UNIVR: Unified RL via Implicit Value Regularization

Haoran Xu[1,†], **Liyuan Mao**[2,†], **Hui Jin**[4,†], **Weinan Zhang**[2], **Xianyuan Zhan**[3], **Amy Zhang**[1]
haoran.xu@utexas.edu

[1]**University of Texas at Austin** [2]**Shanghai Jiao Tong University** [3]**Tsinghua University**
[4]**Nanyang Technological University**

[†] Equal contribution. Correspondence to Haoran Xu.

Abstract

The practical implementations of reinforcement learning (RL) often face diverse settings, such as online, offline, and offline-to-online learning. Instead of developing separate algorithms for each setting, we propose UNIVR, a unified model-free RL framework that addresses all these scenarios within a single formulation. UNIVR builds on the Implicit Value Regularization (IVR) framework (Xu et al., 2023) and generalizes its dataset behavior constraint to the constraint w.r.t. a reference policy, yielding a unified value learning objective for general settings. The reference policy is chosen to be the target policy in the online setting and the behavior policy in the offline setting. Using an iteratively refined behavior policy solves the over-conservative issue of directly applying IVR in the online setting, it provides an implicit trust-region style update through the value function while being off-policy. UNIVR also introduces a unified policy extraction objective that estimates in-sample policy gradient using only actions from the reference policy. This not only supports various policy classes, but also theoretically guarantees less value estimation error and larger performance improvement over the reference policy. We evaluate UNIVR on a range of standard RL benchmarks across online, offline, and offline-to-online settings. In online RL, UNIVR achieves higher sample efficiency than both off-policy methods without trust-region updates and on-policy methods with trust-region updates. In offline RL, UNIVR retains the benefits of in-sample learning while outperforming IVR through better policy extraction. In offline-to-online RL, UNIVR beats both constraint-based methods and unconstrained approaches by effectively balancing stability and adaptability.

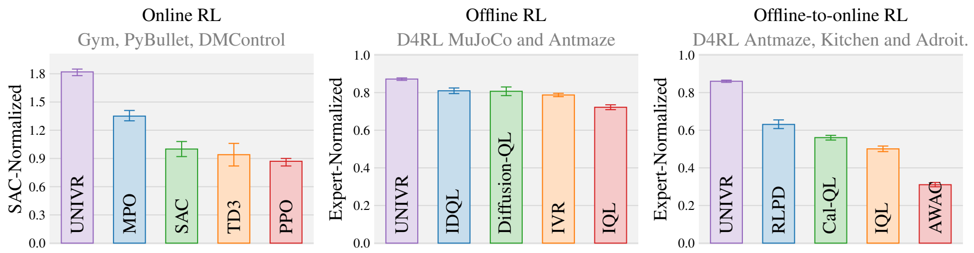

Figure 1: **Summary of results.** Aggregate mean performance across six common RL benchmarks and 23 environments with diverse characteristics. UNIVR achieves competitive performance compared to state-of-the-art baselines across online, offline, and offline-to-online settings.

1 Introduction

Reinforcement learning (RL) has achieved impressive results across a wide range of domains, from playing complex games like Go (Mnih et al., 2013; Silver et al., 2017) to robotic manipulation tasks (Levine et al., 2016). However, the practical deployment of RL demands adaptability to diverse learning scenarios, such as online, offline, and offline-to-online settings. In the classical online RL paradigm, agents interact continuously with the environment, gaining experience and improving policy performance via trial-and-error approaches such as policy-gradient methods (Schulman et al., 2015; 2017) and actor-critic algorithms (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018). However, in safety-critical or high-cost domains, such as healthcare (Gottesman et al., 2018) and industrial control (Zhan et al., 2022; 2025), direct online exploration can be prohibitively risky or expensive, necessitating offline RL approaches that leverage pre-collected datasets without further environment interaction. Recent advances in offline RL have introduced conservative algorithms (Kumar et al., 2020; Fujimoto et al., 2019) and regularization-based methods (Wu et al., 2019; Xu et al., 2023) designed specifically to mitigate extrapolation errors arising from dataset distribution shifts. Moreover, many practical applications adopt an offline-to-online pipeline, initially leveraging offline datasets for safe and effective policy initialization, followed by online fine-tuning to explore more high-quality data to overcome the suboptimality of offline data (Nair et al., 2020; Lee et al., 2021b).

Existing RL algorithms are typically specialized for just one setting. For example, online algorithms (*e.g.*, PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018)) struggle with offline data due to distribution shift. Offline methods (*e.g.*, BCQ (Fujimoto et al., 2019), CQL (Kumar et al., 2020), IVR (Xu et al., 2023)) tend to be overly conservative or generalize poorly when given online access. Offline-to-online methods (*e.g.*, Off2On (Lee et al., 2021b), Cal-QL (Nakamoto et al., 2023)) often require special design choices, making them difficult to generalize to other settings. Designing different algorithms for each setting separately greatly limits the scalability and widespread adoption of RL in real-world applications. Therefore, a critical research question emerges:

Can we design a scalable RL framework that unifies online, offline, and offline-to-online settings?

A Unified Reinforcement Learning Framework. A desirable unified framework should fulfill several crucial properties: **(1) Adaptability**, seamlessly adapt to different settings, including online, offline, and offline-to-online, without changing the learning objective; **(2) Sample efficiency**, exhibit high sample efficiency, minimizing required interactions with the environment; and **(3) Scalability**, scale with growing volumes of data and learn effectively in large and complex environments. We consider the constraint optimization problem with a reference policy μ:

$$\texttt{UNIVR} \qquad \pi^* = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim d^\pi}\big[r(s,a)\big] \;\text{ s.t. }\; \mathbb{E}_{s\sim d^\pi}\big[D_f\left(\pi(\cdot|s)\|\mu(\cdot|s)\right)\big] \le \epsilon, \tag{1}$$

where $D_f(p\|q) = \mathbb{E}_{x\sim q}[f(\frac{p(x)}{q(x)})]$ is the f-divergence (Boyd et al., 2004). By converting the constraint problem into an unconstrained, regularized one, we get a learning objective that imposes a policy-level value regularization by adding the f-divergence regularization term to the reward (Geist et al., 2019). This learning objective generalizes the Implicit Value Regularization (IVR) framework (Xu et al., 2023) from the dataset behavior constraint to the constraint w.r.t. the reference policy μ, yielding **Unified IVR (UNIVR)**, a unified framework for general settings: For *offline RL*, μ is set to be the behavior policy $\pi_{\mathcal{D}}$ of the dataset. For *online RL*, we set μ to be the target policy $\bar{\pi}$ which is periodically or softly updated towards the current policy π. For *offline-to-online RL*, μ is set to be $\pi_{\mathcal{D}}$ in the offline pre-training stage while changing to $\bar{\pi}$ in the online stage. This framework is fully off-policy: offline RL can be considered as a one-step version of online RL (Brandfonbrener et al., 2021), and offline-to-online RL uses offline data to provide a good initialization for online RL.

One issue with Eq. (1) is the reliance on the probability distributions of π and μ, which limits the scalability of using advanced generative models such as diffusion and flow matching models (Song et al., 2020; Lipman et al., 2022) as the policy. However, we show later that by using techniques from duality (Xu et al., 2023; Sikchi et al., 2023), we can get the closed-form solution of the optimal policy π^*, and the value function of π^* can be learned "in-sample" (using samples from the reference policy). By doing so, UNIVR provides an implicit trust-region style update through the value function, and this implicit update results in an off-policy learning scheme that previous trust-region based methods could not achieve (Schulman et al., 2015; 2017). The implicit trust-region update not only increases sample-efficiency, but also solves the over-conservative issue when directly applying IVR in the online setting. Furthermore, it enables smooth dataset constraint relaxation in offline-to-online RL, preventing early performance drops (Li et al., 2023). The in-sample learning scheme in UNIVR also isolates the process of value learning and policy extraction, offering better learning stability. We further introduce a unified policy extraction objective that estimates in-sample policy gradient using only actions from the reference policy. This design scales successfully to more powerful policy classes beyond Gaussian distributions, and theoretically guarantees less value estimation error and larger performance improvement over the reference policy.

We evaluate UNIVR on 6 widely used RL benchmarks and 23 environments across online, offline, and offline-to-online settings, achieving superior or competitive performance against state-of-the-art domain-specific baselines. In online RL, UNIVR achieves higher sample efficiency than both off-policy methods without trust-region updates and on-policy methods with trust-region updates. In offline RL, UNIVR retains the benefits of in-sample learning while outperforming IVR through better policy extraction. In offline-to-online RL, UNIVR beats both constraint-based methods and unconstrained approaches by effectively balancing stability and adaptability.

