Bayesian Reparameterization of Reward-Conditioned Reinforcement Learning with Energy-based Models

Wenhao Ding [*1] **Tong Che** [*2] **Ding Zhao** [1] **Marco Pavone** [2 3]

Abstract

Recently, reward-conditioned reinforcement learning (RCRL) has gained popularity due to its simplicity, flexibility, and off-policy nature. However, we will show that current RCRL approaches are fundamentally limited and fail to address two critical challenges of RCRL – improving generalization on high reward-to-go (RTG) inputs, and avoiding out-of-distribution (OOD) RTG queries during testing time. To address these challenges when training vanilla RCRL architectures, we propose Bayesian Reparameterized RCRL (BR-RCRL), a novel set of inductive biases for RCRL inspired by Bayes' theorem. BR-RCRL removes a core obstacle preventing vanilla RCRL from generalizing on high RTG inputs – a tendency that the model treats different RTG inputs as independent values, which we term "RTG Independence". BR-RCRL also allows us to design an accompanying adaptive inference method, which maximizes total returns while avoiding OOD queries that yield unpredictable behaviors in vanilla RCRL methods. We show that BR-RCRL achieves state-of-the-art performance on the Gym-Mujoco and Atari offline RL benchmarks, improving upon vanilla RCRL by up to 11%.

1. Introduction

Reinforcement learning (RL) aims at learning policies to maximize cumulative rewards by trial and error (Sutton et al., 1998). It was shown that when combined with deep neural networks, deep RL is able to learn powerful policies for complex decision-making tasks using only self-generated data (Mnih et al., 2013; Akkaya et al., 2019; Kiran et al., 2021). RL algorithms can be divided into two classes in terms of data usage: on-policy RL (Schulman et al., 2017; 2015) algorithms have to be trained on data that the current policy learner generates, while off-policy RL algorithms (Haarnoja et al., 2018; Lillicrap et al., 2015) can be trained on data that is generated by some different policies. It is a common belief that off-policy algorithms are more sample efficient than on-policy algorithms (Prudencio et al., 2022).

One recent proposed off-policy RL paradigm is reward-conditioned reinforcement learning (RCRL) (Kumar et al., 2019b; Chen et al., 2021; Janner et al., 2021; Srivastava et al., 2019; Ajay et al., 2022), which transforms the RL problem into a conditional sequence modeling problem. The general idea of RCRL is straightforward: we train an RTG (reward-to-go)-conditioned generative model using off-policy data and then set a target RTG when we roll out the policy during training to collect more data (Kumar et al., 2019b) or during testing (Chen et al., 2021; Janner et al., 2021). RCRL has gained popularity due to its conceptual simplicity, flexibility, and off-policy nature. Moreover, its usage allows us to leverage powerful neural architectures (e.g., Transformers (Vaswani et al., 2017)) and large generative models (e.g., diffusion models (Yang et al., 2022) or masked language models (Ghazvininejad et al., 2019)), which have been a massive success in other parts of artificial intelligence, such as natural language processing (Brown et al., 2020; Ouyang et al., 2022) and computer vision (Dosovitskiy et al., 2020; Ramesh et al., 2022).

Vanilla RCRL paradigms treat RTGs as standard inputs in addition to the states and actions of the neural network. This design choice makes it easy to employ modern architectures such as Transformers (Vaswani et al., 2017). However, this practice, although plausible, has largely ignored the central challenge of RCRL. Training the RCRL model optimizes a policy to fit a dataset or data buffer. However, during policy rollout, we usually want to set a high target RTG in the hope that the model can achieve higher performance than the generating policy of the dataset. In other words, the central challenge of RCRL methods is to achieve generalization from low-return regions to high-return regions of the state

[*]Equal contribution [1]Carnegie Mellon University, Pittsburgh, PA, US [2]NVIDIA Research, Santa Clara, CA, US [3]Stanford University, Palo Alto, CA, US. Correspondence to: Wenhao Ding <wenhaod@andrew.cmu.edu>, Tong Che <tongc@nvidia.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

space. One core generalization obstacle in vanilla RCRL is that the RTG inputs carry too little information. The model tends to treat different RTGs inputs as independent and unrelated tasks and then fails to find internal connections between trajectories with different RTG levels. In most cases, when we want to learn from sub-optimal datasets or data buffers, high total return trajectories are scarce. Thus, this problem makes learning and generalization in high-return regions extremely difficult.

In this paper, we propose Bayesian Reparameterized RCRL (BR-RCRL). Our main intuition behind the design is that (1) in order to facilitate generalization from low-return regions to high-return regions, one needs to encode more inductive biases into the model, especially on how to deal with RTGs. (2) one needs to modify the rollout procedure of RCRL to filter out completely out-of-distribution inputs. BR-RCRL is a novel set of inductive biases to parameterize reward-conditioned policies inspired by Bayes' theorem. In BR-RCRL, we explicitly impose the prior knowledge that different RTGs are competitive with each other, not independent. From a causality perspective, BR-RCRL can also be viewed as a *causal generative model* (Peters et al., 2017) that respects the ground-truth causal relationships between random variables. This causal viewpoint provides a concrete explanation of why BR-RCRL generalizes better than vanilla RCRL models when distribution shifts occur.

In the experiment section, we show that BR-RCRL dramatically boosts the performance of many RCRL algorithms across commonly used offline RL benchmarks.

2. Related Works and Preliminaries

2.1. Off-policy and Offline RL

Reinforcement learning studies learning problems in the setting of a Markov Decision Process (MDP) described by the tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$. This tuple consists of action $a \in \mathcal{A}$, state $s \in \mathcal{S}$, transition probability $P(s'|s,a)$, and reward function $r = \mathcal{R}(s,a)$. The goal of MDP is to maximize the expected return $\mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t]$, where we denote γ the discount factor and a_t, s_t, $r_t = \mathcal{R}(s_t, a_t)$ the action, state, and reward at timestep t, respectively.

In the on-policy RL setting (Schulman et al., 2017; 2015), an agent interacts with the environment and updates its current policy using experiences gathered using the same current policy. In off-policy RL (Mnih et al., 2013; Lillicrap et al., 2015), the agent still interacts with the environment, but can update its current policy using experiences collected from any past policies as well. The off-policy framework brings us two advantages: (1) More sample-efficient training since the agent does not have to discard all previous transitions and can instead maintain a buffer where transitions can be reused multiple times. (2) Better state space exploration since the sample collection follows a behavior policy different from the target policy.

