Latent-Variable Advantage-Weighted Policy Optimization for Offline Reinforcement Learning

Xi Chen[1] **Ali Ghadirzadeh**[2] **Tianhe Yu**[2] **Yuan Gao**[3] **Jianhao Wang**[1]
Wenzhe Li[1] **Bin Liang**[1] **Chelsea Finn**[2] **Chongjie Zhang**[1]

Abstract

Offline reinforcement learning methods hold the promise of learning policies from pre-collected datasets without the need to query the environment for new samples. This setting is particularly well-suited for continuous control robotic applications for which online data collection based on trial-and-error is costly and potentially unsafe. In practice, offline datasets are often *heterogeneous*, i.e., collected in a variety of scenarios, such as data from several human demonstrators or from policies that act with different purposes. Unfortunately, such datasets often contain action distributions with multiple modes and, in some cases, lack a sufficient number of high-reward trajectories, which render offline policy training inefficient. To address this challenge, we propose to leverage latent-variable generative model to represent high-advantage state-action pairs leading to better adherence to data distributions that contributes to solving the task, while maximizing reward via a policy over the latent variable. As we empirically show on a range of simulated locomotion, navigation, and manipulation tasks, our method referred to as latent-variable advantage-weighted policy optimization (LAPO), improves the average performance of the next best-performing offline reinforcement learning methods by 49% on heterogeneous datasets, and by 8% on datasets with narrow and biased distributions.

1 Introduction

Offline reinforcement learning (RL), also known as batch RL [26], addresses the problem of learning an effective policy from a static fixed-sized dataset without interacting with the environment to collect new data. This formulation is especially important for robotics, as it avoids costly and unsafe trial-and-error and provides an alternative way of leveraging a pre-collected dataset. However, in practical settings, such offline datasets are often heterogeneous and are collected using different policies, leading to a data distribution with multiple modes. These data-collection policies may aim to accomplish tasks that are not necessarily aligned with the target task or may accomplish the same task but provide conflicting solutions. In contrast to the prior works that have focused on the distributional shift problem [24, 9, 25], in this paper, we address the problem of learning from heterogeneous data.

The main challenges in learning from heterogeneous settings are the existence of conflicting actions in the dataset, and the lack of sufficient high-return trajectories as the dataset is constructed by re-labeling state-action pairs from some different tasks. Learning from heterogeneous datasets with conflicting actions is particularly challenging for implicit policy constraint methods that formulate a supervised learning objective function based on the forward Kullback–Leibler (KL)-divergence between the parametric policy being learned and the closed-form optimal policy found through advantage-weighted behavior cloning [38, 40, 43]. Figure 1 illustrates an example of offline heterogeneous settings in which state-conditional distributions learned directly over the action space

[1]Tsinghua University, [2] Stanford University, [3] Shenzhen Institute of Artificial Intelligence and Robotics for Society, CUHK Shenzhen.

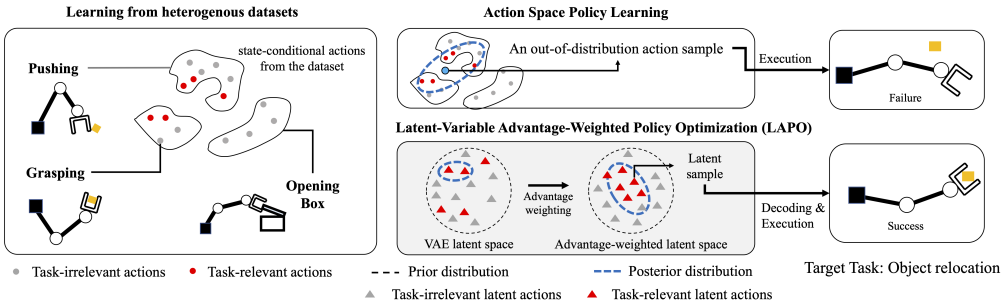

Figure 1: An example scenario of learning from data with heterogeneous action distributions. (left) The dataset includes data from three different tasks, pushing, grasping, and opening boxes. (right) The learned task is to relocate an object to a goal position. (middle up) Learning policies directly in the action space can result in sampling from out-of-distribution actions which subsequently can fail the learning task. (middle bottom) LAPO constructs a state-conditional latent space and learns a latent policy to only select in distribution high-return actions.

using the forward KL-divergence objective may assign high probability to out-of-distribution (OOD) actions which results in sub-optimal policies. Replacing the forward-KL-divergence with a reverse KL-divergence can avoid this issue in theory, as it forces the policy to only learn a single mode of the distribution. But in practice, it requires querying the behavior policy which is unknown, and using an erroneous approximation of the behavior policy can negatively affect the performance ([38]). A similar problem has been reported when explicitly constraining the policy on multi-modal distributions using maximum mean discrepancy (MMD) distances [54]. This problem however may seem to be solved by using more expressive policies such as variational auto-encoders (VAEs) or Gaussian mixture models (GMMs). However, in practice this does not help in heterogeneous settings with a low density of high-return trajectories. In such settings, adhering to the data distribution using policy constraint methods can result in sub-optimal performances.

We illustrate this problem with a toy navigation task in which an agent navigates to a goal position while avoiding an obstacle. In this example, the demonstrated expert actions have two prominent modes at the initial state corresponding to moving to either the right or left side of the obstacle. We assume moving to the right leads to higher returns, but we have a lower density of such actions in the dataset. Figure 2.a illustrates the histogram of actions sampled from a trained policy with AWAC [38], a policy constraint method. The method fails in this setting since the learned action distribution includes OOD actions which result in a collision with the obstacle. Learning a GMM policy with AWAC can avoid the OOD action problem but still assigns a high probability distribution over low-return actions (Figure 2.b). Other offline RL methods, including BCQ [9] and PLAS [54], which learn a VAE policy, also fail to represent the high-return actions in this setting. PLAS can only model the low-return but high-density samples in the dataset (Figure 2.c).

Figure 1 also illustrates a potential solution to the problem of learning from heterogeneous datasets. The intuition is to construct a state-conditioned latent space that represents high-return actions. In this case, we can learn simple state-conditional distributions such as Gaussian distributions over the latent space which, as shown in Figure 2.d, can capture task-relevant and high-reward actions without including out-of-distribution samples. Based on this intuition, we propose to learn a latent-space policy by alternating between training the policy and maximizing the advantage-weighted log-likelihood of data. This biases the RL policy to choose actions supported by the training data while effectively solving the target task.

