Discovering Multiple Solutions from a Single Task in Offline Reinforcement Learning

Takayuki Osa [1 2] **Tatsuya Harada** [1 2]

Abstract

Recent studies on online reinforcement learning (RL) have demonstrated the advantages of learning multiple behaviors from a single task, as in the case of few-shot adaptation to a new environment. Although this approach is expected to yield similar benefits in offline RL, appropriate methods for learning multiple solutions have not been fully investigated in previous studies. In this study, we therefore addressed the problem of finding multiple solutions from a single task in offline RL. We propose algorithms that can learn multiple solutions in offline RL, and empirically investigate their performance. Our experimental results show that the proposed algorithm learns multiple qualitatively and quantitatively distinctive solutions in offline RL.

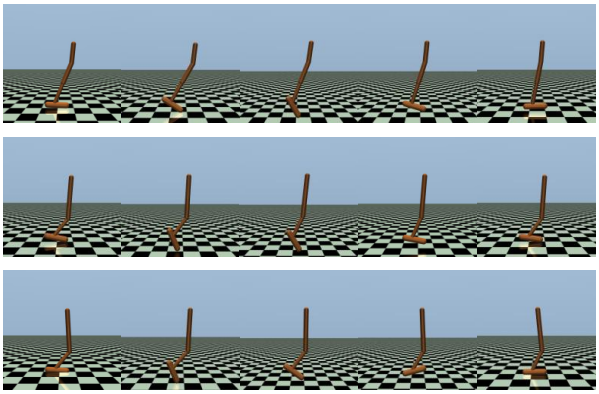


Figure 1. Sequential snapshots of multiple locomotion behaviors obtained by proposed method for the hopper agent.

1. Introduction

The benefits of discovering diverse solutions have been demonstrated in literature pertaining to online reinforcement learning (RL), as in the cases of few-shot adaptation to changes in the environment (Kumar et al., 2019) and composition of complex motion by sequencing different behavioral styles (Kumar et al., 2020a). For example, a locomotion task may encompass multiple motion styles using different postures. Although the discovery of diverse solutions in offline RL is expected to be equally beneficial in practice, it has not been fully investigated in previous studies. Accordingly, we address the problem of discovering diverse solutions in offline RL.

The primary challenge of learning multiple solutions in offline RL is the problem of learning the latent skill space. In the case of online RL, diverse behaviors are typically modeled with a latent-conditioned policy wherein the value of the latent variable is stored alongside the state action pairs during the data collection process. Although offline RL employs a dataset containing the state-action pairs, this dataset does not contain information that indicates the skill type, such as the value of the latent variable. Therefore, it is essential to learn the latent skill representations in an unsupervised manner.

In this study, we introduce a novel algorithm designed to uncover multiple solutions through the unsupervised learning of latent skill representations. We developed our algorithm to tackle the challenge of identifying multiple solutions in offline RL through a coordinate ascent approach, resembling the expectation-maximization (EM) algorithm. In our algorithm, we iteratively update a policy conditioned on latent variables to capture multiple behaviors, along with a posterior distribution that models latent skill representations. To evaluate our algorithm, we constructed datasets that contain diverse behaviors using the D4RL framework (Fu et al., 2020), with locomotion tasks based on MuJoCo (Todorov et al., 2012). The experimental results demonstrate that the proposed algorithm discovers multiple solutions as shown in Figure 1, and that few-shot adaptation to new environments can be performed by learning multiple behaviors in offline RL.

[1]The University of Tokyo, Japan [2]RIKEN Center for Advanced Intelligence Project. Correspondence to: Takayuki Osa <osa@mi.t.u-tokyo.ac.jp>.

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

2. Background

2.1. Related Work

In literature pertaining RL, it is known that multiple optimal policies may elicit the optimal value function (Sutton & Barto, 2018). The discovery of diverse solutions has been actively investigated in the evolutionary computation community, with such algorithms often referred to as quality-diversity (QD) algorithms (Chatzilygeroudis et al., 2021; Cully & Demiris, 2018; Parker-Holder et al., 2020; Pugh et al., 2016). These QD algorithms can be adapted to offline RL settings by learning dynamics models as proposed in (Hein et al., 2018a;b). In the context of online RL, studies such as (Kumar et al., 2020b; Osa et al., 2022; Sharma et al., 2020) reported that multiple solutions can be obtained from a single task, enabling few-shot adaptation to new environments and composition of complex behavior by sequencing learned behaviors. Recent studies on robotics have also reported that multiple solutions frequently manifest in robotics problems (Osa, 2022; Toussaint et al., 2018).

The task of learning latent representations of the state-action space is closely related to that of discovering multiple solutions in RL. Recent studies pertaining to offline RL have investigated the leveraging of latent representations of the state-action space (Chen et al., 2022; Zhou et al., 2020; Osa et al., 2023), revealing this task to improve the stability of the learning process. The study by (Ajay et al., 2021) proposed a framework to learn primitive skills from offline data and utilize them for downstream tasks. Although the focus of the study presented by (Ajay et al., 2021) is utilizing the primitive skills for downstream tasks, their work is relevant to ours in a sense that how to extract multiple behaviors in offline RL is investigated. Liu et al. (2023) recently proposed a framework to leverage expert demonstrations in offline RL. While the framework proposed by Liu et al. (2023) is not specifically designed for finding multiple behaviors, they also demonstrated that multiple behaviors can be found by their method in a unsupervised manner.

Our algorithm utilizes coordinate ascent, drawing inspiration from the EM algorithm (Bishop, 2006). While the policy update in the E-step can be perceived as an extension of MPO (Abdolmaleki et al., 2018) or AWAC (Nair et al., 2020), it is essential to note that our algorithm distinguishes itself by the alternate updating of the latent-conditioned policy and posterior distribution. This distinctive approach represents a novel algorithm designed specifically for learning multiple solutions in offline RL, setting it apart from existing methodologies.

A concurrent work by Mao et al. (2024) recently addressed the problem of finding multiple solutions in offline RL and proposed a method called Stylized Offline RL (SORL). Mao et al. (2024) formulated the problem of finding multiple solutions in offline RL as a clustering problem, where the latent variable that characterizes the behavior style is discrete, and the number of solutions obtained by SORL is finite. In contrast, our formulation is more general in the sense that the latent variable can be either continuous or discrete. While we report results based on the continuous latent variable in this study, the discrete latent variable can also be learned using techniques such as the Gumbel-softmax trick (Jang et al., 2017; Maddison et al., 2017).

2.2. Reinforcement Learning

In RL, we consider a Markov decision process (MDP) that consists of a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, d)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the transition probability density, $r(\boldsymbol{s}, \boldsymbol{a})$ is the reward function, γ is the discount factor, and $d(\boldsymbol{s}_0)$ is the probability density of the initial state. A policy $\pi(\boldsymbol{a}|\boldsymbol{s}) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is defined as the conditional probability density over actions given the states. The aim of RL is to learn a policy that maximizes the expected return $\mathbb{E}[R_0|\pi]$ where $R_t = \sum_{k=t}^{T} \gamma^{k-t} r(\boldsymbol{s}_k, \boldsymbol{a}_k)$.

In offline RL, it is assumed that the learning agent is provided with a fixed dataset, $\mathcal{D} = \{(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}'_i)\}_{i=1}^{N}$, comprising states, actions, rewards, and next states collected by an unknown behavior policy $\beta(\boldsymbol{a}|\boldsymbol{s})$. For convenience, we denote by $d^{\beta}(\boldsymbol{s})$ the state distribution obtained by executing the behavior policy β. The objective of offline RL is to obtain a policy that maximizes the expected return using $\mathcal{D}$ without online interactions with the environment during the learning process.

2.3. Latent-Conditioned Policy

A latent-conditioned policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ is typically used to model multiple behaviors with a single model, as demonstrated by prior studies on online RL (Kumar et al., 2020b; Sharma et al., 2020). Here, the latent variable $\boldsymbol{z}$, which may be continuous or discrete, is used to control the behavior type of the latent-conditioned policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$. We assumed that the value of the latent variable is sampled at the beginning of an episode and fixed until the episode's end. When the dataset contains multiple behaviors, we assume that the behavior policy can be written as $\beta(\boldsymbol{a}|\boldsymbol{s}) = \int \beta(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z}$, where $p(\boldsymbol{z})$ is the prior distribution of the latent variable, and the value of $\boldsymbol{z}$ controls the behavior style of the behavior policy. As the value of $\boldsymbol{z}$ is unknown, the primary challenge of learning multiple behaviors in offline RL is to estimate the latent representations.

In the context of online RL, a latent-conditioned action-value function is typically estimated to evaluate the quality of different behaviors. The latent-conditioned action-value

function is defined as

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = \mathbb{E}_\pi \left[R | \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z} \right], \tag{1}$$

which models the expected return when taking action $\boldsymbol{a}$ under state $\boldsymbol{s}$ with the latent variable $\boldsymbol{z}$, and then following policy π. The value of the latent variable $\boldsymbol{z}$ corresponds to a type of the behavior encoded in the latent-conditioned policy π. Similarly, a latent-conditioned state-value function is defined as

$$V^\pi(\boldsymbol{s}, \boldsymbol{z}) = \mathbb{E}_\pi \left[R | \boldsymbol{s}, \boldsymbol{z} \right], \tag{2}$$

which models the expected return when starting from state $\boldsymbol{s}$ with the latent variable $\boldsymbol{z}$, and then following policy π. In our work, we consider the advantage function given by $A^\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = Q^\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) - V^\pi(\boldsymbol{s}, \boldsymbol{z})$.

3. Algorithm to Learn Diverse Behaviors in Offline RL

When a latent-conditioned policy is trained in an online RL manner, the value of the latent variable is known during the rollout, being stored alongside states, actions, and rewards. However, in the context of offline RL, we assume that the value of the latent variable corresponding to a state-action pair is unknown. To train a latent-conditioned policy, it is therefore essential to estimate the value of the latent variable in an unsupervised learning manner. In addition, as the performance of the latent-conditioned policy relies on the learned latent representations, the encoder that estimates the latent variable corresponding to a state-action pair must reflect the behavior of the policy. Therefore, it is essential to jointly train the latent-conditioned policy and encoder.

In this section, we describe our method for obtaining latent representations of the state action space, and explain how to train a latent-conditioned policy using the learned representations.