Offline RL (Prudencio et al., 2022; Levine et al., 2020), also known as Batch RL, moves one step further and becomes truly "off-policy" by learning only from static dataset $\{s_i, a_i, s'_i, r_i\}_{i=1}^N$ collected from arbitrary policies. This offline setting can be extremely valuable when an online interaction is impractical due to expensive or dangerous data collection such as robotics (Singh et al., 2021), healthcare (Liu et al., 2020b), and autonomous driving (Kiran et al., 2021). However, the main challenge in offline RL is the distributional shift between the dataset and the environment. This challenge is either addressed by constraining the learned policy to the behavior policy used to collect the dataset (Fujimoto et al., 2019; Kumar et al., 2019a; Wu et al., 2019) or estimating a conservative value function (Kumar et al., 2020; Yu et al., 2021).

2.2. Reward-Conditioned RL

A popular off-policy learning paradigm is Reward-Conditioned Reinforcement Learning, which has been studied across multiple contexts (Janner et al., 2021; Chen et al., 2021; Kumar et al., 2019b; Emmons et al., 2021; Srivastava et al., 2019; Ajay et al., 2022). We denote the data-generating behavior policy to be β, and then we define the random variable of total return (RTG) after taking action a at state s as

$$Z^{\beta}(s,a) = \sum_{t \geq 0} \gamma^t r(a_t, s_t)|_{a_0=a, s_0=s}. \tag{1}$$

RCRL tries to learn the RTG-conditioned policy $\bar{\beta}_\theta(a|R,s)$ parameterized by θ to match the ground-truth posterior policy $\beta[a|s, Z^{\beta}(s,a) = R]$, where R is a target RTG. In vanilla RCRL settings, the RTGs are treated as an input variable and directly fed into a neural network, which could be an MLP (Emmons et al., 2021), a Transformer (Chen et al., 2021; Janner et al., 2021), or a diffusion model (Ajay et al., 2022). Namely, the learned policy $\bar{\beta}_\theta(a|R,s)$ takes R and current state s and outputs a distribution of actions that matches the posterior distribution of data generating policy $\beta(a|s,R)$. More formally, given a data buffer $\{s_i, a_i, s'_i, R_i\}_{i=1}^N$, RCRL optimizes the following loss function:

$$\mathcal{L}_{\text{RCRL}}(\theta) = -\sum_i \log \bar{\beta}_\theta(a_i|s_i, R_i) \tag{2}$$

In an online RCRL setting (Kumar et al., 2019b), one interleaves the policy fitting with data collection and dynamically expands the dataset with new data collected.

2.3. Energy-based Models

Energy-based Models (EBMs) (LeCun et al., 2006; Song & Kingma, 2021), also known as non-normalized prob-

abilistic models, are flexible and can model expressive distributions since they do not have a restriction on the tractability of the normalizing constant. The density given by an EBM is $p_\theta(\boldsymbol{x}) = \exp(-E_\theta(\boldsymbol{x}))/Z_\theta$, where energy $E_\theta(x)$ is a nonlinear function parameterized by θ and $Z_\theta = \int \exp(-E_\theta(\boldsymbol{x}))d\boldsymbol{x}$ is a constant w.r.t $\boldsymbol{x}$. Although the likelihood of EBMs cannot be directly maximized, three surrogate principles for learning EBMs are usually considered. Firstly, we can estimate the gradient of the log-likelihood with MCMC approaches (Neal et al., 2011; Welling & Teh, 2011), which use the fact that the gradient of the log-probability w.r.t. $\boldsymbol{x}$ equals the gradient of the energy. Secondly, one can learn an EBM by matching the first derivatives of the density function and the data distribution (Song & Ermon, 2019; Song et al., 2020). Finally, an EBM can be learned by contrasting it with another distribution with a known density using Noise Contrastive Estimation (NCE) (Gutmann & Hyvärinen, 2010). InfoNCE (Oord et al., 2018), inspired by NCE, uses categorical cross-entropy loss to identify the positive sample $\boldsymbol{x}$ amongst a set of unrelated noise samples X'.

$$\mathcal{L}_{\text{infoNCE}} = -\mathbb{E}\left[\log \frac{f(\boldsymbol{x}, \boldsymbol{c})}{\sum_{\boldsymbol{x}' \in X'} f(\boldsymbol{x}', \boldsymbol{c})}\right] \tag{3}$$

where $\boldsymbol{c}$ is the context indicating the label of $\boldsymbol{x}$ and the scoring function is $f(\boldsymbol{x}, \boldsymbol{c}) \propto \frac{p(\boldsymbol{x}|\boldsymbol{c})}{p(\boldsymbol{x})}$.

Recently, EBM formulation has been considered the policy representation (Haarnoja et al., 2017). Existing works also use EBMs in a model-based planning framework (Boney et al., 2020) or imitation learning (Liu et al., 2020a) with an on-policy algorithm. Another trend to combine EBM and RL is utilizing an EBM as part of the RL framework (Kostrikov et al., 2021; Nachum & Yang, 2021).

3. Limitations of Vanilla RCRL

The pipeline of vanilla RCRL has two fundamental limitations. Although the loss $\mathcal{L}_{\text{RCRL}}$ is a reasonable surrogate of $-\mathbb{E}_\beta[\log \bar{\beta}(a|s, R)]$, optimizing such a loss is equivalent to maximizing the average likelihood under the distribution induced by β. However, we aim to achieve a higher reward than we can get from β during test time. The mismatch between the training and testing input distribution becomes more severe when we try to set a higher RTG for the model. In many cases, the input RTGs can become so high that they turn out to be out-of-distribution inputs (c.f. Figure 2). Thus, we argue that a fundamental challenge for RCRL is to improve the model's generalization performance on higher RTG inputs while avoiding unpredictable behavior caused by OOD inputs.

However, vanilla RCRL models lack appropriate inductive biases to facilitate such generalization. One core issue that makes the generalization to high RTG region extremely

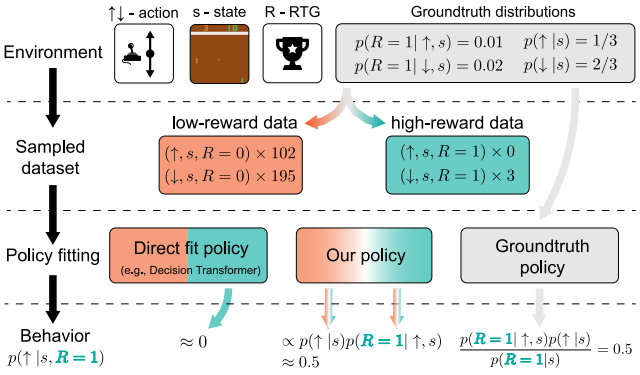

Figure 1: Illustration of the **Sampling Bias Dominance** problem. When conditioned on $R = 1$, only 3 noisy samples are available; thus, a huge sampling variance is introduced. Our policy uses an amortized estimation of $p(a|s)$ and $p(R|a, s)$, so it is less affected by the high sampling variance.

difficult is that the model could treat inputs with specific RTGs as independent, unrelated prediction problems and fail to discover the connections across trajectories with different RTGs (for which we term as **RTG Independence**). Since RTGs contain minimal information, in our experiments, we found that in contrast with what one may hope, many vanilla RCRL models (Emmons et al., 2021; Chen et al., 2021) have a tendency to treat each different RTG input simply as one independent prediction task. This phenomenon prohibits the model from learning useful information in low RTG regions and tries to generalize to high RTG regions. (The generalization is notoriously hard because, in almost all RL tasks, low-reward trajectories look very different from high-reward ones.)

