# Reinforcement Learning from Bagged Reward

**Yuting Tang**[1,2*] **Xin-Qiang Cai**[1*] **Yao-Xiang Ding**[3]
**Qiyu Wu**[1] **Guoqing Liu**[4] **Masashi Sugiyama**[2,1]
[1]The University of Tokyo, Japan
[2]RIKEN Center for Advanced Intelligence Project, Japan
[3]Zhejiang University, China
[4]Microsoft Research AI4Science, China

## 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, immediate reward signals are not obtainable; 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 *Reinforcement Learning from Bagged Reward* (RLBR), where sequences of data are treated as *bags* with non-Markovian bagged rewards. We provide a theoretical study to establish the connection between RLBR and standard RL in Markov Decision Processes (MDPs). To effectively explore the reward distributions within these bags and enhance policy training, we propose a Transformer-based reward model, the Reward Bag Transformer, which employs a bidirectional attention mechanism to interpret contextual nuances and temporal dependencies within each bag. Our empirical evaluations reveal that the challenge intensifies as the bag length increases, leading to the performance degradation due to reduced informational granularity. Nevertheless, our approach consistently outperforms existing methods, demonstrating the least decline in efficacy across varying bag lengths and excelling in approximating the original MDP's reward distribution. 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 [23], continuous control [4–6], complex game playing [39, 3], and financial trading [47]. 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, this assumption is violated in many real-world applications. Recognizing this gap, numerous studies [45, 21, 14, 34, 50] have explored the concept of delayed rewards, primarily focusing on trajectory feedback where rewards are allocated at the end of a sequence. Similarly, sparse reward settings, where agents receive infrequent and instance-specific feedback, present significant challenges in the well-known exploration-exploitation trade-off [36, 9, 8].

On the other hand, real-world applications, such as autonomous driving (see Fig. 1), often feature complex non-immediate reward structures that neither sparse rewards nor trajectory feedback can fully capture. Providing reward for every action is impractical, and focusing only on end goals ignores crucial aspects of the journey. Typically, rewards are linked to completing specific tasks or sequences of actions, rather than individual actions or the final objective [11, 15]. However, previous studies (see Fig. 2) mainly focused on learning desirable policies with immediate rewards [40, 44],

[*] Equal contribution.

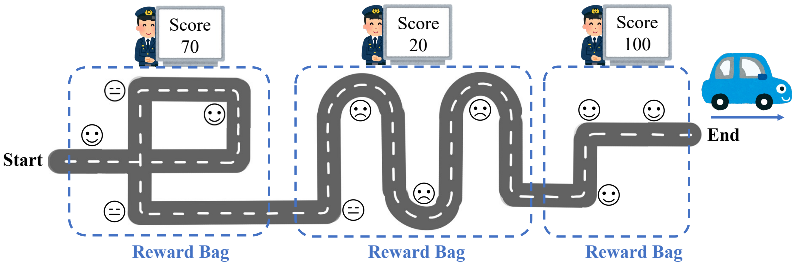


Figure 1: An example of the reward bag structure on an autonomous driving trajectory. Each segment of the driving sequence is evaluated and assigned a score by an evaluator, and the score for each sequence is integrated based on the performance at each step.

trajectory feedback [45, 1], or sparse rewards where feedback is infrequent and tied only to the current instance [36, 9, 35], tending to fail under such scenarios.

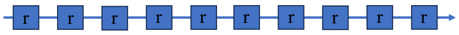
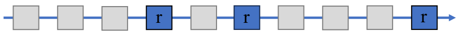
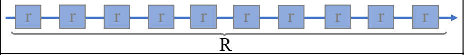
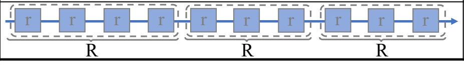
To address these challenges, we introduce *Reinforcement Learning from Bagged Rewards* (RLBR), which better aligns with real-world scenarios by considering the cumulative effect of a series of state-action pairs. In RLBR, we define sequences of state-action pairs as *bags*, each associated with a cumulative *bagged reward*. This framework includes both the traditional RL setting, where each bag only contains a single instance (the first line of Fig. 2), and the trajectory feedback setting, where a bag spans an entire trajectory (the third line of Fig. 2). 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 are time-dependent on each other and their contributions to the bagged reward vary. Given the importance of context in RLBR, we turn to the bidirectional attention mechanism [38, 43, 10], 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 context 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) [16], for environments with bagged rewards.

Our research contributes to the field in several ways. First, we establish the RLBR framework as a general problem setting and connect it theoretically to traditional Markov Decision Processes (MDPs), elaborated in Section 3. In Section 4, we introduce a Transformer-based reward model designed to assign rewards to individual instances while capturing environmental dynamics. Additionally, we propose an algorithm that alternates between optimizing this reward model and the policy, thereby enhancing the effectiveness of both components. Finally, in Section 5, our experiments show that the performance of baseline methods drops as the bag length increases, indicating that the larger bag length will increase the learning difficulty. Furthermore, we experimentally demonstrate the superiority of our method through comparative performance analyses and validate the ability of the proposed model to mimic the reward distribution of the ground truth MDP, highlighting its contextual understanding and adaptability to environmental dynamics.

## 2 Related Work

**RL with Trajectory Feedback.** RL with Trajectory Feedback (RLTF), termed episodic rewards or delayed rewards in some works (see the third line in Fig. 2), has become increasingly prominent in many applications [45, 1, 50]. A key approach to this challenge is reward redistribution, aiming to assign rewards to individual instances more effectively. Return Decomposition for Delayed Rewards (RUDDER) [1] used a return-equivalent formulation for precise credit assignment, with advancements incorporating expert demonstrations and language models [32, 25, 46]. Iterative Relative Credit Refinement (IRCR) [14] presented a uniform reward redistribution model based on equal contributions from all state-action pairs, while Randomized Return Decomposition (RRD) [34] proposed a novel upper bound for return-equivalent assumptions, integrating return decomposition with uniform reward redistribution. Additionally, Han et al. [17] modified RL algorithms to use sequence-level information, helping agents learn from broader structures and long-term outcomes. However, our focus is on

| Types of Reward | Toy Examples of Trajectories (Length $N$) | Markovian | Reward Frequency |
|---|---|---|---|
| Immediate Reward (Traditional RL) |  | Yes | $N$ |
| Sparse Reward |  | Yes | $[1, N]$ |
| Trajectory Feedback |  | No | 1 |
| Bagged Reward |  | No | $[1, N]$ |

r Instance with observed reward. r Instance with latent reward. ☐ Instance with no reward.