2 Preliminaries

Reinforcement Learning We consider the RL problem presented as a Markov Decision Process (MDP) (Sutton et al., 1998), which is specified by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, d_0, r, \gamma \rangle$. Here $\mathcal{S}$ and $\mathcal{A}$ are state and action space, $\mathcal{P}(s'|s,a)$ and d_0 denote transition dynamics and initial state distribution, $r(s,a)$ and γ represent reward function and discount factor, respectively. The goal of RL is to find a policy $\pi(a|s)$ which maximizes expected return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot r(s_t, a_t)]$. Offline RL considers the setting where interaction with the environment is prohibited, and one needs to learn the optimal π from a static replay buffer $\mathcal{D} = \{s_i, a_i, r_i, s_i'\}_{i=1}^N$. We also refer to $\mathcal{D}$ as the online replay buffer that is updated by filling in new transitions in the online or offline-to-online setting. Let V^π and Q^π be the state and state-action value function of π, where $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s\right]$ and $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\right]$. The visitation distribution d^π is defined as $d^\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \Pr\left(s_t = s, a_t = a \mid s_0 \sim d_0, \forall t, a_t \sim \pi\left(s_t\right), s_{t+1} \sim \mathcal{P}\left(s_t, a_t\right)\right)$, which measures how likely π is to encounter (s,a) when interacting with the environment, averaging over time via γ-discounting. Let V^*, Q^* and d^* denote the value functions and visitation distribution corresponding to the regularized optimal policy π^*. We denote the empirical visitation distribution of $\mathcal{D}$ as $d^{\mathcal{D}}$ and the empirical behavior policy of $\mathcal{D}$ as $\pi_{\mathcal{D}}$, which represents the conditional distribution $p(a|s)$ observed in the dataset. Let $\mathcal{T}^\pi$ be the Bellman operator with policy π such that $(\mathcal{T}^\pi Q)(s,a) := r(s,a) + \gamma \mathbb{E}_{s'|s,a} \mathbb{E}_{a' \sim \pi}\left[Q(s', a')\right]$ and $(\mathcal{T}^\pi V)(s) := \mathbb{E}_{a \sim \pi}\left[r(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V(s')\right]\right]$.

3 UNIVR: Unified RL via Implicit Value Regularization

We give a detailed introduction of UNIVR in this section. We begin by recalling how the dual form of Eq. (1) provides implicit value regularization, which is a unified, in-sample value learning objective. We then show a naïve extension of IVR to the online setting suffers from suboptimality, and address it with an iteratively improved behavior policy, resulting in an implicit trust-region style update. Finally, we propose a unified policy extraction method that is scalable and versatile to use across different policy classes. We theoretically prove that it enjoys lower value estimation error and larger performance improvement over the reference policy than previous policy extraction methods.

3.1 Towards Unified Value Learning

Implicit Value Regularization. Note that the Lagrangian relaxation of Eq. (1) is equal to

$$\pi^*_{\text{IVR}} = \arg\max_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, a_t) - \alpha \cdot g\Big(\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}\Big)\Big)\right], \tag{2}$$

where $g(x) = f(x)/x$ is differentiable and satisfies $g(1) = 0$. Eq. (2) can be thought of as solving a *behavior-regularized* MDP problem with a modified reward function (Vieillard et al., 2020; Xu et al., 2023). In this behavior-regularized MDP, the Bellman operator is changed to $\mathcal{T}_f^\pi$ such that

$$(\mathcal{T}_f^\pi)Q(s,a) := r(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V(s')\right]$$
$$V(s) = \mathbb{E}_{a\sim\pi}\left[Q(s,a) - \alpha \cdot g\Big(\frac{\pi(a|s)}{\mu(a|s)}\Big)\right].$$

Compared with the original Bellman operator $\mathcal{T}^\pi$, $\mathcal{T}_f^\pi$ is actually applying a value regularization to the Q-function. This transfers the greedy-max policy π^* to a softened max (depending on f) over the reference policy μ, which enables a scalable in-sample learning scheme[1].

Lemma 1 (Results in IVR)**.** *Using duality, the optimal value function Q^* and V^* can be solved by*

$$\min_V \mathbb{E}_{s\sim\mathcal{D},a\sim\mu}\Big[V(s) + \alpha \cdot f_{\text{IVR}}\big(\left[Q(s,a) - V(s)\right]/\alpha\big)\Big] \tag{3}$$

$$\min_Q \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\big(r(s,a) + \gamma V(s') - Q(s,a)\big)^2\right], \tag{4}$$

where $f_{\text{IVR}} = \exp(x)$ if D_f is the KL divergence and $\max(x^2/4+x+1, 0)$ if D_f is the χ^2 divergence. Note that Eq. (3) only uses samples from the reference policy distribution, without needing the knowledge of $\pi(a|s)$ and $\mu(a|s)$. This makes the usage of advanced generative models possible in the online setting, and avoids the need to use additional models to fit a dataset behavior distribution in the offline setting. We can recover IVR by setting μ to be $\pi_\mathcal{D}$, where offline actions serve as $\pi_\mathcal{D}$.

Does IVR work in the online setting? We first examine whether IVR can be naturally extended to the online setting. In online IVR, the offline dataset will be the replay buffer collected so far. Unlike the offline setting, the replay buffer is collected by the learned policy that is periodically updated, which means it may contain a large portion of suboptimal or random data previously collected. Learning from a highly suboptimal dataset is known to be hard in offline RL due to the tendency to anchor the learned policy to the dataset behavior policy caused by regularization (Xu et al., 2025). In online IVR, the optimality of the regularized policy also highly depends on the replay buffer, and we need to break the regularization barrier to get the optimal value function and policy.

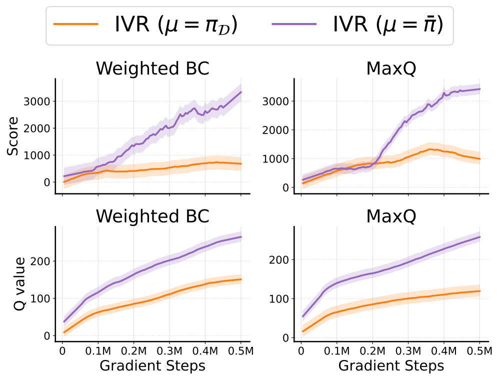

Figure 2: Using an **iteratively refined** behavior policy improves IVR in the online setting.

One way to do so is by filtering the offline dataset iteratively according to the learned optimal policy at each iteration, and gradually optimizing towards the optimal policy using the filtered dataset, as done in Xu et al. (2025). However, this dataset filtering process is costly in the online setting. We thus propose a more lightweight option, where instead of iteratively refining the dataset, we maintain an iteratively refined reference policy $\bar{\pi}$, which is updated towards the current policy π and improves beyond $\pi_\mathcal{D}$. This yields the learning objective in the online or offline-to-online setting.

$$\min_V \mathbb{E}_{s\sim\mathcal{D},a\sim\bar{\pi}}\Big[V(s) + \alpha \cdot f_{\text{IVR}}\big(\left[Q(s,a) - V(s)\right]/\alpha\big)\Big] \text{ with } \bar{\pi} \leftarrow \lambda\pi + (1-\lambda)\bar{\pi}, \tag{5}$$

where λ denotes the soft updating frequency, and if $\lambda = 1$, this amounts to assign $\bar{\pi}$ with the previous policy π_{k-1}, at iteration k. Using $\bar{\pi}$ as the reference policy gives a trust-region style update with an improved μ, solving the over-constrained problem of directly applying IVR in the online setting. Figure 2 illustrates this, UNIVR achieves higher Q values and better sample efficiency than

[1]Here in-sample means samples from the reference policy μ.

IVR across different policy extraction methods. Since the trust-region update is implicitly imposed through the value learning, the algorithm remains off-policy. This improves sample-efficiency over previous trust-region methods, which are all on-policy (Schulman et al., 2015; 2017), and exhibits less value overestimation and more stability compared to existing off-policy methods (Fujimoto et al., 2018; Haarnoja et al., 2018; Abdolmaleki et al., 2018b). Furthermore, it enables smooth dataset constraint relaxation in offline-to-online RL, preventing the finetuned policy from early performance dropping (Nair et al., 2020; Nakamoto et al., 2023; Li et al., 2023).

3.2 Towards Unified Policy Extraction

After introducing the unified value learning objective and demonstrating how UNIVR addresses issues with IVR in the online setting, we now focus on providing a unified policy extraction scheme to effectively extract the best policy from the value functions learned with UNIVR.