The main contributions of this work is the introduction of a new method, which we refer to it as latent-variable advantage-weighted policy optimization (LAPO), that can efficiently solve heterogeneous offline RL tasks. LAPO learns an advantage function and a state-conditional latent space in which high-advantage action samples are generated by sampling from a prior distribution over the latent space. Furthermore, following a prior work, [54], we also train a latent policy that obtains state-conditioned latent values which result in higher reward outcomes compared to latent samples directly

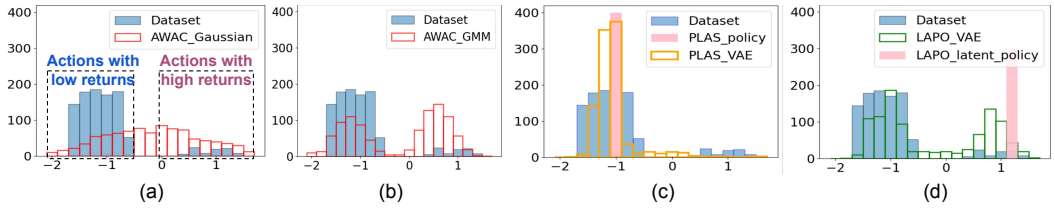

Figure 2: The histogram of actions for the toy navigation task at the initial state in the offline dataset (in blue), and the histogram of actions sampled from (a) the AWAC Gaussian policy (in red), (b) the AWAC GMM policy (in red), (c) the PLAS generative model (in yellow) and the PLAS latent policy (in pink), and (d) the LAPO generative model (in green), and the LAPO latent policy (in pink).

drawn from a prior distribution. We compare LAPO to vanilla behavior cloning, BCQ [9], PLAS [54], AWAC [38], IQL [23], and CQL [25] on a variety of simulated locomotion, navigation, and manipulation tasks, provided heterogeneous offline datasets and also biased datasets with narrow data distribution in standard offline RL benchmarks. LAPO is the only method that yields good performance across all of these tasks and on average improves by 49% over the next best method for heterogeneous datasets, and by 8% on other offline RL tasks with narrow data distributions.

2 Preliminaries

The goal of reinforcement learning is to obtain a policy that maximizes a notion of accumulated reward for a task as a Markov decision process (MDP). In offline RL settings, we assume that we are given a dataset of tuples $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}$, where, $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ denote the state and action at the time step t, $r_t = r(s_t, a_t)$ is the reward given by a bounded reward function, and s_{t+1} is sampled from a fixed transition probability distribution $p(s_{t+1}|s_t, a_t)$. Note that actions can come from a mixture of policies which is referred to as the behavior policy. The policies may try to accomplish different tasks and hence generate trajectories unrelated to the target task.

Given an initial state distribution $\mu_0(s)$ and a discount factor $\gamma \in (0, 1)$, the RL agent optimizes a policy $\pi(a|s)$ to maximize the expected cumulative reward

$$J(\pi) = \mathbb{E}_{s_0 \sim \mu_0, s_{t+1} \sim p(\cdot|s_t, \pi(s_t))} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \right]. \tag{1}$$

The state action value function (Q function), of the policy is defined as $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$, which can be approximated with the Bellman equation:

$$\phi^* = \operatorname*{argmin}_\phi \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}} \left[r_t + \gamma V(s_{t+1}) - Q_\phi(s_t, a_t) \right)^2], \tag{2}$$

where, ϕ denotes the parameters of the Q-function, V denotes the value function $V(s) = \mathbb{E}_{\pi(a|s)}[Q_{\phi'}(s, a)]$ and can be approximated by sampling actions from the policy distribution and averaging the corresponding Q value. ϕ' denotes the parameters of the Q-function at the previous iteration. The advantage of a pair of state and action is then defined as $A(s, a) = Q_\phi(s, a) - V(s)$.

3 Latent-Variable Advantage-Weighted Policy Optimization

In this work, we aim to address the problem of learning from heterogeneous offline datasets collected by policies with different purposes. The challenge arises from the fact that the dataset contains conflicting actions distributed with multiple modes. Besides, the dataset often does not include a sufficient number of high-return trajectories to learn the task properly, as the dataset may be constructed by re-labeling rewards for state-action pairs collected for some other tasks that not necessarily aligned with the target task. We argue that in this setting we not only need a more expressive policy class to represent the multi-modal action distributions, but also need a mechanism that can select which samples to use for the given target task.

We present a novel method, named *latent-space advantage-weighted policy training* (LAPO), which learns a policy by maximizing the Q function over a latent space that is trained to approach the

distribution of actions that leads to higher returns in solving the task. At a high-level, the overall objective of the LAPO can be represented as follows:

$$\underset{\vartheta,\theta}{\operatorname{argmax}}\, \mathbb{E}_{\substack{z\sim\pi_\vartheta(\cdot|s)\\ a\sim p_{\theta'}(\cdot|s,z)\\ s\sim\mathcal{D}}} [Q(s,a)] + \mathbb{E}_{(s,a)\sim\mathcal{D}} [\omega * \log[p_\theta(a|s)]] \,. \tag{3}$$

LAPO has two key components. On the right term, it learns a latent variable state-conditioned generative model $p_\theta(a|s) = \mathbb{E}_{z\sim p(z)}[p_\theta(a|s,z)]$ to regress the actions in the dataset with an importance weight ω. Here $p(z)$ is a specified prior distribution and $p_\theta(a|s,z)$ is a decoder. The importance weight ω indicates how significant an action is to solving the task, which can be computed in a similar way as other weighted-imitation learning methods [38, 40, 43]. On the left term of the objective function, LAPO learns a *latent policy* $\pi_\vartheta(z|s)$ to generate latent values that can be decoded into in-distribution actions that maximize the state-action Q values, via the decoder $p_{\theta'}(a|s,z)$. The parameter θ' denotes the parameter of the decoder in earlier versions, which do not provide gradient information and therefore do not interfere with the optimization of the first component. All components are trained iteratively, and the parameter θ' is updated after one or more iterations.

Overall, the latent variable generative model learns to represent the multi-modal distribution included in the dataset, while actively prioritize regions containing actions that lead to higher returns through the importance weights. Using the decoder, the latent policy will naturally avoid sampling OOD actions, and can learn to optimize Q values without being further constrained by the data distribution, leading to better performance (in Figure. 2.d). Next, we will describe each of these components in detail.

Maximizing the weighted log-likelihood: To learn high-return actions while being conservative to the dataset, LAPO incrementally maximizes the weighted log-likelihood of the data by alternating two steps: (1) estimating the importance weight of each action, and (2) regressing the actions in the dataset with the importance weight of each action. In practice, to characterize the importance of high-return trajectories, we can use the exponential advantages as the importance weight, $\omega = \exp(A(s,a)/\lambda)$. Following [21], we derive a weighted variational lower bound that maximizes the weighted log-likelihood of data:

$$\max_{\theta,\psi} \mathbb{E}_{s,a\sim\mathcal{D}}[\omega\, \mathbb{E}_{q_\psi(z|s,a)}[\log(p_\theta(a|s,z)) \; - \beta\, \mathrm{D}_{\mathrm{KL}}(q_\psi(z|s,a)\,||\,p(z))\,]], \tag{4}$$

where $q_\psi(z|s,a)$ is the variational posterior distribution, $p(z)$ is the prior distribution over the latent variable and typically modelled by a standard normal distribution, and β is a hyper-parameter to balance the two loss terms [17].

Policy evaluation: As explained earlier, the estimated advantage of each state-action pair is used to update the decoder. These advantage estimations are incrementally found through a policy evaluation process in which the overall policy, formed by combining the latent policy and the decoder, is being evaluated. The state-conditional action distribution of the overall policy $\pi_{\vartheta,\theta'}(a|s)$, is found by first sampling a latent value from the latent policy $z \sim \pi_\vartheta(z|s)$, and then feeding the latent value to the decoder to sample the action $a \sim p_{\theta'}(a|s,z)$. In the policy evaluation step, we update the value function by minimizing the squared temporal difference error using Equation 2. To calculate the expected value of a state, we sample actions from the overall policy, i.e.,

$$V(s) = \mathbb{E}_{a\sim p_{\theta'}(\cdot|s,z), z\sim\pi_\vartheta(\cdot|s)}[Q_{\phi'}(s,a)]. \tag{5}$$

Since the latent values z will be used as the input to the decoder, an unbounded z can lead to generating out-of-distribution actions. Therefore, to better ensure sampling in-distribution actions, $\pi_\vartheta(z|s)$ can be modeled as a truncated Gaussian [5]. For deterministic versions of the latent policy, the policy output can be limited to $[-z_{\max}, z_{\max}]$ using a $\tanh$ function.