In fact, these issues result in two serious problems for RCRL during both training and testing. The first problem is **Sampling Bias Dominance**. When conditioned on higher RTGs, the training samples that can attain these high RTGs become fewer. This means that empirical distribution in the dataset, when conditioned on a high RTG, can look very different from the ground-truth data-generating distribution because of the large sampling variance. The sampling variance, when combined with the RTG Independence problem, makes the generalization on high RTG regions notoriously difficult. Consider a simple example environment in Figure 1, where we have 2 possible RTG values for a given action. The RTG input tokens carry only one bit of information, and it does not provide any information or hints about how the target RTGs $R = 1$ and $R = 0$ are related to each other. The model has no choice but to view $p(a|R = 1, s)$ and $p(a|R = 0, s)$ as independent and unrelated prediction problems. The model has no idea about simple facts such as these two options $R = 0$ and $R = 1$ should be mutually exclusive. What makes things worse, high RTG samples

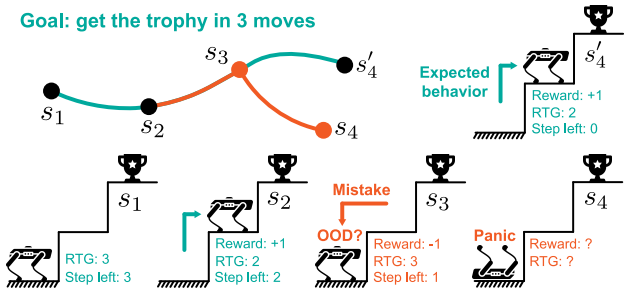

Figure 2: Illustration of the **OOD Conditioning** problem, where an agent accidentally takes a bad move and encounters an OOD, it then gets stuck there and is "panic". Our algorithm can dynamically select the RTG to maximize the return when such bad things happen.

are rare. In this case, $R = 1$ consists of only 3 samples. Thus, the empirical distribution $p_e(\uparrow |R = 1, s) = 0$ based on the dataset is quite different from the ground-truth distribution $p(\uparrow |R = 1, s) = 0.5$. The prediction problem $f : S \to \Delta(\mathcal{A})$ where $f(s) = p(a|R = 1, s)$ is thus too noisy to be trained on this dataset due to sampling bias. Therefore, Vanilla RCRL methods have difficulties in generalization to high reward regions because of such large sampling noises in the dataset.

The second problem, which mainly occurs in testing time, is what we term as **Out-of-Distribution (OOD) Conditioning**. We provide a stair-climbing example in Figure 2 for illustration. During testing time, a typical practice is first to choose a high target RTG and then dynamically decrease the RTG according to immediate rewards from the environment. In this example, if the robot takes one bad move in s_2, then quickly the conditioning RTG would not be achievable from the next state s_3 (because the goal can never be achieved with 1 remaining step), which means the RTG condition turns into an OOD input to the policy model. The behavior of the trained neural network is undefined on such inputs, bringing in severe performance drops.

4. Methodology

4.1. Bayesian Reparameterization of RCRL

In our method, the goal is to learn a probabilistic model to better approximate the posterior policy $\beta[a|s, Z^\beta(s, a) = R]$. Instead of directly taking RTGs as inputs to the neural network, our observation is that one could encode more prior knowledge into the model. A core inductive bias we introduce is that different RTGs should be competitive, not independent of each other. In order to encode this competition between different RTGs into the model, we no longer feed RTGs into the model as an extra input variable. Instead, the RTG mechanism is replaced by an energy function defined by two amortized neural network estimations of $\beta(a|s)$ and $\beta(R|s, a)$.

More precisely, thanks to the Bayes formula, the posterior policy can be written as:

$$\beta[a|s, Z^\beta(s, a) = R] \propto \beta(a|s)\beta[Z^\beta(s, a) = R|s, a]. \quad (4)$$

Inspired by the Bayesian representation of the posterior policy, we define an energy-based model

$$E_\theta(a|s, R) = -\log \bar{\beta}_\theta(a|s) - \log \bar{\beta}_\theta(R|s, a), \quad (5)$$

in which $\bar{\beta}_\theta(\cdot|s)$ and $\bar{\beta}_\theta(\cdot|s, a)$ are represented as two parameterized neural networks. In this work, we further assume that $\bar{\beta}_\theta(a|s)$ is easy to sample from, and its likelihood can be computed exactly. Instead of directly fitting a conditional model like vanilla RCRL, we reparameterize RCRL with the policy defined by this energy function $\bar{\beta}_\theta(a|s, R) = \exp(-E_\theta(a|s, R))/Z$ as an approximation to real $\beta(a|s, R)$, where $Z_\theta = \sum_a \exp(-E_\theta(a|s, R))$.

In vanilla RCRL, we optimize the following loss function:

$$\mathcal{L}_0(\theta) = -\sum_i \log \bar{\beta}_\theta(a_i|s_i, R_i), \quad (6)$$

where $\bar{\beta}_\theta(a|s, R)$ is parameterized as a normal neural network with input s, R and output a. In our model, we still optimize the above loss function but with a Bayesian way of parameterization:

$$\bar{\beta}_\theta(a|s, R) = \frac{\exp(-E_\theta(a|s, R))}{Z_\theta} = \frac{\bar{\beta}_\theta(a|s)\bar{\beta}_\theta(R|s, a)}{Z_\theta} \quad (7)$$

In order to learn such a model, we need to calculate the normalizing constant Z. Here we use samples from $\bar{\beta}_\theta(a|s)$ to estimate it:

$$Z_\theta = \sum_a \bar{\beta}_\theta(a|s)\bar{\beta}_\theta(R|s, a) = \mathbb{E}_{a\sim\bar{\beta}_\theta}[\bar{\beta}_\theta(R|s, a)]. \quad (8)$$

