Reinforcement Learning from Bagged Reward

Anonymous authors
Paper under double-blind review

Abstract

In Reinforcement Learning (RL), it is commonly assumed that an immediate reward signal is generated for each action taken by the agent, helping the agent maximize cumulative rewards to obtain the optimal policy. However, in many real-world scenarios, designing immediate reward signals is difficult; instead, agents receive a single reward that is contingent upon a partial sequence or a complete trajectory. In this work, we define this challenging problem as *RL from Bagged Reward* (RLBR), where sequences of data are treated as *bags* with non-Markovian bagged rewards, leading to the formulation of Bagged Reward Markov Decision Processes (BRMDPs). Theoretically, we demonstrate that RLBR can be addressed by solving a standard MDP with properly redistributed bagged rewards allocated to each instance within a bag. Empirically, we find that reward redistribution becomes more challenging as the bag length increases, due to reduced informational granularity. Existing reward redistribution methods are insufficient to address these challenges. Therefore, we propose a novel reward redistribution method equipped with a bidirectional attention mechanism, enabling the accurate interpretation of contextual nuances and temporal dependencies within each bag. We experimentally demonstrate that our proposed method consistently outperforms existing approaches. The code is available at an anonymous link: `https://anonymous.4open.science/r/RLBR-F66E/`.

1 Introduction

Reinforcement Learning (RL) has achieved remarkable success in various domains, including autonomous driving (Kiran et al., 2021), healthcare (Yu et al., 2021), complex game playing (Silver et al., 2016; Wurman et al., 2022), and financial trading (Yang et al., 2020). One common and essential assumption for most RL algorithms is the availability of immediate reward feedback at each time step of the decision-making process. However, designing such immediate reward feedback is quite difficult and may mislead the policy learning in many real-world applications (Kwon et al., 2023; Lee et al., 2023). Recognizing this gap, numerous studies (Watkins, 1989; Ke et al., 2018; Arjona-Medina et al., 2019; Gangwani et al., 2020; Ren et al., 2021; Zhang et al., 2023) have explored the concept of RL with Trajectory Feedback (RLTF), primarily focusing on trajectory feedback where rewards are allocated at the end of a sequence.

However, real-world applications often feature complex non-immediate reward structures that traditional RLTF cannot fully capture. For instance, in autonomous driving (see Fig.1), rewards are typically linked to completing specific tasks by sequences of actions rather than individual actions or just the final objective (Early et al., 2022; Gaon & Brafman, 2020). Similarly, in mobile health interventions (Gao et al., 2024), the reward is defined by sustained daily commitment to being active throughout the intervention period, with multiple activity suggestions each day influencing this outcome. In such scenarios, providing rewards for every action is impractical, while focusing only on the final goal overlooks important aspects of the decision-making process. Previous studies have explored learning desirable policies using immediate rewards(Sutton & Barto, 1998; Vinyals et al., 2019) or trajectory-based feedback (Arjona-Medina et al., 2019; Gangwani et al., 2020; Ren et al., 2021) are not ideal methods for such scenarios.

To address these challenges, we introduce *RL from Bagged Rewards* (RLBR), a framework that better aligns with real-world scenarios by considering the cumulative effect of a series of instances. In RLBR, sequences of instances are defined as *bags*, each associated with a *bagged reward*. This framework encompasses both the

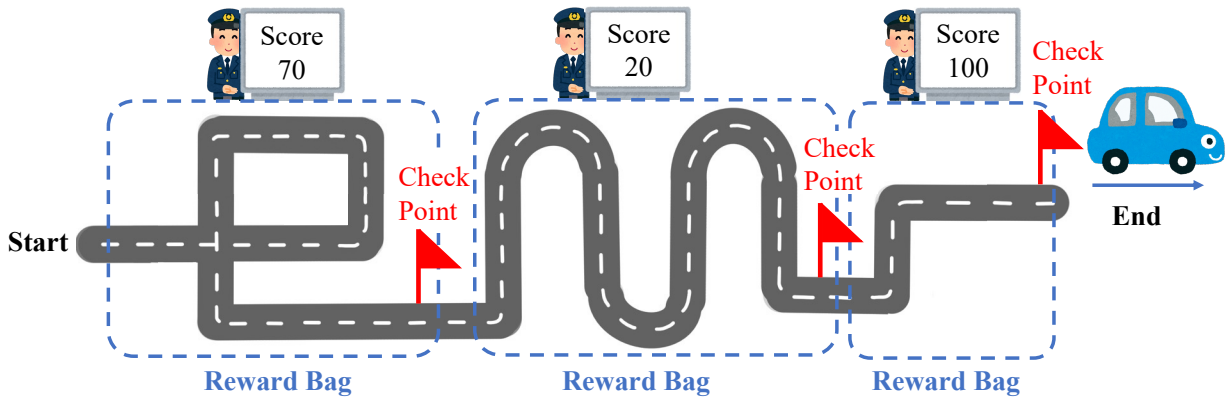

Figure 1: An example of the reward bag structure on an autonomous driving trajectory. Each segment of the driving sequence is evaluated and assigned a score by an evaluator.

traditional RL setting, where each bag contains only a single instance, and the trajectory feedback setting, where a bag spans the entire trajectory, as special cases. Furthermore, RLBR offers the potential to reduce the labeling workload by lessening the frequency of reward annotations. However, this benefit is balanced by increased learning complexity due to the reduced granularity of information.

In the RLBR framework, our focus is on leveraging bagged reward information to discern the significance of each instance within a bag and to understand the relationships among different bags. The challenge lies in accurately interpreting the contextual nuances within individual bags, as the instances within a bag would be time-dependent on each other and their contributions to the bagged reward vary. Given the importance of contexts in RLBR, we turn to the bidirectional attention mechanism (Seo et al., 2016; Vaswani et al., 2017; Devlin et al., 2019), renowned for its effectiveness in contextual understanding, especially for time-dependent data. Specifically, we propose a Transformer-based reward model, leveraging the bidirectional attention mechanism to adeptly interpret contexts within a bag and allocate rewards to each instance accurately. This model can be utilized to enhance general RL algorithms, such as Soft Actor-Critic (SAC) (Haarnoja et al., 2018), for environments with bagged rewards.

In summary, our contributions are threefold:

1. We introduce the RL from Bagged Rewards (RLBR) framework and formalize it as Bagged Reward MDPs (BRMDPs), bridging the gap between instance-level and trajectory-level rewards and establishing a theoretical foundation for reward redistribution in non-Markovian settings (Section 3).
2. We establish a theoretical connection between MDPs after reward redistribution and the original BRMDPs, providing a solid foundation for policy learning under bagged rewards. Building on this, we propose a Transformer-based reward model with a bidirectional attention mechanism that effectively captures temporal and contextual dependencies, combined with an alternating optimization algorithm to jointly enhance reward redistribution and policy learning (Section 4).
3. We demonstrate through extensive experiments our method not only outperforms existing approaches but also maintains robust performance as bag length increases, highlighting its ability to mimic ground truth reward distributions while exhibiting contextual understanding and adaptability to environmental dynamics (Section 5).

2 Related Works

With the growing attention on RLTF, where trajectory feedback is often considered the sum of rewards over the entire trajectory, new methods specifically designed for RLTF have emerged. Return Decomposition for Delayed Rewards (RUDDER) (Arjona-Medina et al., 2019) used a return-equivalent formulation for precise

credit assignment, with advancements incorporating expert demonstrations and language models (Liu et al., 2019; Widrich et al., 2021; Patil et al., 2022). Both Iterative Relative Credit Refinement (IRCR) (Gangwani et al., 2020) and Randomized Return Decomposition (RRD) (Ren et al., 2021) assumed that each state-action pair contributes equally to the trajectory feedback. Specifically, IRCR presented a uniform reward redistribution model, while RRD proposed a novel upper bound for return-equivalent assumptions, integrating return decomposition with uniform reward redistribution. Additionally, Han et al. (2022) proposed modifying RL algorithms to incorporate sequence-level information, enabling agents to learn from broader structures and long-term outcomes. On the other hand, some studies on sparse rewards, such as reward shaping (Ng et al., 1999; Tambwekar et al., 2019; Hu et al., 2020) and intrinsic reward methods (Sorg et al., 2010; Pathak et al., 2017; Zheng et al., 2018; 2021), could be directly applied to RLTF. However, these approaches primarily focused on addressing the exploration-exploitation trade-off in sparse reward settings, which made them less ideal for solving RLTF problems. A detailed discussion on the sparse reward settings and methods is provided in Appendix B.

However, previous RLTF works that assume each state-action pair contributes equally to the trajectory feedback did not consider that instances contributing more to the feedback should receive greater attention. On the other hand, methods that consider sequence information struggled to capture effective information from long trajectories, resulting in suboptimal training outcomes, as evidenced by prior studies (Ren et al., 2021; Zhang et al., 2023). Building on previous methods, we adopt a reward redistribution learning strategy to enhance policy learning in the context of reward bags. The proposed method aims to extract information from bag-level rewards, taking into account the contribution of each instance, ultimately learning a reliable and well-performing policy. Experimental results (see Section 5) demonstrate that the proposed method can accurately capture information from long sequences, and the rich information in the bags further enhances its effectiveness.

3 Reinforcement Learning from Bagged Reward

In this section, we first provide preliminaries on RL with immediate rewards and trajectory feedback, and then formulate the RLBR problem with an extension of the traditional MDPs, the BRMDPs.