3.1. Problem Formulation

To acquire multiple solutions, we train a latent-conditioned policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ parametrized by vector $\boldsymbol{\theta}$. Concurrently, we train a posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ parametrized by vector $\boldsymbol{\phi}$ to learn latent skill representations. Our objective in solving a given task is to optimize $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ to maximize the expected return. In addition, we also maximize the mutual information between the latent variable and the state-action pair, $I_\pi(\boldsymbol{z}; \boldsymbol{s}, \boldsymbol{a})$, to enhance the diversity of the behavior encoded in the policy. Moreover, to mitigate the generation of out-of-distribution actions in offline RL, we impose a constraint based on the Kullback-Leibler (KL) divergence between the policy $\pi_{\boldsymbol{\theta}}$ and the behavior policy β. Consequently, we formulate the problem of discovering multiple solutions in offline RL as the following optimization problem:

$$\max_{\pi, q} \left(\mathbb{E}_{\boldsymbol{s} \sim d^\beta, \boldsymbol{a} \sim \pi_{\boldsymbol{\theta}}, \boldsymbol{z} \sim q_{\boldsymbol{\phi}}} \left[A^\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \right] + \lambda I_\pi(\boldsymbol{z}; \boldsymbol{s}, \boldsymbol{a}) \right) \tag{3}$$

$$\text{s.t. } \mathbb{E}_{\boldsymbol{s} \sim d^\beta, \boldsymbol{z} \sim q_{\boldsymbol{\phi}}} \left[D_{\text{KL}} \left(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}, \boldsymbol{z}) || \beta(\cdot|\boldsymbol{s}, \boldsymbol{z}) \right) \right] \leq \epsilon_\pi, \tag{4}$$

where λ is a coefficient to balance the weight on each term. The proposed algorithm addresses maximization of the first term in (3) through a procedure resembling the EM algorithm. In the E-step, we update the latent-conditioned policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ based on the posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$. In the M-step, we update the posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ given the latent-conditioned policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$. Subsequently, both $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ are updated to maximize the variational lower bound of the mutual information $I_\pi(\boldsymbol{z}; \boldsymbol{s}, \boldsymbol{a})$. Our algorithm can be viewed as a form of coordinate ascent, wherein the latent-conditioned policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ and the posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ are alternately updated. In the following section, we provide a detailed description of each step.

3.2. E-Step

Given the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$, we solve the following optimization problem in E-step:

$$\max_{\pi} \mathbb{E}_{\boldsymbol{s} \sim d^\beta, \boldsymbol{a} \sim \pi_{\boldsymbol{\theta}}, \boldsymbol{z} \sim q_{\boldsymbol{\phi}}} \left[A^\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \right] \tag{5}$$

$$\text{s.t. } \mathbb{E}_{\boldsymbol{s} \sim d^\beta, \boldsymbol{z} \sim q_{\boldsymbol{\phi}}} \left[D_{\text{KL}} \left(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}, \boldsymbol{z}) || \beta(\cdot|\boldsymbol{s}, \boldsymbol{z}) \right) \right] \leq \epsilon_\pi. \tag{6}$$

The KL divergence constraint in (6) encourages the policy close to the data distribution, enabling us to avoid generating out-of-distribution actions. In addition, when a dataset contains diverse behaviors, the constraint in the above problem maintains the diversity of the learned policy π.

The solution to this problem is given in a nonparametric form as

$$\pi^*(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z}) = \frac{1}{Z_\pi} \beta(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z}) \exp\left(\frac{1}{\alpha_\pi} A^\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \right), \tag{7}$$

where Z_π is the partition function. To approximate the nonparametric solution in (7) with a parameterized model, we minimize the KL divergence between $\pi^*(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ in (7) and $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$:

$$\mathbb{E}_{\boldsymbol{s}, \boldsymbol{z}} \left[D_{\text{KL}} \left(\pi^*(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z}) || \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z}) \right) \right]. \tag{8}$$

The minimizer of the above KL divergence can be obtained by maximizing the weighted log-likelihood:

$$\begin{aligned} &\mathcal{L}_{\text{E-step}}(\boldsymbol{\theta}) \\ &= \mathbb{E}_{\boldsymbol{s}, \boldsymbol{a} \sim d^\beta, \beta} \mathbb{E}_{\boldsymbol{z} \sim q} \left[W_\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z}) \right], \end{aligned} \tag{9}$$

where the weight $W_\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$ is given by

$$W_\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = \exp\left(\frac{1}{\alpha_\pi} A^\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \right). \tag{10}$$

The derivation of (7) and (9) is presented in Appendix A. According to the derived results, the latent-conditioned policy is updated using (9) in the E-step. The policy update in (9) can be understood as an extension of MPO (Abdolmaleki et al., 2018) or AWAC (Nair et al., 2020).

3.3. M-Step

In the M-step, we introduce a constraint on the update of the posterior distribution $q_{\boldsymbol{\phi}}$ to enhance the stability of the learning process. Specifically, we impose an upper bound on the KL divergence between $q_{\boldsymbol{\phi}}$ before and after the update. Denoting the parameter of the posterior distribution before the update as $\boldsymbol{\phi}_{\text{old}}$, the optimization problem in the M-step can be expressed as follows:

$$\max_{q} \mathbb{E}_{\boldsymbol{z}\sim q}\left[A^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right] \tag{11}$$

$$\text{s.t. } D_{\text{KL}}\left(q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})||q_{\boldsymbol{\phi}_{\text{old}}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})\right) \leq \epsilon_q. \tag{12}$$

The constraint in (12) can be viewed as a trust region constraint. As in the optimization problem in the E-step, the solution to this problem is given in a nonparametric form as

$$q^*(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a}) = \frac{1}{Z_q} q_{\boldsymbol{\phi}_{\text{old}}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a}) \exp\left(\frac{1}{\alpha_q} A^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right), \tag{13}$$

where Z_q is the partition function. In the M-step, the parameterized posterior distribution $q_{\boldsymbol{\phi}}$ is trained to approximate the optimal posterior distribution $q^*(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ as defined in (13). Although the solution in (13) is independent of the distributions of $\boldsymbol{s}$ and $\boldsymbol{a}$, we train our model with respect to $d^{\beta}(\boldsymbol{s})$ and $\pi(\boldsymbol{a}|\boldsymbol{s})$. This choice is motivated by our objective to obtain a solution for the problem described in (4).

Given a dataset $\mathcal{D}$ sampled from $p^{\beta}(\boldsymbol{s},\boldsymbol{a}) = d^{\beta}(\boldsymbol{s})\beta(\boldsymbol{a}|\boldsymbol{s})$ and $\pi_{\boldsymbol{\theta}}$, we update $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ to approximate the density induced by the optimal posterior distribution. Assuming $p(\boldsymbol{z}|\boldsymbol{s}) = p(\boldsymbol{z})$, we approximate $\pi(\boldsymbol{a}|\boldsymbol{s})$ as

$$d^{\beta}(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s}) = d^{\beta}(\boldsymbol{s}) \int \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}. \tag{14}$$

Denoting $p^{\beta,\pi}(\boldsymbol{s},\boldsymbol{a}) = d^{\beta}(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s})$, we minimize the following KL divergence to update the parameterized posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$:

$$\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim p^{\beta,\pi}}\left[D_{\text{KL}}\left(q^*(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})||q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})\right)\right] \tag{15}$$

The minimizer of the above KL divergence is given by the weighted log-likelihood:

$$\mathcal{L}_{\text{posterior}}(\boldsymbol{\phi}) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim p^{\beta,\pi}}\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}_{\text{old}}}}\left[W_q \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})\right], \tag{16}$$

where the weight $W_q(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$ is given by

$$W_q(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}) = \frac{1}{Z_q}\exp\left(\frac{1}{\alpha_q}A^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right). \tag{17}$$

While the form of the solution in (16) is analogous to that in (9), the temperature parameters α_{π} and α_q are different because they are separately derived from constraints in (6) and (12). Thus, α_{π} and α_q should be tuned separately in practice.

In addition, to assure the consistency between the trained posterior distribution and policy, we also train the posterior distribution so as to approximate the density induced by the optimal policy in (7). For this purpose, we introduce a likelihood $p_{\boldsymbol{\psi}}(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{z})$ parameterized with a vector $\boldsymbol{\psi}$.

Given a posterior distribution $q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$, the density $p^*(\boldsymbol{s},\boldsymbol{a})$ induced by the optimal policy in (7) is given by

$$p^*(\boldsymbol{s},\boldsymbol{a}) = d^{\beta}(\boldsymbol{s})\pi^*(\boldsymbol{a}|\boldsymbol{s}) \tag{18}$$

$$= \int d^{\beta}(\boldsymbol{s}))p(\boldsymbol{z}|\boldsymbol{s})\pi^*(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})d\boldsymbol{z} \tag{19}$$

$$= \int d^{\beta}(\boldsymbol{s})p(\boldsymbol{z}|\boldsymbol{s})\beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\frac{1}{Z_{\pi}}\exp\left(\frac{1}{\alpha_{\pi}}A^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)d\boldsymbol{z} \tag{20}$$

$$= \frac{1}{Z_{\pi}}\int d^{\beta}(\boldsymbol{s})\beta(\boldsymbol{a}|\boldsymbol{s})q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})\exp\left(\frac{1}{\alpha_{\pi}}A^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)d\boldsymbol{z} \tag{21}$$

The result in (21) indicates that the density $p^*(\boldsymbol{s},\boldsymbol{a})$ can be approximated by 1) sampling the latent variable $\boldsymbol{z}$ with q, 2) set the importance weight with $\exp\left(\frac{1}{\alpha_{\pi}}A^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)$, and 3) marginalize over the latent variable $\boldsymbol{z}$.

To train the parameterized posterior distribution and likelihood, we use the variational lower bound used in variational autoencoder (VAE) (Kingma & Welling, 2014), which is given by

$$\log p(\boldsymbol{s}_i,\boldsymbol{a}_i) \geq \ell_{\boldsymbol{\phi},\boldsymbol{\psi}}(\boldsymbol{s}_i,\boldsymbol{a}_i) \tag{22}$$

$$= -D_{\text{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}_i,\boldsymbol{a}_i)||p(\boldsymbol{z})\right) + \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\psi}}(\boldsymbol{s}_i,\boldsymbol{a}_i|\boldsymbol{z})\right]. \tag{23}$$

Combining (21) and (23), we maximize the following objective:

$$\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim p^*}\left[\log p(\boldsymbol{s},\boldsymbol{a})\right]$$

$$= \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim d^{\beta},\beta}\mathbb{E}_{\boldsymbol{z}\sim q}\left[W_{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\log p(\boldsymbol{s},\boldsymbol{a})\right] \tag{24}$$

$$\geq \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim d^{\beta},\beta}\mathbb{E}_{\boldsymbol{z}\sim q}\left[W_{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\ell_{\boldsymbol{\phi},\boldsymbol{\psi}}(\boldsymbol{s},\boldsymbol{a})\right] \tag{25}$$

$$\approx \frac{1}{N\cdot N_z}\sum_{i=1}^{N}\sum_{j=1}^{N_z} W_{\pi}(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{z}_j)\ell_{\boldsymbol{\phi},\boldsymbol{\psi}}(\boldsymbol{s}_i,\boldsymbol{a}_i) \tag{26}$$

$$:= \mathcal{L}_{\text{w-vae}}(\boldsymbol{\phi},\boldsymbol{\psi}) \tag{27}$$

where $W_{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$ is given as in (10).