We then use InfoNCE (Oord et al., 2018) loss to optimize objective (6) after rewriting the model as

$$\bar{\beta}_\theta(a|s, R, A') = \frac{\bar{\beta}_\theta(a|s)\bar{\beta}_\theta(R|s, a)}{\sum_{a'\in A'} \bar{\beta}_\theta(R|s, a')}, \quad (9)$$

where negative samples $a' \in A'$. Thus, our loss function for RCRL can be summarized as

$$\mathcal{L}_0(\theta) = -\sum_i \left[\log \bar{\beta}_\theta(a_i|s_i) + \log \frac{\bar{\beta}_\theta(R_i|s_i, a_i)}{\sum_{a'_i\in A'_i} \bar{\beta}_\theta(R_i|s_i, a'_i)}\right], \quad (10)$$

where a'_i is sampled from $\bar{\beta}_\theta(a|s_i)$. In addition to the $\mathcal{L}_0(\theta)$ loss that has the same goal as vanilla RCRL, we add term $\mathcal{L}_1(\theta)$, which is the log-likelihood of the RTG:

$$\mathcal{L}_1(\theta) = -\sum_i \log \bar{\beta}_\theta(R_i|s_i, a_i). \quad (11)$$

This term is essential for our model since it addresses the RTG independence problem. After having this loss, the model is forced to capture the critical prior knowledge of how different RTG inputs depend on each other. Finally, our objective is to maximize the combination of the two losses with an adjustable parameter λ:

$$\mathcal{L}_{\text{BR-RCRL}}(\theta) = \mathcal{L}_0(\theta) + \lambda \mathcal{L}_1(\theta). \tag{12}$$

4.2. Adaptive Inference

In order to address the OOD Conditioning problem, we propose a novel adaptive inference method to ensure that our query is always in training distribution. After training $\bar{\beta}_\theta(a|s, R)$, we aim to deduce a new policy that can perform better than the data-generating policy β. To do so, we write $Z^{\bar{\beta}}(s) = \sum_{t\geq 0} \gamma^t r(a_t, s_t)|_{s_0=s}$ the expected total return under $\bar{\beta}$. For a given $\delta \in (0, 1)$, we define a threshold function

$$\theta_\delta(s) = \max_r \{r \in \mathbb{R} | P(Z^{\bar{\beta}}(s) \geq r) \geq \delta\}. \tag{13}$$

Then, we define the new policy as

$$\pi^\delta(a|s) = \bar{\beta}(a|Z^{\bar{\beta}}(s, a) \geq \theta_\delta(s), s). \tag{14}$$

During testing time, we sample from $\pi^\delta(a|s)$. Intuitively, this sampling method tries to dynamically adjust the RTG as high as possible while preserving feasibility.

By definition we have $\bar{\beta}_\theta(a|s, R) = \exp(-E_\theta(a|s, R))/Z$. Combing these formulas, we have the following proposition:

Proposition 1. *Define a new energy function :* $E^\delta_\theta(a|s) = -\log \bar{\beta}_\theta(a|s) - \log \bar{\beta}_\theta(R > \theta_\delta(s)|s, a)$*, where*

$$\bar{\beta}_\theta(R > \theta_\delta(s)|s, a) = \sum_{r>\theta_\delta(s)} \bar{\beta}_\theta(r|s, a) \tag{15}$$

then we have $\pi^\delta(a|s) \propto E^\delta_\theta(a|s)$*. Namely,* $\pi^\delta(a|s)$ *is the Boltzmann distribution of energy function* $E^\delta_\theta(a|s)$*.*

Proof. It directly follows from rewriting $\bar{\beta}(a|Z^{\bar{\beta}_\theta}(s, a) \geq \theta_\delta(s), s)$ with Bayes formula. □

In our implementation, $\theta_\delta(s)$ is not exactly computable, so we approximate it using samples. We first sample a batch of actions $\{a_i\}_{i=1}^N \sim \bar{\beta}_\theta(a|s)$, then use these samples to compute an estimate of the distribution $\bar{\beta}_\theta(R|s) = \frac{1}{N}\sum_i [\bar{\beta}_\theta(R|s, a_i)]$. Then we use the threshold δ to select the threshold θ_δ such that

$$\theta_\delta(s) = \max_r \{r \in \mathbb{R} | P(R \geq r|s) \geq \delta\} \tag{16}$$

In summary, we illustrate the proposed adaptive inference procedure in Algorithm 1.

Algorithm 1 Adaptive Inference for BR-RCRL

Input: threshold $\delta > 0$, trained policy network $\bar{\beta}_\theta$
Initialize $s = s_0$.
while s is not a terminal state **do**
  Sample a batch of actions $\{a_i\}_{i=1}^N \sim \bar{\beta}_\theta(a|s)$.
  Compute $\bar{\beta}_\theta(R|s) = \frac{1}{N}\sum[\bar{\beta}_\theta(R|s, a_i)]$
  Compute $\theta_\delta(s) = \max_r\{r \in \mathbb{R} | P(R \geq r|s) \geq \delta\}$
  Perform iterative inference on energy function $E^\delta_\theta(a|s)$ and return the best action a'
  Execute a' in the environment and get s'
  Update $s' \to s$
end while

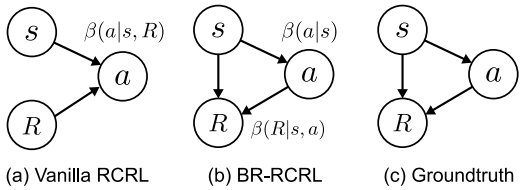

Figure 3: Generative models of Vanilla RCRL (a), BR-RCRL (b), and the ground-truth causal graphical model of RTG generation (c).

4.3. Discrete Action

In the discrete action case, our model can be considerably simplified. Following (Bellemare et al., 2017), we model the distribution of RTG using a discrete distribution parametrized by $V_{\min} \in \mathbb{R}$, $V_{\max} \in \mathbb{R}$, and $N \in \mathbb{N}$. Specifically, we use a set of bucket B:

$$\{b_i = V_{\min} + i\Delta b : i \in [0, N)\}, \Delta b = \frac{V_{\max} - V_{\min}}{N - 1}, \tag{17}$$

to represent the discretized RTG. In this case, we can train a joint probabilistic model $\bar{\beta}_\theta(a, R|s)$, which is parameterized as $f : S \to \Delta(\mathcal{A} \times \mathbb{R})$. The set of joint distributions $\Delta(\mathcal{A} \times \mathbb{R})$ is represented using a $|A||B|$-way Softmax function (Bridle, 1989), where $|A|$ is the number of actions and $|B|$ is the number of reward buckets.

In this parameterization, the $\mathcal{L}_1(\theta)$ loss stays the same but the InfoNCE (Oord et al., 2018) loss can be replaced by a normal Softmax loss $\mathcal{L}_0(\theta) = -\sum_i \log \bar{\beta}_\theta(a_i|s_i, R_i)$ because now the conditional distribution $\bar{\beta}_\theta(a|s, R)$ can be easily computed.