Figure 2: Comparison of four different types of reward settings in RL. In traditional RL, a reward is given based on each instance [40, 44], whereas in sparse reward settings, only some instances receive rewards [36, 9, 35]. Both types of rewards are Markovian. Trajectory feedback and the proposed bagged reward both address non-Markovian situations; however, trajectory feedback provides only one reward signal for the entire trajectory [45, 1], while bagged rewards can include multiple reward signals within the sequence.

the original, unobservable rewards within reward bags, not the aggregated bagged rewards. Our experiments (see Section 5) show that the method by Han et al. [17] is ineffective for long sequences.

Following the previous methodology, we adopt a reward redistribution learning strategy to enhance policy learning in the context of reward bags. However, conventional reward redistribution methods fail to effectively extract information from bag-level rewards, as they did not take the structure of the *bag* into consideration. Our experimental results demonstrate that directly applying previous methods to the reward bag setting does not yield results as promising as those achieved by our algorithm.

**RL with Sparse Rewards.** The sparse reward setting (see the second line in Fig. 2) 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 [29, 18, 41] added rewards to actions in a way that guides the agent towards better policies without altering the original reward function. Curiosity-driven methods [31, 37] encouraged agents to explore the environment by visiting unseen states, potentially solving tasks with sparse rewards. Additionally, curriculum learning in RL [12, 36] 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.

In contrast to sparse rewards, which are typically infrequent Markovian signals associated with individual state-action pairs, bagged rewards depend on the cumulative effect of sequences of state-action pairs and are non-Markovian, requiring context from the entire sequence to accurately assess the contribution of each instance.

**Transformers for RL.** Transformers, introduced by Vaswani et al. [43], have proven effective in RL environments requiring high sample efficiency and generalization, such as StarCraft [44, 48] and DMLab30 [30]. They have also been used for sequential modeling in offline RL [7, 20] and as reward models in policy learning from offline datasets [22]. Luo et al. [27] combined deep convolution transfer learning models and inverse RL for reward function acquisition, while Zhang et al. [49] transformed non-Markovian reward processes into Markovian ones, enhancing online interaction efficiency. For reward bags, where rewards are linked to sequences of state-action pairs, Transformers are ideal due to their ability to handle long-range dependencies. Our approach leverages a Transformer-based model to redistribute bag-level rewards into instance-level rewards, capitalizing on the Transformer's proficiency in sequential data analysis and attention mechanisms.

## 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 MDP, the *Bagged Reward MDP* (BRMDP). We further conduct a theoretical analysis of the relationship between the BRMDP and the traditional MDP, showing that solving RLBR problems not only relaxes the demands on reward acquisition but also guarantees finding the optimal policy even with limited reward information.

### 3.1 Preliminaries

We consider finite-horizon RL in this paper, which is traditionally modeled using a finite MDP, where rewards are promptly provided for each state-action pair [40]. This paradigm is encapsulated in a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \mu)$, with $s \in \mathcal{S}$ and $a \in \mathcal{A}$ as the sets of states and actions, $P$ as the state transition probability function, $r$ as the immediate reward function, and $\mu$ as the initial state distribution. The primary objective in this framework is to discover a policy $\pi = p(a|s)$ that maximizes the cumulative sum of rewards over a horizon length $T$:

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

Distinct from this traditional approach, RLTF offers feedback only after a complete sequence of actions, or a trajectory [1, 50]. A trajectory $\tau = \{(s_0, a_0), (s_1, a_1), \ldots, (s_{T-1}, a_{T-1})\}$ includes $T$ state-action pairs, with a reward $R_{\mathrm{traj}}(\tau)$ that is the sum of latent immediate rewards $\sum_{t=0}^{T-1} r(s_t, a_t)$, but only the cumulative reward is observable at the end of the trajectory. Denoting by $\mathcal{T}(\pi)$ as the distribution of trajectories induced by $\pi, P, \mu$, the learning objective in RLTF is to maximize the expected trajectory-based reward:

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

### 3.2 Problem Formulation

We formally define the setting of RLBR, which has a granularity of rewards in between the above two settings. First, we define the notion of *bags*, which are sub-pieces of complete trajectories. A trajectory $\tau$ is divided into several neighboring bags, and a bag of size $n_i$, which starts from time $i$, is defined as $B_{i,n_i} = \{(s_i, a_i), \ldots, (s_{i+n_i-1}, a_{i+n_i-1})\}, 0 \le i \le i + n_i - 1 \le T - 1$. 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 sets of states and actions.*
- $P$ *is the state transition probability function.*
- $R$ *denotes the reward function over bagged reward:* $R(B_{i,n_i}) = \sum_{t=i}^{i+n_i-1} r(s_t, a_t)$*.*
- $\mu$ *represents the initial state distribution.*

The essential properties of BRMDP are two-fold: 1) only bagged rewards can be obtained by the learner, not immediate rewards; 2) the bagged reward is the accumulated sum of immediate rewards within the bag. In the RLBR framework, a bag $B_{i,n_i}$ metaphorically aggregates individual rewards from a contiguous sequence of state-action pairs into a unified reward unit. A trajectory $\tau$ 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: $\mathcal{G} : \tau \to \mathcal{B}_\tau$, 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, if each bag comprises a single instance ($n_i = 1, \forall 0 \le i \le T - 1$), RLBR simplifies to standard RL. Conversely, if a single bag encompasses the entire trajectory ($n_0 = T$), RLBR reduces to RLTF. This adaptability highlights the capacity of the RLBR framework to accommodate varying reward structure scenarios.