Note that in IVR (Eq. (2)), we have a closed-form solution for the ratio of the *optimal regularized policy* π^*_{IVR} to the reference policy μ (Xu et al., 2023), which can be expressed as

$$w^*_{\text{IVR}}(s,a) = \frac{\pi^*_{\text{IVR}}(a|s)}{\mu(a|s)} = \max\Big(0, (f')^{-1}\big(\left(Q^*(s,a) - V^*(s)\right)/\alpha\big)\Big). \tag{6}$$

Previous works try to extract this policy by using either Forward KL or Reverse KL divergence.

(1) $D_{\text{KL}}\left(\pi^*_{\text{IVR}} \| \pi\right) \rightarrow$ **weighted BC**

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D},\boldsymbol{a\sim\mu}}\big[w^*_{\text{IVR}}(s,a)\cdot \log\pi(a|s)\big].$$

Using Forward KL divergence tends to be mode-covering, resulting in a weighted behavior cloning style loss where action is sampled from the reference policy. Although it queries the Q function with only in-sample actions (since Q is trained using actions from μ), it prefers to cover all modes of the target distribution, including those with low probability mass (Park et al., 2024; Xu et al., 2025).

(2) $D_{\text{KL}}\left(\pi \| \pi^*_{\text{IVR}}\right) \rightarrow$ **MaxQ+BC**

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D},\boldsymbol{a\sim\pi}}\big[\log w^*_{\text{IVR}}(s,a) + \log\mu(a|s) - \log\pi(a|s)\big].$$

Using Reverse KL divergence gives a MaxQ+BC style loss (Tomar et al., 2020; Fujimoto & Gu, 2021; Mao et al., 2024b). Reverse KL divergence is mode-seeking, however, actions are sampled from the policy π, which are potentially out-of-distribution and causes over-estimation errors to the Q function. Futhermore, this loss needs an explicit $\mu(a|s)$, which might not be available in the offline setting, and $\mathbb{E}_{a\sim\pi}[\log\pi(a|s)]$ can be hard to estimate when π is parameterized with generative models such as diffusion or flow matching models (Kong et al., 2023).

In summary, weighted BC has lower overestimation error and scales well across different settings and policy classes, while remaining mode-covering. In contrast, MaxQ+BC is mode-seeking but suffers from inaccurate gradient estimation and limited scalability. *Can we design a policy extraction method that shares the best of both worlds?*

Policy extraction via In-sample Policy Gradient. Our key intuition is that we still try to maintain the weighted BC style loss due to its stability and scalability, but we want to incorporate the first-order gradient information from policy gradient into the zero-order gradient induced by the weighted BC loss. To achieve that, we introduce In-sample Policy Gradient (**InPG**) where we ignore the original IVR weights $w^*_{\text{IVR}}(s,a)$ and instead think of those weights as learnable variables. In-sample PG projects the MaxQ gradient into the weighted BC gradient by learning the new **optimal weights** $w^*_{\text{InPG}}(s,a)$ such that

$$\begin{aligned} w^*_{\text{InPG}}(s,a) &= \arg\min_{w\geq 0}\ \mathbb{E}_{s\sim\mathcal{D}}\Big[\big(G_{\text{PG}} - G_{\text{BC}}(w)\big)^2\Big] \\ \text{where}\quad G_{\text{PG}} &= \nabla_\theta\, \mathbb{E}_{a\sim\pi_\theta}\left[Q(s,a)\right] \ \text{ and }\ G_{\text{BC}}(w) = \mathbb{E}_{a\sim\mu}\left[w(s,a)\nabla_\theta\log\pi_\theta(a|s)\right]. \end{aligned} \tag{7}$$

$G_{\text{BC}}(w)$ is the gradient vector of using weighted BC, we can modify the direction of this vector by changing the weight value on different (s, a), and we want to find a non-negative w metric to make the resulting vector stay as close as possible to the MaxQ gradient vector. Eq.(7) is a constrained quadratic programming (QP) problem (Nocedal & Wright, 2006), $w^*_{\text{InPG}}(s, a)$ can be solved using off-the-shelf QP solvers (Stellato et al., 2020) or by gradient descent using a neural network and clipping the output (we use this way in the paper). After getting $w^*_{\text{InPG}}(s, a)$, the policy is extracted by

$$\pi^*_{\text{InPG}} = \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D},a\sim\mu}\big[w^*_{\text{InPG}}(s,a)\cdot\log\pi(a|s)\big]. \tag{8}$$

Using $w^*_{\text{InPG}}(s, a)$ as the BC weight enables better utilization of the Q function since using $w^*_{\text{IVR}}(s, a)$ cannot guarantee the policy is maximizing the Q function. It also makes the new weighted BC loss more mode-seeking, enabling policy extraction under a support constraint since the BC actions are always within the support of the reference policy.

Theoretically, we find InPG could potentially enable a larger **performance improvement** than π^*_{IVR} by increasing $J(\pi)$. In fact, InPG is optimizing towards the constraint learning objective from Eq.(1) while π^*_{IVR} is the solution of Eq. (2), the relaxed Lagrangian of Eq. (1).

Theorem 1. *Define* $w_{\text{min}} = \min_x w(x)$, $w_{\text{max}} = \max_x w(x)$ *and* $Z = \mathbb{E}_{x\sim\mu}[w(x)] \in [w_{\text{min}}, w_{\text{max}}]$, *the* f*-divergence between* π^*_{InPG} *and* μ *is bounded by*

$$f(\frac{w_{\text{min}}}{Z}) \le D_f(\pi^*_{\text{InPG}}\|\mu) \le f(\frac{w_{\text{max}}}{Z}).$$

Theorem 2. *There exists a range of* w_{min} *and* w_{max} *that satisfies* $J(\pi^*_{\text{InPG}}) \ge J(\pi^*_{\text{IVR}})$.

We give the proof of these two theorems in Appendix A. These two theorems reveal that using in-sample policy gradient is actually finding the best policy (*i.e.,* maximizes the Q function) within a fixed constraint (depends on w_{min} and w_{max}) rather than treating that constraint as a penalty, which is more aligned with the original optimization problem Eq. (1). This gives more freedom and allows for better policy extraction. The following theorem further demonstrates the necessity of applying the in-sample projection to the policy gradient, since directly using policy gradient has no improvement guarantee over the reference policy (also known to suffer from overestimation error in offline RL).

Theorem 3. *Define* $\pi^*_{\text{PG}} = \arg\max_\pi \mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[Q(s, a)]$, *there is no guarantee that* $J(\pi^*_{\text{PG}}) \ge J(\mu)$.

We use the `maze2d-umaze` dataset in D4RL (Fu et al., 2020) as an example to demonstrate the effect of using InPG. The Maze2D domain is a navigation task in which a 2D agent must reach a fixed goal location as quickly as possible; the action corresponds to the velocity along the (x, y) axes. We visualize actions sampled from policies learned with IVR ($w^*_{\text{IVR}}(s, a)$) and UNIVR ($w^*_{\text{InPG}}(s, a)$) in Figure 3. As shown, IVR tends to produce actions that are centered around the offline action distribution due to its mode-covering nature. In contrast, UNIVR generates better actions that are near the boundaries of the action space by effectively using the value function. Intuitively, weighted behavior cloning can also exhibit mode-seeking behavior if appropriate weights are assigned to different actions. Notably, in-sample policy gradient does not require sensitive hyperparameter tuning like MaxQ+BC (Tarasov et al., 2024a; Park et al., 2025); it only requires setting a lower and upper weight threshold and we use the same value across all experiments.

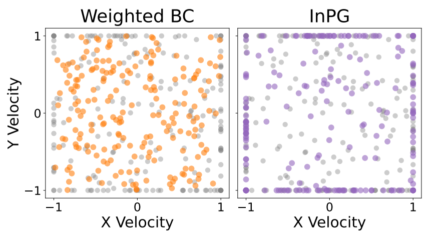

Figure 3: **InPG** discovers better actions from $\mathcal{D}$ (grey points) than **weighted BC**.