4.4. Causal Perspective of BR-RCRL

We provide another explanation of why BR-RCRL can facilitate better generalization over vanilla RCRL. Consider the ground-truth causality relationships between three random variables (s, a, R) generated by the data-generating policy. It is obvious that s is a direct cause of a since a is generated

by the behavior policy β. It is also apparent that both s and a are immediate causes of the total return R, leading to the true causal graph as shown in Figure 3(c). BR-RCRL tries to fit two neural networks $\bar{\beta}_\theta(a|s)$ and $\bar{\beta}_\theta(R|s,a)$ that are aligned with the true causal model. It is well-known that generative models that respect the causality relationships are more robust to distribution shifts because they can avoid learning spurious relationships between random variables (Arjovsky et al., 2019; Schölkopf et al., 2021; Lu et al., 2021; Ding et al., 2022). This explains why our BR-RCRL generalizes much better than vanilla RCRL when the distribution shifts because of the user-specified value of R, which can be viewed as an intervention (Eberhardt & Scheines, 2007) from a causality point of view.

4.5. Theoretical Analysis

Now we provide an analysis of the proposed algorithm. In the following, we sometimes write $\pi = \pi^\delta$ for short if there is no confusion. Theoretically, we expect our new policy π^δ satisfy two important properties:

(1) Its trajectory distribution should not diverge from β, which means the model should not take OOD actions. This can be quantitatively measured by KL divergence $\text{KL}[\pi^\delta(\mathcal{T})||\beta(\mathcal{T})]$. If the KL divergence is bounded, each sample trajectory in π^δ has a positive probability in β, thus eliminating the OOD problem.
(2) Policy π^δ should be guaranteed to have better performance than β. Otherwise, one can do behavior cloning and ignore the reward information.

In summary, we have the following theorem to justify our algorithm. The proof can be found in Appendix B.1.

Theorem 1. *Let $\delta \in (0,1)$, β be the data generating policy and π^δ be the conditional policy $\pi^\delta(a|s) = \beta(a|Z(s,a) \geq \theta_\delta(s), s)$. Then we have KL$[\pi^\delta(\mathcal{T})||\beta(\mathcal{T})] \leq -N \log \delta$. On the other hand, we have $V^\beta(s) \leq V^\pi(s), \forall s \in S$. $V^\beta(s) = \mathbb{E}_{a\sim\beta,\beta}[Z^\beta(s,a)]$, and $V^\pi(s) = \mathbb{E}_{a\sim\pi,\pi}[Z^\pi(s,a)]$.*

5. Experiment

In this section, we conduct several experiments on two standard benchmarks to answer the following questions:

- **Q1:** How is the performance of our proposed method compared to existing offline RL methods?
- **Q2:** How do different target RTG strategies during inference influence the results?
- **Q3:** How does the observed RTG match the target RTG during the inference stage?
- **Q4:** How do different components in BR-RCRL influence the performance?

We first briefly introduce the datasets and settings used in the experiment, then provide answers to the above questions and additional analyses.

5.1. Benchmarks and Datasets

We evaluate our method in 9 Gym-MuJoCo tasks (Fu et al., 2020) and 4 Atari games (Mnih et al., 2013), which are both standard offline RL benchmarks and cover continuous and discrete action spaces. Results of baselines are obtained from the original papers except for DT on Atari because the reported score is obtained with the 1% buffer dataset.

Datasets of the Gym-MuJoCo tasks are collected in locomotion environments (HalfCheetah, Hopper, and Walker2D) with three different data buffers. `Medium` is generated by first training an online Soft Actor-Critic (SAC) (Haarnoja et al., 2018) model, early-stopping the training, and collecting 1 million samples from this partially-trained policy. `Medium-Replay` consists of recording all samples in the replay buffer observed during training until the policy reaches the "medium" level of performance. `Medium-Expert` mixes one million expert demonstrations and one million suboptimal data generated by a partially trained policy or by unrolling a uniform-at-random policy. The results are normalized to ensure that the well-trained SAC model has a 100 score and the random policy has a 0 score.

The Atari benchmark is more difficult due to the high-dimensional state space and the long-horizon delayed reward. The offline dataset of this benchmark is collected from the replay buffer of an online DQN agent (Mnih et al., 2015). The entire dataset has 50 million transitions, but we follow the setting in (Kumar et al., 2020) and use 10% of the buffer (5 million transitions). Following (Hafner et al., 2020), we report the normalized score where the random policy is 0 and the human performance is 100.

5.2. Overall Performance (Q1)

The overall performance in Gym-MuJoCo and Atari benchmarks is reported in Table 1 and Table 2 with the comparison of three types of baselines:

- **Offline TD learning** adds constraints to online RL methods that use TD error, leading to a pessimistic behavior policy or a conservative value function. In this paper, we compare with QR-DQN (Dabney et al., 2018), REM (Agarwal et al., 2020), IQL (Kostrikov et al., 2021), CQL (Kumar et al., 2020), and BEAR (Kumar et al., 2019a).
- **Reward-conditioned RL** takes state and RTG (or reward) as input and predicts actions for the next step. We consider four representative works in the experiment: Trajectory Transformer (TT) (Janner et al., 2021), Decision Transformer (DT) (Chen et al., 2021), Decision Diffuser (DD) (Ajay et al., 2022), and RvS (Emmons et al., 2021).
- **Imitation learning** uses supervised learning to train

Table 1: Normalized Scores on Gym-MuJoCo tasks. The results of our method are averaged over 5 random seeds.

Dataset	Environment	Ours	DD	TT	DT	RvS	BC	10%BC	IBC	TD3+BC	IQL	CQL	BEAR
Med-Expert	HalfCheetah	**95.2**±0.8	90.6	95.0	86.8	92.2	55.2	92.9	34.8	90.7	86.7	91.6	53.4
Med-Expert	Hopper	**112.9**±0.9	111.8	110.0	107.6	101.7	52.5	110.9	27.5	98.0	91.5	105.4	96.3
Med-Expert	Walker2d	**111.0**±0.4	108.8	101.9	108.1	106.0	107.5	109.0	16.2	110.1	109.6	108.8	40.1
Medium	HalfCheetah	48.6±1.1	**49.1**	46.9	42.6	41.6	42.6	42.5	35.2	48.3	47.4	44.0	41.7
Medium	Hopper	78.0±1.3	**79.3**	61.1	67.6	60.2	52.9	56.9	75.3	59.3	66.3	58.5	52.1
Medium	Walker2d	82.3±1.7	82.5	79.0	74.0	71.7	75.3	75.0	14.7	**83.7**	78.3	72.5	59.1
Med-Replay	HalfCheetah	42.3±3.3	39.3	41.9	36.6	38.0	36.6	40.6	24.5	44.6	44.2	**45.5**	38.6
Med-Replay	Hopper	98.3±2.6	**100**	91.5	82.7	73.5	18.1	75.9	12.4	60.9	94.7	95.0	33.7
Med-Replay	Walker2d	80.6±2.5	75	**82.6**	66.6	60.0	26.0	62.5	9.4	81.8	73.9	77.2	19.2
Average Score		**83.2**	81.8	78.9	74.7	71.7	51.9	74.0	27.8	75.3	77.0	77.6	48.2

policy, which mimics the state-action pairs in the dataset and usually ignores the reward information. We consider Behavior Cloning (BC) (Pomerleau, 1988), BC with top 10% data (10%BC), Implicit BC (IBC) (Florence et al., 2022), and TD3+BC (Fujimoto & Gu, 2021) as our baselines.