Based on the result in (16) and (27), we maximize the following objective function in the M-step:

$$\mathcal{L}_{\text{M-step}}(\boldsymbol{\phi},\boldsymbol{\psi}) = \mathcal{L}_{\text{posterior}}(\boldsymbol{\phi}) + \mathcal{L}_{\text{w-vae}}(\boldsymbol{\phi},\boldsymbol{\psi}). \tag{28}$$

3.4. Learning the Latent Skill Space via Mutual Information Maximization

Herein, we discuss how to maximize the mutual information term in (4). For this purpose, we utilize the variational lower bound of the mutual information. We consider mutual information $I(\boldsymbol{z};\boldsymbol{s},\boldsymbol{a})$ between the latent variable $\boldsymbol{z}$ and a state-action pair $(\boldsymbol{s},\boldsymbol{a})$ as

$$I(\boldsymbol{z};\boldsymbol{s},\boldsymbol{a}) = \iiint p(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}) \log \frac{p(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})}{p(\boldsymbol{s},\boldsymbol{a})p(\boldsymbol{z})} d\boldsymbol{z}d\boldsymbol{a}d\boldsymbol{s}. \tag{29}$$

The variational lower bound of $I(\boldsymbol{z};\boldsymbol{s},\boldsymbol{a})$ is then given as follows (Barber & Agakov, 2003; Osa et al., 2022):

$$I(\boldsymbol{z};\boldsymbol{s},\boldsymbol{a}) \geq \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\sim p}\left[\log q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})\right] + H(\boldsymbol{z}), \tag{30}$$

where $H(\boldsymbol{z})$ is the entropy of the latent variable $\boldsymbol{z}$. As $H(\boldsymbol{z})$ is independent of the policy, encoder, and decoder parameters, only the first term of the right-hand side in (30) must be maximized.

As we maximize the lower bound with respect to $p^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}) = d^{\beta}(\boldsymbol{s})p(\boldsymbol{z}|\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})$, we maximize the following term:

$$\begin{aligned}\mathcal{L}_{\text{info}}(\boldsymbol{\theta},\boldsymbol{\phi}) &= \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\sim p^{\pi}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})\right] \\ &= \int d^{\beta}(\boldsymbol{s})p(\boldsymbol{z}|\boldsymbol{s})\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) \log q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{z}d\boldsymbol{a}d\boldsymbol{s}\end{aligned} \tag{31}$$

As shown by previous studies (Kumar et al., 2020b; Osa et al., 2022), maximizing the variational lower bound of mutual information $I(\boldsymbol{z};\boldsymbol{s},\boldsymbol{a})$ encourages the latent-conditioned policy to generate different actions for different values of the latent variable. Consequently, the diversity of actions is enhanced by maximizing this term. Additionally, when maximizing $\mathbb{E}_{\tilde{\boldsymbol{a}}\sim\pi_{\boldsymbol{\theta}}}\left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\tilde{\boldsymbol{a}})\right]$, the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ recovers the value of the latent variable from the state and action generated by the latent-conditioned policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})$. Thus, the encoder is updated to learn latent representations that are consistent with the behavior of the latent-conditioned policy.

Combining the objectives for the M-step and mutual information maximization, we jointly update the policy, posterior distribution, and likelihood by maximizing the following objective function with the regularization based on mutual information maximization:

$$\mathcal{L}_{\text{M-info}}(\boldsymbol{\phi},\boldsymbol{\psi},\boldsymbol{\theta}) = \mathcal{L}_{\text{M-step}}(\boldsymbol{\phi},\boldsymbol{\psi}) + \lambda\mathcal{L}_{\text{info}}(\boldsymbol{\theta},\boldsymbol{\phi}), \tag{32}$$

where λ is a constant that balances the two terms.

3.5. Practical Algorithm

The proposed algorithm, referred to as Learning Diverse Behaviors in Offline RL (DiveOff), is summarized in Algorithm 1. We initialize the posterior $q_{\boldsymbol{\phi}}$ and likelihood $p_{\boldsymbol{\psi}}$ using the standard VAE objective function. We then iterate the E-step and M-step described in the previous section to update the policy and posterior distribution. Whereas Algorithm 1 is described assuming that the critic is trained with double-clipped Q-learning (Fujimoto et al., 2018), alternative methods can be employed, such as expectile regression as in IQL.

Algorithm 1 Learning Diverse Behaviors in Offline RL (DiveOff)

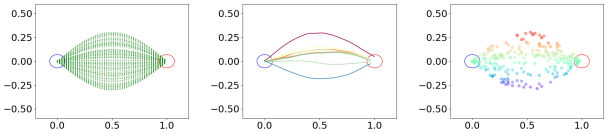
Input: dataset $\mathcal{D} = \{(\boldsymbol{s}_i,\boldsymbol{a}_i,r_i,\boldsymbol{s}'_i)\}_{i=1}^N$
Initialize the policy $\pi_{\boldsymbol{\theta}}$, critic $Q_{\boldsymbol{w}_j}$ for $j=1,2$, posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$, and likelihood $p_{\boldsymbol{\psi}}(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{z})$
Train the posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$, and likelihood $p_{\boldsymbol{\psi}}(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{z})$ with $\mathcal{D}$ using the standard VAE loss
for $t=1$ **to** T **do**
  Sample a minibatch $\{(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}'_i,r_i)\}_{i=1}^N$ from $\mathcal{D}$
  Sample the latent variable $\boldsymbol{z}_i \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}_i,\boldsymbol{a}_i)$
  Compute the target value:
    $y_i = r + \gamma \min_{j=1,2} Q_{\boldsymbol{w}_j}(\boldsymbol{s}_i,\boldsymbol{a}',\boldsymbol{z}_i)$,
    where $\boldsymbol{a}' \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}'|\boldsymbol{s}'_i,\boldsymbol{z}_i)$
  Update the critic:
    $N^{-1}\sum_{i=1}^N \left(y_i - Q_{\boldsymbol{w}_j}(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{z})\right)^2$ for $j=1,2$
  if $t \bmod d_{\text{interval}} = 0$ **then**
    Update the policy by maximizing $\mathcal{L}_{\text{E-step}}(\boldsymbol{\theta})$ in (9)
    Update the policy, posterior, and likelihood by maximizing $\mathcal{L}_{\text{M-info}}(\boldsymbol{\phi},\boldsymbol{\psi},\boldsymbol{\theta})$ in (32)
  end if
end for

4. Experiments

To analyze the performance of the proposed method, we initially assess it on a two-dimensional toy task. Following this, we conduct evaluations on locomotion tasks in Mujoco, utilizing our original datasets that encompass a wide range of diverse behaviors.

4.1. Evaluation on Toy Task

To visualize the behavior of the proposed method in a simple task, we evaluated the performance of our method in a path planning problem on two dimensional space. In this toy task, the agent is represented as a point mass, the state is the position of the point mass in two-dimensional space, and the action is the small displacement of the point mass. The blue circle denotes the starting position, and the agent receives the reward if it reaches the red circle. Figure 2(a) shows the data samples in the dataset. Figure 2(b) shows the multiple solutions learned by the proposed method. The proposed method may not necessarily cover all solutions, but it can be confirmed that multiple solutions have been discovered by the proposed approach.



(a) Task setting and data samples. (b) Trajectories generated by the proposed method. (c) The latent variable learned with the proposed method.

Figure 2. Path planning task on two-dimensional space.

Figure 2(c) shows the latent variable learned by the proposed method, which learns the latent variable consistent with trajectories or type of solutions. In (c), the color indicates the mean of the latent variable, $\boldsymbol{\mu}$, where the posterior distribution $q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ is given by Gaussian $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma})$. The colors are separately assigned along the vertical axis, because the vertical positions characterize the behavior of the latent-conditioned policy. This result indicates that the proposed method learns the latent representations that are consistent with the types of solutions.

4.2. Performance on D4RL Tasks with Diverse Behaviors

4.2.1. Datasets with Diverse Behaviors

We constructed datasets that contain diverse locomotion behaviors based on the D4RL framework. To emphasize the existence of diverse solutions, we modified the velocity term of the reward function from that in the original tasks in the D4RL framework, as in (Kumar et al., 2020a):

$$r_{\text{vel}} = \min\left((x_t - x_{t-1})/\Delta t, v_{\max}\right), \tag{33}$$

where x_t represents the horizontal position of the agent at time t, and $v_{\max}$ is a constant term that defines the upper bound of r_{vel}. We refer to tasks with the above reward term as xxVel, e.g., Walker2dVel. To construct datasets with diverse locomotion behaviors, a latent-conditioned policy was trained with the method presented in (Osa et al., 2022), and we collected samples by executing the trained policy with random values of the latent variable. We repeated this process five times with different random seeds, and aggregated the collected samples into a dataset. Following the convention of the D4RL framework, we prepared four different types of datasets based on the policy level used to collect samples: "expert," "medium-expert," "medium," and "medium-replay." We describe the tasks in detail in Appendix B. Codes and videos are available in `https://takaosa.github.io/project_diveoff.html`.

4.2.2. Comparison with Baseline Methods

As a baseline method, we evaluated CLUE, which was proposed by Liu et al. (2023). To investigate the effect of the components of the proposed algorithm, we devised several methods for comparison ourselves and compared them with the proposed method. The proposed policy update rule presented in (10) can be combined with other methodologies for learning the latent representations. We refer to baseline methods based on (10) as AWAC-L. As a baseline method for learning the latent representations, we evaluated methods based on VAE. In a baseline method using VAE, VAE was trained with a given dataset $\mathcal{D}$. The latent variable estimated by VAE is augmented with the state as $\tilde{\boldsymbol{s}} = [\boldsymbol{s}, \boldsymbol{z}]$, and the policy that takes in $\tilde{\boldsymbol{s}}$ was trained with AWAC-L. We refer to this baseline method as AWAC-L+VAE. We also evaluated a variant of AWAC-L+VAE where we trained the critic with the expectile regression as in IQL (Kostrikov et al., 2022). We refer to this variant of AWAC-L+VAE as IQL-L+VAE. In our implementation of baseline methods except IQL-L, we used the double-clipped Q-learning as in TD3+BC (Fujimoto & Gu, 2021). In our implementation of AWAC-L+VAE and IQL-L+VAE, VAE was implemented with two hidden fully-connected layers.