Policy improvement: LAPO optimizes the latent policy $\pi_\vartheta(z|s)$ in every iteration to directly maximize the return. It is trained based on standard RL approaches such as DDPG [29] or TD3 [8] to maximize the RL objective in Equation 1, by learning latent actions which result in high returns after being converted to the original action space using the decoder.

The latent policy is updated by maximizing the Q-function over states sampled from the dataset, latent variables from the latent policy, and actions from the decoder $p_{\theta'}$:

$$\underset{\pi_\vartheta}{\operatorname{argmax}}\, \mathbb{E}_{a\sim p_{\theta'}(\cdot|s,z), z\sim\pi_\vartheta(\cdot|s), s\sim\mathcal{D}}[Q(a,s)]. \tag{6}$$

Algorithm 1 Latent-Variable Advantage-Weighted Policy Optimization

Input: $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}, i = 1, \ldots, N$.
Initialize: $\phi, \theta, \psi, \vartheta$ randomly, $\theta' = \theta$, and $\omega = 1.0$.
repeat
  Update the decoder p_θ using Equation 4.
  Update the Q-function Q_ϕ using Equations 2 and 5.
  Estimate the advantage weights for every state-action pair using $\omega = \exp(\frac{Q_\phi(s,a) - V(s)}{\lambda})$
  Update the latent policy π_ϑ using Equation 6
  Update θ' according to θ
until M iterations completed

A summary of our method is presented in Algorithm 1. We randomly initialize the parameters of the state-action Q function, the action policy, the amortized variational distribution, and the latent policy. In every training iteration, we first update the decoder and the variational distribution q using Equation 4 given the latest values of the advantage weights. Then, we update the Q function using Equations 2 and 5 provided the updated action and latent policies. Then, for every state-action pair in the dataset, we compute the advantage using the updated Q function, the latent policy and the decoder, and then estimate the weights as the exponential of the scaled advantages, with the scale factor λ. Finally, we update the latent policy using Equation 6.

4 Related Works

Offline Reinforcement Learning: Offline RL methods generally address the problem of distribution shift between the behavior policy and the policy being learned [9, 24], which can cause issues due to out-of-distribution actions sampled from the learned policy and passed into the learned critic. To address this issue, prior methods constrain the learned policy to stay close to the behavior policy via explicit policy regularization [31, 18, 49, 24, 22, 13], via implicit policy constraints [43, 48, 41, 40, 38, 22, 23, 50, 34], by regularizing based on importance sampling [36, 30], by learning of conservative value functions [25, 45], by leveraging auxiliary behavioral cloning losses [7, 37], and through model-based training with conservative penalties [53, 20, 3, 46, 35, 28, 52]. Compared to these prior works, we opt to develop a method that uses an implicit policy constraint. However, prior implicit policy constraint methods have been largely limited to Gaussian policies, leading to OOD action samplings and sub-optimal performance on heterogeneous datasets, We overcome this challenge by introducing a new method that leverages latent variable models. As we will find in Section 5, our method also outperforms state-of-the-art prior methods in other categories, particularly when learning from heterogeneous datasets.

Prior works [1, 19, 51] also studied the problem of learning from heterogeneous dataset in the offline RL with heterogeneous dynamics setting [1] and multi-task offline RL with data sharing scenario [19, 51] respectively. Both settings require knowledge of the agent/task identifier. Here, we study a more general problem setting including datasets with conflicting actions without assuming access to the ground-truth identifier of the source of the heterogeneity.

RL with Generative Models: Generative models have been used by prior works for improving training performance [44, 11, 12], enabling transfer learning [4, 10], implementing hierarchical RL [2, 32, 33, 39], avoiding distributional shift in offline RL settings [9, 54], and learning dynamics models [27, 16, 42, 15]. Our proposed method resembles PLAS [54], which learns a generative model and latent policy; but, unlike this prior work, our method trains an advantage-weighted generative model by alternating between learning the generative model, the advantage function, and the latent policy. This allows LAPO to capture different high-reward solutions using a simple Gaussian distribution in the latent space. As we will find in Section 5, this distinction is crucial, as LAPO significantly outperforms PLAS on heterogeneous datasets.

5 Experiments

Our experiments aim to answer the following questions: (1) How does LAPO compare to other offline RL methods on a set of standard offline RL tasks, including learning from heterogeneous

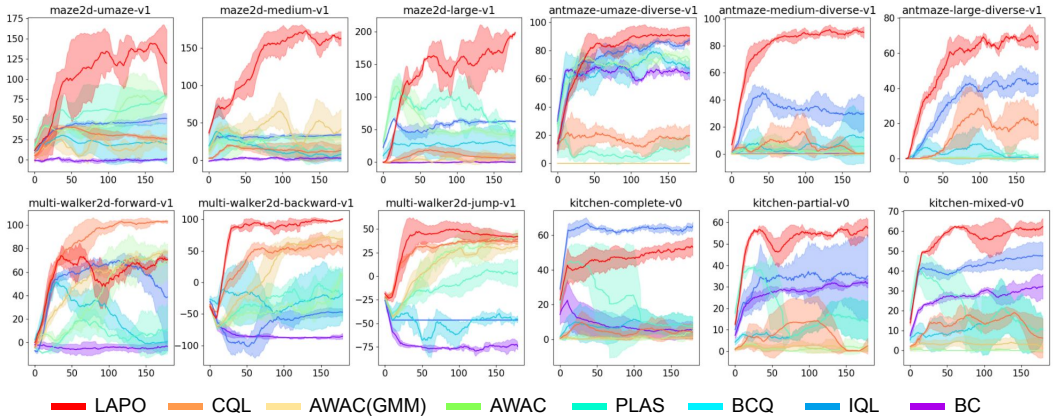

Figure 3: The learning curve of LAPO with 95% confidence interval for heterogeneous tasks. The x-axis is the training epochs, each containing 5,000 gradient updates, and the y-axis is the normalized score. The learning curves for random/narrow and bias datasets are included in the Appendix C

and homogeneous offline datasets? (2) How does LAPO compare to prior methods implemented with GMM policies when learning behaviors offline from heterogeneous datasets? and (3) In which setting does LAPO benefit from the latent policy training, versus only using the tenerative model? Can LAPO learn without constraining the latent values?

We study the first question by evaluating LAPO and comparing it to several prior methods on offline RL benchmarks with heterogeneous datasets. We consider a range of simulated robotic tasks, including navigation, locomotion, and manipulation, each with a corresponding static dataset with a heterogeneous data distribution. We also consider datasets with narrow and biased data distributions, containing near-optimal or random trajectories. To answer the second question, we include a comparison of AWAC [38] with a GMM policy, since AWAC trains using a similar advantage-weighted objective as the actor policy in LAPO. This comparison is only performed on heterogeneous datasets, since this is where we expect to see the most improvement from a multi-modal policy. Finally, the questions in (3), we conduct two ablation studies in which we train LAPO without learning the latent policy, i.e. evaluating only LAPO's action policy, and without limiting the latent values to the range $[-z_{\max}, z_{\max}]$. We compare this version of the method to the original LAPO on four offline RL tasks that differ in the amount of high-performing task-relevant data contained in the offline dataset.