According to Table 1, our method outperforms all baselines in terms of the average score. We achieve the highest score in 3 out of 9 Gym-MuJoCo tasks and are very close to the highest score (within standard derivation) in the remaining tasks. Unsurprisingly, our method works well in sub-optimal datasets (i.e., `Medium` and `Medium-Replay`) since the Bayesian reparametrization generalizes well even though only trained with low-reward data. Compared to strong architectures, i.e., Transformers (DT and TT) and diffusion models (DD), our method still achieves improvement in most dataset settings, demonstrating the advantages of generalizability.

In the Atari benchmark, as shown in Table 2, our method has the highest score in 3 out of 4 games and achieves a significant improvement in the `Breakout` game over other methods. We find that all methods have poor performance in the `Seaquest` game compared to human players (score of 100). The potential explanation for the low reward in this game might be the complex rules of the game and the low quality of the 10% dataset, both of which make the model only see the low-reward region.

5.3. Different Target RTG Strategies (Q2)

The second problem we want to investigate is the influence of target RTG during the inference stage. The results are reported in Table 3. The easiest way to set the target RTG is using a fixed value, e.g., the max value of RTG in the dataset (named Max in Table 3). This may cause a severe mismatch because lots of states correspond to low RTG. Therefore, DT sets an initial target RTG with the max value and gradually reduces it by subtracting the observed reward

Table 2: Normalized Score on Atari with the 10% dataset. The results of our method are averaged over 5 random seeds.

Method	Breakout	Q*bert	Pong	Seaquest
BC	136.5	38.3	1.9	1.6
QR-DQN	496.7	52.1	119.3	14.5
REM	282.4	63.7	98.7	**19.2**
CQL	889.0	103.0	130.7	17.9
DT	293.6	60.1	113.0	7.2
Ours	**1239.2**±104.2	**117.4**±13.5	**138.0**±2.2	7.1±3.1

Table 3: Comparison between different inference strategies in Walker2D task.

Target RTG	Med-Reply	Medium	Med-Expert
Max	69.4	77.3	110.2
DT-Scheduler	72.2	78.1	109.6
Ours ($\delta = 0.1$)	80.6	82.3	111.0

(named DT-Scheduler in Table 3). However, this scheduler cannot avoid the OOD problem, where the RTG is unreachable since $p(R|s) = 0$. As shown in the results, our method achieves better performance than using both max value and DT scheduler. The reason is that we select the target RTG according to the distribution $p(R|s)$, which generalizes well to different states s.

To further explore the selection of target RTG, we conduct an ablation study of the critical threshold δ. We plot the results in Figure 4 with Walker2D and Breakout environments. We can see that reducing the value of δ consistently improves the performance, which is in line with our design that a small δ corresponds to a high target RTG. region. However, setting λ to a too-small value still has the risk of causing the OOD conditioning problem.

5.4. Target RTG v.s. Observed RTG (Q3)

After analyzing the design choice of the target RTG, we now look at the relationship between the target RTG and

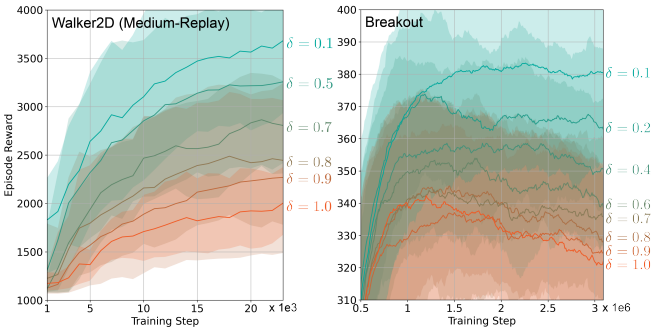

Figure 4: Raw episode reward of different values of δ during the inference stage in the Walker2D (`Medium-Replay`) task and the Breakout game. As the value of δ decreases, the performance improves.

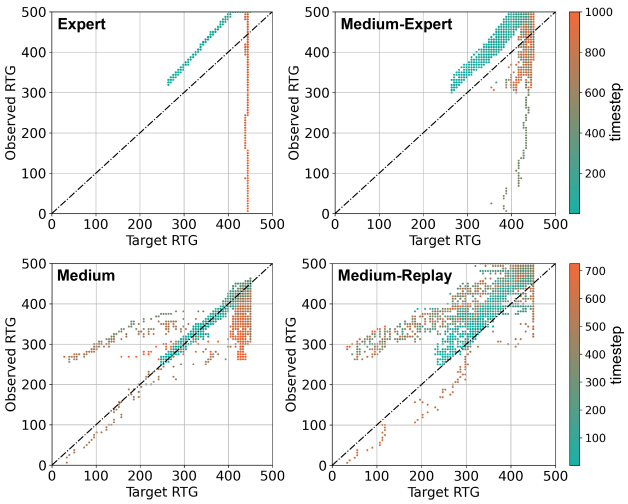

Figure 5: The relationship between target RTG and observed RTG in the Walker2D task with 4 datasets. Color represents the timestep in the trajectory.

the observed RTG, which further reveals the behavior of our method. The results in Figure 5 indicate that the observed RTG can generally match the same value of the target RTG. We find that target RTG usually starts from a medium value ($250 \sim 300$) rather than a high value used in DT. The reason is that the robot has not begun to move at the beginning, thus leading to a medium RTG. As timestep increases, the target RTG increases to the high RTG region, meaning that the robot reaches the states corresponding to high RTG in the dataset.

At the end of trajectories, there are some mismatch cases where the observed RTG is lower or higher than the target RTG. One explanation for the case `observed RTG` $>$ `target RTG` is that the model $p(R|s)$ underestimates the value of RTG due to the sub-optimality of the `Medium-Replay` dataset. In contrast, `observed RTG` $<$ `target RTG` is usually caused by the maximum episode length, which compulsorily terminates a good state that should have had a high RTG. This phenomenon happens when the quality of the dataset is high, for example, the Expert dataset in the left-top corner of Figure 5.