3.1 Preliminaries

In traditional RL problems, which are modeled using MDPs, each state-action pair is promptly associated with a reward (Sutton & Barto, 1998). This paradigm is encapsulated in a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \mu)$, with $\mathcal{S} \ni s$ and $\mathcal{A} \ni a$ as finite sets of states and actions, $P = p(s'|s, a)$ as a state transition probability function to state s' when action a is taken at state s, $r(s, a)$ as the immediate reward function when action a is taken at state s, and $\mu = p(s)$ as the initial state distribution. The primary objective in this framework is to discover a policy $\pi = p(a|s)$, which is a conditional probability of taking action a given state s, that maximizes the cumulative sum of rewards over a horizon length T:

$$J(\pi) = \mathbb{E}_{\pi,P,\mu}\Big[\sum_{t=0}^{T-1} r(s_t, a_t)\Big]. \tag{1}$$

Given that associating each state-action pair with a reward is challenging in real-world scenarios and involves significant labeling costs, RLTF, also known as episodic or delayed rewards in some works (Watkins, 1989; Arjona-Medina et al., 2019; Zhang et al., 2023), has become increasingly prominent in many applications (Agarwal et al., 2021; Chen et al., 2024). Distinct from traditional RL, RLTF offers only one feedback after a complete trajectory (Arjona-Medina et al., 2019; Zhang et al., 2023). A trajectory $\tau = \{(s_0, a_0), (s_1, a_1), \ldots, (s_{T-1}, a_{T-1})\}$ consists of T state-action pairs, with a cumulative reward $R_{\text{traj}}(\tau)$ that is the sum of latent immediate rewards $\sum_{t=0}^{T-1} r(s_t, a_t)$, observable only at the end. Denoting by $\mathcal{T}(\pi)$ as the distribution of trajectories induced by π, P, μ, the learning objective in RLTF is to maximize the expected trajectory-based reward:

$$J_{\text{traj}}(\pi) = \mathbb{E}_{\mathcal{T}(\pi)}\big[R_{\text{traj}}(\tau)\big]. \tag{2}$$

3.2 Problem Formulation

We formulate the RLBR problem as an extension of the traditional MDP, referred to as the BRMDP, which features a reward granularity that lies between the two aforementioned settings. First, we define the notion of *bags*, which are sub-pieces of complete trajectories. A trajectory τ is divided into several neighboring bags, and a bag of size n_i, which starts from time i, is defined as $B_{i,n_i} = \{(s_i, a_i), \ldots, (s_{i+n_i-1}, a_{i+n_i-1})\}, 0 \leq i \leq i + n_i - 1 \leq T - 1$. Afterward, we define the BRMDP to navigate the complexities of the aggregated non-Markovian reward signals:

Definition 1 (BRMDP)**.** *A BRMDP is defined by the tuple* $(\mathcal{S}, \mathcal{A}, P, R, \mu)$*, where*

- $\mathcal{S}$ *and* $\mathcal{A}$ *are finite sets of states and actions.*
- P *is the state transition probability function.*
- R *denotes the bagged reward function, which defines the reward over a bag:* $R(B_{i,n_i})$*.*
- μ *represents the initial state distribution.*

In the RLBR framework, a bag B_{i,n_i} metaphorically a unified reward unit for a contiguous sequence of state-action pairs. A trajectory τ is a composite of a set of bags, denoted as $\mathcal{B}_\tau$, which ensures that each trajectory includes at least one reward bag. We further assume a bag partition function defined by the environment: $\mathcal{G} : \tau \mapsto \mathcal{B}_\tau$, which is a task-dependent function for generating bags given an input trajectory. Consequently, the learning objective of the policy is to maximize the accumulated bagged rewards:

$$J_{\mathrm{B}}(\pi) = \mathbb{E}_{\mathcal{T}(\pi)}\left[\sum_{B \in \mathcal{B}_\tau} R(B) \middle| \mathcal{G}\right]. \tag{3}$$

Notably, when each bag consists a single instance ($n_i = 1$ for $i = 0, 1, \ldots, T-1$), RLBR simplifies to traditional RL. Conversely, when a single bag covers the entire trajectory ($n_0 = T$), RLBR reduces to RLTF. This adaptability highlights the flexibility of the RLBR framework in accommodating various reward structure scenarios.

4 Reward Redistribution for RLBR

This section delves into the proposed reward redistribution method for RLBR. We aim for the reward redistribution to meet the following criteria: first, the policies learned from the MDP after reward redistribution should be consistent with those learned from the BRMDP; second, the redistribution process should reflect each instance's contribution based on the sequence information. The subsequent subsections are structured to first theoretically prove the rationality of reward redistribution, followed by proposing the use of a bidirectional attention mechanism for this redistribution. Finally, we outline a comprehensive algorithm for developing efficient policies within the BRMDP framework.

4.1 Equivalence of Optimal Policies in the BRMDP and the MDP after Reward Redistribution

In the BRMDP, agents do not have direct access to the immediate rewards associated with each instance. To address this, we adopt a common approach used in trajectory feedback scenarios, modeling the cumulative sum form as the reward redistribution strategy to convert bagged rewards into instance-level rewards (Ren et al., 2021; Arjona-Medina et al., 2019; Zhang et al., 2023). Specifically, we define $R(B_{i,n_i}) = \sum_{t=i}^{i+n_i-1} \hat{r}(s_t, a_t)$, where $\hat{r}(s_t, a_t)$ represents the redistributed reward. However, a critical question arises: How can we ensure that the policy learned with redistributed rewards performs well in the original BRMDP? To address this, we provide the following theorem:

Theorem 1. *Consider a BRMDP where only the sequence-level bagged reward is available. Suppose the bagged reward is redistributed across instances while preserving its total sum within each bag. Under this assumption, the set of optimal policies* Π *learned in the resulting MDP is identical to the set of optimal policies* Π_{B} *in the BRMDP, meaning that* $\Pi = \Pi_{\mathrm{B}}$.

Proof. Over a complete trajectory τ, the cumulative reward in BRMDP can be expressed as the sum of the rewards from all the bags along the trajectory:

$$\sum_{B \in \mathcal{B}_\tau} R(B) = \sum_{i \in \mathcal{I}_\tau} R(B_{i,n_i}) = \sum_{i \in \mathcal{I}_\tau} \left(\sum_{t=i}^{i+n_i-1} \hat{r}(s_t, a_t) \right) = \sum_{t=0}^{T-1} \hat{r}(s_t, a_t),$$

where $\mathcal{I}_\tau$ denotes the set of initial timestep indices for $B \in \mathcal{B}_\tau$.

The policy optimization objective in BRMDP, $J_{\mathrm{B}}(\pi)$, aims to maximize the expected sum of bagged rewards along the trajectory from $t = 0$, which is equivalent to maximizing the cumulative reward in redistributed reward MDP, $J(\pi)$:

$$J_{\mathrm{B}}(\pi) = \mathbb{E}_{\mathcal{T}(\pi)} \left[\sum_{B \in \mathcal{B}_\tau} R(B) \middle| \mathcal{G} \right] = \mathbb{E}_{\pi, P, \mu} \left[\sum_{t=0}^{T-1} \hat{r}(s_t, a_t) \right] = J(\pi).$$

Given that the expected cumulative rewards for any policy in the original BRMDP and redistributed reward MDP frameworks are equivalent, under the condition of infinite exploration or exhaustive sampling within the state-action space, the sets of optimal policies for each framework also coincide, implying that $\Pi = \Pi_{\mathrm{B}}$.

□

Under this theorem, directly using bagged rewards or applying an average reward redistribution strategy can, in theory, lead to an optimal policy. However, directly using bagged rewards introduces delayed feedback, which slows down policy convergence, as demonstrated in Arjona-Medina et al. (2019). Meanwhile, average reward redistribution assumes equal contributions from all instances within a bag, potentially diluting critical decision points and consequently affecting policy learning (Gangwani et al., 2020; Ren et al., 2021). Therefore, an effective reward redistribution method should capture the actual contributions of individual state-action pairs within each bag. By assigning rewards in a way that reflects their true impact on bagged rewards, we can enhance credit assignment accuracy and improve the quality of policy optimization. By this theorem, we ensure that optimizing a policy on an MDP after rewards redistribution is equivalent to optimizing on the original BRMDP. Below, we introduce our specific reward redistribution approach.

4.2 Reward Redistribution based on Bidirectional Attention Mechanism

Building on Theorem 1, which establishes that optimal policies in the BRMDP and the MDP after reward redistribution are equivalent, our focus shifts to the crucial process of redistributing rewards within a bag. To capture the contextual influence of each instance within the sequence, the Causal Transformer (Vaswani et al., 2017) naturally serves as a suitable sequential prediction model. Traditionally, Transformers in RL have been used in a unidirectional manner (Chen et al., 2021; Janner et al., 2021; Micheli et al., 2023), where only previous instances influence the current prediction due to the unobservability of future instances. However, since our bagged rewards are generally non-Markovian, both preceding and subsequent instances can influence the contribution of the current instance to the bagged reward. Therefore, understanding the relationships among instances within a bag is therefore pivotal. This insight leads us to employ a *bidirectional* attention mechanism (Seo et al., 2016; Vaswani et al., 2017; Devlin et al., 2019) to capture these relationships. This mechanism connects both past and future instances within a bag, enabling a more comprehensive understanding of contextual influences. By quantitatively evaluating the contribution of each instance, the bidirectional attention mechanism facilitates nuanced reward redistribution, as demonstrated experimentally in Section 5.

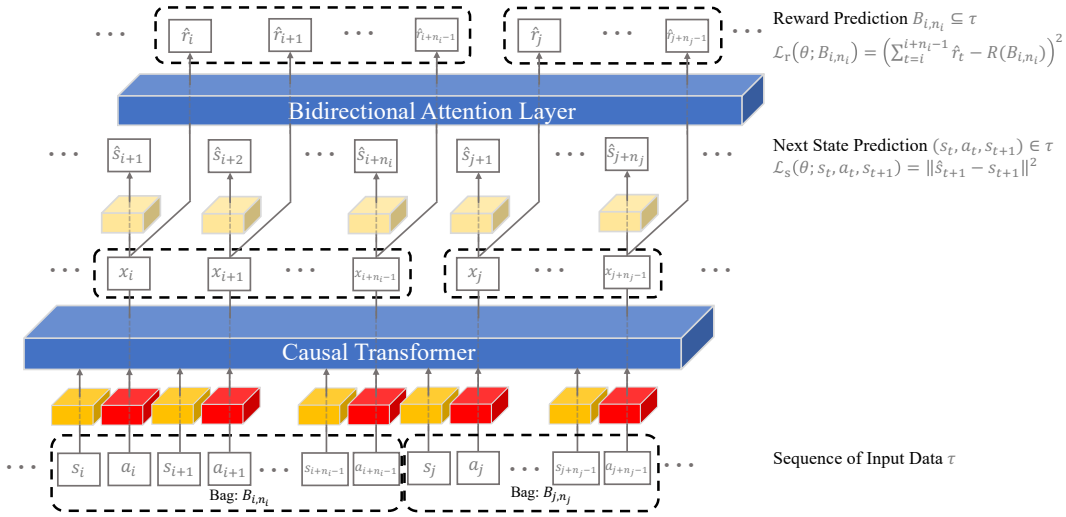

Figure 2: The illustration of the Reward Bag Transformer (RBT) architecture. The Causal Transformer is used for reward representation by processing sequences of input data consisting of state-action pairs. The bidirectional attention layer is used for reward redistribution, utilizing the outputs of the Causal Transformer to predict instance-level rewards. The next state prediction helps the model understand the environment, thereby improving reward prediction.

4.3 Reward Bag Transformer

We introduce the *Reward Bag Transformer* (RBT), a novel reward redistribution method for the BRMDP. The RBT is engineered to comprehend the complex dynamics of the environment through bags and to precisely predict instance-level rewards, facilitating effective reward redistribution.