As a variant of the AWAC-L+VAE, we also evaluated VAE with LSTM layers, which can consider temporal sequences for learning the latent representations. We refer to this variant of AWAC-L+VAE as AWAC-L+VAE with LSTM. In OPAL proposed by Ajay et al. (2021), the latent variables are learned with a recurrent neural network using a VAE-like loss function, and the latent-conditioned policy learned. In a sense that the latent variable encodes temporal information, AWAC-L+VAE with LSTM is similar to OPAL. The state normalization was employed in all the methods, including the proposed methods.

In the context of online RL, maximizing the mutual information between the state and latent variables is a prevalent approach to obtain diverse behaviors. As a baseline, we evaluated a variant of AWAC that maximizes the mutual information term in addition to the task reward as in (Kumar et al., 2020b; Sharma et al., 2020). When introducing a model that approximates the posterior distribution $q(\boldsymbol{z}|\boldsymbol{s})$, we can consider the auxiliary reward term given by

$$r_{\text{info}} = \log q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{s}), \tag{34}$$

where $q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{s})$is the approximated posterior distribution. The auxiliary reward term in (34) was used in DIAYN (Eysenbach et al., 2019), and it was later used by Kumar et al. (2020b); Sharma et al. (2020). We refer to the variant of AWAC-L+VAE that maximizes the reward term in (34) as AWAC-L+VAE+DIAYN.

To assess the impact of the components in our proposed method, we conducted an evaluation on a variant of DiveOff. In this variant, the posterior distribution was trained as part of a VAE, and the objective function in (28) was replaced with the standard VAE loss. We denote this variant as DiveOff+VAE. In DiveOff+VAE, the posterior distribution and the policy are alternately updated. SORL proposed by

Table 1. Normalized scores of methods that train latent-conditioned policies. WK = Walker2dVel, HP = HopperVel, HC = HalfcheetahVel, AN=AntVel. Best results are denoted in bold.

		TD3+BC	CLUE	AWAC-L + VAE	IQL-L + VAE	AWAC-L + VAE w/ LSTM	AWAC-L + VAE + DIAYN	DIVEOFF w/ VAE (OURS)	DIVEOFF (OURS)
DIV.-EXP.	WK	75.5±31.9	59.1±6.7	79.9±35.1	43.5±38.7	82.2±26.7	86.7±15.5	77.2 32.7	**93.3±9.9**
	HP	94.6±11.5	39.5±17.8	**98.6±3.0**	4.9±3.8	94.0±3.8	**98.0±3.3**	**99.7±0.6**	81.2 33.8
	HC	97.6±0.2	95.8±0.8	**96.5±0.4**	94.0±5.3	**96.6±0.2**	96.3±0.5	**97.0±0.2**	**96.8±0.3**
	AN	98.5±2.8	78.2±7.8	94.5±1.1	**97.2±3.4**	**95.4±0.4**	86.8±5.4	**95.8±0.8**	**95.4±1.2**
DIV.-EXP.-MED.	WK	96.5±7.0	54.7±15.3	93.7±9.1	42.0±27.4	62.9±6.9	**97.1±2.2**	95.8±5.9	80.5 17.7
	HP	92.1±14.0	74.8±22.8	94.8±7.2	89.3±23.4	79.7±24.3	78.7±36.5	94.8±9.2	**98.4±4.7**
	HC	97.1±4.2	**96.2±0.4**	**96.0±0.5**	91.8±14.2	**96.9±0.2**	**96.2±0.7**	**96.3±0.2**	**96.1±0.6**
	AN	96.2±3.8	79.2±10.6	84.8±6.2	**95.3±3.0**	87.4±3.8	82.1±6.3	89.8±0.6	88.5±1.9
DIV.-MED.	WK	77.0±13.7	47.4±14.2	**84.6±4.9**	74.3±23.2	57.3±15.9	74.8±11.1	79.1±19.3	67.9±17.4
	HP	88.1±9.0	65.0±7.2	93.1±5.3	87.1±9.1	82.2±15.6	92.4±6.3	**94.5±3.4**	83.1±18.0
	HC	92.3±2.3	92.3±3.0	**93.4±0.3**	**93.5±4.0**	**93.0±0.4**	**93.3±0.6**	93.3±0.4	**93.2±0.5**
	AN	64.3±64.1	66.3±5.4	80.5±2.4	**92.7±0.9**	76.5±5.1	75.3±5.7	81.8±2.6	81.1±2.3
DIV.-MED.-REP.	WK	80.6±28.4	17.5±13.0	26.8±9.3	**88.3±9.7**	43.5±26.8	51.4±5.9	27.5±29.0	39.3±11.6
	HP	67.9±23.1	93.3±7.4	96.6±5.8	**101.0±0.1**	6.6±3.9	58.9±42.4	99.1±3.9	**101.0±0.1**
	HC	93.2±0.6	89.9±2.5	**91.1±3.0**	**92.7±0.7**	**92.2±0.7**	89.3±3.8	80.4±25.0	89.8±6.4
	AN	91.9±3.2	18.5±3.8	34.6±2.3	**94.6±0.8**	35.5±1.6	34.7±3.0	33.9±2.3	32.9±2.9

Mao et al. (2024) iterates the clustering and policy improvement, without advantage-based weights in the clustering. In this sense, DiveOff+VAE can be interpreted as a variant of SORL. To quantify the diversity of the learned solutions, we employed the diversity metric used by Parker-Holder et al. (2020). We provide details on how to compute the diversity score in Appendix C. In the following, we report the mean and standard deviation over the 10 test episodes and five seeds unless otherwise stated. The value of the latent variable was sampled from the uniform distribution $U(-1, 1)$ at the beginning of the test episodes. The latent variable was continuous and two-dimensional in this experiment.

The normalized scores attained by the proposed method's variants are listed Table 1. Our algorithm consistently achieved high d4rl scores, demonstrating the stability of the learning process. A comparison with baseline methods that learn a single solution is provided in Appendix D. The results of a variant of DiveOff based on the expectile regression as in IQL is also provided in Appendix E.

As outlined in Table 2, our algorithm demonstrated a high diversity score. Notably, our method stands out as the sole approach capable of achieving both a high diversity score and a high d4rl score across tasks. Furthermore, the contrast between DiveOff+VAE and DiveOff underscores the efficacy of our proposed methodology in learning latent skill representations, thereby enhancing the diversity of the acquired policy.

While it was demonstrated that diverse behaviors can be learned with CLUE, the performance on the training tasks were not stable, compared with DiveOff. It is worth noting that CLUE was originally developed to leverage expert demonstrations in offline RL, and it is not specifically designed to discover multiple solutions from a single task in offline RL. When CLUE is applied to unsupervised RL, Liu et al. (2023) proposed to use the discrete latent representation, learned by a clustering algorithm such as K-means. However, unlike our method, important information such as the Q-values is not incorporated with such latent representations. Therefore, it is natural that DiveOff outperforms CLUE in our problem setting.

The comparison between AWAC-L+VAE+DIAYN and DiveOff with VAE in Tables 1 and 2 indicates the effect of strategies for maximizing mutual information. DIAYN is a prevalent method to encourage the diversity of behaviors in online RL, maximizing an auxiliary reward given by the lower bound of the mutual information between the state and latent variables. The results indicate that our strategy proposed in Section 3.4 is more suitable for our framework than DIAYN.

The comparison between AWAC-L+VAE and AWAC-L+VAE with LSTM shows that the use of LSTM in the posterior distribution does not improve performance in our problem setting. While AWAC-L+VAE with LSTM demonstrated reasonable performance with the "diverse-expert" datasets, the performance with "diverse-medium" and "diverse-medium-replay" datasets was relatively worse. The "diverse-medium" and "diverse-medium-replay" datasets do not contain expert behaviors, and it is considered that the "stitching" capability is more crucial for learning from these datasets.

Table 2. Diversity scores of methods that train latent-conditioned policies. WK = Walker2dVel, HP = HopperVel, HC = HalfcheetahVel, AN=AntVel. Best results are denoted in bold. The results for cases where the D4RL scores fall below 20.0 compared to the best score are presented in gray text. The results for cases where the D4RL scores are the best among the compared methods are presented with underlines.

		CLUE	AWAC-L + VAE	IQL-L + VAE	AWAC-L + VAE w/ LSTM	AWAC-L + VAE + DIAYN	DIVEOFF + VAE (OURS)	DIVEOFF (OURS)
DIV.-EXP.	WK	0.95±0.10	0.13±0.12	0.20±0.40	0.35±0.36	**0.48±0.33**	0.18±0.34	0.22±0.23
	HP	0.99±0.01	0.65±0.33	0.06±0.08	0.55±0.28	0.54±0.33	0.28±0.32	**0.88±0.11**
	HC	**0.99±0.01**	**0.96±0.06**	0.20±0.15	0.62±0.38	0.90±0.16	0.85±0.11	0.77±0.24
	AN	**0.71±0.32**	**0.20±0.34**	0.02±0.02	0.19±0.32	0.03±0.03	0.06±0.09	0.09±0.11
DIV.-EXP.-MED.	WK	0.98±0.02	0.38±0.39	0.432±0.38	0.81±0.23	0.21±0.22	0.29±0.26	**0.75±0.20**
	HP	0.99±0.01	0.05±0.07	0.36±0.39	0.23±0.22	0.21±0.40	0.31±0.36	**0.88±00.19**
	HC	**0.99±0.01**	0.72±0.20	0.40±0.48	0.81±0.22	0.62±0.26	0.65±0.30	**0.96±0.04**
	AN	0.42±0.25	0.38±0.27	0.04±0.05	0.18±0.28	0.407±0.22	0.38±0.29	**0.61±0.27**
DIV.-MED.	WK	0.99±0.01	0.25±0.25	**0.43±0.38**	0.44±0.29	0.31±0.25	0.34±0.37	**0.36±0.33**
	HP	0.99±0.01	0.46±0.28	**0.79±0.25**	0.60±0.34	0.61±0.36	0.45±0.33	**0.79±0.34**
	HC	**0.99±0.01**	0.52±0.25	0.05±0.04	0.73±0.25	0.57±0.32	0.75±0.22	0.81±0.13
	AN	0.84±0.17	0.56±0.35	0.02±0.04	0.15±0.23	0.56±0.31	0.38±0.29	**0.65±0.26**
DIV.-MED.-REP.	WK	0.81±0.33	0.00±0.00	**0.36±0.40**	0.39±0.48	0.32±0.39	0.01±0.02	0.01±0.01
	HP	**0.99±0.01**	0.00±0.02	0.00±0.00	0.46±0.39	0.78±0.27	0.00±0.00	0.02±0.03
	HC	**0.98±0.02**	0.01±0.03	0.14±0.16	0.04±0.08	0.01±0.02	0.00±0.00	0.00±0.00
	AN	0.33±0.41	0.24±0.21	0.0±0.0	0.43±0.47	0.40±0.49	0.42±0.48	0.06±0.12

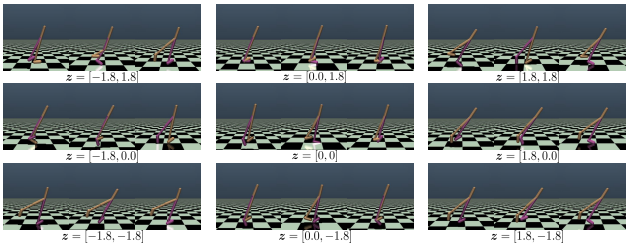


Figure 3. Sequential snapshots of locomotion behaviors obtained by DiveOff on the walker2dvel-diverse-expert task.