Scalable to advanced generative models. In-sample policy gradient can easily scale to the usage of generative models like diffusion and flow matching models. Maximizing the log-likelihood objective in InPG is equivalent to minimizing the matching loss of different generative models (Song et al., 2021). For the use of diffusion models, we have the following matching loss $L_{\text{Diffusion}}(\theta)$ based on Ho et al. (2020).

$$\mathbb{E}_{s\sim\mathcal{D},a\sim\mu,\epsilon\sim\mathcal{N}(0,I),t\sim\text{Unif}([1,T])}\left[\|\epsilon-\epsilon_\theta\left(\sqrt{\bar\alpha_t}a+\sqrt{1-\bar\alpha_t}\epsilon,s,t\right)\|^2\right]. \tag{9}$$

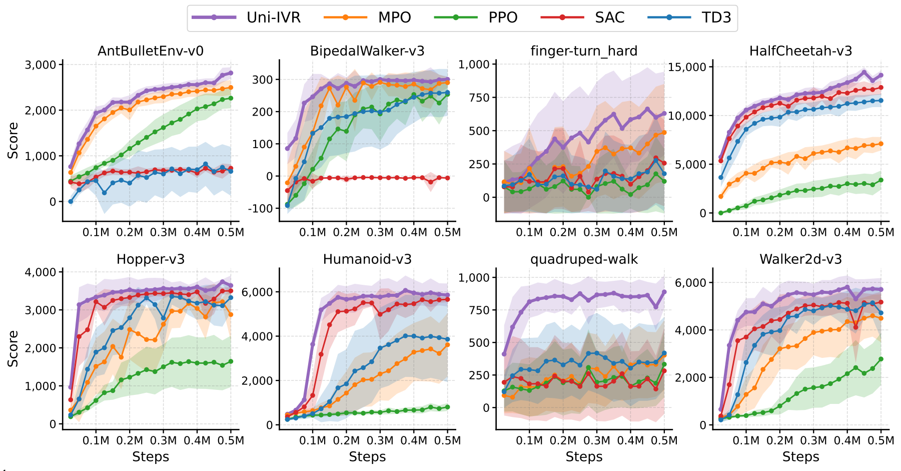

Figure 4: **Results in online RL.** Learning curves are plotted across 5 seeds with a smoothing window of 5000. Shading represents one standard deviation.

In summary, InPG provides a unified policy extraction method based on the weighted BC objective, which uses zero-order gradient while being mode-seeking to maximize Q values. InPG avoids value overestimation errors and scales to implicit behavior cloning through generative modeling, bringing better generalization. The full UNIVR algorithm combines the unified policy extraction module with the unified value learning module. The pseudocode of UNIVR is provided in Algorithm 1.

4 Experiments

Algorithm 1 Unified Implicit Value Regularization

```
Require: $\mathcal{D}$, $\alpha$.
 1: Initialize $Q$, $V$, $\pi$, $\bar{\pi}$
 2: ▷ Unified value and policy learning
 3: procedure UNIVR ($\pi, \mu, \mathcal{D}$)
 4:     Sample transitions $(s, a, r, s') \sim \mathcal{D}$
 5:     Update $V$ by Eq.(3) with $a \sim \mu$, $Q$ and $\mathcal{D}$
 6:     Update $Q$ by Eq.(4) with $V$ and $\mathcal{D}$
 7:     Update $\pi$ by Eq.(7,8) with $a \sim \mu$, $Q$ and $\mathcal{D}$
 8: ▷ Offline training (offline and offline-to-online)
 9: for $t = 1, 2, \cdots, M$ do
10:     UNIVR ($\pi, \pi_{\mathcal{D}}, \mathcal{D}$)
11: ▷ Online finetuning (online and offline-to-online)
12: for $t = 1, 2, \cdots, N$ do
13:     Explore using $\pi$ and append $(s, a, r, s')$ to $\mathcal{D}$
14:     UNIVR ($\pi, \bar{\pi}, \mathcal{D}$)
15:     Update $\bar{\pi}$ by Eq.(5)
```

Online RL. We first test UNIVR in the online setting. We choose 8 environments from OpenAI Gym, DeepMind Control Suite, and PyBullet, representing a large and diverse set of domains based on Box2D, MuJoCo and Bullet physics engines. To demonstrate the effectiveness of our method, we compare UNIVR with several state-of-the-art model-free online RL algorithms including TD3, SAC, PPO and MPO. All baselines use a Gaussian policy, so we also use a Gaussian policy with UNIVR for a fair comparison. The solid curve represents the average return, and the transparent shaded region represents the standard deviation. Each experiment was conducted over 5e5 training steps. According to the learning curves in Figure 2, UNIVR achieves state-of-the-art performance and sample efficiency compared to the other four baseline algorithms, especially in DMControl.

Offline RL. In the offline setting, we evaluate UNIVR on the D4RL benchmark (Fu et al., 2020) and compare it with several state-of-the-art algorithms. For the evaluation tasks, we select MuJoCo locomotion tasks and AntMaze navigation tasks which require both locomotion and navigation. While MuJoCo tasks are popular in offline RL, AntMaze tasks are more challenging due to their stronger need for selecting optimal parts of different trajectories to perform stitching. For baseline algorithms, we selected state-of-the-art methods not only from traditional methods that use a Gaussian policy but also methods that use diffusion models. Gaussian-policy-based baselines include 10%BC, BCQ, TD3+BC, CQL, IQL and IVR. Diffusion-policy-based baselines include Diffusion-QL and IDQL.

We implement UNIVR with both Gaussian policy and Diffusion policy as in Eq.(9). The results in Table 1 show that UNIVR with Gaussian policy already matches or outperforms most baseline

Table 1: **Results in offline RL**. Scores are averaged over the final 10 evaluations across 5 seeds with standard deviation reported, we highlight the best score in integer-level.

Dataset	10%BC	TD3+BC	CQL	IQL	IVR	Diffusion-QL	IDQL	**UNIVR (Gaussian)**	**UNIVR (Diffusion)**
halfcheetah-m	42.5	48.3	44.0 ±0.8	47.4 ±0.2	48.3±0.2	51.1 ±0.5	51.0	49.4±0.2	58.0 ±0.6
hopper-m	56.9	59.3	58.5 ±2.1	66.3 ±5.7	75.5±3.4	90.5 ±4.6	65.4	102.1 ±0.1	102.5 ±0.2
walker2d-m	75.0	83.7	72.5 ±0.8	72.5 ±8.7	84.2±4.6	87.0 ±0.9	82.5	89.3±0.3	92.3 ±0.1
halfcheetah-m-r	40.6	44.6	45.5 ±0.5	44.2 ±1.2	44.8±0.7	47.8 ±0.3	45.9	45.3±0.3	48.4 ±0.9
hopper-m-r	75.9	60.9	95.0 ±6.4	95.2 ±8.6	99.7±3.3	101.3 ±0.6	92.1	101.1 ±2.7	101.3 ±2.1
walker2d-m-r	62.5	81.8	77.2±5.5	76.1 ±7.3	81.2±3.8	95.5 ±1.5	85.1	86.6±1.1	90.8±1.6
halfcheetah-m-e	92.9	90.7	90.7±4.3	86.7±5.3	94.0±0.4	96.8 ±0.3	95.9	94.2±0.6	97.3 ±0.6
hopper-m-e	110.9	98.0	105.4±6.8	101.5 ±7.3	111.8 ±2.2	111.1 ±1.3	108.6	111.0 ±0.6	111.2 ±0.3
walker2d-m-e	109.0	110.1	109.6±0.7	110.6±1.0	110.0±0.8	110.1 ±0.3	112.7	110.8±0.2	114.1±0.5
antmaze-u	62.8	78.6	84.8±2.3	85.5 ±1.9	92.2±1.4	93.4 ±3.4	94.0	94.1±1.6	98.1 ±1.8
antmaze-u-d	50.2	71.4	43.4±5.4	66.7 ±4.0	74.0±2.3	66.2 ±8.6	80.2	80.4±2.3	82.0 ±1.4
antmaze-m-p	5.4	10.6	65.2±4.8	72.2 ±5.3	80.2±3.7	76.6 ±10.8	84.5	86.0±2.6	88.5 ±3.1
antmaze-m-d	9.8	3.0	54.0±11.7	71.0 ±3.2	79.1±4.2	78.6 ±10.3	84.8	82.7±3.4	89.7 ±2.8
antmaze-l-p	0.0	0.2	38.4±12.3	39.6 ±4.5	53.2±4.8	46.4 ±8.3	63.5	59.9±2.9	68.6 ±3.6
antmaze-l-d	6.0	0.0	31.6±9.5	47.5 ±4.4	52.3±5.2	56.6 ±7.6	67.9	60.2±3.8	69.0 ±4.5

