

Towards Robust Zero-Shot Reinforcement Learning

Kexin Zheng^{1*†}, Lauriane Teyssier^{2*}, Yinan Zheng², Yu Luo³, Xianyuan Zhan^{2,4†}

¹ The Chinese University of Hong Kong ² Tsinghua University

³ Huawei Noah’s Ark Lab ⁴ Shanghai Artificial Intelligence Laboratory

1155173723@link.cuhk.edu.hk, zhanxianyuan@air.tsinghua.edu.cn

Abstract

The recent development of zero-shot reinforcement learning (RL) has opened a new avenue for learning pre-trained generalist policies that can adapt to arbitrary new tasks in a zero-shot manner. While the popular Forward-Backward representations (FB) and related methods have shown promise in zero-shot RL, we empirically found that their modeling lacks expressivity and that extrapolation errors caused by out-of-distribution (OOD) actions during offline learning sometimes lead to biased representations, ultimately resulting in suboptimal performance. To address these issues, we propose *Behavior-REgularizEd Zero-shot RL with Expressivity enhancement* (BREEZE), an upgraded FB-based framework that simultaneously enhances learning stability, policy extraction capability, and representation learning quality. BREEZE introduces behavioral regularization in zero-shot RL policy learning, transforming policy optimization into a stable in-sample learning paradigm. Additionally, BREEZE extracts the policy using a task-conditioned diffusion model, enabling the generation of high-quality and multimodal action distributions in zero-shot RL settings. Moreover, BREEZE employs expressive attention-based architectures for representation modeling to capture the complex relationships between environmental dynamics. Extensive experiments on ExORL and D4RL Kitchen demonstrate that BREEZE achieves the best or near-the-best performance while exhibiting superior robustness compared to prior offline zero-shot RL methods. The official implementation is available at: <https://github.com/Whiterrrrr/BREEZE>.

1 Introduction

Reinforcement learning (RL) has become a cornerstone of artificial intelligence, enabling transformative advances in robotics [53], autonomous systems [29], industrial control [71], and large language models (LLM) [60]. However, its real-world adoption faces two persistent challenges: the reliance on human-provided reward functions and its task-specific learning paradigm, which limits adaptability to novel or multiple tasks. These challenges sparked growing interest in zero-shot RL [57, 58, 44, 26, 4, 23, 56, 13], which enables learning a versatile agent through pretraining on reward-free transitions and then zero-shot adaptation to arbitrary reward functions during inference. This opens up new possibilities for developing general-purpose RL agents capable of generalizing across diverse tasks in open-world scenarios.

Existing zero-shot RL approaches mainly fall into two categories: task/skill-conditioned RL [13, 23] and dynamic representation-based methods [4, 3, 58, 26, 56, 44]. The first category encodes demonstrations or reward functions into embeddings as conditioning signals for policy learning. This preserves the task generalization capability of policies, but often results in heavy, manually

*Equal contribution.

†Corresponding author.

[‡]Work done during internships at Institute for AI Industry Research (AIR), Tsinghua University.

designed pretraining without optimality guarantees. The second category of methods adopts a more principled approach by decomposing the problem into dynamic representations that can be recomposed for novel tasks without retraining. Among dynamic representation-based methods, Forward-Backward representations (FB) [57, 58] have recently attracted notable attention, factorizing occupancy measures into two components: a forward representation that captures policy dynamics and a backward representation that encodes global state information. Through offline, unsupervised pretraining, the FB framework learns linearized representations that approximate value functions for arbitrary tasks, holding great promise for zero-shot generalization.

However, despite the elegant theoretical framework provided by the FB representations, our empirical studies reveal that the successor measures learned through existing FB-based methods are often inconsistent and biased, thereby compromising the stability and overall performance (see Section 3 for details). In this paper, we identify two causes for the shortcomings of existing FB-based methods. First, learning complex approximators and multimodal behaviors requires highly expressive models, which current FB methods lack in both their representations and policy. Second, the offline, unsupervised pretraining stage suffers from extrapolation error due to out-of-distribution (OOD) actions, which is a similar problem to offline RL [17, 32, 33], but exhibits more complex behavior. As our results show, naively integrating value constraints can be ineffective [26], indicating a need for a more delicate OOD regularization mechanism.