Notably, the methods evaluated in this study often failed to extract diverse behaviors from the "diverse-medium-replay" datasets. While it is believed that the "diverse-medium-replay" data may have the most diverse actions, the diversity of successful states is considered to be much less than that of the "diverse-expert" dataset. If the successful states are not diverse, the actions within those states are also not diverse. It is considered that this characteristic of the "diverse-medium-replay" datasets is attributed to the failure observed in these datasets.

As a qualitative result, Figure 3 visualizes the locomotion behaviors obtained by DiveOff on the walker2dvel-diverse-expert tasks. Here, the agent performed one-leg hopping when $\boldsymbol{z} = [-1.8, -1.8]$, and walked with two legs when $\boldsymbol{z} = [1.8, 1.8]$. As the policy is conditioned on a continuous latent variable, we can continuously change the locomotion behavior by manipulating the latent variable's value.

A limitation of this study is that the learned behavior varies with different random seeds. This is a common characteristic in existing deep reinforcement learning algorithms. However, we posit that addressing this issue could be feasible if it were possible to learn all learnable behaviors. Consequently, there is room for future development of algorithms with the potential to acquire a more diverse range of behaviors.

4.3. Few-shot Adaptation to Unseen Tasks

The purpose of the few-shot adaptation experiment was to demonstrate that learning multiple behaviors from a single task in a training environment allows us to perform few-shot adaptation to a new environment by selecting a usable behavior from behaviors obtained in the training environment. We prepared two environments for the few-shot adaptation experiment shown in Figure 4. The embodiment of the agent, such as lengths of links, is changed on these tasks.

In the few-shot adaptation experiment, policies trained in offline RL were adapted to a test environment different from the one used for training. The protocol of the few-shot adaptation experiment was based on the one reported in (Kumar et al., 2020b). Given a latent-conditioned policy trained via offline RL, we sampled the value of the latent variable uniformly and executed the latent-conditioned policy in a test environment with the sampled value of the latent variable. In this phase, we tested each policy only once and evaluated the policy with K different values of the latent variable. In (Kumar et al., 2020b), K is called the budget for

Table 3. Results of few-shot adaptation with the hopper agent.

TRAIN. DATASET	TEST AGENT	AWAC-L + VAE	AWAC-L + VAE + DIAYN	DIVEOFF W/ VAE (OURS)	DIVEOFF (OURS)
DIV.-EXP.	LOWKNEE	766.4±631.4	817.7±623.8	**1374.76±663.6**	764.7±330.1
	LONGHEAD	1139.7±468.0	**1440.6±563.7**	1363.06±423.0	**1401.1±458.9**
DIV.-EXP.-MED.	LOWKNEE	1898.8±54.1	1689.5±304.2	**1950.46±37.0**	**1892.8±161.7**
	LONGHEAD	1645.3±543.2	1498.2±727.3	**1948.96±46.9**	988.6±597.0
DIV.-MEDIUM	LOWKNEE	1176.4±619.8	**1797.9±166.7**	1626.66±274.7	1359.1±380.0
	LONGHEAD	1587.8±235.1	1737.9±88.7	1652.26±285.6	**1859.5±154.0**
DIV.-MED.-REPL.	LOWKNEE	1593.2±439.9	1142.4±469.4	**1780.16±336.1**	1594.1±321.3
	LONGHEAD	1492.4±453.2	1146.1±584.8	**1958.26±33.1**	**1941.8±62.2**

(a) Hopper-LowKnee. (b) Hopper-LongHead.

Figure 4. Tasks for few-shot adaptation. Links modified from the original agent are indicated as blue.

few-shot adaptation, and we set $K = 25$ in our experiment. The latent variable was continuous and two-dimensional in this experiment, and we uniformly sampled the value of the latent variable during the adaptation phase. Additional details pertaining these tasks are provided in Appendix F.

The results of the few-shot adaptation experiments are summarized in Table 3. The proposed methods outperformed baseline methods in this experiment. These results correlate with the diversity scores shown in Table 2, demonstrating that the proposed method successfully enables few-shot adaptation to new environments by learning diverse behaviors from a single task in offline RL. These results also indicate that the proposed method discovers more diverse behaviors than the compared baseline methods.

5. Conclusions

In this study, we tackled the challenge of obtaining multiple solution in offline RL. We introduced a novel algorithm designed to learn diverse behaviors from individual tasks in offline RL, resembling a coordinate ascent approach akin to the EM algorithm. Experimental results illustrate that our proposed algorithm effectively learns multiple solutions in offline RL and that our method learn more diverse solutions than baseline methods. Additionally, the findings suggest that the acquired multiple behaviors enables few-shot adaptation to new environments with limited test samples. While we have presented a framework for formulating the problem of learning multiple solutions in offline RL, there is still room for improvement in enhancing the diversity of behaviors. We hope to inspire future work to investigate alternative approaches to discover diverse solutions in offline RL.

Impact Statement

This study was undertaken with the aim of contributing to the field of reinforcement learning. Reinforcement learning has various applications in society, and its impact is considered diverse. However, regarding this study, we believe there are no notable ethical issues or concerns.

Acknowledgements

This work was partially supported by JST Moonshot R&D Grant Number JPMJPS2011, CREST Grant Number JPMJCR2015, JSPS KAKENHI Grant Number JP23K18476 and Basic Research Grant (Super AI) of Institute for AI and Beyond of the University of Tokyo.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Barber, D. and Agakov, F. The im algorithm : A variationalapproach to information maximization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 16, pp. 201–208, 2003.

Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006.

Chatzilygeroudis, K., Cully, A., Vassiliades, V., and Mouret,

J.-B. Quality-diversity optimization: A novel branch of stochastic optimization. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, 2021.

Chen, X., Ghadirzadeh, A., Yu, T., Wang, J., Gao, Y., Li, W., Liang, B., Finn, C., and Zhang, C. Lapo: Latent-variable advantage-weighted policyoptimization for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Cully, A. and Demiris, Y. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2018. doi: 10.1109/TEVC.2017.2704781.

Deisenroth, M. P., Neumann, G., and Peters, J. A survey on policy search for robotics. *Foundations and Trends®in Robotics*, 2013.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

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

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:20132–20145, 2021.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1587–1596, 2018.

Hein, D., Udluft, S., and Runkler, T. A. Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence*, 76: 158–169, 2018a.

Hein, D., Udluft, S., and Runkler, T. A. Generating interpretable fuzzy controllers using particle swarm optimization and genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018b.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 11784–11794, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1179–1191, 2020a.

Kumar, S., Kumar, A., Levine, S., and Finn, C. One solution is not all you need: Few-shot extrapolation via structured MaxEnt RL. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 8198–8210, 2020b.

Liu, J., Zu, L., He, L., and Wang, D. Clue: Calibrated latent guidance for offline reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 229, pp. 906–927, 2023.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Mao, Y., Wu, C., Chen, X., Hu, H., Jiang, J., Zhou, T., Lv, T., Fan, C., Hu, Z., Wu, Y., Hu, Y., and Zhang, C. Stylized offline reinforcement learning: Extracting diverse high-quality behaviors from heterogeneous datasets. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Nair, A., Gupta, A., Dalal, M., and Levine, S. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv*, arXiv:2006.09359, 2020.

Osa, T. Motion planning by learning the solution manifold in trajectory optimization. *The International Journal of Robotics Research*, 41(3):291–311, 2022.

Osa, T., Tangkaratt, V., and Sugiyama, M. Discovering diverse solutions in deep reinforcement learning by maximizing state-action-based mutual information. *Neural Networks*, 152:90–104, 2022.

Osa, T., Hayashi, A., Deo, P., Morihira, N., and Yoshiike, T. Offline reinforcement learning with mixture of deterministic policies. *Transactions on Machine Learning Research*, 2023.

Parker-Holder, J., Pacchiano, A., Choromanski, K., and Roberts, S. Effective diversity in population based reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 18050–18062, 2020.

Peters, J., Mulling, K., and Altün, Y. Relative entropy policy search. In *n Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

Pugh, J. K., Soros, L. B., and Stanley, K. O. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3(40), 2016.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, Second edition, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Toussaint, M. A., Allen, K. R., Smith, K. A., and Tenenbaum, J. B. Differentiable physics and stable modes for tool-use and manipulation planning. In *Proceedings of Robotics: Science and Systems Foundation*, 2018.

Zhou, W., Bajracharya, S., and Held, D. Plas: Latent action space for offline reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 155, pp. 1719–1735, 2020.

A. Derivation of the latent-conditioned policy update

We can show the relationship between (7) and (8) in the main manuscript by extending the results presented in (Peters et al., 2010; Deisenroth et al., 2013; Nair et al., 2020). We consider an offline RL problem where a dataset is collected by a behavior policy that exhibits multiple behaviors. Here, we assume that the behavior policy is given by

$$\beta(\boldsymbol{a}|\boldsymbol{s}) = \int \beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}, \tag{35}$$

where $p(\boldsymbol{z})$ represents the prior distribution of the latent variable, and the value of $\boldsymbol{z}$ determines the behavior style of the behavior policy. A fixed dataset can be regarded as samples generated from the distribution $d^\beta(\boldsymbol{s})\beta(\boldsymbol{a}|\boldsymbol{s})$, where $d^\beta(\boldsymbol{s})$ represents the stationary distribution of the state obtained by executing the behavior policy $\beta(\boldsymbol{a}|\boldsymbol{s})$. For example, when a dataset is constructed by executing two distinctive policies, the latent variable is discrete, e.g. $z \in \{0, 1\}$, and the prior distribution $p(\boldsymbol{z})$ defines the ratio of executing these policies.