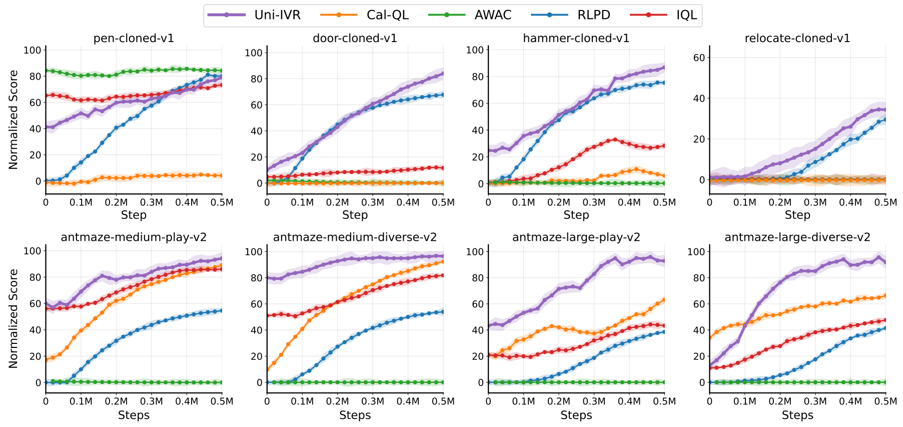

Figure 5: **Results in offline-to-online RL.** Learning Curves is plotted after 250k offline pretraining (5 seeds).

algorithms, especially on MuJoCo medium, medium-replay datasets, and AntMaze datasets. The use of diffusion models in UNIVR further enhances its performance on tasks with multi-modal datasets, demonstrating the scalability of UNIVR. Additionally, the consistently better performance of UNIVR over IVR demonstrates the benefit of policy extraction using in-sample policy gradient.

Offline-to-online RL. In the offline-to-online setting, we conduct extensive experiments on AntMaze and Adroit tasks with D4RL datasets to demonstrate the stable, optimal policy learning and adaptability of UNIVR. We compare UNIVR with the following baselines: *(i)* AWAC: an offline-to-online method that learns the finetuning policy using AWR-style policy loss. *(ii)* IQL: a SOTA offline RL approach based on weighted BC that can directly transfer to online finetuning. *(iii)* Cal-QL: a SOTA offline-to-online approach specially designed based on CQL offline training. *(iv)* RLPD: a method that uses offline data to accelerate online training, it ignores the offline pretraining stage and learns from scratch. Note that we remove the high update-to-data trick in RLPD (*i.e.,* adding layer normalization and ensembles to the Q-function) for a fair comparasion.

Figure 5 shows that existing constraint-based approaches (IQL, AWAC) in most cases only marginally improve the offline pretrained policy, due to the over-conservatism introduced by the constraint w.r.t. the offline dataset. This is especially pronounced when the offline dataset or pretrained policy is highly-suboptimal such as in Adroit and Antmaze tasks. In contrast, UNIVR enjoys stable initial finetuning and superior final performance owing to the iteratively refined policy regularization. Cal-QL is limited to CQL, making it hard to yield reasonable performance when the tasks are too difficult for CQL to obtain good results. While RLPD achieves appealing results in some tasks, the sample efficiency is greatly limited in tasks with diverse or good offline data.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Jonas Degrave, Steven Bohez, Yuval Tassa, Dan Belov, Nicolas Heess, and Martin Riedmiller. Relative entropy regularized policy iteration. *arXiv preprint arXiv: 1812.02256*, 2018a.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. In *Proc. of ICLR*, 2018b.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Proc. of NeurIPS*, 2021.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Proc. of NeurIPS*, 2021.

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

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *ArXiv preprint*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proc. of ICML*, pp. 1582–1591, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proc. of ICML*, pp. 2052–2062, 2019.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *Proc. of ICML*, pp. 2160–2169, 2019.

Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, et al. Evaluating reinforcement learning algorithms in observational health settings. *arXiv preprint arXiv:1805.12298*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. of ICML*, pp. 1856–1865, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Matthew W. Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Nikola Momchev, Danila Sinopalnikov, Piotr Stańczyk, Sabela Ramos, Anton Raichuk, Damien Vincent, Léonard Hussenot, Robert Dadashi, Gabriel Dulac-Arnold, Manu Orsini, Alexis Jacq, Johan Ferret, Nino Vieillard, Seyed Kamyar Seyed Ghasemipour, Sertan Girgin, Olivier Pietquin, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Abe Friesen, Ruba Haroun, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv: 2006.00979*, 2020.

Sham M. Kakade. A natural policy gradient. In *Proc. of NeurIPS*, pp. 1531–1538, 2001.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

Xianghao Kong, Rob Brekelmans, and Greg Ver Steeg. Information-theoretic diffusion. *arXiv preprint arXiv:2302.03792*, 2023.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *Proc. of ICML*, pp. 5774–5783, 2021a.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *ArXiv preprint*, 2021b.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Proc. of NeurIPS*, pp. 11761–11771, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proc. of NeurIPS*, 2020.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *Proc. of ICML*, pp. 6120–6130, 2021a.

Seunghyun Lee, Younggyo Seo, Kimin Lee, P. Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. *Conference on Robot Learning*, 2021b.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Kun Lei, Zhengmao He, Chenhao Lu, Kaizhe Hu, Yang Gao, and Huazhe Xu. Uni-o4: Unifying online and offline deep reinforcement learning with multi-step on-policy optimization. *arXiv preprint arXiv:2311.03351*, 2023.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 2016.

Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, and Ya-Qin Zhang. Proto: Iterative policy regularized offline-to-online reinforcement learning. *arXiv preprint arXiv:2305.15669*, 2023.

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. In *Proc. of ICLR*, 2016.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liyuan Mao, Haoran Xu, Weinan Zhang, and Xianyuan Zhan. Odice: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. *arXiv preprint arXiv:2402.00348*, 2024a.

Liyuan Mao, Haoran Xu, Weinan Zhang, Xianyuan Zhan, and Amy Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. *arXiv preprint arXiv:2407.20109*, 2024b.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *ArXiv preprint*, 2019.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *ArXiv preprint*, 2020.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer Science & Business Media, 2nd edition, 2006. ISBN 9780387303031.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *arXiv preprint arXiv:2406.09329*, 2024.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proc. of ICML*, pp. 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. *arXiv preprint arXiv:2302.08560*, 2023.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 2017.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12: 637–672, 2020. DOI: 10.1007/s12532-020-00179-2.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024a.

Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. Corl: Research-oriented deep offline reinforcement learning library. *Advances in Neural Information Processing Systems*, 36, 2024b.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12163–12174, 2020.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 1992.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv preprint*, 2019.

Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M. Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *Proc. of ICML*, pp. 11319–11328, 2021.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023.

Haoran Xu, Shuozhe Li, Harshit Sikchi, Scott Niekum, and Amy Zhang. An optimal discriminator weighted imitation perspective for reinforcement learning. *arXiv preprint arXiv:2504.13368*, 2025.

Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4680–4688, 2022.

Xianyuan Zhan, Xiangyu Zhu, Peng Cheng, Xiao Hu, Ziteng He, Hanfei Geng, Jichao Leng, Huiwen Zheng, Chenhui Liu, Tianshun Hong, et al. Data center cooling system optimization using offline reinforcement learning. *International Conference on Learning Representations*, 2025.

Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. *arXiv preprint arXiv:2302.00935*, 2023.

Yi Zhao, Rinu Boney, Alexander Ilin, Juho Kannala, and Joni Pajarinen. Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning. *arXiv preprint arXiv:2210.13846*, 2022.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Limitations and Future Work

In this paper, we propose UNIVR, a scalable framework that unifies different reinforcement learning settings, including online RL, offline RL, and offline-to-online RL. UNIVR builds on the Implicit Value Regularization framework but generalizes the offline data constraint to a reference policy constraint, resulting in unified value learning and policy extraction objectives based on in-sample learning. UNIVR is simple, effective, and scalable, and achieves superior performance across diverse RL settings. UNIVR only considers the model-free setting and future directions include incorporating UNIVR with model-based RL methods, and extending it to the LLM+RL setting. We believe that UNIVR represents a concrete step toward building scalable algorithms for unified reinforcement learning.

B Prior work

Online RL. Model-free online RL algorithms can be categorized into on-policy and off-policy methods. While on-policy methods only use experience from the current policy to perform updates (Kakade, 2001; Mnih et al., 2016), off-policy methods, in general, can utilize experience from any arbitrary policy (Watkins & Dayan, 1992; Haarnoja et al., 2018; Fujimoto et al., 2018). To stabilize training, previous methods (Schulman et al., 2015; 2017) add an explicit trust-region policy constraint that limits the derivation from the previous policy. However, this results in on-policy learning, while UNIVR uses an implicit value regularization that is off-policy. Some works also use a weighted BC style loss (Abdolmaleki et al., 2018b;a; Oh et al., 2018; Tomar et al., 2020) in the policy extraction step. However, they don't impose regularization in the value learning. Their value learning and policy learning are coupled, causing the overestimation issue in the offline setting. Note that UNIVR decouples the value and policy learning owing to the implicit value regularization.