5.1 Comparisons

We compare LAPO to 7 offline RL approaches: (1) vanilla behavioral cloning (BC), (2) the BCQ method [9], which approximates the dataset distribution with a generative model and manually selects the action with the maximum Q value among a set of generated actions, (3) the PLAS method [54], which learns a generative model via maximum likelihood and a latent-space policy to maximize the RL objective in the latent space, (4) the AWAC method [38], which performs advantage-weighted imitation learning with a forward-KL objective for the policy projection step, (5) the IQL method [23], which learns the value function using expectile regression without an explicit policy, and then extracts the policy using advantage weighted regression (AWR), (6) the CQL method [25] which learns a conservative Q-function, and (7) AWAC $_{\text{(GMM)}}$ which trains GMM policies using AWAC.

In our experiments, we train each method three times using three different random seeds each for one million steps, and report the return averaged over 10 test episodes from the trained policies. We used the implementation provided by [47] for the BCQ and AWAC, the original implementation for the PLAS, IQL and CQL, and our implementation for the BC method. We report the implementation details in the Appendix D.

5.2 Tasks and Datasets

We compare LAPO to the introduced prior methods on learning from both heterogeneous and homogeneous offline datasets. We evaluate the methods on three simulated task domains, locomotion,

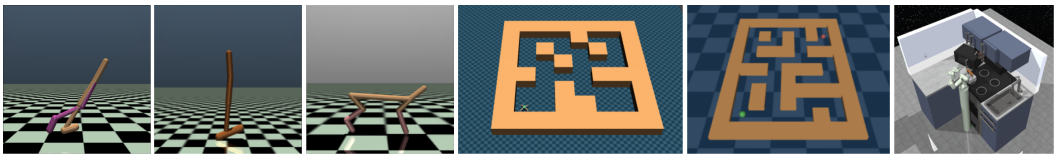

Figure 4: D4RL environments used in our evaluations: (from left to right) Walker 2D, Hopper, Halfcheetah, Antmaze, Maze2d, and Kitchen environments.

navigation, and manipulation domains (Figure 4), with heterogeneous data distributions, and also a locomotion task with random, narrow and biased data distribution. We consider two sources of heterogeneous datasets: (1) data collected by policies accomplishing different tasks, and (2) data collected by different policies that accomplish one task but in different ways. For (1), we use the multi-task heterogeneous datasets introduced by [51], and for (2), we leverage heterogeneous datasets from the standard D4RL benchmark [6]. More details about our tasks and datasets are provided in the Appendix E.

5.3 Experimental Results

Heterogeneous datasets: Table 4 reports the results of our experiments on heterogeneous datasets, and in Figure 3, we present the learning performance of LAPO with 95% confidence interval on Heterogeneous tasks. Among the 12 tasks introduced in the previous section, LAPO (our method) achieves the best performance on nine tasks and the second-best performance on one task. Besides, on average, LAPO improves by 49% over the next best method on the heterogeneous datasets.

The heterogeneous locomotion tasks can be accomplished more easily by the prior methods since still 33% of the data comes from the target task. However, from the prior works, AWAC and CQL are the only methods that can accomplish the tasks. LAPO performs well on all tasks and on average yields the best performance on one out of the three tasks. This shows that LAPO is capable of learning from heterogeneity introduced by multi-task datasets.

The navigation tasks are in general more challenging especially for those with medium and large map sizes. The main challenge is to learn policies for long-horizon planning from datasets that do not contain optimal trajectories, and there are only very few states with rewards. LAPO significantly outperforms all prior works for all of the navigation tasks. Similarly, the manipulation tasks are also challenging, as they require the assembly of sub-trajectories related to completing a given task consisting of multiple sub-tasks. In addition, the agent has access to fewer training samples while having to learn to interact with complicated dynamics. LAPO outperforms previous approaches by a large margin on Kitchen-partial and Kitchen-mixed for which only a few optimal trajectories are provided by the dataset. It also performs competitively on the Kitchen-complete task and yields the second-best performance.

Surprisingly, we observe that GMM policy training does not perform well on heterogeneous offline RL settings. The AWAC method with GMM policies yields similar or even worse performance compared to the original AWAC. These results suggest that latent variable models are better candidates to effectively model the high-reward regions of data especially when given data with multiple modes.

Random/Narrow datasets: Table 5 reports the results of the locomotion tasks with random, narrow and biased datasets. Our method yields the best performance for four out of nine tasks. It also achieves competitive performance on the rest of the tasks, showing that LAPO is applicable to general offline RL settings and is not limited to learning from heterogeneous datasets. However, the most significant gain is when it is applied to offline settings with heterogeneous datasets. Learning curves and results with 95%-confidence interval are reported in Appendix C.

5.4 Ablations

Latent Policy Training: To study the importance of the latent policy training in achieving good performances, we conduct an ablation study on 4 different tasks by eliminating the latent policy training from LAPO. The main motivation is that, as described in Section 3, the latent variable generative model is trained to incrementally learn the action distribution that lead to higher return, hence training the latent policy on top of that may seem redundant. In this case, another option for

Table 1: The normalized performance of all methods on tasks with heterogeneous dataset. 0 represents the performance of a random policy and 100 represents the performance of an expert policy. The scores are averaged over the final 10 evaluations and 3 seeds. LAPO achieves the best performance on 9 tasks and achieves competitive performance on the rest 3 tasks. Results with 95%-confidence interval are reported in Appendix C.2

Task ID	BC	BCQ	PLAS	AWAC	AWAC $_{(GMM)}$	IQL	CQL	LAPO$_{(Ours)}$
Walker2d-mix-forward-v1	-5.65	0.91	22.91	71.89	78.09	28.25	**102.75**	74.17
Walker2d-mix-backward-v1	-84.56	-1.37	-27.58	-7.95	71.33	-46.25	66.64	**99.22**
Walker2d-mix-jump-v1	-72.92	-41.43	0.56	**51.08**	28.01	-46.41	37.28	43.20
Maze2d-umaze-v1	0.99	18.91	80.12	94.53	19.45	51.00	22.86	**118.86**
Maze2d-medium-v1	3.34	12.79	5.19	31.40	46.53	33.26	12.25	**142.75**
Maze2d-large-v1	-1.14	27.17	45.80	43.85	9.04	64.30	7.00	**200.56**
Antmaze-umaze-diverse-v1	60.00	62.00	7.00	72.00	0.00	69.33	16.71	**91.33**
Antmaze-medium-diverse-v1	0.00	11.33	8.67	0.33	0.00	73.00	1.00	**85.67**
Antmaze-large-diverse-v1	0.00	0.67	1.33	0.00	0.00	48.00	11.89	**61.67**
Kitchen-complete-v0	4.50	9.08	38.08	3.83	1.08	**66.67**	4.67	53.17
Kitchen-partial-v0	31.67	17.58	27.00	0.25	0.42	32.33	0.55	**53.67**
Kitchen-mixed-v0	30.00	11.50	29.92	0.00	3.92	49.92	1.86	**62.42**