According to the above assumption, we consider the problem formulated as follows:

$$\max_\pi \mathbb{E}_{\boldsymbol{a}\sim\pi}\left[A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right] \tag{36}$$

$$\text{s.t.} \quad D_{\text{KL}}\left(\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})||\beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\right) \leq \epsilon \tag{37}$$

$$\int \pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})d\boldsymbol{a} = 1 \tag{38}$$

Here, we assume that the approximated posterior distribution $q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ is given. The problem in (38) is equivalent to (7) in the main manuscript when taking the expectation with respect to $d^\beta(\boldsymbol{s})$ and $q(\boldsymbol{z}|\boldsymbol{s})$. The Lagrangian of the constraint problem in (38) is given by

$$\mathcal{L}(\pi) = \mathbb{E}_{\boldsymbol{a}\sim\pi}\left[A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right] + \alpha\left(\epsilon - D_{\text{KL}}\left(\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})||\beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\right)\right) + \lambda\left(1 - \int \pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})d\boldsymbol{a}\right). \tag{39}$$

By taking the partial derivative of $\mathcal{L}(\pi)$ with respect to π, we obtain the following equation:

$$\frac{\partial\mathcal{L}(\pi)}{\partial\pi} = A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}) + \alpha\left(-\log\beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) + \log\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) + 1\right) - \lambda, \tag{40}$$

When we set $\frac{\partial\mathcal{L}(\pi)}{\partial\pi} = 0$, we obtain a non-parametric solution to the problem in (38):

$$\pi^* \propto \beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\exp\left(\frac{1}{\alpha}A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right). \tag{41}$$

Given a dataset $\mathcal{D} = \{(\boldsymbol{s}_i,\boldsymbol{a}_i)\}_{i=1}^N \sim d^\beta(\boldsymbol{s})\beta(\boldsymbol{a}|\boldsymbol{s})$ and the posterior distribution $q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$, we project the non-parametric solution to the parametric model $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})$ by minimizing the KL divergence:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}\sim d^\beta,\boldsymbol{z}\sim q}\left[D_{\text{KL}}\left(\beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\exp\left(\frac{1}{\alpha}A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)\middle\|\,\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\right)\right]. \tag{42}$$

The KL divergence in (42) can be rewritten as

$$\begin{aligned}
&D_{\text{KL}}\left(\beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\exp\left(\frac{1}{\alpha}A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)\middle\|\,\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\right)\\
&= \int \beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\exp\left(\frac{1}{\alpha}A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)\log\beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\exp\left(\frac{1}{\alpha}A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)d\boldsymbol{a}\\
&\quad - \int \beta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\exp\left(\frac{1}{\alpha}A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})d\boldsymbol{a}.
\end{aligned} \tag{43}$$

As the first term in the right-hand side of (43) is independent of the policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})$, the problem in (42) can be rewritten as

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}\sim d^\beta,\boldsymbol{a}\sim\beta,\boldsymbol{z}\sim q}\left[\exp\left(\frac{1}{\alpha}A_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\right)\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\right]. \tag{44}$$

Thus, we can obtain the relationship between (7) and (8) in the main manuscript.

B. Details of diverse datasets

B.1. Details of tasks in diverse datasets

In the reward function of the original tasks in D4RL, a velocity term r_{vel} is used to encourage the agent to walk faster, and r_{vel} is given by

$$r_{\text{vel}} = (x_t - x_{t-1}))/\Delta t, \tag{45}$$

where x_t represents the horizontal position of the agent at time t, and Δt represents the time step size in the simulation. To emphasize the existence of diverse solutions, we modified the velocity term of the reward function from that in the original tasks in the D4RL framework, as in (Kumar et al., 2020a):

$$r_{\text{vel}} = \min\left((x_t - x_{t-1})/\Delta t, v_{\text{max}}\right), \tag{46}$$

where x_t represents the horizontal position of the agent at time t, and v_{max} is a constant term that defines the upper bound of r_{vel}. The value of v_{max} used in each agent is presented in Table 4.

The maximum return obtained using the trained policy and the minimum return obtained using a random policy are presented in Table 5.

Table 4. Upper bound for the velocity term in our dataset

Agent	v_{max}
Walker	2
Hopper	1
Halfcheetah	2
Ant	1.5

Table 5. Maximum returns obtained using the trained policy and the minimum return obtained using a random policy

Agent	Max. return	Min. return
Walker	2860	-4.02
Hopper	1962	6.84
Halfcheetah	1869	-324.79
Ant	2265	-379.33

B.2. How to construct the diverse dataset

We collected the samples by running LTD3 proposed in (Osa et al., 2022) according to the author implementation in `https://github.com/TakaOsa/LTD3`. To create diverse datasets, we ran LTD3 five times with different random seeds with the two-dimensional continuous latent variable. For "diverse-expert" and "diverse-medium" datasets, we collected samples by running the trained policies with uniformly generated values of the latent variable. As the trained policy occasionally resulted in an episode with a low return, we removed such episodes from the "diverse-expert" and "diverse-medium" datasets. The minimum return to be included in a dataset is presented in Table 6. The approximate number of samples in the "diverse-expert" and "diverse-medium" datasets was 300,000, although there was a variance in the number of samples due to the variance of the number of the steps in episodes. The number of time steps for training policies for each dataset is presented in Table 7. The "medium-expert" datasets were constructed by concatenating the "expert" and "medium" datasets. Therefore, the approximate number of samples in the "diverse-medium-expert" datasets was 600,000.

To construct the "medium-replay" datasets, we aggregated the data in replay buffers from five different random seeds. To avoid making the datasets too large, we did not include all the samples of the replay buffers in the "medium-replay" datasets . We selected and included one episode after every N episodes from the replay buffer, and the data from five random seeds were aggregated. The value of N was changed for each agent owing to the file size of the dataset. The values of N are presented in Table 8. The hyperparameters of LTD3 used to train policies for constructing our datasets are presented in

Table 10. We also provide visualization of the locomotion behaviors in the expert-diverse datasets in Figure 5, which we omitted from the main manuscript owing to the page limitation.

Table 6. Minimum return for each dataset

Agent	"expert"	"medium"
Walker	2000	500
Hopper	1700	500
Halfcheetah	1650	500
Ant	1700	500

Table 7. Training time steps to obtain policies for each dataset

Agent	"expert"	"medium"	"medium-replay
Walker	3 million	1 million	1 million
Hopper	3 million	1 million	1 million
Halfcheetah	2 million	250,000	250,000
Ant	3 million	1 million	1 million

Table 8. Interval to be included in the medium-replay dataset

Agent	N
Walker	5
Hopper	5
Halfcheetah	2
Ant	5

B.3. Diversity of the samples in our datasets

To quantify sample diversity within the datasets, we evaluated the entropy of their states. However, as the datasets' sample distributions are unknown, the entropy estimation is not a trivial task. Instead of directly estimating the entropy, we therefore evaluated its upper bound. When the density is approximated with a Gaussian mixture model as $p(\boldsymbol{s}) = \sum_{i=1}^{M} w_i \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the upper bound of the entropy $\mathcal{H}(\boldsymbol{s})$ is given as follows:

$$\mathcal{H}(\boldsymbol{s}) \leq \sum_{i=1}^{L} w_i \left(\frac{D}{2} \log(1 + 2\pi) + \frac{1}{2} \log \det(\boldsymbol{\Sigma}) - \log w_i \right) \tag{47}$$

Table 9. Diversity of datasets based on the state entropy. WK = Walker2d, HP = Hopper, HC = Halfcheetah.

	diverse-expert-v0 (ours)	expert-v2 (d4rl)	diverse-medium-v0 (ours)	medium-v2 (d4rl)
WK	**5.97±1.19**	-5.77±1.94	**5.73±1.96**	2.06±2.35
HP	**1.76±0.78**	-2.29±1.03	**5.0±1.03**	-0.44±1.28
HC	**13.73±0.26**	1.34±0.4	**17.01±0.6**	11.96±1.36

	diverse-medium-expert-v0 (ours)	medium-expert-v2 (d4rl)	diverse-medium-replay-v0 (ours)	medium-replay-v2 (d4rl)
WK	**6.26±1.67**	0.76±2.03	8.41±1.72	12.65±0.84
HP	**4.35±0.68**	2.25±0.68	6.31±1.17	6.88±0.67
HC	**14.41±0.19**	7.86±1.34	**18.93±0.47**	14.11±0.44

To compute the upper bound of $\mathcal{H}(\boldsymbol{s})$ in (47), we selected a batch of samples from the dataset and fitted a Gaussian mixture model. The number of Gaussian components was determined based on the Bayesian information criteria.

We compared the upper bound of the entropy with that for existing D4RL datasets, with results listed in Tablee 9. As shown, our datasets exhibit higher values compared with existing D4RL datasets, indicating higher behavioral diversity.

We also provide visualization of the locomotion behaviors in the expert-diverse datasets in Figure 5, which we omitted from the main manuscript owing to the page limitation.

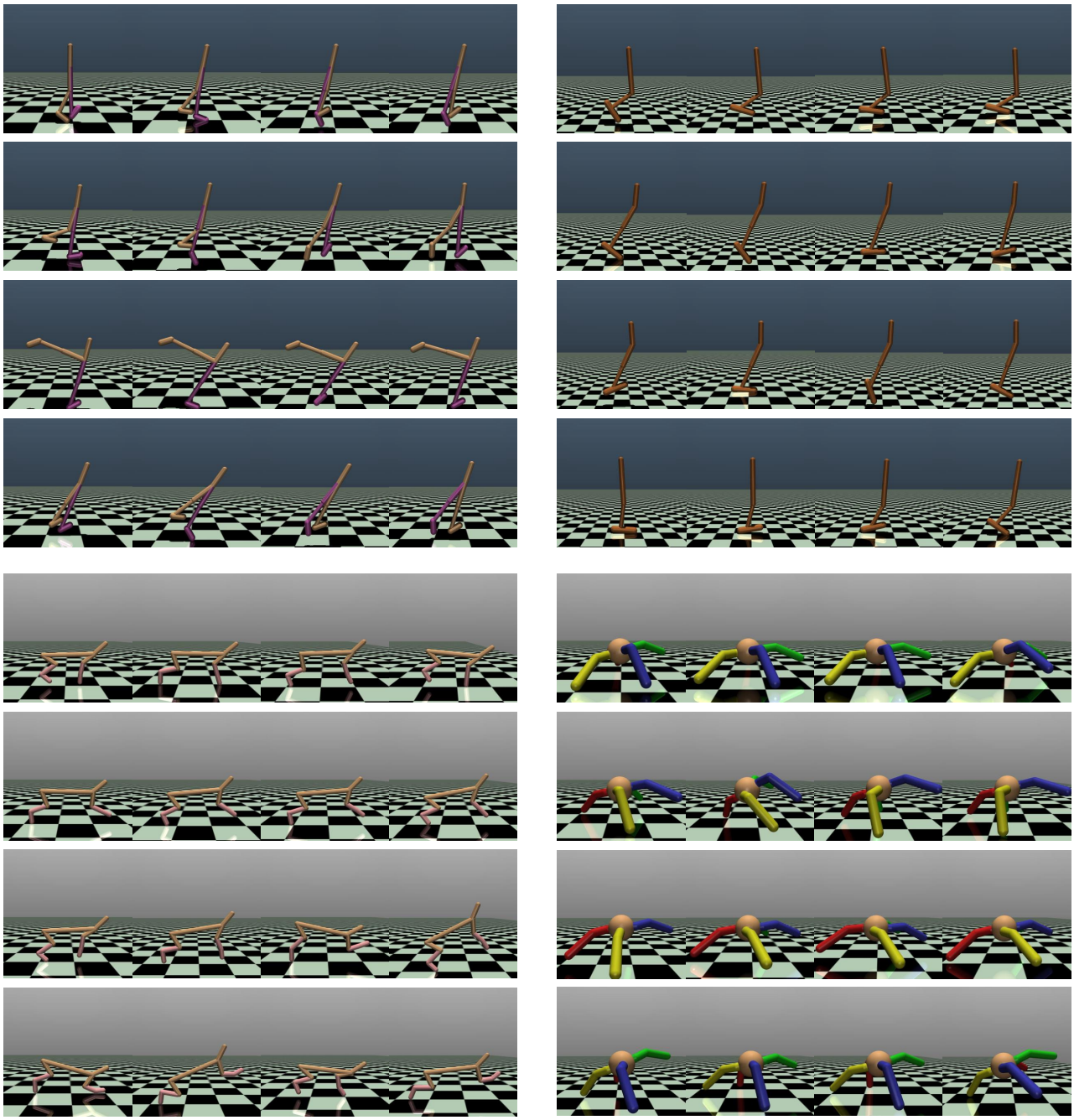


Figure 5. Examples of locomotion behaviors contained in the expert-diverse datasets. The color of the legs of the ant agent was changed for visibility.