4.3.1 Causal Transformer for Reward Representation

Referring to Fig. 2, the RBT comprises a Causal Transformer (Vaswani et al., 2017; Radford et al., 2018), which maintains the chronological order of state-action pairs (Chen et al., 2021; Janner et al., 2021). For each time step t in a sequence of M time steps, the Causal Transformer, represented as a function f, processes the input sequence $\sigma = \{s_0, a_0, \ldots, s_{M-1}, a_{M-1}\}$, generating the output $\{x_t\}_{t=0}^{M-1} = f(\sigma)$. By aligning the output head x_t with the action token a_t, we directly model the consequence of actions, which are pivotal in computing immediate rewards and predicting subsequent states, thereby helping the model better understand environmental dynamics.

4.3.2 Bidirectional Attention Layer for Reward Redistribution

Once we have obtained the output embeddings $\{x_t\}_{t=0}^{M-1}$, for reward prediction, they pass through a bidirectional attention layer to produce $\{\hat{r}_t\}_{t=0}^{M-1}$, where $\hat{r}_t \equiv \hat{r}_\theta(s_t, a_t)$ with θ being the RBT parameters. This layer addresses the unidirectional limitation of the Causal Transformer architecture (Vaswani et al., 2017; Radford et al., 2018), integrating past and future contexts for enhanced reward prediction accuracy. For state prediction, since the state at step $t+1$ depends only on the previous steps, x_t is input into a state linear decoder, yielding the predicted next state $\hat{s}_{t+1} \equiv \hat{s}_\theta(s_t, a_t)$.

The core of the RBT architecture is its bidirectional attention mechanism. For each output embedding x_t, we apply three different linear transformations to obtain the query embedding $\mathbf{q}_t \in \mathbb{R}^d$, key embedding $\mathbf{k}_t \in \mathbb{R}^d$, and value embedding $v_t \in \mathbb{R}$, where d is the embedding dimension of the key. The instance-level reward is then calculated by

$$\hat{r}_t = \sum_{\ell=0}^{M-1} \texttt{softmax}\left(\frac{\{\langle \mathbf{q}_t, \mathbf{k}_{t'} \rangle\}_{t'=0}^{M-1}}{\sqrt{d}}\right)_\ell \cdot v_\ell. \tag{4}$$

Algorithm 1 Policy Optimization with RBT

1: Initialize replay buffer $\mathcal{D}$, RBT parameters θ.
2: **for** trajectory τ collected from the environment **do**
3: Store trajectory τ with bag information $\{(B_{i,n_i}, R(B_{i,n_i}))\}_{B_{i,n_i} \in \mathcal{B}_\tau}$ in $\mathcal{D}$.
4: Sample batches from $\mathcal{D}$.
5: Estimate bag loss based on Eq. equation 7.
6: Update RBT parameters θ based on the loss.
7: Relabel rewards in $\mathcal{D}$ using the updated RBT.
8: Optimize policy using the relabeled data by off-the-shelf RL algorithms (such as SAC (Haarnoja et al., 2018)).
9: **end for**

The rescaling operation is used to prevent extremely small gradients as in Vaswani et al. (2017). This mechanism enables the RBT to consider both the immediate and contextual relevance of each instance in a bag when predicting rewards.

4.4 Learning Objectives

The learning objectives of the RBT are twofold: (1) *reward prediction* within each reward bag and (2) *state transition forecasting*. These objectives are critical for enabling the model to navigate the complex dynamics of BRMDP environments.

4.4.1 Reward Prediction

The RBT is trained to ensure that, for each reward bag, the sum of predicted instance-level rewards matches the total bagged reward. This is vital for maintaining the integrity of the reward structure in the BRMDP framework. The loss function is expressed as

$$\mathcal{L}_{\mathrm{r}}(\theta; B_{i,n_i}) = \left(\sum_{t=i}^{i+n_i-1} \hat{r}_t - R(B_{i,n_i}) \right)^2. \tag{5}$$

This loss function encourages RBT to learn a meaningful distribution of rewards across instances within a bag while ensuring that the total redistributed reward remains consistent with the original bagged reward. By enforcing this consistency, the model aligns with the conditions established in Theorem 1.

4.4.2 State Transition Forecasting

Alongside reward prediction, the RBT is tasked with accurately predicting the next state in the environment given the current state and action. This capability is crucial for understanding the dynamics of the environment, which in turn enhances the accuracy of reward prediction. The corresponding loss function is

$$\mathcal{L}_{\mathrm{s}}(\theta; s_t, a_t, s_{t+1}) = \|\hat{s}_{t+1} - s_{t+1}\|^2, \tag{6}$$

where $\|\cdot\|$ denotes the ℓ^2-norm. This loss emphasizes the model's understanding of dynamics.

4.4.3 Composite Loss

The final learning objective combines the reward and state prediction losses:

$$\begin{aligned} \mathcal{L}_{\mathrm{bag}}(\theta) = & \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \big[\mathcal{L}_{\mathrm{r}}(\theta; B_{i,n_i}) \big| B_{i,n_i} \in \mathcal{B}_\tau \big] \\ & + \beta \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \big[\mathcal{L}_{\mathrm{s}}(\theta; s_t, a_t, s_{t+1}) \big| (s_t, a_t, s_{t+1}) \in \tau \big], \end{aligned} \tag{7}$$

where $\beta > 0$ is a balancing coefficient between the two loss components, and $\mathcal{D}$ denotes the replay buffer.

The RBT's dual predictive capacity is its key advantage, enabling precise reward redistribution to individual instances and forecasting the next state. This leverages environmental dynamics for enhanced reward distribution as experimentally evidenced in Section 5. Integrated with off-the-shelf RL algorithms such as SAC (Haarnoja et al., 2018), the RBT can enhance policy learning within the BRMDP framework, as outlined in Algorithm 1.

5 Experiment

In the following experiment section, we scrutinize the efficacy of our proposed method using benchmark tasks from both the MuJoCo (Brockman et al., 2016) and the DeepMind Control Suite (Tassa et al., 2018) environments, focusing on scenarios with bagged rewards. Initially, we assess the performance of our method to understand its overall effectiveness. Subsequently, we examine whether the proposed RBT reward model accurately predicts rewards. Finally, we evaluate the indispensability of each component of the reward model, questioning if every part is essential.

5.1 Performance Comparison

To evaluate the performance of our method, we conduct a detailed comparison across multiple baselines. This analysis helps us understand how our approach fares against existing methods in handling scenarios with bagged rewards.

5.1.1 Experiment Setting

We evaluated our method on benchmark tasks from the MuJoCo locomotion suite (Ant-v2, Hopper-v2, HalfCheetah-v2, and Walker2d-v2) and the DeepMind Control Suite (cheetah-run, quadruped-walk, fish-upright, cartpole-swingup, ball_in_cup-catch, and reacher-hard).

In standard reinforcement learning settings, rewards are assigned at each step based on immediate state-action transitions. However, in our setup, only the bagged reward is observable, while the individual rewards for each instance remain unknown. Bagged rewards are constructed by aggregating the rewards of all state-action pairs within a predefined sequence. Specifically, in the main text, we present experimental results using summation-based aggregation, while additional results for more complex reward structures are provided in Appendix C. To implement this reward structure, the bagged reward is assigned to the last instance within the bag, while all other state-action pairs within the bag receive a reward of zero. This setting effectively simulates delayed or sparse reward scenarios, making the learning problem more challenging due to the lack of direct instance-level reward supervision. For each task, we collected 1e6 time steps, while the maximum episode length was fixed at 1,000 steps, ensuring consistency across experiments.

5.1.2 Baselines

In the comparative analysis, our framework was rigorously evaluated against several leading algorithms in the domain of RLTF:

- **SAC** (Haarnoja et al., 2018): It directly utilized the original bagged reward information for policy training using the SAC algorithm.
- **IRCR** (Gangwani et al., 2020): It adopted a non-parametric uniform reward redistribution approach. We have adapted IRCR for bagged reward setting.
- **RRD** (Ren et al., 2021): It employed a reward model trained with a randomized return decomposition loss. We have adapted RRD for bagged reward setting.
- **HC** (Han et al., 2022): The HC-decomposition framework was utilized to train the policy using a value function that operates on sequences of data. We employed the code as provided by the original paper.

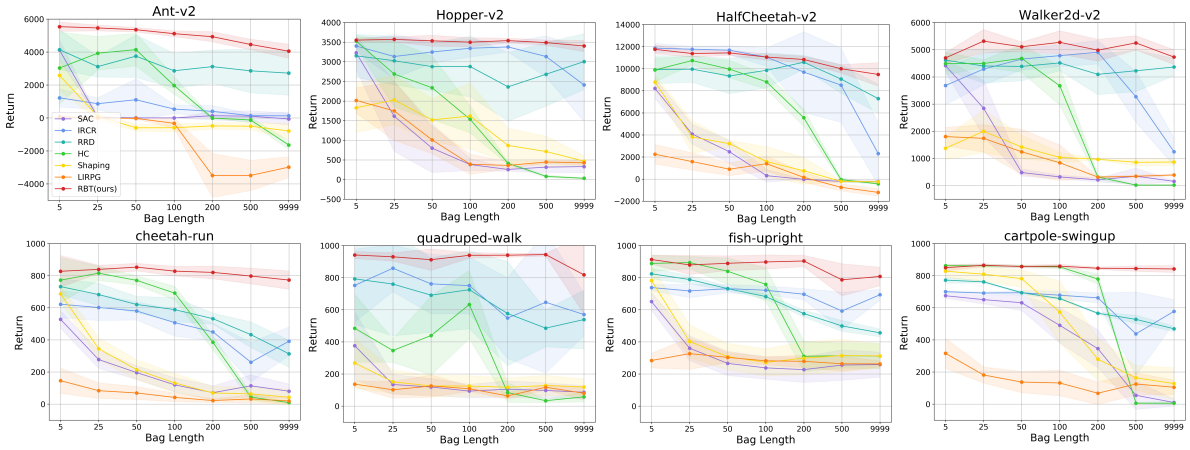

Figure 3: Performance comparison across fixed-length reward bag in MuJoCo (top row) and DeepMind Control Suite (bottom row) environments with six different length bag settings (5, 25, 50, 100, 200, and 500) and trajectory feedback (labeled as 9999). The mean and standard deviation are displayed over 6 trials with different random seeds across a total of 1e6 time steps.