### 3.3 Equivalence of Optimal Policies in the BRMDP and the Original MDP

In the BRMDP, agents are unable to directly observe the immediate reward linked to each state-action pair. Nonetheless, we aim for agents to optimize policies in environments where bagged rewards are accessible and to perform well on the original task. To unify the learning objectives of the BRMDP

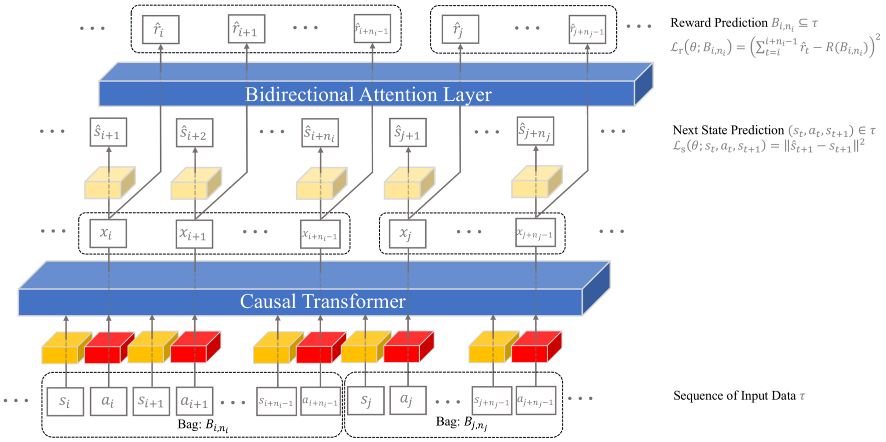

$\hat{r}_i$
$\hat{r}_{i+1}$
$\hat{r}_{i+n_i-1}$
$\hat{r}_j$
$\hat{r}_{j+n_j-1}$
Reward Prediction $B_{i,n_i} \subseteq \tau$
$\mathcal{L}_{\mathrm{r}}(\theta; B_{i,n_i}) = \left(\sum_{t=i}^{i+n_i-1} \hat{r}_t - R(B_{i,n_i})\right)^2$
Bidirectional Attention Layer
$\hat{s}_{i+1}$
$\hat{s}_{i+2}$
$\hat{s}_{i+n_i}$
$\hat{s}_{j+1}$
$\hat{s}_{j+n_j}$
Next State Prediction $(s_t, a_t, s_{t+1}) \in \tau$
$\mathcal{L}_{\mathrm{s}}(\theta; s_t, a_t, s_{t+1}) = \|\hat{s}_{t+1} - s_{t+1}\|^2$
$x_i$
$x_{i+1}$
$x_{i+n_i-1}$
$x_j$
$x_{j+n_j-1}$
Causal Transformer
$s_i$
$a_i$
$s_{i+1}$
$a_{i+1}$
$s_{i+n_i-1}$
$a_{i+n_i-1}$
$s_j$
$a_j$
$s_{j+n_j-1}$
$a_{j+n_j-1}$
Bag: $B_{i,n_i}$
Bag: $B_{j,n_j}$
Sequence of Input Data $\tau$

Figure 3: 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. Predicting the next state helps the model understand the environment, thereby improving reward prediction.

with those of the traditional MDP, we formulate a theorem that crystallizes this relationship. The significance of this theorem lies in its assertion that the optimal policy derived in a BRMDP setting is congruent with that of an MDP when the bagged rewards are considered as cumulative sums. This finding not only validates the theoretical underpinnings of our approach but also affirms its practical relevance across various RL paradigms.

**Theorem 1.** *Consider a BRMDP where the bagged reward is defined as the sum of rewards for state-action pairs from the corresponding MDP contained within the same bag. In this context, the set of optimal policies for the standard MDP* $\Pi$*, aligns with that of the BRMDP* $\Pi_{\mathrm{B}}$*, implying that* $\Pi = \Pi_{\mathrm{B}}$.

The detailed proof of Theorem 1 is deferred to Appendix B. By this theorem, we can see that optimizing a policy on a BRMDP is equivalent to optimizing on the original MDP. The crux of the matter is how to accurately redistribute the bagged rewards so that the policies learned under these distributed rewards are more close to the policies on the original MDP.

# 4 Reward Bag Transformer

This section delves into the intricacies of leveraging the bidirectional attention mechanism [38, 43, 10] for reward redistribution, designs a Transformer architecture, and outlines a comprehensive algorithm for the cultivation of efficient policies under the BRMDP framework.

## 4.1 Reward Redistribution based on Bidirectional Attention Mechanism

Building on the foundation of Theorem 1 that optimal policies in the BRMDP and MDP are equivalent, our focus shifts to the crucial process of redistributing rewards within a bag from the BRMDP. To capture the contextual influence of each instance within the sequence, the Causal Transformer [43] is a natural choice as a sequential prediction model. Traditionally, Transformers in RL are used in a unidirectional manner [7, 19, 28], where only previous instances influence the current prediction due to the unobservability of future instances. However, given that our bagged rewards are non-Markovian and both preceding and subsequent instances affect the contribution of the current instance to the bagged reward, a bidirectional attention mechanism [38, 43, 10] becomes pivotal. 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 and effective reward redistribution.

**Algorithm 1** Policy Optimization with RBT

```
Initialize replay buffer $\mathcal{D}$, RBT parameters $\theta$.
for trajectory $\tau$ collected from the environment do
    Store trajectory $\tau$ with bag information $\{(B_{i,n_i}, R(B_{i,n_i}))\}_{B_{i,n_i}\in\mathcal{B}_\tau}$ in $\mathcal{D}$.
    Sample batches from $\mathcal{D}$ and estimate bag loss based on Eq. (7).
    Update RBT parameters $\theta$ based on the estimated loss.
    Relabel rewards in $\mathcal{D}$ using the updated RBT.
    Optimize policy using the relabeled data by off-the-shelf RL algorithms (such as SAC [16]).
end for
```

### 4.2 Proposed Approach

We introduce the *Reward Bag Transformer* (RBT), a novel approach designed for the BRMDP framework. 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.