Table 2: The normalized performance of all methods on locomotion tasks with random, narrow and biased dataset. 0 represents the performance of a random policy and 100 represents the performance of an expert policy. The scores are averaged over the final 10 evaluations and 3 seeds. LAPO achieves the best performance on 4 tasks and achieves competitive performance on the rest of the tasks. Results with 95%-confidence interval are reported in Appendix C.2

Task ID	BC	BCQ	PLAS	AWAC	IQL	CQL	LAPO$_{(Ours)}$
Hopper-random-v2	2.23	7.80	6.68	8.01	7.89	8.33	**23.46**
Walker2d-random-v2	1.11	4.87	**9.17**	0.42	5.41	-0.23	1.28
Halfcheetah-random-v2	2.25	2.25	26.45	15.18	13.11	22.20	**30.55**
Hopper-medium-v2	49.23	56.44	50.96	69.55	65.75	**71.59**	51.63
Walker2d-medium-v2	47.11	73.72	76.47	**84.02**	77.89	82.10	80.75
Halfcheetah-medium-v2	37.84	47.22	44.54	48.13	47.47	**49.76**	45.97
Hopper-expert-v2	76.16	68.86	107.05	**109.32**	**109.36**	102.27	106.76
Walker2d-expert-v2	79.22	110.51	109.56	110.46	109.93	108.76	**112.27**
Halfcheetah-expert-v2	85.63	93.15	93.79	14.01	94.98	87.40	**95.93**

sampling latent variables is to sample from the prior distribution, which changes Equation 5 to:

$$V(s) = \mathbb{E}_{a \sim p_{\theta'}(\cdot|s,z) z \sim p(z)}[Q_{\phi'}(s,a)],$$

where, the latent z is now sampled from a fixed prior.

As shown in Fig. 5, training a latent policy does not significantly contribute to higher performance for tasks such as the Walker2D-medium and Walker2d-mix-backward which contain sufficient task-relevant data. However, latent policy training results in large performance gains when learning from datasets which contain fewer optimal task-related data such as Maze2D and Antmaze. LAPO overcomes this challenge by explicitly optimizing the RL objective through learning the latent policy.

Limiting the latent values: We also study the importance of limiting the latent variable z to be within the range $[-z_{\max}, z_{\max}]$. This is important to understand the limits of using the latent variable decoder as a generative model of in-distribution actions. Figure 6 illustrates the result of LAPO policy training using unbounded latent actions on four tasks. The policy training performance is significantly worse when we do not limit the latent values. This suggests that the the action policy can generate in-distribution actions only when in-distribution latent values (close to samples drawn from the prior distribution $p(z)$) are given as the input.

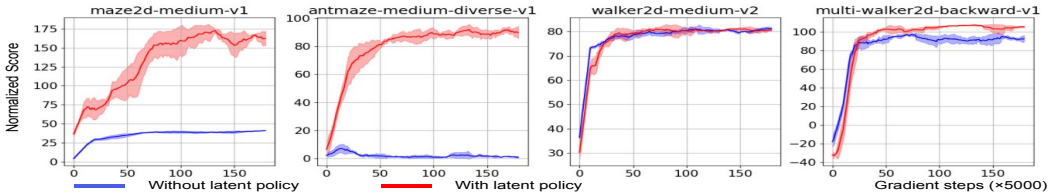

Figure 5: Comparison of LAPO with and without the latent policy on Maze2D-medium, Antmaze-medium-diverse, Walker2D-medium, and Walker2D-mix-backward task. The shaded area represents ±95%-confidence intervals, computed using 10 evaluations and 3 random seeds. For tasks with less optimal task-relevant data like Maze2D-medium, Antmaze-medium-diverse, the learned latent policy results in high-performance gains.

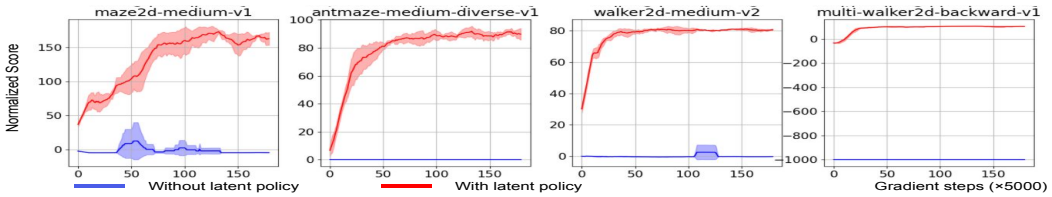

Figure 6: Comparison of LAPO with and without limiting the latent values. The policy performance is significantly worse when the latent values are not limited to the range $[-z_{\max}, z_{\max}]$.

6 Conclusion

In this paper, we study an offline RL setup for learning from heterogeneous datasets where trajectories are collected using policies with different purposes, leading to a multi-modal data distribution, and in some cases, do not contain sufficient high-reward trajectories. Through empirical analysis, we find that in such cases, policies constrained methods may contain out-of-distribution actions and lead to suboptimal performance for continuous control tasks, especially for heterogeneous datasets. To address this challenge, we present the latent-variable advantage-weighted policy optimization (LAPO) algorithm, which learns a latent variable model that generates high-advantage actions when sampling from a prior distribution over the latent space. In addition, we train a latent policy that obtains state-conditioned latent actions which result in higher reward outcomes compared to sampling from the prior distribution. We compare our method to 6 prior methods on a variety of simulated locomotion, navigation, and manipulation tasks provided heterogeneous offline datasets, and also on standard offline RL benchmarks with narrow and biased datasets. We find that our proposed method consistently outperforms prior methods by a large margin on tasks with heterogeneous datasets, while being competitive on other offline RL tasks with narrow data distributions. For our future work, we will extend LAPO to multi-task offline RL settings in which an agent learns multiple RL tasks provided a heterogeneous dataset of diverse behaviors.

References

[1] Anish Agarwal, Abdullah Alomar, Varkey Alumootil, Devavrat Shah, Dennis Shen, Zhi Xu, and Cindy Yang. Persim: Data-efficient offline reinforcement learning with heterogeneous agents via personalized simulators. *arXiv preprint arXiv:2102.06961*, 2021.

[2] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[3] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

[4] Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2725–2731. IEEE, 2020.

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[6] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[7] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

[8] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[9] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[10] Ali Ghadirzadeh, Xi Chen, Petra Poklukar, Chelsea Finn, Mårten Björkman, and Danica Kragic. Bayesian meta-learning for few-shot policy adaptation across robotic platforms. *arXiv preprint arXiv:2103.03697*, 2021.

[11] Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2351–2358. IEEE, 2017.

[12] Ali Ghadirzadeh, Petra Poklukar, Ville Kyrki, Danica Kragic, and Mårten Björkman. Data-efficient visuomotor policy training using reinforcement learning and generative models. *arXiv preprint arXiv:2007.13134*, 2020.

[13] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pages 3682–3691. PMLR, 2021.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[15] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[16] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[18] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[19] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[20] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[22] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

[23] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[24] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[25] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[26] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[27] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

[28] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2020.

