easily exploited by the RL agent. Take Figure 18 as an example: when comparing the behaviors of "Stand still" and "Recline", humans would typically not prefer the latter, as it clearly contradicts the desired behavior of "running forward". However, in our experiments, we observe that the hand-engineered overseer would favor "Recline" over "Stand still", as the hand-engineered preferences only consider the forward velocity and overlook the fact that the agent is lying down while still maintaining some forward velocity. This highlights the advantages of RL with real human feedback over standard RL that training with hand-engineered rewards.

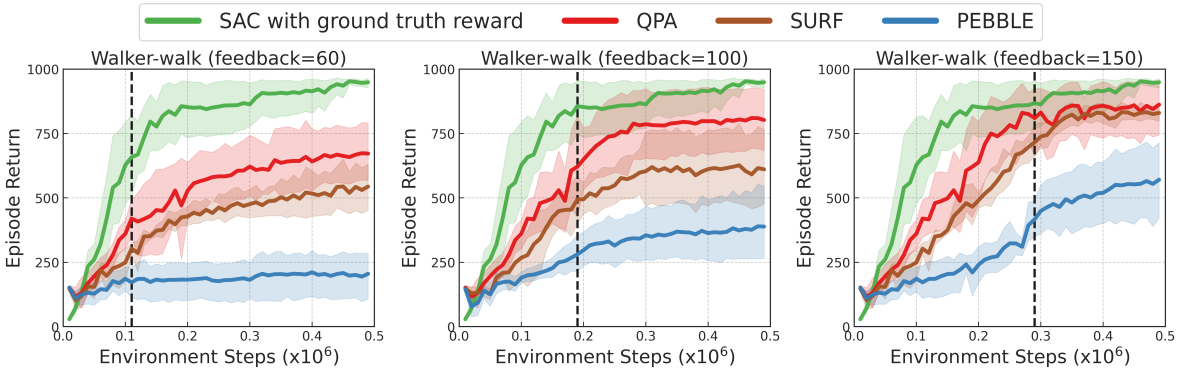


Figure 10: Learning curves on *Walker_walk* under varying total amounts of feedback $\{60, 100, 150\}$. The dashed black line represents the last feedback collection step.

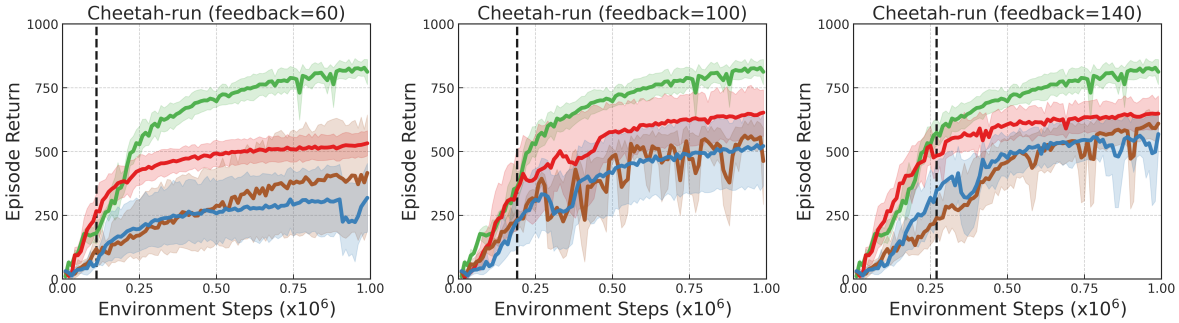


Figure 11: Learning curves on *Cheetah_run* under varying total amounts of feedback $\{60, 100, 140\}$. The dashed black line represents the last feedback collection step.

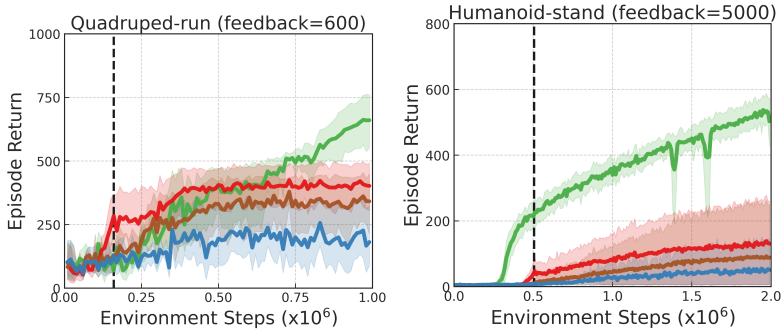


Figure 12: Learning curves on other tasks under different total amounts of feedback. The dashed black line represents the last feedback collection step.