**Causal Transformer for Reward Representation.** Referring to Fig. 3, the RBT comprises a Causal Transformer [43, 33], which maintains the chronological order of state-action pairs [7, 19]. For each time step $t$ in a sequence of $M$ time steps, the Causal Transformer, represented as a function $f$, processes the input sequence $\sigma = \{s_0, a_0, \dots, s_{M-1}, a_{M-1}\}$, 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, which in turn aids the model in better understanding the environmental dynamics.

**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 $\theta$ being the RBT parameters. This layer addresses the unidirectional limitation of the Causal Transformer architecture [43, 33], integrating past and future contexts for enhanced reward prediction accuracy. For state prediction, $x_t$ 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 embeddings for query $\mathbf{q}_t \in \mathbb{R}^d$, key $\mathbf{k}_t \in \mathbb{R}^d$, and value $v_t \in \mathbb{R}$. Then the instance-level reward is calculated by

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

where $d$ is the embedding dimension of the key. The rescaling operation is used to prevent extremely small gradients as in Vaswani et al. [43]. This mechanism enables the RBT to consider both the immediate and contextual relevance of each state-action pair in the trajectory when predicting rewards.

### 4.3 Learning Objectives

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

**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 for this objective is expressed as

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

where $B_{i,n_i}$ represents the reward bag starting at time step $i$ with length $n_i$, and $R(B_{i,n_i})$ is the total reward for this bag. This formulation encourages the RBT to learn a nuanced distribution of rewards across states and actions within a bag. At the same time, it ensures that the sum of redistributed rewards matches the total bagged reward, maintaining consistency as per Theorem 1.

**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. 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 $\ell^2$-norm. This loss emphasizes the model's understanding of dynamics.

**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 the coefficient $\beta > 0$ balances 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 the ablation study in Appendix D.2. Integrated with off-the-shelf RL algorithms such as SAC [16], the RBT can enhance policy learning within the BRMDP framework, as outlined in Algorithm 1.

## 5 Experiment

In the following experimental section, we scrutinize the efficacy of our proposed method using benchmark tasks from both the MuJoCo [2] and the DeepMind Control Suite [42] environments, focusing on scenarios with bagged rewards. We assess the performance of our method to understand its overall effectiveness and examine whether the RBT reward model accurately predicts rewards.

### 5.1 Compare with SOTA Methods

**Experiment Setting.** We evaluated our method on eight benchmark tasks from both the MuJoCo locomotion suite, including Ant-v2, Hopper-v2, HalfCheetah-v2, and Walker2d-v2, and the DeepMind Control Suite, including cheetah-run, quadruped-walk, fish-upright, and cartpole-swingup. Differing from standard environments where rewards are assigned at each step, our approach involved assigning a cumulative reward at the end of each bag while assigning a reward of zero to all other state-action pairs within the bag. The maximum length for each episode was fixed at 1000 steps across all tasks.

**Baselines.** In the comparative analysis, our framework was rigorously evaluated against several leading algorithms in the domain of RL with delayed reward:

- **SAC** [16]: It directly utilized the original bagged reward information for policy training using the SAC algorithm.
- **IRCR** [14]: It adopted a non-parametric uniform reward redistribution approach by using the sum of immediate per-step rewards as a stand-in for trajectory returns. We have adapted IRCR for bagged reward setting.
- **RRD** [34]: It employed a reward model trained with a randomized return decomposition loss. We have adapted RRD for bagged reward setting.
- **LIRPG** [51]: 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.
- **HC** [17]: The HC-decomposition framework was utilized to train the policy using a value function that operates on sequences of data. This approach decoupled the value function approximation task for the current step from the historical trajectory. We employed the code as provided by the original paper.

While methods like RUDDER [1] and Align-RUDDER [32] are known for addressing the problem of trajectory feedback, previous studies [14, 34, 50] 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 [32], we ultimately excluded both methods from our comparison. Besides, detailed descriptions of the model parameters and hyper-parameters used during training are provided in Appendix C. More experimental results are included in Appendix D.1.

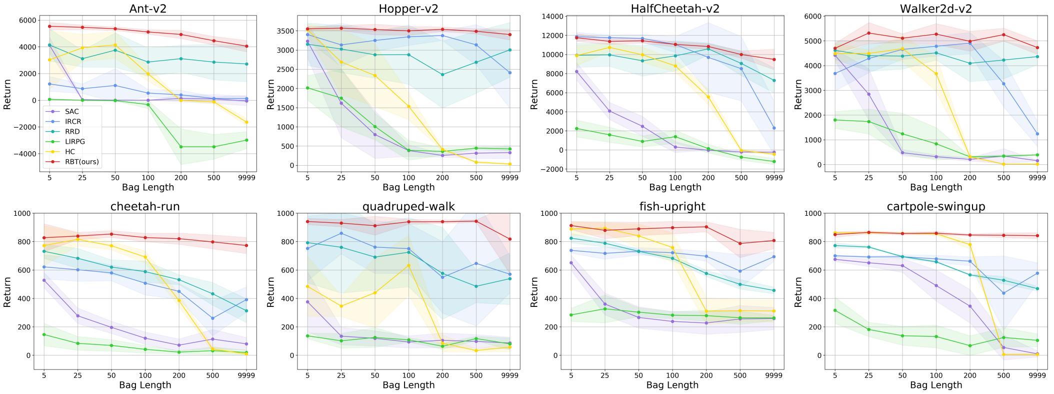


Figure 4: Performance comparison in MuJoCo (top row) and DeepMind Control Suite (bottom row) environments with six different fixed-length reward bag settings (5, 25, 50, 100, 200, and 500) and trajectory feedback (labeled as 9999). The mean and standard deviation are displayed over 6 trials with different random seeds across a total of 1e6 time steps.