Table 10. Hyperparameters used for LTD3 to construct our datasets

Parameter	Value
Optimizer	Adam
Learning rate	$3 \cdot 10^{-4}$
Discount factor γ	0.99
Replay buffer size	10^6
Number of hidden layers	2
Number of hidden units per layer	256
Number of samples per minibatch	256
Activation function	Relu
Target smoothing coefficient	0.005
gradient steps per time step	1
Clipping param. for IW	0.3
interval for maximizing $\mathcal{J}_{\mathrm{Q}}(\boldsymbol{\theta})$	2
interval for maximizing $\mathcal{J}_{\mathrm{info}}(\boldsymbol{\theta})$	5 for walker, 4 for others

C. Diversity score

To quantify the diversity of the learned solutions, we employed the diversity metric used in (Parker-Holder et al., 2020). Given a set of policies $\{\pi_i\}_{i=1}^{M}$, the diversity metric is expressed as

$$D_{\text{div}} = \det\left(K(\boldsymbol{\phi}(\pi_i), \boldsymbol{\phi}(\pi_j))_{i,j=1}^{M}\right), \tag{48}$$

where $\boldsymbol{\phi}(\pi) \in \mathbb{R}^l$ is the behavior embedding of policy π, and $K : \mathbb{R}^l \times \mathbb{R}^l \mapsto \mathbb{R}$ is a kernel function. Specifically, we used the squared-exponential kernel function given by

$$k(\boldsymbol{\phi}(\pi_i), \boldsymbol{\phi}(\pi_j)) = \exp\left(-\frac{\|\boldsymbol{\phi}(\pi_i) - \boldsymbol{\phi}(\pi_j)\|^2}{2h^2}\right), \tag{49}$$

where $\boldsymbol{\phi}(\pi_i)$ is the policy embedding given by the mean of the states visited by the policies as $\boldsymbol{\phi}(\pi_i) = \mathbb{E}_{\boldsymbol{s}\sim\pi_i,\mathcal{P}}[\boldsymbol{s}]$. We uniformly sampled the value of the latent variable and computed the policy embedding, with $M = 9$. In the main manuscript, we reported the average and the standard deviation over five different random seeds.

D. Comparison with methods that learn a single solution

We evaluated the performance of prevalent offline RL methods that finds a single solution on our datasets. Specifically, we considered behavior cloning, TD3+BC (Fujimoto & Gu, 2021), IQL (Kostrikov et al., 2022), AWAC (Nair et al., 2020), and LAPO (Chen et al., 2022) as baseline methods that learn a single behavior. For TD3+BC, we used the author-provided implementations. In our implementation of AWAC, we used the state normalization and double-clipped Q-learning as in TD3+BC. We implemented IQL by adopting the author-provided implementation of IQL to pytorch. We report the performance of policies after training with 1 million steps unless otherwise stated. For DiveOff, a learning rate for $\mathcal{L}_{\text{M-info}}$ was $6e-4$ for all the task, and the D4RL score is different from Table 1.

In Table 11, DiveOff (ave.) is the average over different behaviors, while DiveOff (best) represents the results when we report the results of the latent variables that led to the highest return. To select the latent variable for DiveOff (best), we uniformly sampled the value of the latent variable 25 times, and the value that led to the highest total returns in three test episodes was used as the best value.

As shown, DiveOff (best) outperforms DiveOff (ave), and DiveOff (best) achieved a result comparable or superior to prevalent offline RL methods that learn a single policy. Although learning multiple behaviors potentially increase the computational and algorithmic complexity, the performance of the proposed method in terms of the normalized score was found to be comparable to that of the baseline methods. LAPO is designed to deal with multimodal distributions in a given dataset, and outperformed other baselines in diverse-expert datasets. Notably, DiveOff achieved a performance comparable to LAPO on diverse-expert datasets, and the overall performance of DiveOff was better than LAPO.

Interestingly, the performance of IQL on the walker2d-diverse-expert and hopper-diverse-expert datasets was lower than that of other baseline methods, whereas IQL often achieved state-of-the-art performance on the expert dataset in D4RL. However,

Table 11. Results on diverse dataset. WK = Walker2dVel, HP = HopperVel, HC = HalfcheetahVel, AN = AntVel. Best results are denoted in bold. The average and standard deviation of normalized scores over the 10 test episodes and five seeds are shown.

		BC	TD3+BC	IQL	AWAC	LAPO	DIVEOFF (AVE.)	DIVEOFF (BEST)
DIV.-EXP.	WK	95.0±4.8	75.5±31.9	23.5±31.0	90.8±3.3	93.1±7.4	98.5±2.2	**99.8±0.5**
	HP	**99.2±0.8**	94.6±11.5	66.5±39.4	92.5±14.1	**100.6±0.1**	96.4±3.8	99.7±0.6
	HC	95.9±0.7	97.6±0.2	**98.4±0.5**	96.5±0.2	97.5±0.2	96.8±0.1	96.7±0.3
	AN	93.6±2.6	**98.5±2.8**	97.1±2.1	93.8±2.4	96.6±1.0	95.4±1.2	95.9±0.6
DIV.-MED.-EXP.	WK	63.5±24.8	96.5±7.0	49.1±36.5	74.7±22.2	68.5±21.5	**98.0±2.5**	**98.8±1.8**
	HP	98.1±4.9	92.1±14.0	**101.1±0.0**	79.1±23.9	100.2±0.5	98.9±2.2	98.9±1.7
	HC	95.8±0.8	**97.3±0.2**	**97.1±4.2**	95.9±0.6	**97.4±0.2**	96.2±0.1	96.4±0.2
	AN	79.5±6.7	**96.2±3.8**	91.3±5.2	85.1±3.7	94.8±1.2	88.5±1.9	88.9±1.1
DIV.-MED.	WK	76.3±2.6	77.0±13.7	80.1±8.1	76.9±11.3	**90.3±4.8**	**88.4±9.3**	**89.1±5.6**
	HP	73.5±24.4	88.1±9.0	94.5±6.2	**96.9±3.0**	**98.2±3.4**	95.1±6.2	95.6±5.3
	HC	92.6±0.2	92.3±2.3	**94.5±1.8**	92.9±0.4	**94.4±0.2**	93.2±0.1	93.3±0.3
	AN	75.6±4.3	64.3±64.1	**91.9±0.3**	79.9±3.4	83.3±3.7	81.1±2.3	81.8±1.0
DIV.-MED.-REP.	WK	26.0±22.9	80.6±28.4	**83.4±20.1**	32.8±15.5	46.7±18.0	44.2±18.2	52.5±23.0
	HP	26.5±5.2	67.9±23.1	**101.1±0.0**	100.9±0.2	77.0±6.1	**101.1±0.1**	**101.1±0.1**
	HC	14.6±0.2	**93.2±0.6**	**92.8±0.4**	86.5±7.3	44.6±24.8	85.6±11.0	87.0±7.5
	AN	32.9±1.7	**91.9±3.2**	91.4±0.6	35.0±2.5	37.6±6.4	32.9±2.9	36.0±2.5

IQL significantly outperformed other baseline methods on diverse-medium-replay datasets, indicating the "stitching" ability of IQL.

E. Comparison of variants of the baseline and proposed method

As additional baseline, we evaluated the policy update based on TD3+BC, which we refer to as TD3-L+BC+VAE. In TD3-L+BC+VAE, we used the latent-conditioned behavioral cloning term, and the objective function for TD3-L+BC+VAE is given by

$$\begin{aligned}\mathcal{L}_{\text{TD3+VAE}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s}\sim\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim q}\left[Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z}), \boldsymbol{z})\right] \\ -\lambda\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim q}\left[\left(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z}) - \boldsymbol{a}\right)^2\right],\end{aligned} \tag{50}$$

where λ is a constant, and q is the posterior distribution as a part of VAE. To implement TD3-L+BC+VAE, we mofieid the author-provided implementation of TD3+BC. We evaluated TD3-L+BC+VAE with $\lambda \in 1.0, 2.5, 5.0, 10.0$ in (50), but did not observe significant differences. Therefore, we report the results with $\lambda = 2.5$ as in the default setting in TD3+BC.

While we reported the performance of DiveOff based on the double-clipped Q-learning in the main text, alternative methods can be employed. To investigate the effect of the algorithm for learning the critic, we evaluate a variant of DiveOff using the quantile regression as in IQL (Kostrikov et al., 2022).

The results are reported in Tables 12 and 13.

The results indicate that replacing double-clipped-Q-learning with IQL improves the performance on the replay dataset. However, the performance on some expert datasets got worse. This effect can also be observed by comparing AWAC-L+VAE and IQL-L+VAE in Table 1. Previous studies indicate that IQL has better stitching capability compared to double-clipped-Q-learning. The performance gain on the replay datasets implies that the stitching capability is necessary to learn from the replay datasets.

F. Details of few-shot adaptation experiments

F.1. Environments

The purpose of the few-shot adaptation experiment was to demonstrate that learning multiple behaviors from a single task in a training environment allows us to perform few-shot adaptation to a new environment by selecting a usable

Table 12. Normalized scores of methods that train latent-conditioned policies. WK = Walker2dVel, HP = HopperVel, HC = HalfcheetahVel, AN=AntVel. Best results are denoted in bold.