[29] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[30] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

[31] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

[32] Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. Carl: Controllable agent with reinforcement learning for quadruped locomotion. *ACM Transactions on Graphics (TOG)*, 39(4):38–1, 2020.

[33] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020.

[34] Xiaoteng Ma, Yiqin Yang, Hao Hu, Qihan Liu, Jun Yang, Chongjie Zhang, Qianchuan Zhao, and Bin Liang. Offline reinforcement learning with value-based episodic memory. *arXiv preprint arXiv:2110.09796*, 2021.

[35] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[36] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *arXiv preprint arXiv:1906.04733*, 2019.

[37] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[38] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. 2020.

[39] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems*, 32:3686–3697, 2019.

[40] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[41] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[42] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[43] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[44] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

[45] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning. *arXiv preprint arXiv:2103.06326*, 2021.

[46] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104:104366, 2021.

[47] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021.

[48] Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In *NeurIPS*, pages 6291–6300, 2018.

[49] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[50] Yiqin Yang, Xiaoteng Ma, Li Chenghao, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[52] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

[53] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[54] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.

Latent-Variable Advantage-Weighted Policy Optimization for Offline Reinforcement Learning (Appendix)

A Connections to other advantage-weighted behavior cloning methods

The right term of the objective function Equation 3 can be derived from the following constrained policy optimization formulation used in [38, 40, 43], by replacing the policy to a latent variable generative model $\hat{\pi}_\theta$ where actions are generated by first sampling a latent value $\hat{z}$ from a prior distribution $p(z)$, and then sampling an action from a latent variable decoder $p_\theta(a|s,\hat{z})$ with the given $\hat{z}$.

$$\begin{aligned} &\underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{a\sim\hat{\pi}_\theta(a|s),s\sim\mathcal{D}}[A(s,a)], \\ &\text{s.t. } \mathbb{E}_{s\sim\mathcal{D}}\Big[D_{KL}\big(\hat{\pi}_\theta(a|s)||\pi_\beta(a|s)\big)\Big] < \epsilon, \end{aligned} \tag{7}$$

where $\hat{\pi}_\theta(a|s) = \mathbb{E}_{\hat{z}\sim p(z)}[p_\theta(a|s,\hat{z})]$, $\pi_\beta(a|s)$ denotes the unknown empirical conditional action distribution of the dataset, and ϵ is a threshold parameter. In case, we use the KL-divergence as the divergence in Equation 7, the optimal $\hat{\pi}^*$ can be expressed as

$$\hat{\pi}^*(a|s) \propto \pi_\beta(a|s)\exp(A(s,a)/\lambda), \tag{8}$$

where, λ is a temperature parameter that depends on the ϵ.

Prior work [38, 43, 40] suggested to incrementally solve Equation 8 by representing the optimal policy $\hat{\pi}^*(a|s)$ as a non-parametric policy, and then project it onto the parametric policy $\hat{\pi}_\theta(a|s)$ by minimizing the KL-divergence:

$$\begin{aligned} &\underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{s\sim\mathcal{D}}\Big[\,\mathrm{D}_{\mathrm{KL}}\left(\hat{\pi}^*(a|s)||\hat{\pi}_\theta(a|s)\right)\Big] \\ =&\underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{s\sim\mathcal{D}}\Big[\ \mathbb{E}_{a\sim\pi^*(a|s)}\left[\log\hat{\pi}_\theta(a|s)\right]\Big] \\ =&\underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\omega * \log\hat{\pi}_\theta(a|s)\Big] \\ =&\underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\omega * \log\mathbb{E}_{\hat{z}\sim p(z)}\left[p_\theta(a|s,\hat{z})\right]\Big] \\ \geq&\underset{\theta,\psi}{\operatorname{argmax}}\, \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\omega * \mathbb{E}_{\hat{z}\sim q_\psi(z|s,a)}\left[\log(p_\theta(a|s,\hat{z})) - \beta\,\mathrm{D}_{\mathrm{KL}}(q_\psi(\hat{z}|s,a)\,||\,p(z))\,\right]\Big], \end{aligned} \tag{9}$$

where the importance weight $\omega = \exp(A(s,a)/\lambda)$, and $q_\psi(z|s,a)$ is an encoder. The generative model $\hat{\pi}_\theta$ is trained to maximize a weighted log-likelihood of the data distribution, and we optimize this term by maximizing its weighted variational lower bound.

A.1 The temperature and the importance weight

We found empirically that the performance of AWAC is sensitive with the choice of the temperature λ, which control how much the policy can be deviated from the behavior policy. A large λ may constrain the policy to be too close to the behavior policy, leading to poor performance if the behavior policy is far from optimal (Figure 2.b). However, if the λ is too small, the computed importance weighted can be very large which cause problem in the training process. Therefore, additional tuning process is needed to find an appropriate λ for different tasks.

In LAPO, we circumvent this problem by learning the latent policy over the latent space to directly maximize the Q value, the closed-form policy $\hat{\pi}^*(a|s)$ does not need to be optimal, and therefore does not require a massive tuning of λ. Inspired by the work [23], in our implementation, we use a fixed importance weight $\omega = 0.9$ for actions with positive advantage and use $\omega = 0.1$ for actions with negative advantage. It simplify the computation and avoid tuning λ for different tasks. As we empirically observed, this technique yields good results for all settings.

B Connections to BCQ and PLAS

Besides constraining the policy by explicit regularization [24] or converting it to a weighted version of BC [38, 40], another branch of offline RL methods such as BCQ [9] and PLAS [54] utilize generative

models to learn batch-constrained policies. LAPO is similar in structure to PLAS as they both learn a generative model and have a policy over the latent space. However, the generative model in BCQ and PLAS is pre-trained to approximate the distribution over the entire data and is fixed when training the latent policy, which can limit the expressiveness of the model on high-return samples when the number of these samples is small in the dataset. In LAPO, the generative model is trained to selectively represent actions that lead to higher returns based on the current value function.

For example, in the toy task shown in Figure 2, 9% of the actions in the dataset are high-return actions. However, when drawing 1,000 z from the prior $p(z)$ of the latent space learned by PLAS, only **4%** of the samples are high-return actions after decoding, while in LAPO, **45%** of the samples represent high-return actions. As we illustrated in Figure 2.c,d, and empirically shown in the experiments, this distinction is crucial, as LAPO significantly outperforms PLAS on nearly all tasks.

C Additional information on experimental results

C.1 Learning curves of random, narrow and bias tasks

We plot the learning curve of random, narrow and bias tasks in Figure 7.