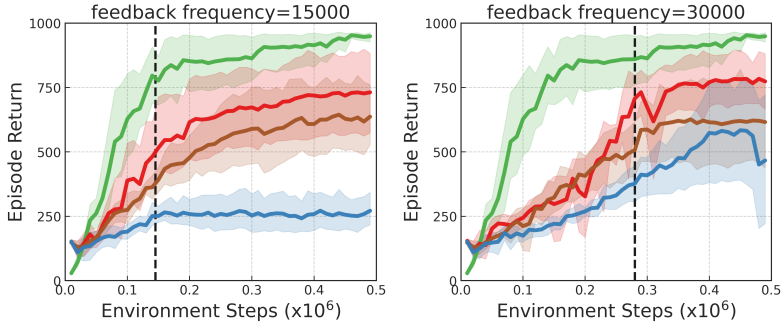
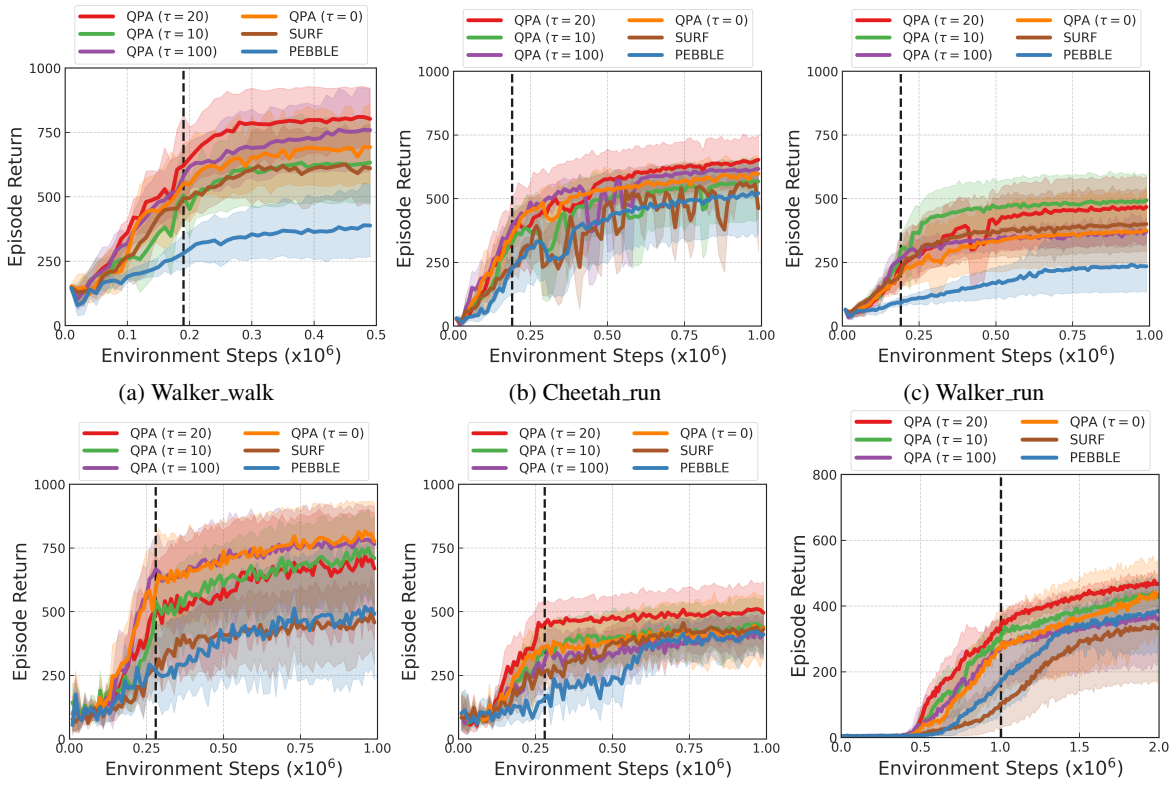