5.5. Influence of Components (Q4)

To study the contribution of each module in our model, we conduct ablation experiments by removing one component at one time and show the results in Table 4. We first test the model without using $\mathcal{L}_1$, which is important to solve the RTG independence problem. We find that removing this term causes a significant performance drop. We also observe that the model with $\mathcal{L}_1$ tends to ignore the RTG condition. We then modify the InfoNCE (Oord et al., 2018) loss by using the uniform distribution instead of sampling from $\bar{\beta}_\theta(a|s)$. This modification also harms the performance since most negative samples from the uniform distribution are far from the valid action space, especially in high-dimensional action space. Finally, we remove the Bayesian reparametrization (BR), which degenerates the model to a vanilla RCRL method. We find that this variant achieves similar performance to the DT model.

Table 4: Ablation study of different components in our method in the Walker2D task.

Model	Med-Reply	Medium	Med-Expert
Full model	80.6	82.3	111.0
w/o $\mathcal{L}_1(\theta)$	72.3	79.5	108.9
w/o $a' \sim \bar{\beta}_\theta(a\|s)$	77.1	80.6	110.3
w/o BR	67.1	75.4	109.1

6. Conclusion

How to design appropriate inductive biases to improve generalization on high RTG inputs during training time and to avoid out-of-distribution RTG queries during the testing time are two core challenges in RCRL that were largely ignored by previous work. This paper addresses these core challenges by proposing a novel set of inductive biases named Bayesian Reparameterized RCRL. Inspired by Bayes' theorem and causal relationships between random variables, our method successfully encodes the critical information that different RTG values are not independent classification problems but competitive. We also provide a causality perspective of our method to show that our parameterization of RCRL is in line with the ground-truth data generation process, which gains robustness to distribution shifts. We demonstrate on standard offline benchmarks how our method significantly improved the generalization performance over previous methods. One potential limitation of our method is the additional computation introduced by the training and inference of the energy-based model compared to discriminative models used in Vanilla RCRL.

Acknowledgements

Wenhao Ding gratefully acknowledges support from the National Science Foundation under grant CAREER CNS-2047454.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T., and Agrawal, P. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. PMLR, 2017.

Boney, R., Kannala, J., and Ilin, A. Regularizing model-based planning with energy-based models. In *Conference on Robot Learning*, pp. 182–191. PMLR, 2020.

Bridle, J. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Dabney, W., Rowland, M., Bellemare, M., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Ding, W., Lin, H., Li, B., and Zhao, D. Generalizing goal-conditioned reinforcement learning with variational causal reasoning. *arXiv preprint arXiv:2207.09081*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Eberhardt, F. and Scheines, R. Interventions and causal inference. *Philosophy of science*, 74(5):981–995, 2007.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. In *Conference on Robot Learning*, pp. 158–168. PMLR, 2022.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019a.

Kumar, A., Peng, X. B., and Levine, S. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019b.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, M., He, T., Xu, M., and Zhang, W. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 2020a.

Liu, S., See, K. C., Ngiam, K. Y., Celi, L. A., Sun, X., Feng, M., et al. Reinforcement learning for clinical decision support in critical care: comprehensive review. *Journal of medical Internet research*, 22(7):e18477, 2020b.

Lu, C., Wu, Y., Hernández-Lobato, J. M., and Schölkopf, B. Invariant causal representation learning for out-of-distribution generalization. In *International Conference on Learning Representations*, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

Nachum, O. and Yang, M. Provable representation learning for imitation with contrastive fourier features. *Advances in Neural Information Processing Systems*, 34:30100–30112, 2021.

Neal, R. M. et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Peters, J., Janzing, D., and Schölkopf, B. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv preprint arXiv:2203.01387*, 2022.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. Toward causal representation learning. *Proceedings of the IEEE*, 109(5): 612–634, 2021.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Singh, B., Kumar, R., and Singh, V. P. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, pp. 1–46, 2021.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Song, Y. and Kingma, D. P. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Sutton, R. S., Barto, A. G., et al. Introduction to reinforcement learning. 1998.

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

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

A. Potential Negative Societal Impacts

The main negative social impact of offline RL is that the learned policy purely relies on the dataset. Therefore, the policy could be subject to any bias in the dataset. Although our proposed method achieves strong out-of-distribution generalization, it may still be influenced by damaged data points. One way to mitigate this problem is to add a sanity check process before the training to ensure that the dataset is safe to use.

B. Theoretical Proof

B.1. Proof of Theorem 1

Proof. For the first claim, we know that $\mathrm{KL}(\pi^\delta(\mathcal{T})||\beta(\mathcal{T})) = \mathbb{E}_{\mathcal{T}\sim\pi^\beta}[\log \frac{\pi^\delta}{\beta}(\mathcal{T})]$. Then, we can get

$$\begin{aligned}
\log \pi^\delta(\mathcal{T})/\beta(\mathcal{T}) &= \sum_{i=1}^{N} \log \pi^\delta(a_i|s_i) - \log \beta(a_i|s_i) \\
&= \sum_{i=1}^{N} \log \beta(Z \geq \theta_\delta|a_i, s_i) - \log \beta(Z \geq \theta_\delta|s_i) \\
&\leq -\sum_{i=1}^{N} \log \beta(Z \geq \theta_\delta|s_i) \\
&\leq -N \log \delta
\end{aligned}$$

Combine these two formulas, we have $\mathrm{KL}(\pi^\delta(\mathcal{T})||\beta(\mathcal{T})) \leq -N \log \delta$.

For the second claim, we first prove the following lemma:

Lemma 1. *X is a random variable, then for any $c \in \mathbb{R}$, we have $\mathbb{E}[X|X \geq c] \geq \mathbb{E}[X]$.*

We know that

$$\mathbb{E}[X|X \geq c] = \mathbb{E}_X[X \cdot \mathbb{1}_{X\geq c}]/P(X \geq c). \tag{18}$$

On the other hand,

$$\begin{aligned}
\mathbb{E}[X] \cdot P(X \geq c) =& \mathbb{E}[X \cdot \mathbb{1}_{X\geq c}]P(X \geq c) + \mathbb{E}[X \cdot \mathbb{1}_{X<c}]P(X \geq c) \\
=& \mathbb{E}[X \cdot \mathbb{1}_{X\geq c}][1 - P(X < c)] + \mathbb{E}[X \cdot \mathbb{1}_{X<c}]P(X \geq c),
\end{aligned} \tag{19}$$

where $\mathbb{1}_{X\geq c}$ is an indicator function that outputs 1 when $X \geq c$ is satisfied. Thus we only need to show

$$\mathbb{E}[X \cdot \mathbb{1}_{X<c}]/P(X < c) \leq \mathbb{E}[X \cdot \mathbb{1}_{X\geq c}]/P(X \geq c). \tag{20}$$