Based on the above observations, we propose ***B*ehavior-**R**egulariz**E**d **Z**ero-shot **R**L with **E**xpressivity **e**nhan**c**ement** (BREEZE), a novel FB-based framework that simultaneously enhances offline learning stability and zero-shot generalization capability. First, we introduce a behavior-regularized reformulation of FB, which mitigates extrapolation errors while preserving the fidelity of representations. Second, we extract the policy using a task-conditioned diffusion model, enabling high-quality multimodal action distributions in zero-shot RL settings. Finally, we employ expressive representation networks based on the attention architecture to capture complicated dynamics. We conducted extensive experiments on the ExORL benchmark [70] and the D4RL Kitchen dataset [14], under both full datasets and small-sample data regimes. The results demonstrate that BREEZE achieves the best or near-the-best performance while exhibiting superior robustness.

2 Preliminary

Reward-free Markov decision process (MDP). Zero-shot RL is typically formulated as a reward-free Markov Decision Process (MDP) [51], defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma)$, consisting of a state space \mathcal{S} , an action space \mathcal{A} , a transition kernel $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$, and a discount factor $\gamma \in (0, 1)$. Given an initial state-action pair $(s_0, a_0) \in \mathcal{S} \times \mathcal{A}$ and a policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, we denote $\Pr(\cdot | s_0, a_0, \pi)$ a probability and $\mathbb{E}[\cdot | s_0, a_0, \pi]$ an expectation under the state-action trajectories $(s_0, a_0, \dots, s_t, a_t)_{t \geq 0}$ generated by sampling $s_t \sim \mathcal{P}(\cdot | s_{t-1}, a_{t-1})$ and $a_t \sim \pi(\cdot | s_t)$. The state transition under π is given by $\mathcal{P}_\pi(ds' | s) = \int \mathcal{P}(ds' | s, a)\pi(da | s)$.

Approximate dynamic programming-based RL [47] uses an action-value Q -function, or optionally a state-value V -function. The Q -function and V -function for π starting at s_0, a_0 under a given reward function $r : \mathcal{S} \rightarrow \mathbb{R}$ are respectively defined as $Q_r^\pi(s_0, a_0) := \sum_{t \geq 0} \gamma^t \mathbb{E}[r(s_{t+1}) | s_0, a_0, \pi]$ and $V_r^\pi(s_0) := \sum_{t \geq 0} \gamma^t \mathbb{E}[r(s_{t+1}) | s_0, \pi]$. The goal of the zero-shot RL problem is to train a task-agnostic policy given an offline dataset $\mathcal{D} = \{s_i, a_i, s_{i+1}\}_{i=1}^N$ generated by an unknown behavior policy $\mu(\cdot | s)$, that can later generalize to any downstream task z defined by a reward function $r_{\text{eval}} : \mathcal{S} \rightarrow \mathbb{R}$, i.e. find $\max_\pi \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{\text{eval}}(s_{t+1}) | s_0, a_0, \pi]$, relying only on a small set of reward-labeled samples with no further finetuning.

Forward-Backward representations (FB). FB introduces a rank- d approximation of the successor measure, which is defined as the expected discounted occurrences of future states $s_+ \in \mathcal{S}_+$ after starting from (s_0, a_0) under a policy π :

$$M^\pi(s_0, a_0, \mathcal{S}_+) := \sum_{t \geq 0} \gamma^t \Pr(s_{t+1} \in \mathcal{S}_+ | s_0, a_0, \pi) \quad \forall \mathcal{S}_+ \subset \mathcal{S}. \quad (1)$$

Following this definition, Q -function under policy π can be expressed as:

$$Q_r^\pi(s, a) = \int_{s_+ \in \mathcal{S}} M^\pi(s, a, ds_+) r(s_+). \quad (2)$$

Given a representation space \mathbb{R}^d , a state distribution ρ , a task vector $z \in \mathbb{R}^d$ and a policy π_z parameterized by z , FB expresses M^{π_z} as the product of a forward representation $F : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ and a backward representation $B : \mathcal{S} \rightarrow \mathbb{R}^d$, resulting in an optimal policy π_z :

$$M^{\pi_z}(s_0, a_0, ds_+) \approx F(s_0, a_0, z)^\top B(s_+) \rho(ds_+), \quad \pi_z(s) := \arg \max_a F(s, a, z)^\top z, \quad (3)$$

where z is defined as $z := \mathbb{E}_{s \sim \rho}[r(s)B(s)]$ from a few samples with a known reward function r . The corresponding Q -function can be obtained immediately as $Q_z(s, a) = F(s, a, z)^\top z$.