Offline RL. To tackle the distributional shift problem, most model-free offline RL methods augment existing off-policy methods with a dataset behavior regularization term. One class of methods imposes behavior regularization explicitly as a divergence penalty (Wu et al., 2019; Kumar et al., 2019; Fujimoto & Gu, 2021) or intervene in the value learning to encourage staying near the behavioral distribution and being pessimistic about unknown state-action pairs (Nachum et al., 2019; Kumar et al., 2020; Kostrikov et al., 2021a; Xu et al., 2022; Wu et al., 2021; An et al., 2021). The other class of methods implicitly impose the behavior regularization through weighted behavior cloning (Kostrikov et al., 2021b; Xu et al., 2023), which filters useful actions to perform behavior cloning based on how advantageous they are. Compared with the first class, UNIVR is inherently more stable due to its imitation-style policy update. Compared with the second class, UNIVR is better at utilizing the value function to avoid suboptimal policy extraction.

Offline-to-online RL. Offline-to-online RL aims to overcome the suboptimality of pure offline learning by collecting more high-quality data with online fine-tuning (Lee et al., 2022; Zhang et al., 2023; Li et al., 2023). To prevent performance drop caused by distribution shift from the offline to online stage, previous methods impose regularization to stay close to the offline data or the offline pretrained policy during online fine-tuning Nair et al. (2020); Kostrikov et al. (2021b); Lee et al. (2022); Nakamoto et al. (2023); Zhao et al. (2022). However, these methods are over-conservative since the offline data could be highly suboptimal. UNIVR solves this issue by enforcing an iteratively refined behavior policy initialized from an offline-pretrained policy, allowing us to gradually release the dataset constraint to continue improving throughout training. There are also some works that try to avoid the performance drop issue by ignoring the offline stage and learning from scratch using offline data (Song et al., 2022; Ball et al., 2023). However, these methods are less sample efficient since they lack the usage of pretrained value functions to perform effective exploration.

Other prior works related to unified RL. There are several other works that try to provide unification for general RL settings. Uni-O4 (Lei et al., 2023) provides unification of offline RL and offline-to-online RL based on on-policy policy gradient, but suffers from sample inefficiency compared to UNIVR, which is off-policy. Policy Agnostic RL (Mark et al., 2024) provides unification for policy extraction methods, whereas UNIVR also achieves a principled unification of value learning for general RL settings. Note that although some DICE-based methods (Lee et al., 2021a; Sikchi et al., 2023; Mao et al., 2024a) have a similar optimization objective with UNIVR, they can't be unified in both the online and offline settings. Specifically, if we apply DICE-based methods to the online setting, sampling from the stationary distribution $d^\mu(s,a)$ of the reference policy is needed. This is intractable in the online setting, making DICE-based methods only work for the offline setting.

C Proof

Theorem 1. *Define $w_{\text{min}} = \min_x w(x)$, $w_{\text{max}} = \max_x w(x)$ and $Z = \mathbb{E}_{x\sim\mu}[w(x)] \in [w_{\text{min}}, w_{\text{max}}]$, the f-divergence between π^*_{InPG} and μ is bounded by*

$$f(\frac{w_{\text{min}}}{Z}) \leq D_f(\pi^*_{\text{InPG}}||\mu) \leq f(\frac{w_{\text{max}}}{Z}).$$

Proof. Remember that π^*_{InPG} has the following expression:

$$\pi^*_{\text{InPG}}(a|s) = \frac{\mu(a|s)w^*(s,a)}{\sum_{a'}\mu(a'|s)w^*(s,a')}.$$

The f-divergence between π and μ is defined as:

$$\begin{aligned} D_f(\pi^*_{\text{InPG}}||\mu) &= \sum_x \mu(x) f\left(\frac{\pi^*_{\text{InPG}}(x)}{\mu(x)}\right) = \sum_x \mu(x) f\left(\frac{\frac{w^*(x)\mu(x)}{Z}}{\mu(x)}\right) \\ &= \sum_x \mu(x) f\left(\frac{w^*(x)}{Z}\right) = \mathbb{E}_{x\sim\mu}\left[f\left(\frac{w^*(x)}{Z}\right)\right]. \end{aligned}$$

Given that $\omega(x)$ is bounded as:

$$w_{\text{min}} \leq w(x) \leq w_{\text{max}}, \quad \forall x,$$

we obtain the following bounds:

$$f\left(\frac{w_{\text{min}}}{Z}\right) \leq \mathbb{E}_{x\sim\mu}\left[f\left(\frac{w(x)}{Z}\right)\right] \leq f\left(\frac{w_{\text{max}}}{Z}\right).$$

Thus, the f-divergence between π^*_{InPG} and μ is bounded. □

Theorem 2. *There exists w_{min} and w_{max} that satisfies $J(\pi^*_{\text{InPG}}) \geq J(\pi^*_{\text{IVR}})$.*

Proof. Remember that IVR considers a new MDP where the reward is augmented with a behavior regularization term ($r(s,a) = r(s,a) - \alpha g(\pi(a|s)/\mu(a|s))$), so the $J(\pi)$ we consider here is equal to the value V^π. And an ideal algorithm wants to make sure that $V^\pi(s) \geq V^\mu(s), s \sim \mathcal{D}$, i.e., safe policy improvement over the behavior policy over the replay buffer.

According to IVR,

$$V(s) = \mathbb{E}_{a\sim\pi}\left[Q(s,a) - \alpha \cdot g\Big(\frac{\pi(a|s)}{\mu(a|s)}\Big)\right].$$

Since π^*_{InPG} is the optimal solution of Eq. (7), i.e., the optimal behavior cloning weight that maximizes the Q function, we have

$$\begin{aligned}
J(\pi^*_{\text{InPG}}) &= \mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{InPG}}}\left[Q(s,a) - \alpha\, g\left(\frac{\pi^*_{\text{InPG}}(a|s)}{\mu(a|s)}\right)\right] \\
&\geq \mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{IVR}}}\left[Q(s,a)\right] - \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{InPG}}}\left[g\left(\frac{\pi^*_{\text{InPG}}(a|s)}{\mu(a|s)}\right)\right] \\
&= \mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{IVR}}(a|s)}\left[Q(s,a) - \alpha\, g\left(\frac{\pi^*_{\text{IVR}}(a|s)}{\mu(a|s)}\right)\right] + \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{IVR}}}\left[g\left(\frac{\pi^*_{\text{IVR}}(a|s)}{\mu(a|s)}\right)\right] \\
&\quad - \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{InPG}}}\left[g\left(\frac{\pi^*_{\text{InPG}}(a|s)}{\mu(a|s)}\right)\right] \\
&= J(\pi^*_{\text{IVR}}) + \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{IVR}}}\left[g\left(\frac{\pi^*_{\text{IVR}}(a|s)}{\mu(a|s)}\right)\right] - \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{InPG}}}\left[g\left(\frac{\pi^*_{\text{InPG}}(a|s)}{\mu(a|s)}\right)\right].
\end{aligned}$$

To make $J(\pi^*_{\text{InPG}}) \geq J(\pi^*_{\text{IVR}})$, one safe condition to satisfy is,

$$\begin{aligned}
\mathbb{E}_{s\sim\mathcal{D}}\left[D_f(\pi^*_{\text{InPG}}(\cdot|s)\|\mu(\cdot|s))\right] &= \mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{InPG}}}\left[g\left(\frac{\pi^*_{\text{InPG}}(a|s)}{\mu(a|s)}\right)\right] \\
&\leq \mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{IVR}}}\left[g\left(\frac{\pi^*_{\text{IVR}}(a|s)}{\mu(a|s)}\right)\right] = \mathbb{E}_{s\sim\mathcal{D}}\left[D_f(\pi^*_{\text{IVR}}(\cdot|s)\|\mu(\cdot|s))\right].
\end{aligned}$$