Table 1: Performance comparison across arbitrary 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 | LIRPG | HC | RBT(ours) |
|---|---|---|---|---|---|---|---|
| Narrow | Ant-v2 | 0.87<br>(2.98) | 368.69<br>(119.74) | 2272.39<br>(835.86) | -756.78<br>(763.66) | 106.92<br>(153.86) | **5122.50**<br>**(206.44)** |
| | Hopper-v2 | 317.72<br>(52.17) | 3353.35<br>(61.97) | 2184.41<br>(807.71) | 126.13<br>(30.18) | 510.66<br>(94.49) | **3499.54**<br>**(76.62)** |
| | HalfCheetah-v2 | 788.45<br>(1737.57) | **10853.85**<br>**(573.72)** | 9709.62<br>(1479.73) | 1101.38<br>(1248.45) | 4027.25<br>(441.01) | **11282.24**<br>**(266.08)** |
| | Walker2d-v2 | 193.07<br>(48.40) | 4144.65<br>(673.66) | 3536.90<br>(546.66) | 123.43<br>(50.97) | 309.19<br>(171.69) | **4983.39**<br>**(311.09)** |
| Broad | Ant-v2 | -3.31<br>(4.15) | 368.69<br>(158.46) | 1323.50<br>(1079.60) | -1264.08<br>(416.86) | 5.97<br>(20.08) | **5167.79**<br>**(303.83)** |
| | Hopper-v2 | 329.48<br>(44.21) | 3296.20<br>(216.35) | 1102.38<br>(892.12) | 203.01<br>(177.80) | 701.84<br>(149.44) | **3499.53**<br>**(94.00)** |
| | HalfCheetah-v2 | 43.96<br>(94.32) | 9158.14<br>(1402.62) | 4199.16<br>(1476.85) | 924.26<br>(1110.97) | 4460.80<br>(518.94) | **10837.15**<br>**(254.99)** |
| | Walker2d-v2 | 176.09<br>(49.81) | 4179.08<br>(937.42) | 330.96<br>(79.26) | 194.95<br>(98.05) | 447.45<br>(155.63) | **5202.38**<br>**(248.35)** |

### 5.1.1 Fixed-Length Reward Bags

In the fixed-length reward bag experiment, we conducted experiments with six bag lengths (5, 25, 50, 100, 200, and 500) and trajectory feedback (labeled as 9999) across each environment. The experimental design followed the problem setting, which assumed that the conclusion of one reward bag directly preceded the beginning of the next. It aimed to illustrate the influence of varying bag lengths on the results, providing insight into how bag size affected the performance of the learning algorithm within these environments.

As shown in Fig. 4, the SAC method, using bagged rewards directly from the environment, suffers from a lack of guidance in agent training due to missing 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 approximate guidance. However, notable variance in their results indicates potential consistency and reliability issues. The LIRPG exhibits subdued performance across tasks, as it is proposed to solve sparse reward problems [51], which is Markovian and does not align the reward bag setting. The HC method excels only with shorter bag lengths, suggesting that this value function modification method struggles to utilize information from longer sequences. The proposed RBT method consistently outperforms the other approaches across all the environments and

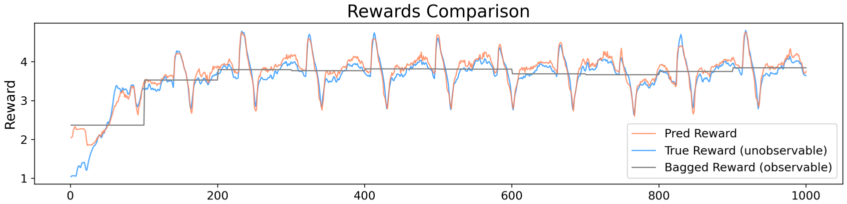


Figure 5: Comparison of predicted rewards with true rewards and aggregated bagged rewards.

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.

### 5.1.2 Arbitrary Reward Bags

To validate the effectiveness of our approach under more complex conditions, we designed an experiment that allowed for overlaps or gaps between reward bags, and the length of each bag was no longer fixed. This setup simulated more realistic scenarios and tested the robustness of our method.

The results, as detailed in Table 1, affirm the superior performance of the proposed RBT method in these complex reward settings. Notably, other baseline methods that rely on reward model training experience a significant drop in effectiveness, primarily due to sample scarcity impacting model accuracy and policy learning. In contrast, the IRCR method, which distributes rewards uniformly without a model, maintains its efficacy in some environments. This outcome suggests that while approximate rewards can still guide policy learning, incorrect rewards can be highly detrimental. The consistent success of our approach in various reward bag scenarios indicates its potential for application to a broader range of reward structures, highlighting its versatility and robustness in handling more intricate 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. As shown in Fig. 5, 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.

## 6 Conclusion

In this paper, we introduce a general learning framework, Learning from Bagged Rewards (RLBR), and make theoretical connections between our learning objectives and traditional MDPs to ensure the justification of our approach. Building on this problem, we propose a Transformer-based reward model, the Reward Bag Transformer (RBT), to efficiently redistribute rewards by interpreting contextual information within bags as well as understanding environmental dynamics. The efficacy of RBT is demonstrated through extensive experiments, outperforming existing delayed-reward methods in various reward bag scenarios. Besides, our case studies highlight the RBT's ability to effectively reallocate rewards, maintaining fidelity to the original MDP structure. While the sum-form bagged reward currently integrated into RLBR addresses most scenarios, as evidenced by previous studies on trajectory feedback [34, 1, 51], there is value in exploring other forms of reward aggregation. Future research will investigate alternative reward structures, such as maximum values or complex combinations of latent rewards, to better capture the nuances of dynamic real-world environments. This exploration aims to enhance the adaptability and effectiveness of the RLBR framework in a broader range of applications.

# References

[1] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] X.-Q. Cai, Y.-X. Ding, Y. Jiang, and Z.-H. Zhou. Imitation learning from pixel-level demonstrations by hashreward. In *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pages 279–287, 2021.