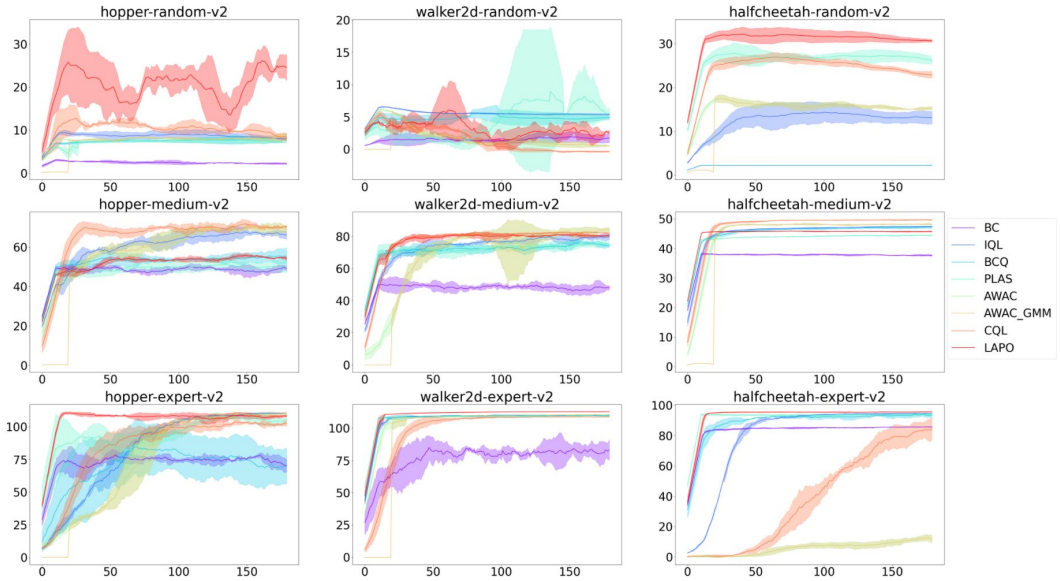

Figure 7: The learning curve of LAPO with 95% confidence interval for random, narrow and bias tasks. The x-axis is the training epochs, each containing 5,000 gradient updates, and the y-axis is the normalized score. For better visualization, the scores are smoothed by a window with length 20.

C.2 Results with confidence interval

Table 4 and 5 report our experimental results with 95%-confidence interval on heterogeneous datasets and random, narrow and bias datasets, respectively.

C.3 Examples trajectories of the navigation tasks

To visualize the performance of the policy learned by LAPO, in figure 8, we plot ten trajectories using the learned policy in the six navigation tasks.

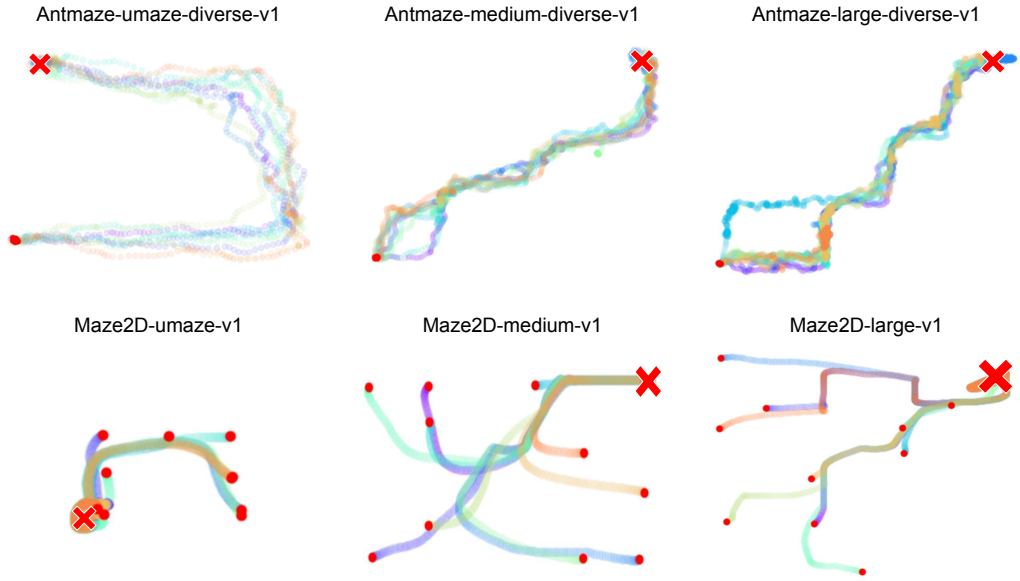

Figure 8: Trajectories learned by LAPO in the navigation tasks. The red dots indicate the starting states of each trajectories, and the red cross indicates the target state. In antmaze environments, the agents are initialized at the same xy position but with different joint values. In maze2D environments, the agents are initialized at different locations.

D Implementation Details

D.1 Implementation of prior methods

The BCQ and AWAC methods are based on the implementations of *d3rlpy*: `https://github.com/takuseno/d3rlpy`, and PLAS, IQL and CQL methods are implemented using the original implementations provided by the authors of the papers: `https://github.com/Wenxuan-Zhou/PLAS`, `https://github.com/ikostrikov/implicit_q_learning`, and `https://github.com/aviralkumar2907/CQL`. For AWAC-GMM method, we extend the policy class implemented in *d3rlpy* AWAC from a single Gaussian policy to 5-head GMM policy.

We used the same hyperparameters as the original paper or the code provided by the author for the prior methods. All models are trained using a NVIDIA P100 GPU.

D.2 Network Hyperparameters

The hyper-parameters used in our experiment are summarized in Table 3. We use $\beta = 0.3$ on Maze2D tasks and $\beta = 1$ for other tasks. We set the dimension of the latent policy to be two times larger than the dimension of the action space. We truncate the output of the latent policy to stay in the range of $[-2.0, 2.0]$. In TD3, we use the clipped double Q-learning ([8]) to compute the state value by combining the outputs of two Q-functions. For Antmaze tasks, we combine them as 0.7*min(Q1, Q2) + 0.3*max(Q1, Q2); for the rest of the tasks, we take the minimum of the two Q values.

D.3 Data pre-processing

We normalized the observations and actions in the dataset for all tasks. For Antmaze, we multiplied Antmaze's reward by 100. For the rest of the tasks, we divided the reward by the maximum reward in the dataset.

Table 3: Hyper-parameters and network architecture for training LAPO

	Hyperparameter	Value
TD3 & LAPO Hyperparameters	Optimizer	Adam
	Critic learning rate	0.0002
	Action policy learning rate	0.0002
	Latent policy learning rate	0.0002
	Mini-batch size	512
	Discount factor	0.99
	Target update rate	0.005
	Policy noise	0.1
	Policy noise clipping	0.3
	VAE β	0.3 for Maze2D, 1.0 for other tasks
	Importance weight ω	0.9 for positive adv, 0.1 for negative adv (A.1)
	Latent action size	$2 \times \|A\|$
	Latent action limit	[-2.0, 2.0]
Architecture	Critic hidden layers	[256, 256, 256]
	Action policy hidden layers	[256, 256, 256]
	Latent policy layers	[256, 256, 256]
	Vae encoder layers	[256, 256, 256]
	Activation function	ReLU

E Tasks and Datasets

Locomotion: We adopt the task settings introduced in [51] to define three locomotion tasks. The datasets are constructed using all of the training data in the replay buffer of three separate policy training sessions each trained for 0.5 million steps using the SAC method [14]. The tasks are to control a Walker2D agent to run forward, backward, and jump. Similar to the original work, we keep a single replay buffer with all of the transitions of all of the three tasks, and form three offline datasets by relabeling the rewards using the reward function provided for each task. We refer to the datasets as *Walker2d-mix-forward*, *Walker2d-mix-backward* and *Walker2d-mix-jump* in the rest of this section.

Navigation: We use the two datasets *Maze2d-sparse* and *Antmaze-diverse* from the D4RL benchmark. The trajectories in the datasets are collected by training goal-reaching policies to navigate to random goals from random initial positions. Provided the pre-collected trajectories, the rewards are relabeled to generate offline data to navigate to different goal positions. Therefore, the task is to learn from data generated by policies that try to accomplish different tasks not aligned with the task at hand. The target task has a sparse binary reward function which gives a reward of *one* only when the agent is close to the goal position, and *zero* otherwise.