- **Shaping** (Hu et al., 2020): A widely used reward shaping method for sparse reward setting, the shaping reward function is adopted from Hu et al. (2020).
- **LIRPG** (Zheng et al., 2018): It learned an intrinsic reward function to complement sparse environmental feedback, training policies to maximize combined extrinsic and intrinsic rewards. We use the same code provided by the paper.

While methods like RUDDER (Arjona-Medina et al., 2019) and Align-RUDDER (Patil et al., 2022) are known for addressing the problem of trajectory feedback, previous studies (Gangwani et al., 2020; Ren et al., 2021; Zhang et al., 2023) have shown superior performance using referenced methods. Additionally, since Align-RUDDER relies on successful trajectories for scoring state-action pairs, which is impractical in MuJoCo (Patil et al., 2022), we ultimately excluded both methods from our comparison. Sparse reward baselines were included in our comparisons to highlight the distinctions and challenges unique to our bagged reward setting. While both settings provide rewards after a sequence, sparse rewards are Markovian, whereas our bagged rewards are non-Markovian. Therefore, including sparse reward baselines illustrates that directly applying sparse reward methods to bagged rewards is possible but not effective. For more discussion on sparse rewards, see Appendix B. Besides, detailed descriptions of the model parameters and hyper-parameters used during training are provided in Appendix A, with more experimental results available in Appendix C.

5.1.3 Evaluation Metric

We report the average accumulative reward across 6 seeds with random initialization to demonstrate the performance of evaluated methods. Higher accumulative reward in evaluation indicates better performance.

5.1.4 Experimental Results

In the **fixed-length** reward bag experiment (see Fig. 3), we conducted experiments with six bag lengths (5, 25, 50, 100, 200, and 500) and trajectory feedback (labeled as 9999) across each environment, where bagged reward is made by accumulating immediate rewards within bags. This aimed to illustrate the influence of varying bag lengths, providing insight into how bag size affected the performance of the learning algorithm. The SAC method, using bagged rewards directly from the environment, suffers from a lack of guidance in agent training due to missing immediate reward information. This issue worsens with longer bag lengths, indicating that increased reward sparsity leads to less effective policy optimization. The IRCR and RRD methods, treating rewards uniformly within a reward bag, outperform SAC, suggesting benefits from even

Table 1: Performance comparison across arbitrary-length reward bag configurations over 6 trials with 1e6 time steps for training, presenting average scores and standard deviations. "Narrow" refers to bags with lengths varying arbitrarily from 25 to 200 and narrow intervals between -10 to 10. "Broad" denotes the setting with bag lengths varying arbitrarily from 100 to 200 and broad interval variations from -40 to 40. The best and comparable methods based on the paired t-test at the significance level 5% are highlighted in boldface.

Bag Setting	Environment	SAC	IRCR	RRD	HC	Shaping	LIRPG	RBT(ours)
Narrow	Ant-v2	0.87 (2.98)	368.69 (119.74)	2272.39 (835.86)	106.92 (153.86)	-732.03 (225.75)	-756.78 (763.66)	**5122.50 (206.44)**
	Hopper-v2	317.72 (52.17)	3353.35 (61.97)	2184.41 (807.71)	510.66 (94.49)	1031.38 (335.05)	126.13 (30.18)	**3499.54 (76.62)**
	HalfCheetah-v2	788.45 (1737.57)	**10853.85 (573.72)**	9709.62 (1479.73)	4027.25 (441.01)	2348.78 (390.48)	1101.38 (1248.45)	**11282.24 (266.08)**
	Walker2d-v2	193.07 (48.40)	4144.65 (673.66)	3536.90 (546.66)	309.19 (171.69)	1022.32 (369.81)	123.43 (50.97)	**4983.39 (311.09)**
Broad	Ant-v2	-3.31 (4.15)	368.69 (158.46)	1323.50 (1079.60)	5.97 (20.08)	-634.69 (161.57)	-1264.08 (416.86)	**5167.79 (303.83)**
	Hopper-v2	329.48 (44.21)	3296.20 (216.35)	1102.38 (892.12)	701.84 (149.44)	1066.51 (429.68)	203.01 (177.80)	**3499.53 (94.00)**
	HalfCheetah-v2	43.96 (94.32)	9158.14 (1402.62)	4199.16 (1476.85)	4460.80 (518.94)	1459.83 (1251.52)	924.26 (1110.97)	**10837.15 (254.99)**
	Walker2d-v2	176.09 (49.81)	4179.08 (937.42)	330.96 (79.26)	447.45 (155.63)	947.87 (90.46)	194.95 (98.05)	**5202.38 (248.35)**

approximate guidance. However, notable variance in their results indicates potential consistency and reliability issues. The HC method excels only with shorter bag lengths, suggesting that this value function modification method struggles to utilize information from longer sequences. The Shaping and LIRPG methods exhibit subdued performance across tasks, as they are designed to solve sparse reward problems with Markovian rewards, which do not align with the reward bag setting. The proposed RBT method consistently outperforms the other approaches across all the environments and bag lengths, showing that it is not only well-suited for environments with short reward bags but also capable of handling large reward bag scenarios. This demonstrated the capability of RBT to learn from the sequence of instances and, by integrating bagged reward information, accurately allocate rewards to instances, thereby guiding better policy training. In addition to experiments conducted under settings consistent with previous work (Gangwani et al., 2020; Ren et al., 2021; Zhang et al., 2023), where the bagged reward is the sum of immediate rewards, we also tested bagged rewards with more complex structures (see Appendix C.2). These results further demonstrated that our method outperforms the baselines, indicating that previous approaches are effective only when bagged rewards are the sum of immediate rewards, highlighting their limitations.

To validate the effectiveness of our approach under more complex conditions, we designed an experiment with **arbitrary-length** reward bags, allowing not only for variations in bag lengths but also for overlaps and gaps between reward bags. This setup simulated more realistic scenarios and tested the robustness of our method. The results, detailed in Table 1, confirm the superior performance of the proposed RBT method in these complex reward settings. These findings suggest RBT's potential for broader applications, demonstrating its versatility and robustness in handling intricate reward dynamics.

We also conducted experiments in **sparse reward environments**, where tasks are rewarded with 1 at the end of the trajectory if completed and 0 otherwise. In this setting, the reward does not accumulate as in our redistribution model. As shown in Table 2, RLTF methods perform poorly in sparse reward environments. In contrast, our proposed method not only outperforms RLTF baselines but also exceeds the performance of sparse reward baselines. From another perspective, we also tested the performance of sparse reward methods in the bagged reward setting (see Appendix B). The results indicated that, due to differences in the reward structure, sparse reward methods do not adapt well in the bagged reward setting either. This demonstrates that our proposed RBT model can effectively capture relationships between instances across different reward structures. By accurately distributing reward signals across instances, it provides more informative feedback, enabling the agent to learn more efficiently and ultimately improving overall policy performance.

Table 2: Performance comparison on sparse reward environments over 6 trials with 1e6 time steps for training, presenting average scores and standard deviations. Rewards are given with 1 at the end of the trajectory if completed and 0 otherwise. The best and comparable methods based on the paired t-test at the significance level 5% are highlighted in boldface.

Environment	SAC	IRCR	RRD	HC	Shaping	LIRPG	RBT(ours)
ball_in_cup-catch	646.90 (231.15)	773.19 (73.06)	666.18 (33.02)	76.19 (67.25)	847.88 (178.01)	125.50 (21.25)	**972.19 (5.50)**
reacher-hard	531.88 (127.94)	584.76 (269.32)	347.16 (106.56)	7.57 (3.50)	664.49 (170.50)	8.74 (2.36)	**735.67 (144.90)**

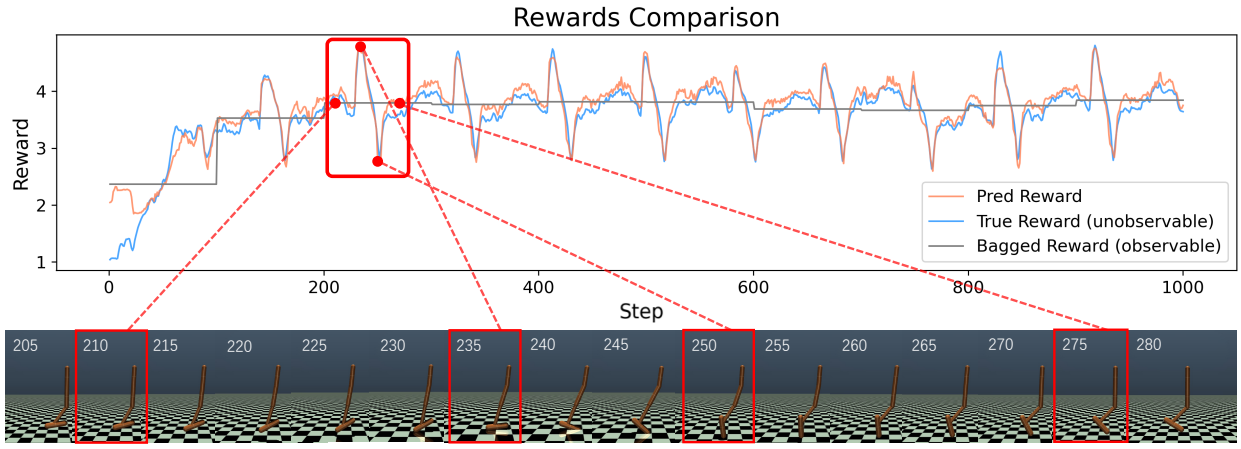

Figure 4: Rewards comparison and agent states in a trajectory with a bag length of 100 in the Hopper-v2 environment. The top graph compares predicted rewards against true rewards and aggregated bagged rewards. The bottom images show the Hopper agent's motion at different time steps, with the highlighted frames representing key moments that provide insight into the relationship between agent behavior and reward dynamics.

5.2 Case Study

The previous experimental results showcase the superiority of RBT over baselines. This led to an intriguing inquiry: Is the RBT reward model proficient in accurately redistributing rewards? To investigate this question, we performed an experiment focused on reward comparison, utilizing a trajectory generated by an agent trained in the Hopper-v2 environment with a bag length of 100, where the bagged reward represents an accumulated reward. As shown in Fig. 4, which spans 1000 steps, RBT-predicted rewards, unobservable true rewards, and observable bagged rewards (presented in a uniform format for better visualization) are compared. The figure indicates that the rewards predicted by the RBT closely match the trends of the true rewards. This observation suggest that the RBT is effective at reconstructing true rewards from bagged rewards, despite the coarse nature of the environmental reward signals.