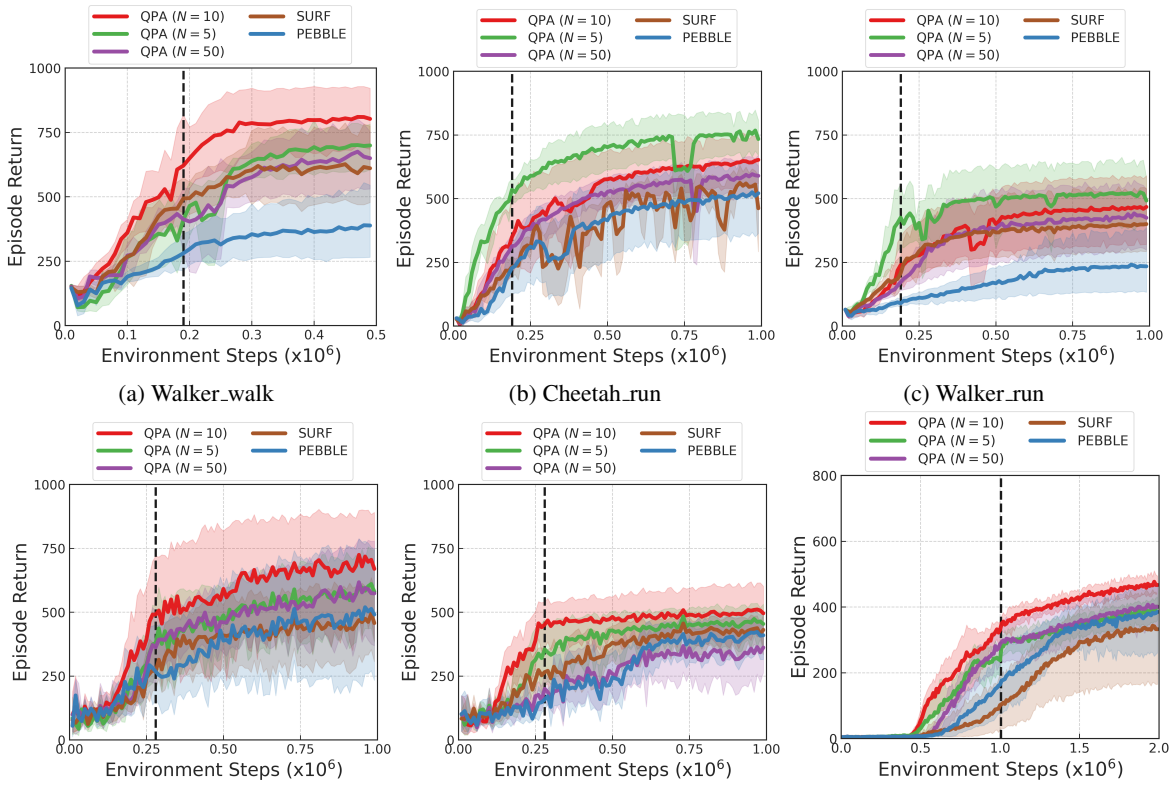
Figure 13: Learning curves on *Walker_walk* under varying feedback frequencies. The dashed black line represents the last feedback collection step.



(a) Walker_walk (b) Cheetah_run (c) Walker_run

(d) Quadruped_walk (e) Quadruped_run (f) Humanoid_stand

Figure 14: Learning curves on locomotion tasks under different data augmentation ratios $\tau \in \{0, 10, 20, 100\}$ of QPA. The dashed black line represents the last feedback collection step.



(a) Walker_walk (b) Cheetah_run (c) Walker_run

(d) Quadruped_walk (e) Quadruped_run (f) Humanoid_stand

Figure 15: Learning curves on locomotion tasks under different sizes of near on-policy buffer $N \in \{5, 10, 50\}$ of QPA. The dashed black line represents the last feedback collection step.