Note that in Theorem 1 we have $f(w_{\text{min}}/Z) \leq D_f(\pi^*_{\text{InPG}}\|\mu) \leq f(w_{\text{max}}/Z)$. Let $f(w_{\text{max}}/Z) \leq \mathbb{E}_s\left[D_f(\pi^*_{\text{IVR}}(\cdot|s)\|\mu(\cdot|s))\right]$ will satisfy this condition. In conclusion, the range of w_{min} and w_{max} to satisfy $J(\pi^*_{\text{InPG}},\mu) \geq J(\pi^*_{\text{IVR}},\mu)$ is

$$\frac{w_{\text{max}}}{Z} \leq f^{-1}\left(\mathbb{E}_{s\sim\mathcal{D}}\left[D_f(\pi^*_{\text{IVR}}(\cdot|s)\|\mu(\cdot|s))\right]\right).$$

Note that some weaker conditions could also satisfy $J(\pi^*_{\text{InPG}}) \geq J(\pi^*_{\text{IVR}})$. For example, we could choose w^*_{InPG} to be close to w^*_{IVR} such that $D_f(\pi^*_{\text{InPG}}(\cdot|s)\|\mu(\cdot|s)) \approx D_f(\pi^*_{\text{IVR}}(\cdot|s)\|\mu(\cdot|s))$. Also, in the worst case, setting $w^*_{\text{InPG}}(s,a) = w^*_{\text{IVR}}(s,a)$ we have $J(\pi^*_{\text{InPG}}) = J(\pi^*_{\text{IVR}})$. □

Theorem 3. *Define $\pi^*_{\text{PG}} = \arg\max_\pi \mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[Q(s,a)]$, there is no guarantee that $J(\pi^*_{\text{PG}}) \geq J(\mu)$.*

Proof. Since π^*_{PG} is the policy that maximizes $\mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[Q(s,a)]$, we have

$$\begin{aligned}
J(\pi^*_{\text{PG}}) &= \mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{PG}}}\left[Q(s,a) - \alpha\, g\left(\frac{\pi^*_{\text{PG}}(a|s)}{\mu(a|s)}\right)\right] \\
&\geq \mathbb{E}_{s\sim\mathcal{D},a\sim\mu}\left[Q(s,a)\right] - \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{PG}}}\left[g\left(\frac{\pi^*_{\text{PG}}(a|s)}{\mu(a|s)}\right)\right] \\
&= \mathbb{E}_{s\sim\mathcal{D},a\sim\mu}\left[Q(s,a) - \alpha\, g\left(\frac{\mu(a|s)}{\mu(a|s)}\right)\right] - \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{PG}}}\left[g\left(\frac{\pi^*_{\text{PG}}(a|s)}{\mu(a|s)}\right)\right] \\
&= J(\mu) - \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi^*_{\text{PG}}}\left[g\left(\frac{\pi^*_{\text{PG}}(a|s)}{\mu(a|s)}\right)\right] = J(\mu) - \alpha\mathbb{E}_{s\sim\mathcal{D}}\left[D_f\left(\pi^*_{\text{PG}}(\cdot|s)\|\mu(\cdot|s)\right)\right].
\end{aligned}$$

This inequality leverages the fact that $g(1) = 1 * f(1) = 0$. Since $D_f(\pi^*_{\text{PG}}(\cdot|s)\|\mu(\cdot|s)) \geq 0$, there is no guarantee that $J(\pi^*_{\text{PG}}) \geq J(\mu)$. □

D Experimental Details

D.1 Online RL experimental details

Environments, Tasks, and Datasets We evaluate UNIVR on 8 standard continuous control environments from OpenAI Gym, DeepMind Control Suite (DM Control), and PyBullet. These

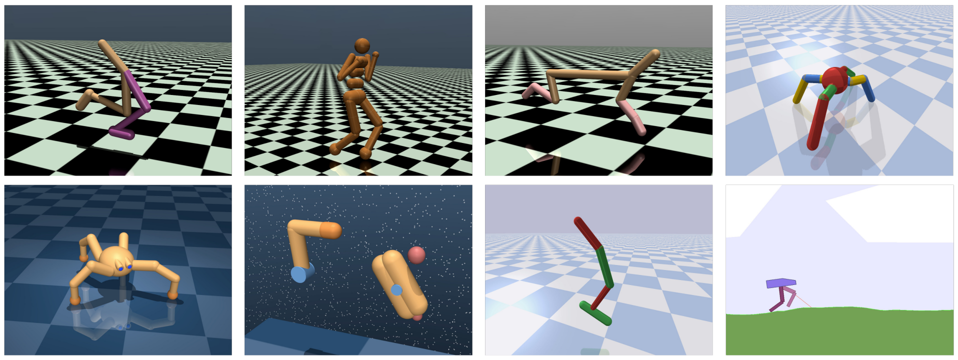

Figure 6: Tasks used in online RL.

tasks vary in agent morphology, dimensionality, and dynamics complexity, ensuring a comprehensive evaluation of our method.

- `AntBulletEnv-v0`: It has a 28-dimensional state space consisting of joint positions, velocities, and body orientation features, and an 8-dimensional action space where each action dimension lies in [-1, 1]. The agent is a quadruped ant-like robot simulated using the PyBullet physics engine. The task requires learning stable and efficient forward locomotion under noisy and contact-rich dynamics.
- `BipedalWalker-v3`: This task has a 24-dimensional state space including hull angle, angular velocity, leg joint positions, and contact sensor readings, and a 4-dimensional continuous action space. The system represents a planar biped that must walk across varying terrain. The challenge lies in foot placement, balance control, and adapting to sparse footholds.
- `finger-turn-hard`: The environment has a 12-dimensional state space and a 6-dimensional action space. It features a robotic finger that must rotate an object to a specific target orientation. The dynamics are sensitive, the rewards are sparse, and the task demands fine motor control and long-horizon reasoning. This environment is from the DeepMind Control Suite.
- `HalfCheetah-v3`: The task includes a 17-dimensional state space representing joint positions and velocities, and a 6-dimensional action space. The system models a planar cheetah-like robot that learns to run forward as fast as possible. It is a widely used benchmark for evaluating the stability and efficiency of learned locomotion policies.
- `Hopper-v3`: The environment has an 11-dimensional state space and a 3-dimensional action space. It consists of a one-legged robot that must learn to hop forward without falling. The task is sensitive to small disturbances and evaluates learning in unstable, underactuated systems.
- `Humanoid-v3`: This is a high-dimensional task with a 376-dimensional state space and a 17-dimensional action space. The agent is a humanoid robot with 21 actuated joints and must learn to walk upright. The complexity of the dynamics and control makes it one of the most challenging continuous control benchmarks.
- `quadruped-walk`: The task includes a 78-dimensional state space representing joint and body kinematics and a 12-dimensional action space. The system models a quadrupedal robot using the DeepMind Control Suite. The goal is to learn stable walking behavior in a realistic 3D setting with proprioceptive sensing.
- `Walker2d-v3`: The environment has a 17-dimensional state space and a 6-dimensional action space. It simulates a planar bipedal robot that must learn to walk forward using two legs with multiple joints. This task provides a balanced challenge, commonly used to evaluate both stability and learning efficiency.

Methods and Hyperparameters In all tasks, we computed the average mean returns over 10 evaluations every $5 \cdot 10^3$ training steps, across 5 different seeds.

For the network, we use 3-layer MLP with 256 hidden units and Adam optimizer (Kingma & Ba, 2015) with a learning rate of 1×10^{-3} for both policy and value functions in all tasks. We also use a target network with soft update weight 5×10^{-3} for Q-function. We clip the output of the weight function by $\max(w(x), 0)$ to ensure a non-negative BC weight. We use 0-1 normalization to the weight in each batch and then clip it to $[w_{\text{min}}, w_{\text{max}}]$ where we set w_{min} to 0.1 and w_{max} to 0.9 through all the datasets.

We implemented UNIVR using PyTorch and ran it on all datasets. We followed the same reporting methods as mentioned earlier. In online experiments, we run baselines using the implementation from ACME (Hoffman et al., 2020)[2]. We use the reported hyperparameters in each paper.

In UNIVR, we have two hyperparameters: regularization weight α and reference policy updating frequency λ. We search α over $[0.1, 0.5, 1.0, 2.0]$ and λ over $[0.01, 0.05, 0.1, 0.2]$ The best value of α and λ for all environments are listed in Table 3.

D.2 Offline and Offline-To-Online RL experimental details

Environments, Tasks, and Datasets In offline and offline-to-online, UNIVR is evaluated on different kinds of datasets from various environments.

For MuJoCo environments, we have the following datasets.