Beneath the figure, a series of images depicts a complete jump cycle by the agent, illustrating its motion sequence: mid-air, landing, jumping, and returning to mid-air. Red boxes highlight specific states that correspond to reward peaks and troughs, representing moments of maximum, minimum, and moderate rewards. In the Hopper-v2 environment, rewards consist of a constant "healthy reward" for operational integrity, a "forward reward" for progress in the positive x-direction, and a "control cost" for penalizing large actions. At peak reward instances, the agent is typically fully grounded in an optimal posture for forward leaping, which maximizes the "forward reward" through pronounced x-direction movement. Concurrently, it sustains the "healthy reward" and minimizes "control cost" through measured, efficient actions. This

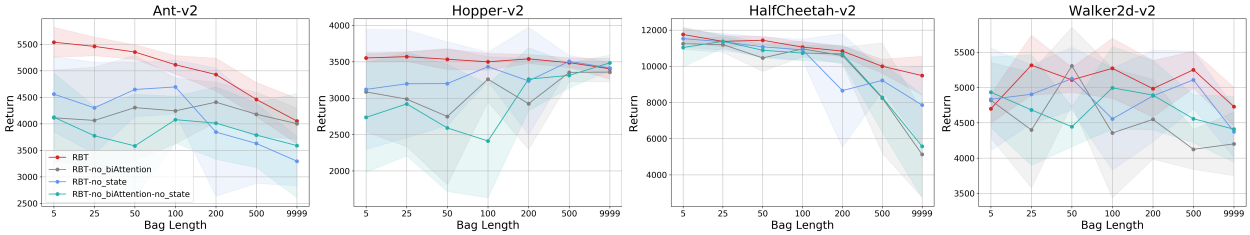

Figure 5: Ablation study of reward model components across various environments. The chart presents mean and standard deviation of rewards over 6 trials with 1e6 timesteps, showcasing the efficacy of the full proposed method relative to its variants without certain features.

analysis underscores that the RBT can adeptly decode the environmental dynamics and the nuanced reward redistribution even under the setting of RLBR.

5.3 Ablation Study

We conducted comparisons to examine the role of RBT's modules. As shown in Fig. 5, the full RBT model consistently outperforms its variants, indicating a synergistic effect when all components are used together. Performance drops significantly when the bidirectional attention mechanism is removed, especially in complex environments like Ant-v2 and HalfCheetah-v2, suggesting its critical role in accurate reward prediction. Additionally, we can observe that removing the next state prediction component weakens the reward model's understanding of environmental dynamics, reducing reward prediction accuracy and hindering policy learning. The greatest performance decline occurs when both the next state prediction and bidirectional self-attention mechanism are absent, underscoring their individual and combined importance in building a robust reward model.

6 Limitation

One major limitation of our approach is the training duration, which is longer compared to methods that do not require a reward model (see Appendix C.5). The increased computational cost primarily arises from the reward relabeling process, where the model updates assigned rewards in the replay buffer based on the learned reward distribution. This step adds significant overhead, making training less efficient, particularly for longer bag lengths, where reward prediction becomes more challenging.

A potential direction for improving computational efficiency is prioritized experience replay, which could selectively relabel more informative samples rather than processing the entire buffer equally, thereby reducing unnecessary computation. Another possible enhancement is a cached reward update mechanism, where only a subset of stored samples is updated per iteration instead of recomputing rewards for the full replay buffer. These optimizations have the potential to reduce training time while maintaining learning performance.

7 Conclusion

In this paper, we introduced a novel learning framework called Reinforcement Learning from Bagged Rewards (RLBR). To address the challenges posed by this framework, we established theoretical connections between Bagged Reward MDPs (BRMDPs) and original MDPs after reward redistribution, providing a solid foundation for our approach. Based on this theory, we proposed a reward model, the Reward Bag Transformer (RBT), designed to redistribute rewards by interpreting contextual information within bags and understanding environmental dynamics. The efficacy of RBT was demonstrated through extensive experiments, where it consistently outperformed existing methods across various reward bag scenarios. Additionally, our case studies highlighted RBT's ability to effectively reallocate rewards while maintaining fidelity to the underlying true reward structure.

References

Pranav Agarwal, Pierre de Beaucorps, and Raoul de Charette. Goal-constrained sparse reinforcement learning for end-to-end driving. *arXiv preprint arXiv:2103.09189*, 2021.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. In *The Thirty-second Annual Conference on Advances in Neural Information Processing Systems*, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *The Seventh International Conference on Learning Representations*, 2019.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *The Thirty-fourth Annual Conference on Advances in Neural Information Processing Systems*, 2021.

Sirui Chen, Zhaowei Zhang, Yaodong Yang, and Yali Du. Stas: Spatial-temporal return decomposition for solving sparse rewards problems in multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17337–17345, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Joseph Early, Tom Bewley, Christine Evers, and Sarvapali Ramchurn. Non-markovian reward modelling from trajectory labels via interpretable multiple instance learning. In *The Thirty-fifth Annual Conference on Advances in Neural Information Processing Systems*, 2022.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *The Thirty-fifth International Conference on Machine Learning*. PMLR, 2018.

Roy Frostig, Matthew James Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 4(9), 2018.

Tanmay Gangwani, Yuan Zhou, and Jian Peng. Learning guidance rewards with trajectory-space smoothing. In *The Thirty-third Annual Conference on Advances in Neural Information Processing Systems*, 2020.

Daiqi Gao, Hsin-Yu Lai, Predrag Klasnja, and Susan A Murphy. Harnessing causality in reinforcement learning with bagged decision times. *arXiv preprint arXiv:2410.14659*, 2024.

Maor Gaon and Ronen Brafman. Reinforcement learning with non-markovian rewards. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, volume 34, 2020.

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

Beining Han, Zhizhou Ren, Zuofan Wu, Yuan Zhou, and Jian Peng. Off-policy reinforcement learning with delayed rewards. In *The Thirty-ninth International Conference on Machine Learning*. PMLR, 2022.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. In *The Thirty-third Annual Conference on Advances in Neural Information Processing Systems*, 2020.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *The Thirty-fourth Annual Conference on Advances in Neural Information Processing Systems*, 2021.

Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding. In *The Thirty-first Annual Conference on Advances in Neural Information Processing Systems*, 2018.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

Ilya Kostrikov. JAXRL: Implementations of Reinforcement Learning algorithms in JAX, 10 2021. URL `https://github.com/ikostrikov/jaxrl`.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback, 2023.

Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *The Sixth International Conference on Learning Representations*, 2018.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *The Sixteenth International Conference on Machine Learning*. PMLR, 1999.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *The Thirty-fourth International Conference on Machine Learning*. PMLR, 2017.

Vihang Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M Blies, Johannes Brandstetter, José Arjona-Medina, and Sepp Hochreiter. Align-rudder: Learning from few demonstrations by reward redistribution. In *The Thirty-ninth International Conference on Machine Learning*. PMLR, 2022.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training, 2018.

Zhizhou Ren, Ruihan Guo, Yuan Zhou, and Jian Peng. Learning long-term reward redistribution via randomized return decomposition. In *The Ninth International Conference on Learning Representations*, 2021.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *The Thirty-fifth International Conference on Machine Learning*. PMLR, 2018.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *The Thirty-seventh International Conference on Machine Learning*. PMLR, 2020.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *The Fourth International Conference on Learning Representations*, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Jonathan Sorg, Richard L Lewis, and Satinder Singh. Reward design via online gradient ascent. In *The Twenty-fourth Annual Conference on Advances in Neural Information Processing Systems*, 2010.

R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019.

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

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *The Seventh International Conference on Learning Representations*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *The Thirtieth Annual Conference on Advances in Neural Information Processing Systems*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. King's College, Cambridge United Kingdom, 1989.

Michael Widrich, Markus Hofmarcher, Vihang Prakash Patil, Angela Bitto-Nemling, and Sepp Hochreiter. Modern hopfield networks for return decomposition for delayed rewards. In *Deep RL Workshop NeurIPS 2021*, 2021.

Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, 2020.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

Yudi Zhang, Yali Du, Biwei Huang, Ziyan Wang, Jun Wang, Meng Fang, and Mykola Pechenizkiy. Interpretable reward redistribution in reinforcement learning: A causal approach. In *The Thirty-sixth Annual Conference on Advances in Neural Information Processing Systems*, 2023.

Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. In *The Thirty-fourth Annual Conference on Advances in Neural Information Processing Systems*, 2021.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. In *The Thirty-first Annual Conference on Advances in Neural Information Processing Systems*, 2018.

A Experiment Settings and Implementation Details

Benchmarks with Bagged Rewards. We introduced a novel problem setting in the suite of MuJoCo and DeepMind Control Suite locomotion benchmark tasks, termed as bagged rewards. Our simulations ran on the OpenAI Gym platform (Brockman et al., 2016) and the DeepMind Control Suite (Tassa et al., 2018), featuring tasks that stretched over long horizons with a set maximum trajectory length of $T = 1000$. We used MuJoCo version 2.0 for our simulations, which is available at `http://www.mujoco.org/`. MuJoCo is licensed under a commercial license, and we have adhered to its terms of service and licensing agreements as stated on the official website. The DeepMind Control Suite is available under an Apache License 2.0, and we have complied with its terms of use.

Reward bag experiments of different bag sizes (5, 25, 50, 100, 200, and 500) and trajectory feedback were set up to verify the effectiveness of the method. To evaluate the efficacy of proposed method, commonly used trajectory feedback algorithms were adapted to fit the bagged reward setting as baselines. In these experiments, each reward bag was treated as an individual trajectory, and these modified algorithms were applied accordingly. Additionally, experiments using standard trajectory feedback were conducted to provide a comparative baseline within the unique setting. The total episodic feedback was computed at the end of the trajectory and was the sum of the per-step rewards the agent had collected throughout the episode. This experiment setting was the same as some previous works for learning from trajectory feedback (Gangwani et al., 2020; Ren et al., 2021).

For the sparse reward experiments, the ball_in_cup-catch task provides a sparse reward of 1 when the ball is in the cup and 0 otherwise; in the reacher-hard task, the reward is 1 when the end effector penetrates the target sphere (Tassa et al., 2018). Therefore, we treated the sparse reward as trajectory feedback and conducted experiments where the entire trajectory was considered as a single reward bag in these environments. The bagged reward was identical to the sparse reward provided by the environment.

Table 3: Hyper-parameters of RBT.

Hyper-parameter	Value
Number of Causal Transformer layers	3
Number of bidirectional attention layers	1
Number of attention heads	4
Embedding dimension	256
Batch size	64
Dropout rate	0.1
Learning rate	0.0001
Optimizer	AdamW (Loshchilov & Hutter, 2018)
Weight decay	0.0001
Warmup steps	100
Total gradient steps	10000

Implementation Details and Hyper-parameter Configuration. In our experiments, the policy optimization module was implemented based on soft actor-critic (SAC) (Haarnoja et al., 2018). We evaluated the performance of our proposed methods with the same configuration of hyper-parameters in all environments. The back-end SAC followed the JaxRL implementation (Kostrikov, 2021), which is available under the MIT License.