[4] X.-Q. Cai, Y.-X. Ding, Z.-X. Chen, Y. Jiang, M. Sugiyama, and Z.-H. Zhou. Seeing differently, acting similarly: Heterogeneously observable imitation learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

[5] X.-Q. Cai, P. Zhang, L. Zhao, B. Jiang, M. Sugiyama, and A. J. Llorens. Distributional pareto-optimal multi-objective reinforcement learning. In *The Thirty-seventh Conference on Neural Information Processing Systems, NeurIPS 2023, New Orleans, US, Dec. 10-16, 2023*, 2023.

[6] X.-Q. Cai, Y.-J. Zhang, C.-K. Chiang, and M. Sugiyama. Imitation learning from vague feedback. In *The Thirty-seventh Conference on Neural Information Processing Systems, NeurIPS 2023, New Orleans, US, Dec. 10-16, 2023*, 2023.

[7] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

[8] Z.-X. Chen, X.-Q. Cai, Y. Jiang, and Z.-H. Zhou. Anomaly guided policy learning from imperfect demonstrations. In *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pages 244–252, 2022.

[9] R. Devidze, P. Kamalaruban, and A. Singla. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in neural information processing systems*, 2022.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *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)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[11] J. Early, T. Bewley, C. Evers, and S. Ramchurn. Non-markovian reward modelling from trajectory labels via interpretable multiple instance learning. *Advances in Neural Information Processing Systems*, 35, 2022.

[12] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*. PMLR, 2018.

[13] R. Frostig, M. J. Johnson, and C. Leary. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 4(9), 2018.

[14] T. Gangwani, Y. Zhou, and J. Peng. Learning guidance rewards with trajectory-space smoothing. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] M. Gaon and R. Brafman. Reinforcement learning with non-markovian rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 2020.

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, 2018.

[17] B. Han, Z. Ren, Z. Wu, Y. Zhou, and J. Peng. Off-policy reinforcement learning with delayed rewards. In *International Conference on Machine Learning*. PMLR, 2022.

[18] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33, 2020.

[19] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.

[20] M. Janner, Q. Li, and I. Mordatch. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*, 2021.

[21] N. R. Ke, A. G. ALIAS PARTH GOYAL, O. Bilaniuk, J. Binas, M. C. Mozer, C. Pal, and Y. Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding. *Advances in neural information processing systems*, 31, 2018.

[22] C. Kim, J. Park, J. Shin, H. Lee, P. Abbeel, and K. Lee. Preference transformer: Modeling human preferences using transformers for RL. In *International Conference on Learning Representations*, 2023.

[23] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

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

[25] Y. Liu, Y. Luo, Y. Zhong, X. Chen, Q. Liu, and J. Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.

[26] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[27] W. Luo, J. Zhang, P. Feng, D. Yu, and Z. Wu. A Deep Transfer-Learning-Based Dynamic Reinforcement Learning for Intelligent Tightening System . *International Journal of Intelligent Systems*, 2021.

[28] V. Micheli, E. Alonso, and F. Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

[29] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*. PMLR, 1999.

[30] E. Parisotto, H. F. Song, J. W. Rae, R. Pascanu, C. Gulcehre, S. M. Jayakumar, M. Jaderberg, R. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. *arXiv preprint arXiv:2006.10729*, 2020.

[31] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*. PMLR, 2017.

[32] V. Patil, M. Hofmarcher, M.-C. Dinu, M. Dorfer, P. M. Blies, J. Brandstetter, J. Arjona-Medina, and S. Hochreiter. Align-rudder: Learning from few demonstrations by reward redistribution. In *International Conference on Machine Learning*. PMLR, 2022.

[33] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[34] Z. Ren, R. Guo, Y. Zhou, and J. Peng. Learning long-term reward redistribution via randomized return decomposition. In *International Conference on Learning Representations*, 2021.

[35] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. *arXiv preprint arXiv:2202.04628*, 2022.

[36] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*. PMLR, 2018.

[37] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*. PMLR, 2020.

[38] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*, 2016.

[39] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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

[41] P. Tambwekar, M. Dhuliawala, L. J. Martin, A. Mehta, B. Harrison, and M. 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.

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

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[44] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.

[45] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[46] M. Widrich, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, and S. Hochreiter. Modern hopfield networks for return decomposition for delayed rewards. In *Deep RL Workshop NeurIPS 2021*, 2021.

[47] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–8, 2020.

[48] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, et al. Deep reinforcement learning with relational inductive biases. *International Conference on Learning Representations*, 2019.

[49] H. Zhang, H. Wang, and Z. Kan. Exploiting transformer in sparse reward reinforcement learning for interpretable temporal logic motion planning. *IEEE Robotics and Automation Letters*, 2023.

[50] Y. Zhang, Y. Du, B. Huang, Z. Wang, J. Wang, M. Fang, and M. Pechenizkiy. Interpretable reward redistribution in reinforcement learning: A causal approach. volume 36, 2023.

[51] Z. Zheng, J. Oh, and S. Singh. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.

# A Broader Impact

In this work, we introduce the problem of Reinforcement Learning from Bagged Reward (RLBR), and propose the Reward Bag Transformer to address it. On the one hand, we recognize that these techniques could raise some potential issues. As collecting bagged rewards is much more convenient and natural than gathering instance-level rewards, this could lead to some risks of abusing unauthorized data. On the other hand, we believe that developing these techniques is still necessary for the importance of solving reinforcement learning tasks with bag-level feedback. Furthermore, there are many techniques for preserving data privacy, which can be compatible with our approach to avoid such problems.

# B Omitted Proof

**Theorem 1.** *Consider a BRMDP where the bagged reward is defined as the sum of rewards for state-action pairs from the corresponding MDP contained within the same bag. In this context, the set of optimal policies for the standard MDP* $\Pi$*, aligns with that of the BRMDP* $\Pi_{\mathrm{B}}$*, implying that* $\Pi = \Pi_{\mathrm{B}}$.