- `halfcheetah/hopper/walker2d-m` (medium): Collected by a policy with moderate performance, typically reaching around one-third of expert returns. These datasets represent structured but suboptimal behavior.
- `halfcheetah/hopper/walker2d-m-r` (medium-replay): Contains the replay buffer of the mediocre SAC policy. It includes a wide range of off-policy transitions, many of which are suboptimal or noisy.
- `halfcheetah/hopper/walker2d-m-e` (medium-expert): A 50-50 mixture of medium and expert trajectories. These datasets are designed to test whether algorithms can leverage near-optimal data when it is partially present.

The AntMaze environments involve a quadruped ant navigating through a 2D maze using sparse goal-based rewards. The agent has a 29-dimensional state space and an 8-dimensional action space, corresponding to joint positions, velocities, and target location encoding. The tasks are particularly challenging due to long-horizon planning and sparse supervision.

- `antmaze-u` (umaze): A small maze where the agent must reach a fixed goal location using sparse rewards. The environment is relatively easy due to short trajectories.
- `antmaze-u-d` (umaze-diverse): Similar to `umaze`, but with broader trajectory diversity collected from random exploration.
- `antmaze-m-p` (medium-play): A medium-sized maze where data is collected via a play policy. The task is harder due to longer horizons and sparse goal rewards.
- `antmaze-m-d` (medium-diverse): Features more diverse and noisy behavior than `medium-play`, increasing exploration coverage but decreasing consistency.
- `antmaze-l-p` (large-play): A large maze with random play data. The agent must navigate long distances, making the task especially difficult under sparse reward signals.
- `antmaze-l-d` (large-diverse): Similar to `large-play`, but with broader and more varied behavior. It is one of the most challenging offline datasets due to the size of the environment and variability in data.

The Adroit environments are high-dimensional dexterous manipulation tasks based on a 24-DoF Shadow Hand. Each environment has a 100-dimensional state space that includes joint angles, velocities, and object pose information, and a 24-dimensional continuous action space controlling

[2] https://github.com/google-deepmind/acme

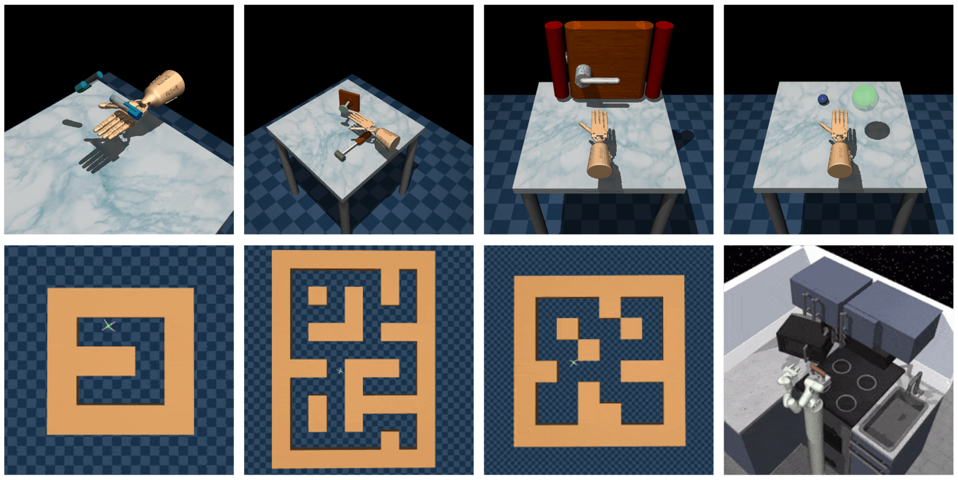

Figure 7: Tasks used in offline and offline-to-online RL.

finger joints. The cloned datasets are generated by behavior cloning from expert demonstrations and represent a moderate level of task success.

- `pen-cloned-v1`: The task requires rotating a pen to a target orientation using a 24-DoF anthropomorphic hand. The dataset is generated by cloning expert demonstrations. It is highly sensitive to precision and coordination.
- `door-cloned-v1`: A dexterous hand must unlock and open a door by grasping and rotating the handle. This task involves contact-rich control and precise force application, with cloned demonstrations as the data source.
- `hammer-cloned-v1`: The agent must use a hammer to drive a nail into a board. This involves both grasping and tool use, making it one of the most complex manipulation tasks in D4RL.
- `relocate-cloned-v1`: The goal is to pick up a ball and move it to a target location using precise grasping and positioning. The cloned dataset reflects human-like strategies but is difficult to exploit due to sparse rewards and high-dimensional dynamics.

Methods and Hyperparameters In Mujoco locomotion tasks, we computed the average mean returns over 10 evaluations every $5 \cdot 10^3$ training steps, across 5 different seeds. For Antmaze and Kitchen tasks, we calculated the average over 50 evaluations every $2 \cdot 10^4$ training steps, also across 5 seeds. We measure binary task success rates (in percentage) for AntMaze and normalized returns for Adroit, following the original evaluation scheme (Fu et al., 2020). Following previous research, we standardized the returns by dividing the difference in returns between the best and worst trajectories in MuJoCo tasks. In AntMaze tasks, we subtracted 1 from the rewards.

For the network, we use 4-layer MLP with 256 hidden units and Adam optimizer (Kingma & Ba, 2015) with a learning rate of 2×10^{-4} for both policy and value functions in all tasks. We also use a target network with soft update weight 5×10^{-3} for Q-function. For UNIVR with Diffusion models as the policy, the score network ϵ_θ we used is based on a U-net architecture, which is fairly common in diffusion-based RL algorithms (Ajay et al., 2022; Mao et al., 2024b). We clip the output of the weight function by $\max(w(x), 0)$ to ensure a non-negative BC weight. We use 0-1 normalization to the weight in each batch and then clip it to $[w_{\text{min}}, w_{\text{max}}]$ where we set w_{min} to 0.1 and w_{max} to 0.9 through all the datasets.

We implemented UNIVR using PyTorch and ran it on all datasets. We followed the same reporting methods as mentioned earlier. In offline experiments, baseline results for other methods were directly sourced from their respective papers. In offline-to-online experiments, we run baselines using

Table 2: Hyperparameters for UNIVR.

Hyperparameter	Value
Learning rate	0.0002 (offline), 0.001 (offline)
Optimizer	Adam
Gradient steps	500000
Minibatch size	256
MLP dimensions	$[256, 256, 256, 256]$
Target network smoothing coefficient	0.005
Discount factor γ	0.99
Diffusion steps (if used)	10
Regularization weight α	Tables 3 and 4
$\bar{\pi}$ updating frequency λ	Tables 3 and 4

the pytorch implementation from CORL (Tarasov et al., 2024b)[3]. Since CORL doesn't have the implementation of RLPD, we re-implement it in the codebase. We use the reported hyperparameters in each paper.

In UNIVR, we have two hyperparameters: regularization weight α and reference policy updating frequency λ. We search α over $[0.1, 0.5, 1.0, 2.0]$ and λ over $[0.01, 0.05, 0.1, 0.2]$ The best value of α and λ for all datasets are listed in Table 4 and Table 5.

Table 3: UNIVR hyperparameters in online RL.

Env	α	λ
AntBulletEnv-v0	1.0	0.1
BipedalWalker-v3	1.0	0.1
finger-turn-hard	0.5	0.1
quadruped-walk	0.5	0.1
HalfCheetah-v3	1.0	0.2
Hopper-v3	1.0	0.2
Humanoid-v3	1.0	0.2
Walker2d-v3	1.0	0.2

[3]https://github.com/corl-team/CORL

Table 4: UNIVR hyperparameters in offline RL.

Env	α
halfcheetah-medium-v2	1.0
hopper-medium-v2	1.0
walker2d-medium-v2	1.0
halfcheetah-medium-replay-v2	1.0
hopper-medium-replay-v2	1.0
walker2d-medium-replay-v2	1.0
halfcheetah-medium-expert-v2	1.0
hopper-medium-expert-v2	1.0
walker2d-medium-expert-v2	1.0
antmaze-umaze-v2	0.1
antmaze-umaze-diverse-v2	2.0
antmaze-medium-play-v2	0.1
antmaze-medium-diverse-v2	0.1
antmaze-large-play-v2	0.1
antmaze-large-diverse-v2	0.1

Table 5: UNIVR hyperparameters in offline-to-online RL.

Env	α	λ
pen-cloned-v1	1.0	0.01
door-cloned-v1	2.0	0.01
hammer-cloned-v1	2.0	0.01
relocate-cloned-v1	2.0	0.01
antmaze-medium-play-v2	0.5	0.01
antmaze-medium-diverse-v2	0.5	0.01
antmaze-large-play-v2	0.5	0.01
antmaze-large-diverse-v2	0.5	0.01