The RBT reward model was developed based on the GPT implementation in JAX (Frostig et al., 2018), which is available under the Apache License 2.0. Our experiments utilized the Causal Transformer with three layers and four self-attention heads, followed by a bidirectional self-attention layer with one self-attention head. For detailed hyper-parameter settings of the RBT, please refer to Table 3.

For the baseline methods, the IRCR (Gangwani et al., 2020) method was implemented based on the descriptions provided in the original paper. The RRD (Ren et al., 2021) and LIRPG (Zheng et al., 2018)

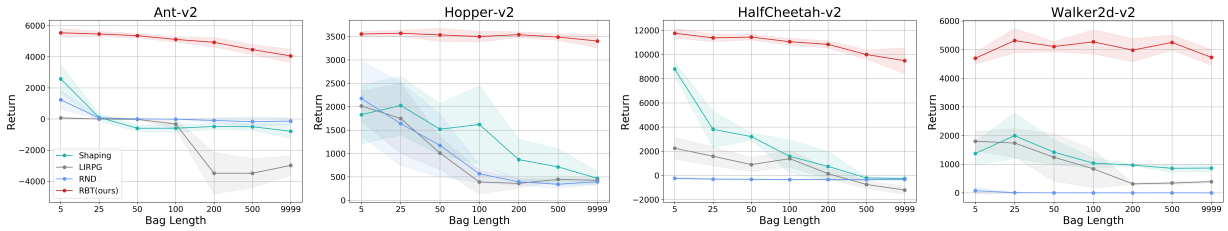

Figure 6: Comparison of proposed method and sparse reward baselines. The mean and standard deviation are displayed over 6 trials with different random seeds across a total of 1e6 time steps.

methods are both licensed under the MIT License. The code of HC (Han et al., 2022) is available in the supplementary material at `https://openreview.net/forum?id=nsjkNB2oKsQ`.

To ensure uniformity in the policy optimization process across all methodologies, each was subjected to 1,000,000 training iterations. For the proposed method, we initially collated a dataset comprising 10,000 time steps to pre-train the reward model. This model then underwent 100 pre-training iterations, a step deemed essential to adequately prepare the reward model before embarking on the principal policy learning phase. Following this initial warm-up period, the reward model was trained for 10 iterations after each new trajectory was incorporated. Moreover, to systematically gauge performance and progress, evaluations were carried out at intervals of every 5,000 time steps. The computational resources for these procedures were NVIDIA GeForce RTX 2080 Ti GPU clusters with 8GB of memory, dedicated to training and evaluating tasks.

B Discussion on Sparse Reward

The sparse reward setting presents significant challenges due to infrequent feedback, making it difficult for agents to effectively explore the environment and discover successful strategies. To address this challenge, various methods have been developed to enhance exploration. Reward shaping strategies (Ng et al., 1999; Hu et al., 2020; Tambwekar et al., 2019) added rewards to actions in a way that guides the agent towards better policies without altering the original reward function. Curiosity-driven methods (Pathak et al., 2017; Sekar et al., 2020) encouraged agents to explore the environment by visiting unseen states, potentially solving tasks with sparse rewards. Additionally, curriculum learning in RL (Florensa et al., 2018; Riedmiller et al., 2018) involved presenting an agent with a sequence of tasks with gradually increasing complexity, allowing the agent to eventually solve the initially given sparse reward task.

Due to the formal similarity between sparse rewards and trajectory feedback, both involving a single reward signal after a sequence, these methods can be directly applied to RLTF. However, the key difference lies in the nature of the rewards: sparse rewards typically focus on specific key points, making the reward Markovian, whereas RLTF evaluates the entire trajectory, indicating a Non-Markovian reward. Furthermore, sparse reward strategies address the exploration-exploitation trade-off inherent in sparse rewards, while RLTF focuses on distributing the evaluation of the entire trajectory across specific state-action pairs to guide learning. Thus, while these methods are applicable, they do not address the unique challenges posed by bagged rewards, as they fail to account for the reward structure within the bags. Therefore, they are not ideal solutions for RLTF (Gangwani et al., 2020; Ren et al., 2021).

In Fig. 6, we present additional results comparing our method with sparse reward baselines. Shaping (Tessler et al., 2019; Hu et al., 2020) and LIRPG (Zheng et al., 2018) are as described in the main paper. The Random Network Distillation (RND) (Burda et al., 2019) method introduces an exploration bonus for deep reinforcement learning by measuring the prediction error of a randomly initialized neural network, which is a widely recognized and utilized approach for addressing sparse reward challenges. We include it as a baseline in our comparison of sparse reward methods for a more comprehensive evaluation.

These results demonstrate that, despite the superficial similarity between sparse rewards and bagged rewards, the internal structure of the rewards differs, leading to suboptimal performance of sparse reward baselines

Table 4: Performance comparison across reward bag with various-length configurations over 3 trials. In this table, "Short" refers to bags with lengths varying from 25 to 200, and "Long" denotes the setting with bag lengths from 100 to 500. The best and comparable methods based on the paired t-test at the significance level 5% are highlighted in boldface.

Bag Setting	Environment	SAC	IRCR	RRD	HC	Shaping	LIRPG	RBT(ours)
Short	Ant-v2	3.21 (1.97)	269.12 (224.66)	2661.56 (1675.65)	20.03 (46.35)	-596.02 (178.75)	-1407.15 (504.59)	**5359.85 (129.28)**
	Hopper-v2	286.53 (72.51)	3275.05 (44.68)	2508.76 (690.07)	474.21 (66.93)	1358.92 (640.58)	287.38 (114.65)	**3433.06 (96.89)**
	HalfCheetah-v2	5.92 (23.06)	10480.33 (202.27)	**10382.80 (516.85)**	3982.43 (433.75)	2010.72 (1085.36)	1225.35 (162.88)	**11073.88 (181.43)**
	Walker2d-v2	222.21 (66.72)	3840.30 (666.24)	3999.14 (561.49)	348.36 (174.28)	926.49 (100.37)	328.97 (107.69)	**5198.09 (225.50)**
Long	Ant-v2	-115.10 (138.93)	215.38 (92.94)	2600.64 (1229.27)	-0.53 (5.07)	-460.51 (258.06)	-2552.99 (419.44)	**4897.50 (292.93)**
	Hopper-v2	360.36 (118.06)	3015.96 (408.08)	3089.52 (433.23)	652.28 (92.41)	729.28 (338.56)	325.84 (50.25)	**3447.64 (83.02)**
	HalfCheetah-v2	-115.56 (35.01)	5944.17 (3421.46)	8591.38 (1048.08)	4563.99 (568.19)	-268.64 (114.37)	-571.35 (374.64)	**10880.76 (441.65)**
	Walker2d-v2	251.11 (144.24)	3397.93 (682.19)	4221.37 (282.75)	497.27 (134.06)	779.43 (219.75)	284.27 (7.54)	**4979.07 (166.95)**

in bagged reward environments. Conversely, as shown in the Section Compare with SOTA Methods of the main paper, our method performs well even in sparse reward environments, surpassing baselines specifically designed for sparse reward scenarios. This highlights the broad applicability and adaptability of our approach across different reward settings.

C Additional Experimental Results

This section provides further analysis and insights through additional experiments to complement the main findings presented in our main paper.

C.1 Experimental Result of Various-Length Reward Bags

In Table 4, we present the experiment results on various-length reward bags. The experiment depict in the table showcases the results of various methods applied across different environments with varying bag lengths of rewards, where bags are one next to another as in the definition of RLBR. This experiment reveals that longer bags tend to degrade the performance of most methods. However, our RBT method appears to be less sensitive to changes in bag length, maintaining robust performance even when the bag length is equal to the full trajectory. This result aligns with the result in the Section Compare with SOTA Methods in our main paper.

C.2 Preference on Handling Complex Delayed Reward Structures

The previous experiments were based on the assumption that a bagged reward is the sum of immediate rewards, which aligns with prior works (Gangwani et al., 2020; Ren et al., 2021; Zhang et al., 2023). However, our approach does not make this assumption, indicating that the proposed method can effectively handle more complex reward structures. In this section, we experimentally validate this claim. Specifically, we included the following complex delayed reward structures computed over B_{i,n_i}:

- **SumSquare**: The delayed reward is the sum of squared step rewards, placing more emphasis on larger rewards: $R_{\text{co}} = \sum_{t=i}^{i+n_i-1} \text{abs}(r_t) \cdot r_t$.

- **SquareSum**: The delayed reward is the square of the sum of step rewards, highlighting overall sequence performance: $R_{\text{co}} = \text{abs}\big(\sum_{t=i}^{i+n_i-1} r_t\big) \cdot \big(\sum_{t=i}^{i+n_i-1} r_t\big)$.
- **Max**: The delayed reward is a softmax-weighted sum of step rewards, giving more attention to the highest rewards: $R_{\text{co}} = \sum_{t=i}^{i+n_i-1} \frac{n_i \cdot e^{\beta r_t}}{\sum_{t'=i}^{i+n_i-1} e^{\beta r_{t'}}} \cdot r_t$,

  where β is a scaling parameter that controls the sharpness of the softmax distribution.

The normalized scores shown in the figures are computed by scaling the obtained rewards against the maximum possible rewards achievable in these bagged reward environments. For SumSquare (see Figure 7), our method consistently outperforms baseline methods across different environments, maintaining strong performance even as the bag length increases. In the SquareSum setting (see Figure 8), the structure of bagged reward is more complex, as it emphasizes the cumulative effect of multiple steps rather than focusing on individual large rewards, making it more difficult to attribute rewards to specific actions. Despite this complexity, our method still holds a clear advantage over most baselines, even as performance slightly decreases with longer bag lengths. In the Max setting (see Figure 9), where rewards are sparse and only a few key steps contribute to the overall outcome, our method manages to learn effectively, showing that it can still identify and focus on the most critical instances in the sequence, even with limited reward signals. Under these conditions, methods relying on the sum-form delayed rewards assumption experience a significant drop in performance. This indicates that these methods are heavily dependent on this assumption. It is also worth noting that the HC method does not rely on this assumption, but its effectiveness is limited to shorter delays. As the length of bag increases, its performance drops sharply.

Overall, our proposed method shows a strong ability to handle various bagged reward structures and adapt to increasing lengths, outperforming baseline methods in most environments. Moreover, these results highlight the importance of further exploring how to learn reward models and train policies in environments with complex bagged reward structure, making it worthy of deeper investigation.