This is obvious because

$$\mathbb{E}[X \cdot \mathbb{1}_{X<c}] \leq c \cdot P(X < c), \tag{21}$$

$$\mathbb{E}[X \cdot \mathbb{1}_{X\geq c}] \geq c \cdot P(X \geq c). \tag{22}$$

Therefore, for a state $s \in S$, we can see that

$$\begin{aligned}
V^\pi(s) =& \mathbb{E}_{a\sim\pi,\pi}[Z^\pi(s,a)] \\
\geq& \mathbb{E}_{a\sim\beta,\beta}[Z^\beta(s,a)|Z^\beta(s,a) \geq \theta_\delta(s,a)] \\
\geq& V^\beta(s)
\end{aligned} \tag{23}$$

□

C. Additional Experiment Results

C.1. RTG Independence Problem in DT

The policy model tends to isolate the prediction problems $a = f_{R_i}(s), i = 1, 2 \cdots$ according to different RTGs R_i, instead of learning a generalizable mapping from R to action. In RCRL, this limitation can briefly be understood as the model can only learn from high RTG samples, the low RTG samples cannot help improve the model's performance in high RTG regions. We confirm this tendency by conducting an additional experiment (results in Table 5). We fit three DT models using different variants of the medium-replay dataset. Top $x\%$ means we only select the top $x\%$ of trajectories, ordered by episode RTG. We observe that these models achieve similar performance, which indicates that the prediction conditioned on high RTG is independent of the training samples with low RTG. In addition, we use the same setting to test our method and find that removing the low RTG samples has a negative influence on the results.

Table 5: Performance on different portions of the dataset.

Dataset	DT (top 100%)	DT (top 50%)	DT (top 20%)	Ours (top 100%)	Ours (top 50%)	Ours (top 20%)
Halfcheetch	36.0	37.2	36.5	42.3	40.5	38.1
Hopper	77.3	78.5	77.1	98.3	94.1	90.3
Walker2D	65.5	64.4	66.2	80.6	76.4	73.2

C.2. Sampling Bias Dominance Problem in DT

Given state s, we assume $c(s)$ is the threshold for our target RTG, which is usually high. The dataset is sub-optimal, so $p_d(R > c(s)|s) < \epsilon$, where ϵ is a small number. Consider the training dataset $D = \{(s_i, a_i, R_i)\}_{i=1}^N$. As we show above, due to RTG independence, the model in fact only can learn from $D_c = \{(s, a, R) \in D | R > c(s)\}$. However $|D_c| << |D|$. Then the trained model will be affected by the large sampling bias due to $|D_c|$ being small.

C.3. OOD Conditioning in DT

Given a state s, during training, we have the distribution of RTG $p_d(R|s)$ while during testing we aim to sample high RTG from $p_t(R|s)$. These two distributions can have non-overlap supports, namely $\mathrm{KL}(p_t||p_d)$ usually can be $+\infty$. During test time, given a state s and target RTG $R_t \sim p_t(R|s)$, the model try to predict $a \sim p(a|s, R = R_t)$, however R_t may be out-of-distribution (OOD) for p_d in the sense that $p_d(R_t|s) = 0$. We provide a concrete example of the OOD target RTG in the Atari Pong game. The image is shown in Figure 6. The game ends when one player gains 20 points. Since the RL agent already loses 19 points, it is almost impossible to obtain $RTG = 20$ in this match. Therefore, $RTG = 20$ is an OOD condition for the DT model.

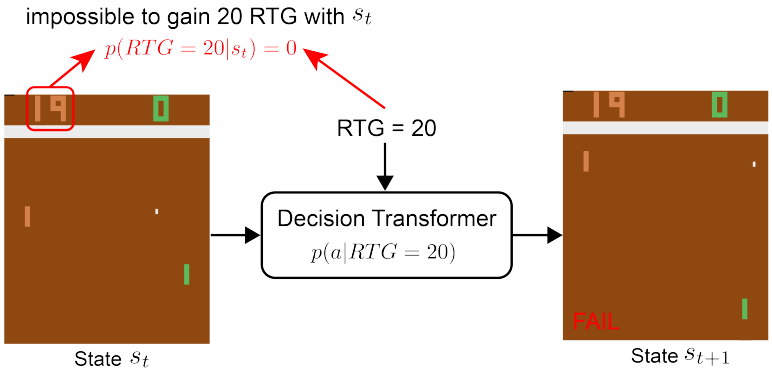

Figure 6: Example of the OOD condition problem.

D. Experiment Details

D.1. Experiment Device

The experiments were conducted on a device with 256GB memory and 2 $\times$ NVIDIA RTX A6000 GPUs. The Atari experiments require $\sim$150GB of memory to load the 10% Atari dataset.

D.2. Inference Optimizer of EBMs

In Gym-Mujoco experiments, we use a derivative-free optimizer (DFO) proposed in (Florence et al., 2022) to infer the energy-based model. As stated in (Florence et al., 2022), other advantaged optimizers such as Langevin MCMC (Welling &

Teh, 2011) could improve the efficiency for high-dimensional cases.

We show the statistic of running time for inference of EBM in Table 6. Although the inference spends more time than directly using conditional policy, the cost is still acceptable since we only need a few iterations.

Table 6: Inference time of EBM.

Environment	Halfcheetch	Hopper	Walker2D	Breakout	Q*bert	Pong	Seaquest
Inference time	0.0082 s	0.0080 s	0.0082 s	0.0029 s	0.0031 s	0.0032 s	0.0031 s

D.3. Hyperparameters

The hyperparameters used in Gym-Mujoco experiments and Atari experiments are summarized in Table 7 and Table 8, respectively. We use the same hyperparameters for all experiments in the same benchmark. The source code of our experiments will be released after the blind review process.

Notation	**Parameter Description**	**Value**
	training iteration	70,000
	learning rate	0.0005
	batch size	512
	action penalty	0.0
λ	weight of $\mathcal{L}_1(\theta)$	1.0
$\|B\|$	number of reward bucket	80
γ	reward discount	0.99
V_{min}	minimal bucket RTG	0
V_{max}	maximal bucket RTG	1,200
$N_{A'}$	number of negative samples during training	256
δ	inference threshold	0.1
	number of episodes for each testing point	10
	Number of iterative in DFO	5
	Number of samples in DFO	65,536
	Noise shrink parameter in DFO	0.9
	Scale of noise in DFO	0.5

Table 7: Hyperparameters for Gym-Mujoco experiments

Notation	**Parameter Description**	**Value**
	training iteration	3,000,000
	learning rate	0.00025
	action penalty	0.5
	target network update frequency	8,000
B	batch size	32
λ	weight of $\mathcal{L}_1(\theta)$	20.0
$\|B\|$	number of reward bucket	51
γ	reward discount	0.95
V_{min}	minimal bucket of RTG	0
V_{max}	maximal bucket RTG	10
δ	inference threshold	0.1
ϵ	exploration ratio during test	0.01
	number of episodes for each testing point	10

Table 8: Hyperparameters for Atari experiments