		TD3+BC+VAE	DIVEOFF (QUANTILE REGRESS.)	DIVEOFF (DOUBLE CLIPPED Q)
DIV.-EXP.	WK	3.7±3.6	27.7±35.9	**93.3±9.9**
	HP	0.7±0.2	6.6±8.4	**81.2±33.8**
	HC	14.6±0.1	94.2±8.7	**96.8±0.3**
	AN	51.5±0.4	**99.1±0.3**	95.4±1.2
DIV.-EXP.-MED.	WK	4.7±3.5	49.4±30.2	**80.5±17.7**
	HP	0.8±0.2	**101.1±0.0**	98.4±4.7
	HC	14.6±0.0	**97.4±4.2**	96.1±0.6
	AN	51.7±0.3	**96.9±0.4**	88.5±1.9
DIV.-MED.	WK	4.1±3.3	62.8±20.9	**67.9±17.4**
	HP	0.8±0.3	**99.7±2.7**	83.1±18.0
	HC	14.6±0.0	**95.5±0.4**	93.2±0.5
	AN	51.7±0.2	**92.9±0.8**	81.1±2.3
DIV.-MED.-REP.	WK	3.4±3.8	**90.8±7.0**	39.3±11.6
	HP	1.1±0.4	**101.1±0.0**	**101.0±0.1**
	HC	14.7±0.0	**93.5±0.5**	89.8±6.4
	AN	51.8±0.3	**95.0±0.7**	32.9±2.9

behavior from behaviors obtained in the training environment. We prepared two environments for the few-shot adaptation experiments. For the hopper agent, we prepared the `HopperLowKnee-v0` and `HopperLongHead-v0` environments. In HopperLowKnee-v0, the position of the knee joint was lower than that of the normal hopper agent. In HopperLongHead-v0, the top link was longer than that of the normal hopper agent.

F.2. Procedure of the few-shot adaptation experiment

In the few-shot adaptation experiment, policies trained in offline RL were adapted to a test environment different from the one used for training. The protocol of the few-shot adaptation experiment was based on the one reported in (Kumar et al., 2020b). Given a latent-conditioned policy trained via offline RL, we sampled the value of the latent variable uniformly and executed the latent-conditioned policy in a test environment with the sampled value of the latent variable. In this phase, we tested each policy only once and evaluated the policy with K different values of the latent variable. In (Kumar et al., 2020b), K is called the budget for few-shot adaptation, and we set $K = 25$ in our experiment. Then, we stored the value of the latent variable $\boldsymbol{z}_{\text{max}}$ that resulted in the maximum return. Subsequently, we ran the latent-conditioned policy with $\boldsymbol{z} = \boldsymbol{z}_{\text{max}}$ in the test environment 10 times and recorded the average return. We repeated the above process five times for different random seeds, and the corresponding average and standard deviation are presented in Table 3.

G. Implementation details and hyperparameters

In DiveOff and AWAC, we used the state normalization employed in TD3+BC (Fujimoto & Gu, 2021) and the double-clipped Q-learning (Fujimoto et al., 2018). The neural network architecture and hyperparameters are presented in Table 14.

H. Effect of hyperarameters

We provide the effect of the learning rate of the optimizer for $\mathcal{L}_{\text{M-info}}$ in Table 18. The rest of the hyperparamters were the same as in Table 14. As we fixed the learning rate for other objective functions, changing the learning rate for $\mathcal{L}_{\text{M-info}}$ corresponds to the change of the weight on $\mathcal{L}_{\text{M-info}}$. The higher learning rate leads to the larger weight on $\mathcal{L}_{\text{M-info}}$. As shown in Table 18, we observed the trade-off between the D4RL score and diversity score.

Table 13. Diversity scores of methods that train latent-conditioned policies. WK = Walker2dVel, HP = HopperVel, HC = HalfcheetahVel, AN=AntVel. Best results are denoted in bold. The results for cases where the D4RL scores fall below 20.0 compared to the best score are presented in gray text. The results for cases where the D4RL scores are the best among the compared methods are presented with underlines.

		TD3+BC + VAE	DIVEOFF (QUANTILE REGRESS.)	DIVEOFF (DOUBLE CLIPPED Q)
DIV.-EXP.	WK	0.24±0.12	0.025±0.05	0.22±0.23
	HP	0.71±0.36	0.07±0.10	**0.88±0.11**
	HC	0.09±0.08	**0.73±0.28**	0.77±0.24
	AN	0.05±0.08	**0.14±0.18**	0.09±0.11
DIV.-EXP.-MED.	WK	0.25±0.21	0.46±0.20	**0.75±0.20**
	HP	0.65±0.37	0.05±0.09	**0.88±00.19**
	HC	0.06±0.06	0.07±0.05	**0.96±0.04**
	AN	0.28±0.30	0.06±0.06	**0.61±0.27**
DIV.-MED.	WK	0.32±0.30	**0.43±0.35**	0.36±0.33
	HP	0.66±0.34	0.68±0.29	**0.79±0.34**
	HC	0.04±0.04	0.12±0.08	0.81±0.13
	AN	0.23±0.29	0.09±0.12	**0.65±0.26**
DIV.-MED.-REP.	WK	0.21±0.39	**0.11±0.14**	0.01±0.01
	HP	0.66±0.37	0.001±0.001	0.02±0.03
	HC	0.004±0.005	**0.29±0.35**	0.00±0.00
	AN	0.07±0.12	0.0±0.0	0.06±0.12

Table 14. Hyperparameters of DiveOff.

Hyperparameter	Value
Optimizer	Adam
Learning rate for critic	3e-4
Learning rate for actor $\pi_{\boldsymbol{\theta}}$	3e-4
Learning rate for posterior distribution $q_{\boldsymbol{\phi}}$	3e-4 (initialization)
Learning rate for Likelihood $p_{\boldsymbol{\psi}}$	3e-4 (initialization)
Learning rate for $\mathcal{L}_{\text{M-info}}$	6e-5 for antvel, 9e-5 for others
Mini-batch size	256
Discount factor	0.99
Target update rate	5e-3
Policy update frequency	2
Score scaling $1/\alpha_\pi$	3.0
Score scaling $1/\alpha_q$	1.0
Critic activation function	ReLU
Actor activation function	ReLU
Encoder hidden dim	256
Encoder hidden layers	2
Encoder activation function	ReLU
Decoder hidden dim	256
Decoder hidden layers	2
Decoder activation function	ReLU
λ	2.0

Table 15. Hyperparameters of TD3-L+BC+VAE.

Hyperparameter	Value
Optimizer	Adam
Critic learning rate	3e-4
Actor learning rate	3e-4
Mini-batch size	256
Discount factor	0.99
Target update rate	5e-3
Policy noise	0.2
Policy noise clipping	(-0.5, 0.5)
Policy update frequency	2
Critic hidden dim	256
Critic hidden layers	2
Critic activation function	ReLU
Actor hidden dim	256
Actor hidden layers	2
Actor activation function	ReLU
Encoder hidden dim	256
Encoder hidden layers	2
Encoder activation function	ReLU
Decoder hidden dim	256
Decoder hidden layers	2
Decoder activation function	ReLU
α	2.5

Table 16. Hyperparameters of AWAC-L+VAE.

AWAC Hyperparameter	Value
Optimizer	Adam
Critic learning rate	3e-4
Actor learning rate	3e-4
Mini-batch size	256
Discount factor	0.99
Target update rate	5e-3
Policy update frequency	2
Score scaling $1/\alpha$	3.0
Critic hidden dim	256
Critic hidden layers	2
Critic activation function	ReLU
Actor hidden dim	256
Actor hidden layers	2
Actor activation function	ReLU
Encoder hidden dim	256
Encoder hidden layers	2
Encoder activation function	ReLU
Decoder hidden dim	256
Decoder hidden layers	2
Decoder activation function	ReLU

Table 17. Hyperparameters of IQL-L+VAE.

IQL Hyperparameter	Value
Optimizer	Adam
Critic learning rate	3e-4
Actor learning rate	3e-4
Mini-batch size	256
Discount factor	0.99
Target update rate	5e-3
Score scaling $1/\alpha$	3.0
Expectile τ	0.7
Critic hidden dim	256
Critic hidden layers	2
Critic activation function	ReLU
Actor hidden dim	256
Actor hidden layers	2
Actor activation function	ReLU
Encoder hidden dim	256
Encoder hidden layers	2
Encoder activation function	ReLU
Decoder hidden dim	256
Decoder hidden layers	2
Decoder activation function	ReLU

Table 18. Effect of the learning rate of $\mathcal{L}_{\text{M-info}}$. WK = Walker2dVel, HP = HopperVel, HC = HalfcheetahVel. AN = AntVel. Best results are denoted in bold. The average and standard deviation of normalized scores over the 10 test episodes and five seeds are shown.

		D4RL score	
		DiveOff lr=9e-5	DiveOff lr=6e-5
Div.-Exp.	WK	93.3±9.9	**98.5±2.2**
	HP	81.2±33.8	**96.4±3.8**
	HC	96.8±0.3	96.8±0.1
	AN	95.6±0.6	95.4±1.2
Div.-Med.-Exp.	WK	80.5±17.7	**98.0±2.5**
	HP	98.4±4.7	98.9±2.2
	HC	96.1±0.6	96.2±0.1
	AN	83.6±3.7	88.5±1.9
Div.-Med.	WK	67.9±17.4	**88.4±9.3**
	HP	83.1±18.0	**95.1±6.2**
	HC	93.2±0.5	93.2±0.1
	AN	76.7±8.9	**81.1±2.3**
Div.-Med.-Rep.	WK	39.3±11.6	44.2±18.2
	HP	101.0±0.1	101.1±0.1
	HC	89.8±6.4	85.6±11.0
	AN	33.9±2.4	32.9±2.9

		Diversity score	
		DiveOff lr=9e-5	DiveOff lr=6e-5
Div.-Exp.	WK	**0.220±0.230**	0.146±0.169
	HP	**0.879±0.114**	0.441±0.331
	HC	0.774±0.241	0.868±0.208
	AN	**0.148±0.142**	0.090±0.111
Div.-Med.-Exp.	WK	**0.751±0.195**	0.132±0.140
	HP	**0.878±0.189**	0.115±0.130
	HC	**0.961±0.041**	0.742±0.297
	AN	0.323±0.317	**0.612±0.267**
Div.-Med.	WK	**0.364±0.327**	0.022±0.019
	HP	0.788±0.338	0.707±0.327
	HC	**0.813±0.131**	0.460±0.187
	AN	0.278±0.221	**0.645±0.259**
Div.-Med.-Rep.	WK	0.007±0.009	**0.083±0.166**
	HP	**0.015±0.030**	0.000±0.000
	HC	0.003±0.003	0.003±0.005
	AN	**0.244±0.386**	0.062±0.120