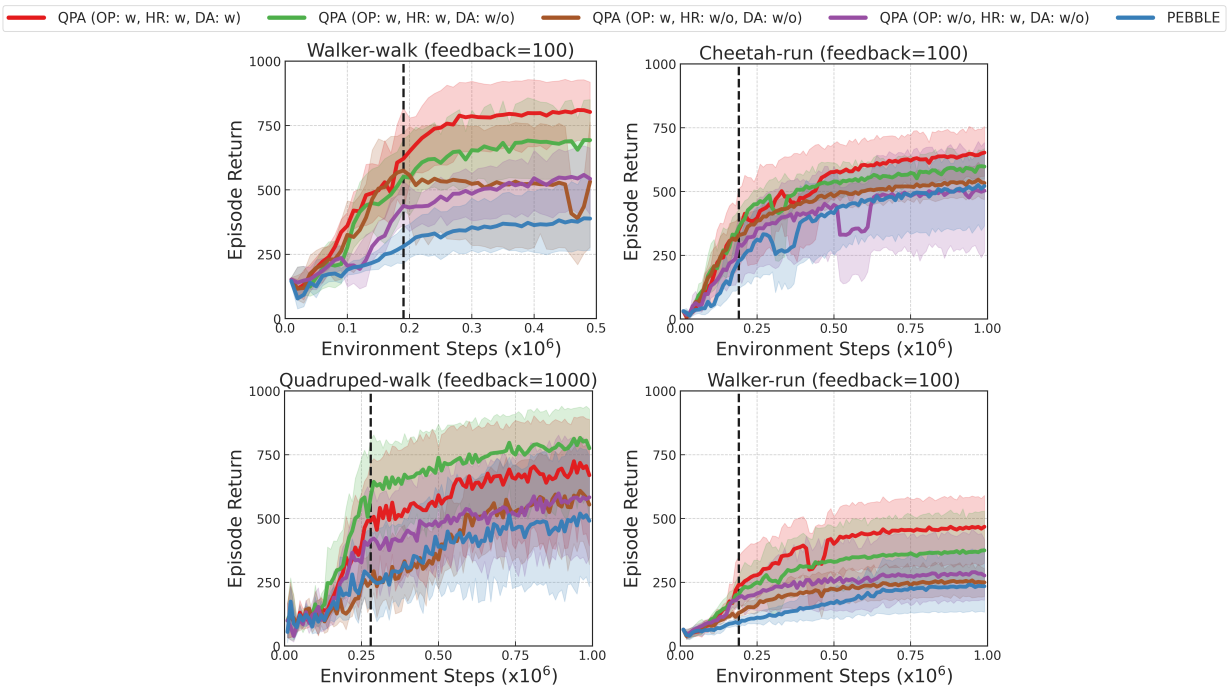
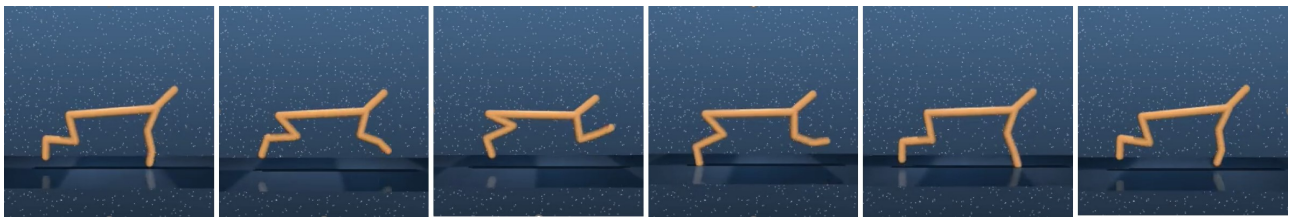
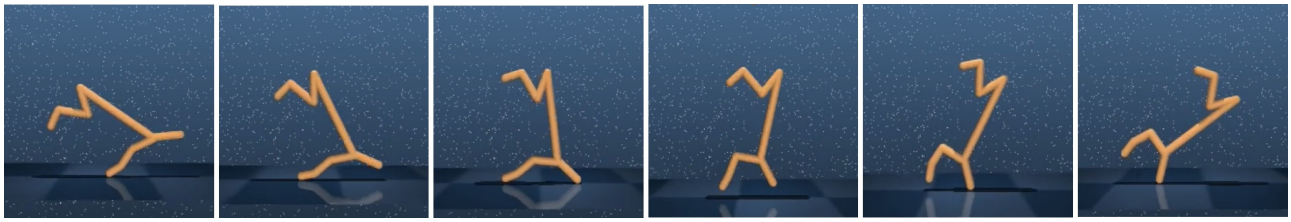


Figure 16: Contribution of each technique in QPA, i.e., near on-policy query (OP), hybrid experience replay (HR), and data augmentation (DA). The dashed black line represents the last feedback collection step.

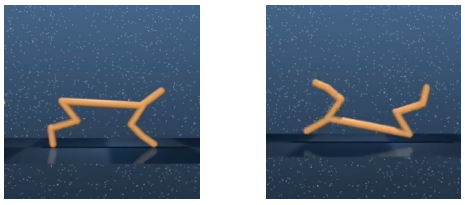


(a) Agent trained with human preferences



(b) Agent trained with hand-engineered preferences

Figure 17: Human experiments on *Cheetah_run* task.



(a) Stand still (b) Recline

Figure 18: A pair of segments.