*Proof.* Let us consider the structure of bagged rewards within BRMDP. For a given bag $B_{i,n_i}$, the bagged reward $R(B_{i,n_i})$ is the sum of the individual rewards of the state-action pairs contained within it, given by:

$$R(B_{i,n_i}) = \sum_{t=i}^{i+n_i-1} r(s_t, a_t).$$

Over a complete trajectory $\tau$, the cumulative reward in BRMDP can be expressed as the sum of the rewards from all the bags along the trajectory:

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

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 standard cumulative reward in MDP, $J(\pi)$:

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

Given that the expected cumulative rewards for any policy in the BRMDP and 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 $\Pi = \Pi_{\mathrm{B}}$. □

# C 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 [2] and the DeepMind Control Suite [42], 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.

In this setting, the agent received a bagged reward at the end of each collected bag. 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 [14, 34].

Table 2: 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 [26] |
| 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) [16]. 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 [24], which is available under the MIT License.

The RBT reward model was developed based on the GPT implementation in JAX [13], 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 2.

For the baseline methods, the IRCR [14] method was implemented based on the descriptions provided in the original paper. The RRD [34] and LIRPG [51] methods are both licensed under the MIT License. The code of HC [17] 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.

# D Additional Experimental Results

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

## D.1 Experimental Result of Various-Length Reward Bags

In Table 3, 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

Table 3: 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 | LIRPG | HC | RBT(ours) |
|---|---|---|---|---|---|---|---|
| Short | Ant-v2 | 3.21<br>(1.97) | 269.12<br>(224.66) | 2661.56<br>(1675.65) | -1407.15<br>(504.59) | 20.03<br>(46.35) | **5359.85**<br>**(129.28)** |
| | Hopper-v2 | 286.53<br>(72.51) | 3275.05<br>(44.68) | 2508.76<br>(690.07) | 287.38<br>(114.65) | 474.21<br>(66.93) | **3433.06**<br>**(96.89)** |
| | HalfCheetah-v2 | 5.92<br>(23.06) | 10480.33<br>(202.27) | **10382.80**<br>**(516.85)** | 1225.35<br>(162.88) | 3982.43<br>(433.75) | **11073.88**<br>**(181.43)** |
| | Walker2d-v2 | 222.21<br>(66.72) | 3840.30<br>(666.24) | 3999.14<br>(561.49) | 328.97<br>(107.69) | 348.36<br>(174.28) | **5198.09**<br>**(225.50)** |
| Long | Ant-v2 | -115.10<br>(138.93) | 215.38<br>(92.94) | 2600.64<br>(1229.27) | -2552.99<br>(419.44) | -0.53<br>(5.07) | **4897.50**<br>**(292.93)** |
| | Hopper-v2 | 360.36<br>(118.06) | 3015.96<br>(408.08) | 3089.52<br>(433.23) | 325.84<br>(50.25) | 652.28<br>(92.41) | **3447.64**<br>**(83.02)** |
| | HalfCheetah-v2 | -115.56<br>(35.01) | 5944.17<br>(3421.46) | 8591.38<br>(1048.08) | -571.35<br>(374.64) | 4563.99<br>(568.19) | **10880.76**<br>**(441.65)** |
| | Walker2d-v2 | 251.11<br>(144.24) | 3397.93<br>(682.19) | 4221.37<br>(282.75) | 284.27<br>(7.54) | 497.27<br>(134.06) | **4979.07**<br>**(166.95)** |

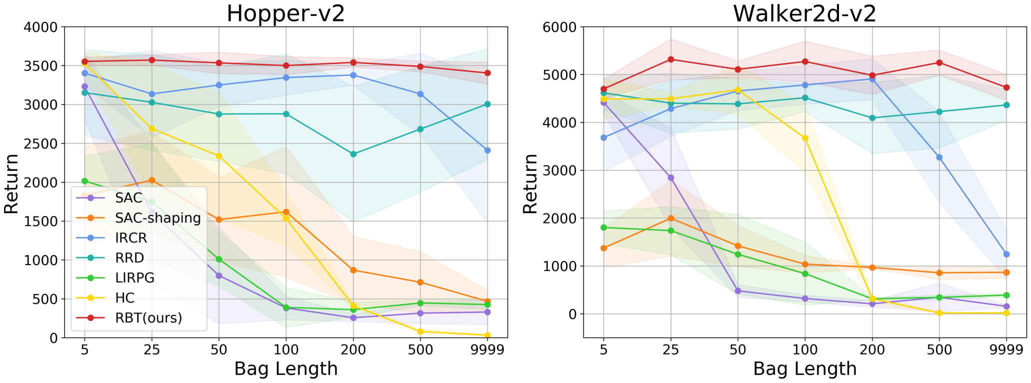


Figure 6: Performance comparison adding SAC with reward shaping (SAC-shaping) in the Hopper-v2 and Walker2d-v2 environments, displaying both the mean and standard deviation over 6 trials with different random seeds. The experiments are conducted across six specific fixed bag lengths: 5, 25, 50, 100, 200, and 500, as well as with trajectory feedback, labeled as 9999, in each environment.

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 Section 5.1.

### D.2 Ablation Study

We conducted comparisons to examine the role of RBT’s modules. As shown in Fig. 7, 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.

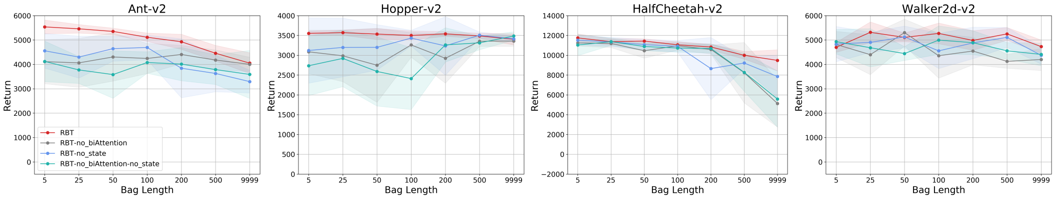


Figure 7: Ablation study of reward model components across various environments. The chart presents mean and standard deviation of rewards over 6 trials with 1e6 timesteps, showcasing the efficacy of the full proposed method relative to its variants without certain features.