C.3 Architecture Sensitivity

Fig. 10 illustrates the sensitivity of our architecture to different input sequence lengths during training (Seq len) and prediction lengths during reward relabeling (Relabel len) in the Ant-v2 environment. The learning curves, based on three independent runs with random initialization, show how varying these parameters affects the performance of the agent.

Although the configuration highlighted by the red box (Seq len of 500 and Relabel len of 100) demonstrates the best performance, the results also show that our proposed model is capable of learning effectively across various other configurations. This analysis underscores the importance of tuning input sequence length during training and prediction length during reward relabeling for optimal performance, while also demonstrating the ability of the RBT model to learn under different parameter settings, showcasing its flexibility and effectiveness in reinforcement learning tasks.

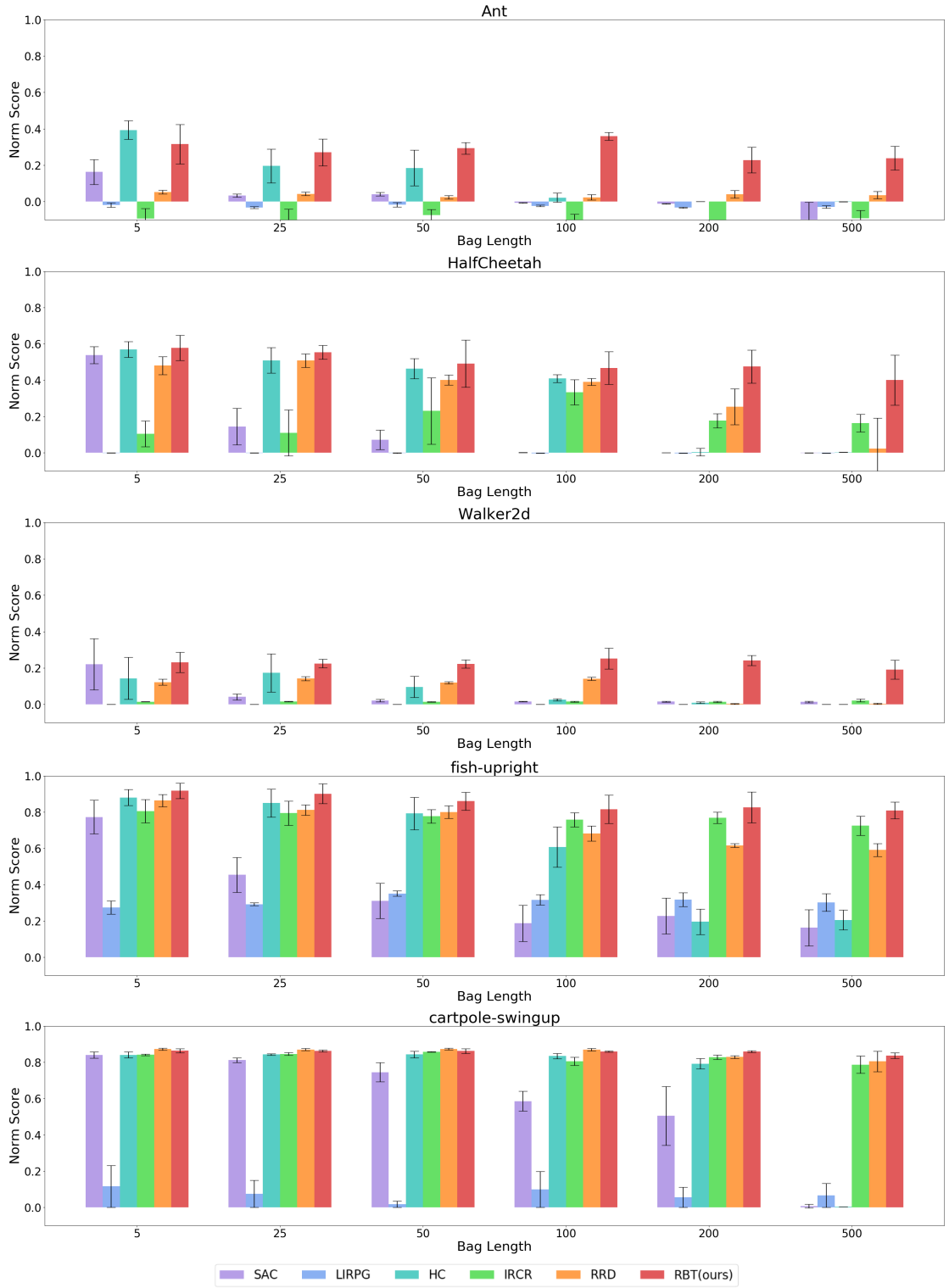

Figure 7: Performance comparison on the SumSquare reward structure with six different bag length settings (5, 25, 50, 100, 200, and 500). The mean and standard deviation are reported over six trials with different random seeds, measured across a total of 1e6 time steps.

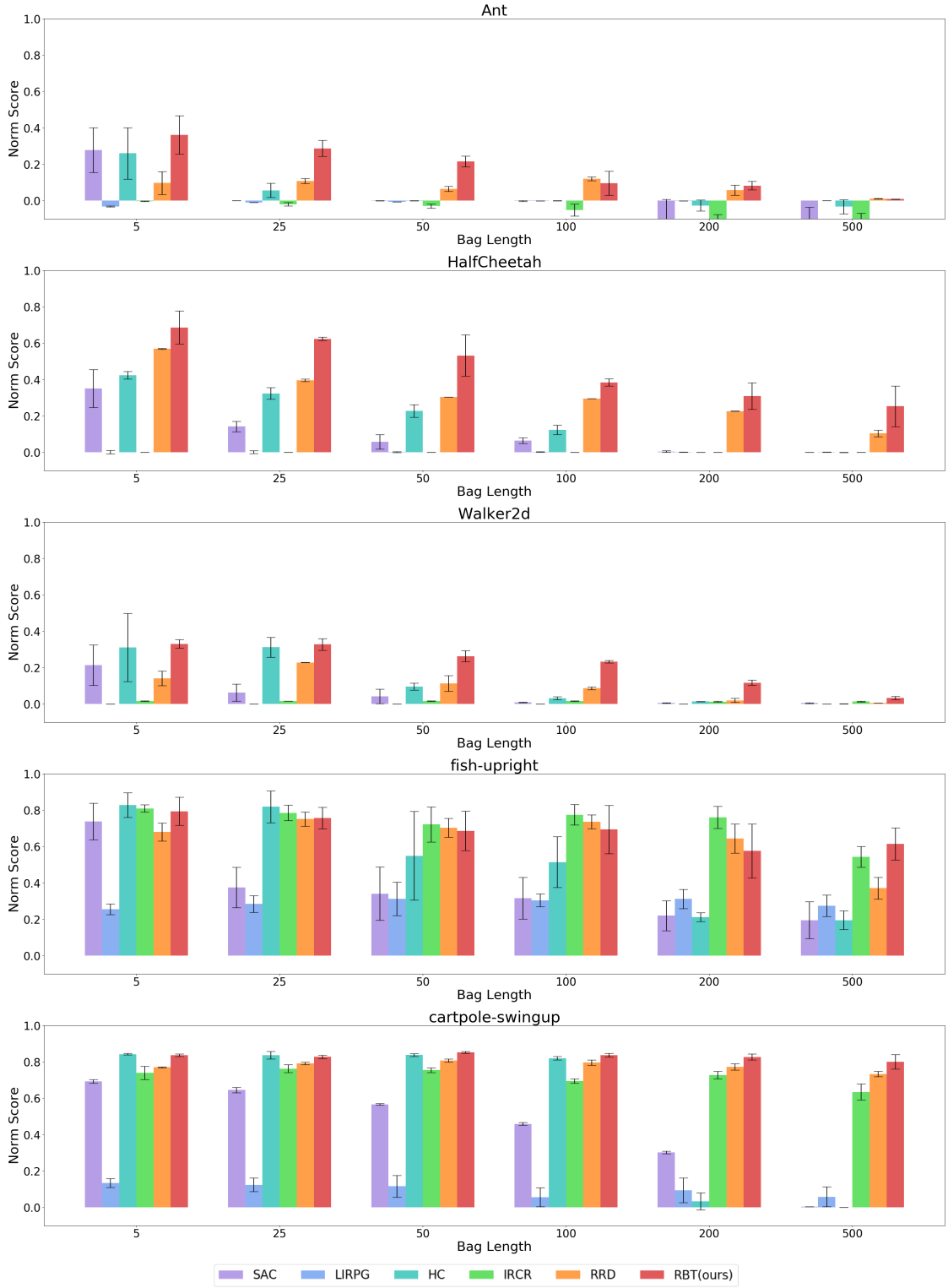

Figure 8: Performance comparison on the SquareSum reward structure with six different bag length settings (5, 25, 50, 100, 200, and 500). The mean and standard deviation are reported over six trials with different random seeds, measured across a total of 1e6 time steps.