Manipulation: For the manipulation domain, we leverage the FrankaKitchen task from the D4RL benchmark. The task is to control a 9-DoF Franka robot to manipulate common household items such as microwave, kettle, and oven, in sequence to reach desired target configuration for several items. There are three datasets collected by human demonstrations: (1) *Kitchen-complete* which consists of successful trajectories that perform tasks in order, (2) *Kitchen-partial* which similar to (1) consists of some successful task-relevant trajectories, but also contains unrelated trajectories that are not necessarily related to reach any target configurations, and (3) *Kitchen-mixed* which consists of partial trajectories that do not solve the entire task, and requires the highest level of generalization from the agent to accomplish the task. The kitchen environment has a sparse reward that is provided whenever an item is at its target configuration.

Locomotion (narrow/random data distribution): For the offline RL task of learning from narrow data distributions, we leverage three Gym-MuJoCo locomotion tasks from the D4RL benchmark with random, narrow and biased data distribution: *Hopper*, *Walker2d* and *Halfcheetah*. Each task contains three datasets collected using a randomly initialized policy ("-random"), a semi-trained policy ("-medium"), and a fully trained policy ("-expert"), respectively. The behavioral policies are trained online using the SAC method.

Task ID	BC	BCQ	PLAS	AWAC	AWAC $_{(GMM)}$	IQL	CQL	LAPO$_{(Ours)}$
Walker2d-mix-forward-v1	-5.65 ± 6.06	0.91 ± 38.43	22.91 ± 7.45	71.89 ± 11.63	78.09 ± 0.93	28.25 ± 6.87	102.75 ± 3.44	74.17 ± 22.64
Walker2d-mix-backward-v1	-84.56 ± 0.57	-1.37 ± 9.48	-27.58 ± 28.29	-7.95 ± 20.63	71.33 ± 28.98	-46.25 ± 14.93	66.64 ± 32.08	99.22 ± 5.19
Walker2d-mix-jump-v1	-72.92 ± 23.35	-41.43 ± 0.02	0.56 ± 16.92	51.08 ± 17.38	28.01 ± 8.23	-46.41 ± 15.03	37.28 ± 3.36	43.2 ± 3.8
Maze2d-umaze-v1	0.99 ± 7.01	18.91 ± 14.89	80.12 ± 29.12	94.53 ± 5.93	19.45 ± 16.91	51.0 ± 10.88	22.86 ± 3.1	118.86 ± 55.66
Maze2d-medium-v1	3.34 ± 8.37	12.79 ± 1.9	5.19 ± 15.1	31.4 ± 16.2	46.53 ± 8.01	33.26 ± 23.9	12.25 ± 12.25	142.75 ± 11.67
Maze2d-large-v1	-1.14 ± 5.42	27.17 ± 5.45	45.8 ± 26.26	43.85 ± 15.87	9.04 ± 14.92	64.3 ± 6.68	7.0 ± 6.26	200.56 ± 18.86
Antmaze-umaze-diverse-v1	60.0 ± 16.0	62.0 ± 5.33	7.0 ± 5.33	72.0 ± 37.34	0.0 ± 5.33	85.67 ± 0.0	16.71 ± 11.14	91.33 ± 10.67
Antmaze-medium-diverse-v1	0.0 ± 0.0	11.33 ± 21.34	8.67 ± 26.67	0.33 ± 5.33	0.0 ± 10.67	9.0 ± 0.0	1.0 ± 5.33	85.67 ± 9.24
Antmaze-large-diverse-v1	0.0 ± 0.0	0.67 ± 9.24	1.33 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	33.67 ± 0.0	11.89 ± 24.45	61.67 ± 21.34
Kitchen-complete-v0	4.5 ± 1.33	9.08 ± 5.81	38.08 ± 6.67	3.83 ± 6.67	1.08 ± 4.0	66.67 ± 5.33	4.67 ± 2.67	53.17 ± 9.62
Kitchen-partial-v0	31.67 ± 4.0	17.58 ± 19.37	27.0 ± 2.31	0.25 ± 2.31	0.42 ± 0.0	32.33 ± 0.0	0.55 ± 6.1	53.67 ± 12.22
Kitchen-mixed-v0	30.0 ± 9.62	11.5 ± 10.59	29.92 ± 6.93	0.0 ± 6.93	3.92 ± 0.0	49.92 ± 1.33	1.86 ± 10.67	62.42 ± 7.06

Table 4: The normalized performance of all methods on tasks with heterogeneous dataset. 0 represents the performance of a random policy and 100 represents the performance of an expert policy. The scores are averaged over the final 10 evaluations and 3 seeds, ± the 95%-confidence interval. LAPO achieves the best performance on 9 tasks and achieves competitive performance on the rest 3 tasks.

Task ID	BC	BCQ	PLAS	AWAC	IQL	CQL	LAPO$_{\text{(Ours)}}$
Hopper-random-v2	2.23 ± 0.18	7.8 ± 0.04	6.68 ± 0.04	8.01 ± 0.04	7.89 ± 1.69	8.33 ± 1.29	23.46 ± 0.62
Walker2d-random-v2	1.11 ± 1.36	4.87 ± 0.13	9.17 ± 0.28	0.42 ± 0.26	5.41 ± 0.39	-0.23 ± 0.12	1.28 ± 2.14
Halfcheetah-random-v2	2.25 ± 0.0	2.25 ± 2.56	26.45 ± 0.0	15.18 ± 0.67	13.11 ± 0.82	22.2 ± 1.36	30.55 ± 0.21
Hopper-medium-v2	49.23 ± 3.74	56.44 ± 5.16	50.96 ± 4.27	69.55 ± 0.45	65.75 ± 4.93	71.59 ± 10.3	51.63 ± 3.27
Walker2d-medium-v2	47.11 ± 2.09	73.72 ± 6.38	76.47 ± 10.61	84.02 ± 1.76	77.89 ± 2.45	82.1 ± 6.26	80.75 ± 0.83
Halfcheetah-medium-v2	37.84 ± 0.17	47.22 ± 0.12	44.54 ± 0.43	48.13 ± 0.39	47.47 ± 0.76	49.76 ± 0.54	45.97 ± 0.32
Hopper-expert-v2	76.16 ± 11.27	68.86 ± 2.19	107.05 ± 11.64	109.32 ± 3.41	109.36 ± 2.33	102.27 ± 7.5	106.76 ± 3.6
Walker2d-expert-v2	79.22 ± 7.22	110.51 ± 0.07	109.56 ± 0.62	110.46 ± 0.34	109.93 ± 0.31	108.76 ± 0.03	112.27 ± 0.08
Halfcheetah-expert-v2	85.63 ± 0.3	93.15 ± 0.17	93.79 ± 7.39	14.01 ± 0.68	94.98 ± 0.82	87.4 ± 20.2	95.93 ± 0.22

Table 5: The normalized performance of all methods on locomotion tasks with random, narrow and biased dataset. 0 represents the performance of a random policy and 100 represents the performance of an expert policy. The scores are averaged over the final 10 evaluations and 3 seeds, ± the 95%-confidence interval.