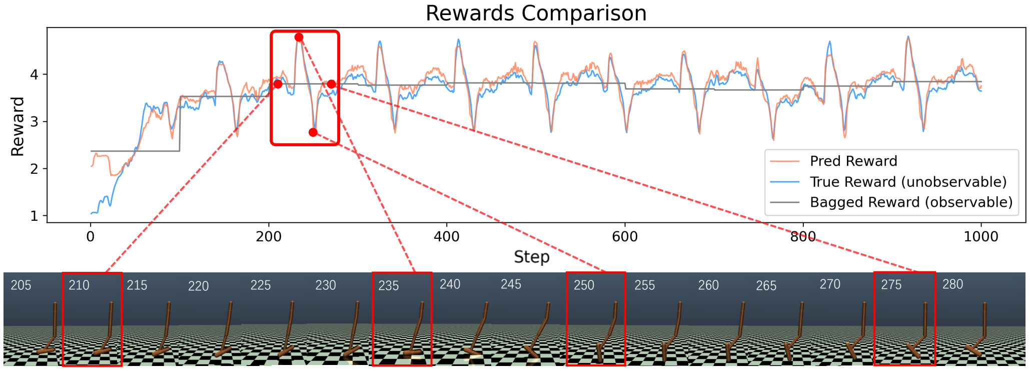


Figure 8: Rewards comparison and agent states in a trajectory with a bag length of 100 in the Hopper-v2 environment. The top graph compares predicted rewards against true rewards and aggregated bagged rewards.

### D.3 Rewards Comparison in Hopper-v2 with Bagged Rewards

Fig. 8 shows a comparison of predicted rewards, true rewards, and aggregated bagged rewards for a trajectory with a bag length of 100 in the Hopper-v2 environment. It shows how well the predicted rewards align with the true and bagged rewards over the course of the trajectory, highlighting the effectiveness of the reward model.

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

### D.4 Reward Shaping

To complement our findings, we conducted additional experiments on reward shaping, following the naive shaping approach described in Hu et al. [18]. These experiments were performed in the Hopper-v2 and Walker2d-v2 environments due to the availability of task-specific weights from the original paper. As shown in the Fig. 6, the performance of SAC with reward shaping (SAC-shaping) varies with different bag lengths.

At shorter bag lengths, SAC-shaping tends to perform worse than standard SAC, possibly because the more frequent but less informative rewards add noise to the training process. However, as bag length increases, SAC-shaping slightly outperforms SAC, likely because the shaped rewards offer clearer, longer-term signals that assist in learning effective policies when rewards are less frequent. Essentially,

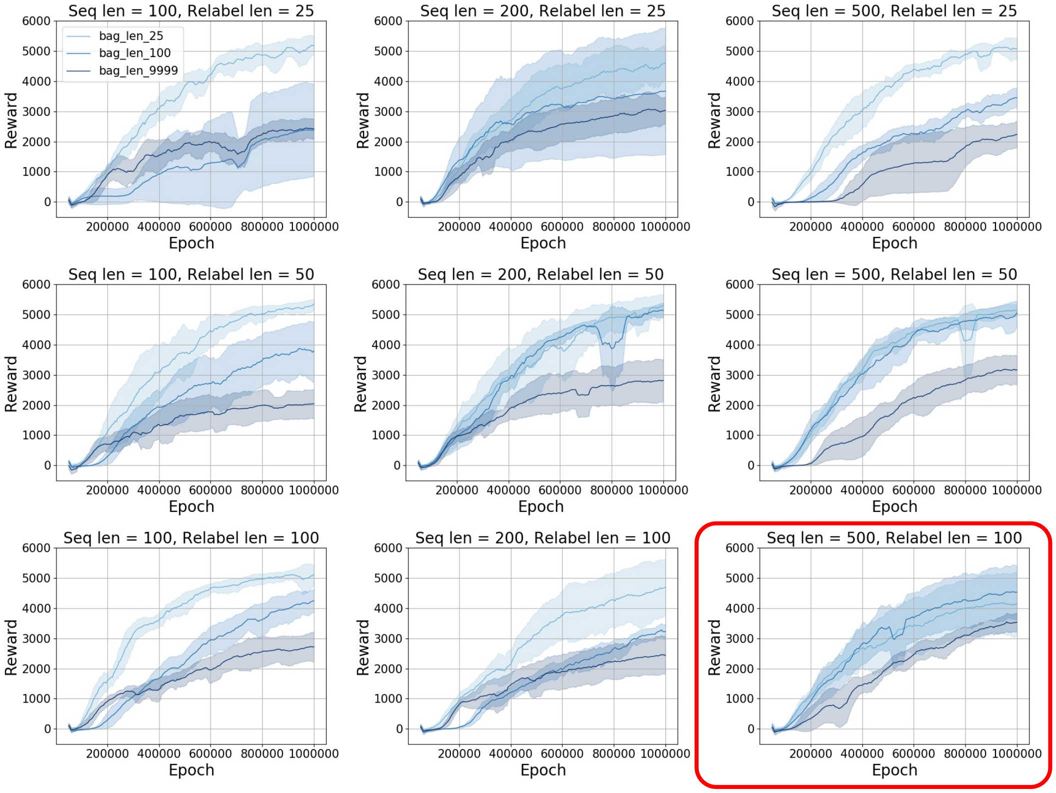


Figure 9: Learning curves on Ant-v2 with different length of input sequence in training (Seq len) and predict length during relabeling process (Relabel len), based on 3 independent runs with random initialization. Within each of the smaller graphs, the curves represent results from experiments with different 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.

reward shaping at longer bag lengths creates a more stable and beneficial learning signal for policy improvement. The figure demonstrates that while reward shaping improves SAC performance with larger bag sizes, our RBT method consistently outperforms both. RBT excels because it not only smooths the rewards but also redistributes them by considering the context within each bag. This comprehensive approach allows RBT to more accurately capture environmental complexities, leading to better performance across different bag lengths, indicating its robustness and efficiency in utilizing extended sequences of data.

### D.5 Architecture Sensitivity

Fig. 9 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.