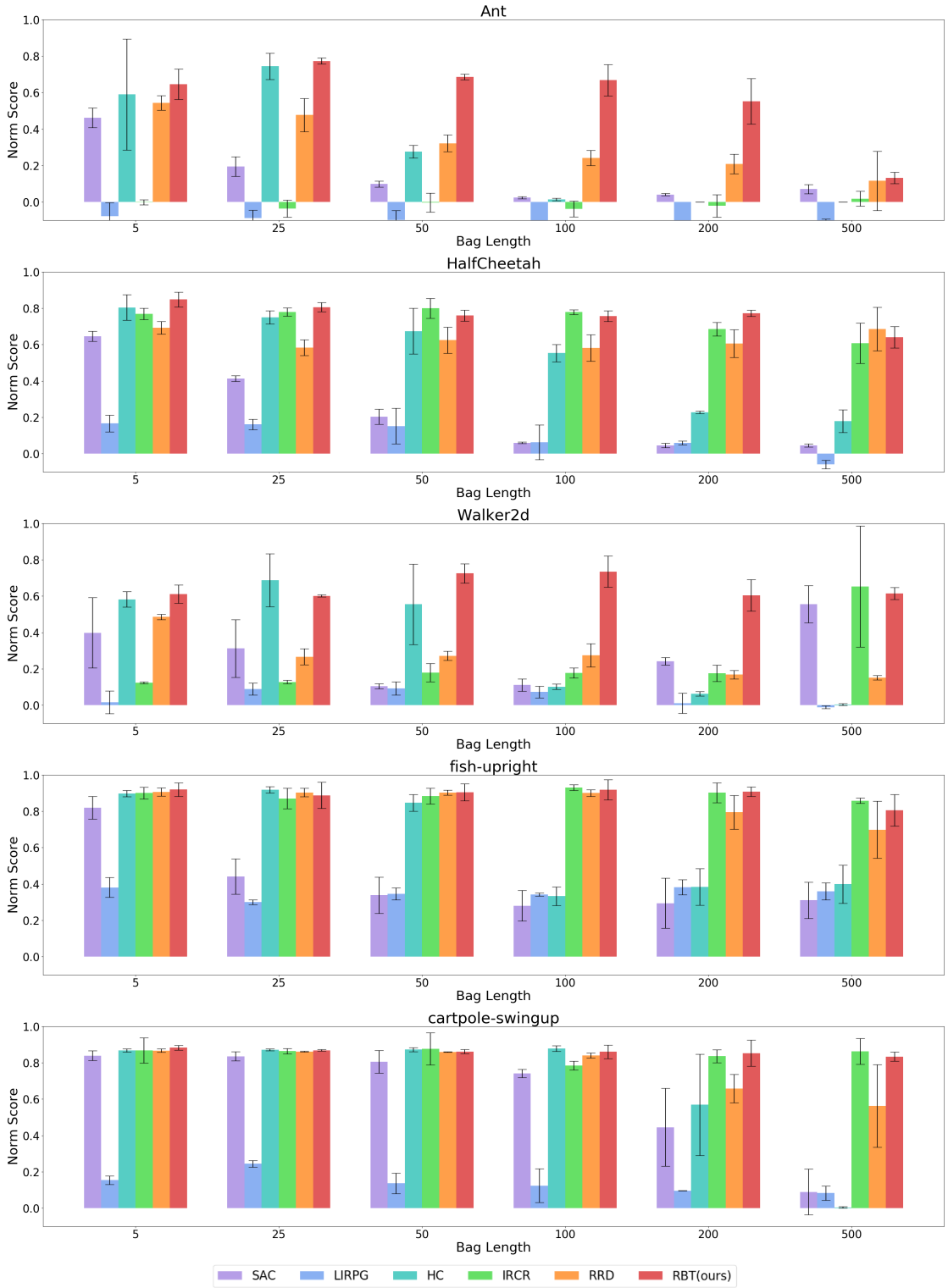

Figure 9: Performance comparison on the Max reward structure with six different bag length settings (5, 25, 50, 100, 200, and 500). The mean and standard deviation are reported over six trials with different random seeds, measured across a total of 1e6 time steps.

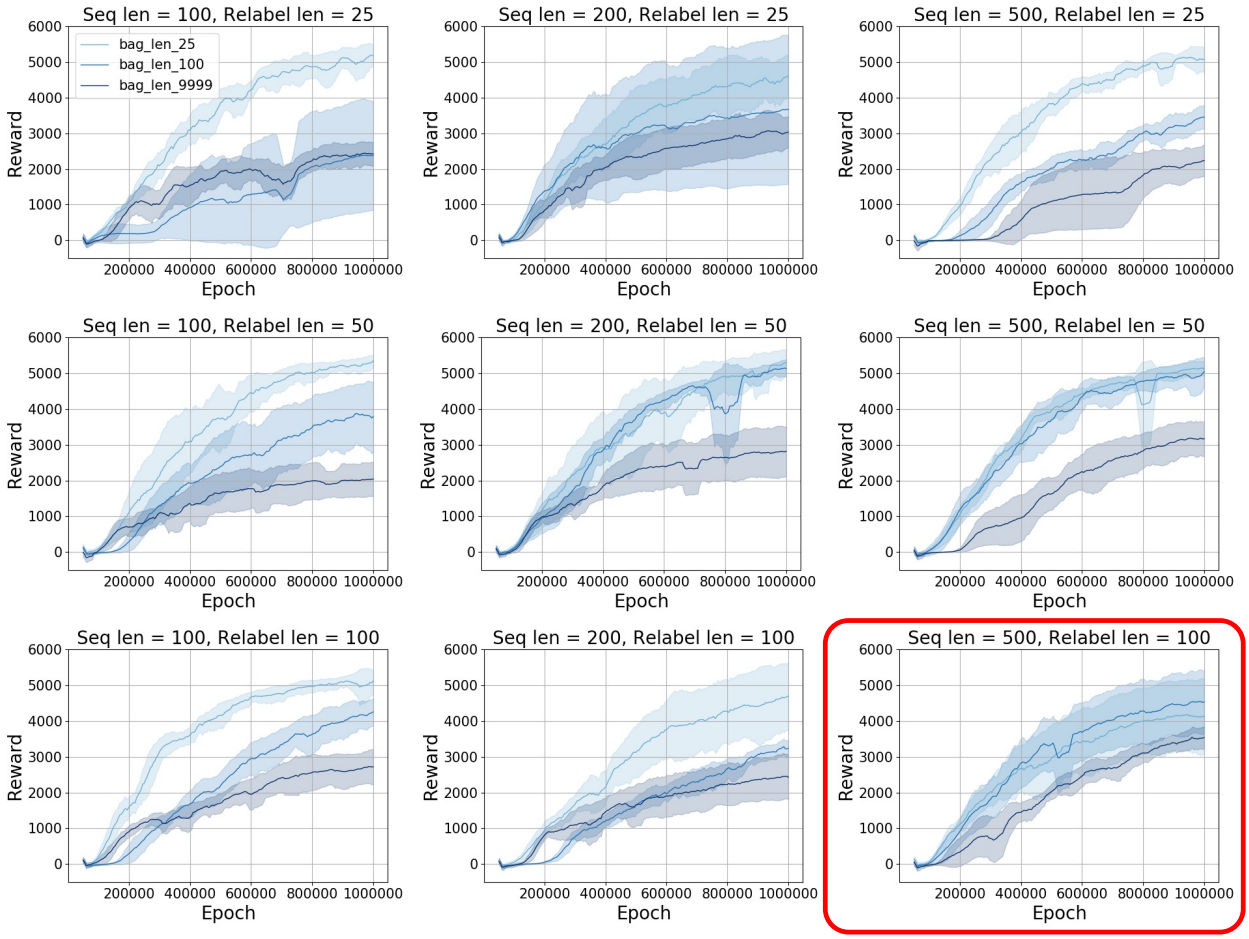

Figure 10: Learning curves on Ant-v2 with different length of input sequence in training (Seq len) and predict length during relabeling process (Relabel len), based on 3 independent runs with random initialization. Within each of the smaller graphs, the curves represent results from experiments with different bag lengths. Specifically, there are three bag lengths evaluated: 25, 100, and what is labeled as 9999, which we interpret as a proxy for trajectory feedback. The graph highlighted by the red box indicates our chosen parameter setting for the experiment, which is a input sequence length of 100 and a predict length of 500.

C.4 Final Training Losses

Table 5: Final training losses on the Ant-v2 environment with different bag lengths, showing the mean and standard deviation over three random seeds.

Bag Length	5	25	50	100	200	500	9999
$\mathcal{L}_{\mathrm{s}}$	0.003 ± 0.002	0.023 ± 0.001	0.031 ± 0.007	0.032 ± 0.013	0.056 ± 0.036	0.053 ± 0.027	0.032 ± 0.011
$\mathcal{L}_{\mathrm{r}}$	0.078 ± 0.003	0.174 ± 0.008	0.310 ± 0.049	0.769 ± 0.061	1.342 ± 0.146	3.066 ± 0.529	6.195 ± 0.406
$\mathcal{L}_{\mathrm{bag}}$	0.081 ± 0.001	0.196 ± 0.008	0.341 ± 0.056	0.801 ± 0.068	1.399 ± 0.116	3.120 ± 0.520	6.227 ± 0.396

The training losses $\mathcal{L}_{\mathrm{s}}$, $\mathcal{L}_{\mathrm{r}}$, and $\mathcal{L}_{\mathrm{bag}}$ vary with different bag lengths, reflecting how the model's learning process is influenced by the difficulty of reward redistribution and state prediction. Since we set the balancing coefficient $\beta = 1$, the composite loss $\mathcal{L}_{\mathrm{bag}}$ is the sum of $\mathcal{L}_{\mathrm{s}}$ and $\mathcal{L}_{\mathrm{r}}$.

The results in Table 5 show that the state prediction loss $\mathcal{L}_{\mathrm{s}}$ remains relatively stable across different bag lengths, suggesting that the model effectively learns environmental dynamics with minimal impact from the bag size. In contrast, the reward prediction loss $\mathcal{L}_{\mathrm{r}}$ increases significantly as the bag length grows, indicating that longer bags introduce higher uncertainty in distributing rewards at the instance level, making the prediction task more challenging. Consequently, since $\mathcal{L}_{\mathrm{bag}}$ is a combination of $\mathcal{L}_{\mathrm{s}}$ and $\mathcal{L}_{\mathrm{r}}$, it is primarily driven by the increasing $\mathcal{L}_{\mathrm{r}}$ at larger bag lengths. This suggests that the main challenge in learning under bagged rewards lies in accurate reward redistribution when fewer direct supervisory signals are available. Future improvements could focus on mitigating the impact of large bag lengths on reward prediction, such as incorporating regularization techniques, hierarchical reward redistribution, or adaptive weighting in $\mathcal{L}_{\mathrm{bag}}$ to improve overall learning efficiency.

C.5 Runtime and Computational Overhead

Table 6: Training runtime (in hours) for different methods.

Method	SAC	IRCR	Shaping	LIRPG	RRD	HC	RBT (ours)
Runtime (hours)	1	1	1	8	21	30	22

The Table 6 presents the training runtime for different methods on an NVIDIA GeForce RTX 2080 Ti GPU cluster with 8GB of memory. Only RBT (ours) and RRD utilize a reward model, which introduces additional computational overhead, resulting in longer training times compared to methods that do not rely on a reward model. HC modifies the value function to adapt to sequence-level rewards, which also increases training time.

Notably, the primary factor contributing to the runtime of our proposed RBT method is not the training of the model itself but rather the relabeling of rewards for samples in the replay buffer. This process adds significant computational overhead, making it an area for potential optimization. A potential improvement is to use prioritized experience replay, where more informative samples are prioritized for relabeling instead of treating all samples equally, thereby improving efficiency while maintaining learning quality. Another approach is to implement a cached reward update mechanism, where only a subset of samples is updated per iteration rather than recomputing rewards for all samples, significantly reducing computational costs. By incorporating these optimizations, the computational efficiency of RBT has the potential to be significantly improved without compromising performance. By incorporating these optimizations, the computational efficiency of RBT has the potential to be significantly improved without compromising performance.