Since the successor measure M^{π_z} satisfies a Bellman-like equation $M^{\pi_z} = \mathcal{P} + \gamma \mathcal{P}_{\pi_z} M^{\pi_z}$, FB derives the temporal difference (TD) objective [50] on M^{π_z} as:

$$\begin{aligned} \mathcal{L}_{\text{FB}} &= \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim D} \left[(F(s_t, a_t, z)^\top B(s_+) - \gamma \bar{F}(s_{t+1}, \pi_z(s_{t+1}), z)^\top \bar{B}(s_+))^2 \right] \\ &\quad - 2 \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim D} [F(s_t, a_t, z)^\top B(s_{t+1})], \end{aligned} \quad (4)$$

and incorporate the following objective on F to enforce the Bellman property on the Q -function:

$$\mathcal{L}_F = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim D} \left[(F(s_t, a_t, z)^\top z - B(s_{t+1})^\top \mathbb{E}_D[BB^\top]^{-1}z - \gamma \bar{F}(s_{t+1}, \pi_z(s_{t+1}), z)^\top z)^2 \right] \quad (5)$$

where $B(s_{t+1})^\top \mathbb{E}_D[BB^\top]^{-1}z$ is the implicit reward estimation and D denote the data distribution.

Recent extensions of FB incorporate additional regularization to improve learning performance. For example, Jeen et al. [26] noted that the use of $\pi_z(s_{t+1})$ in Eq. (4) during offline learning could introduce OOD extrapolation errors. To resolve this problem, they introduce additional CQL-style [33] regularizers on M^{π_z} (MCFB) or Q_z (VCFB) in \mathcal{L}_{FB} for OOD regularization (i.e., decrease values for OOD actions and increase those for data samples).

3 Pitfalls of Existing FB-Based Methods

FB offers an elegant theoretical foundation for zero-shot RL; however, our empirical findings reveal that FB-based methods often produce inconsistent and biased representations in practice. To illustrate this, we evaluate the M^{π_z} - and Q_z -value distributions derived from the learned F and B representations of the vanilla FB [58], the more recent MCFB [26], and our method on ExORL benchmark [70] with RND [5] datasets. For each method, we use its B representation to derive the task vector z under default settings with a fixed number of transitions. The M^{π_z} values are computed from episode-start state-action pairs $(s_0, a_0) \sim \rho_0$ and randomly sampled batch of transitions $s_+ \sim D$, while Q -values are evaluated using policy-induced actions $a \sim \pi_z(s)$. Results across additional environments and with method VCFB [26] are provided in Appendix D.3.

Figure 1 illustrates the value distributions from two tasks in the Walker domain. Although the successor measure M^{π_z} is mathematically a positive quantity representing future state occupancy, the representations learned by existing FB-based methods fail to accurately capture this property. These distributions exhibit two notable discrepancies: a scale mismatch characterized by enormous absolute values, and the presence of a considerable number of invalid negative values. Such inaccuracies in the F and B representations subsequently affect downstream Q -value estimates,

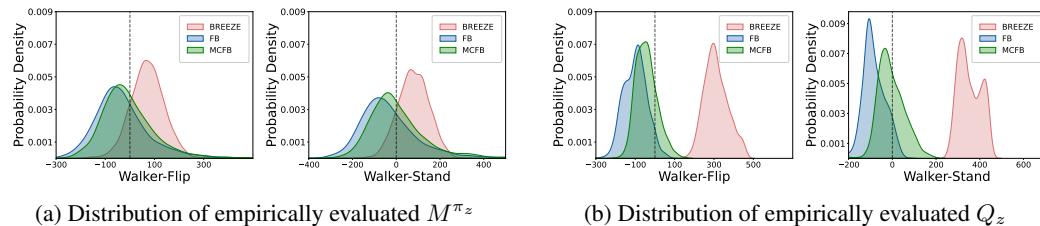


Figure 1: Visualization of the empirical M^{π_z} and Q_z distributions during evaluation stage. We conduct experiments on FB-based methods on two Walker tasks using ExORL [70] RND [5] dataset. We use the learned F, B representations to recover the task vector z and compute $M^{\pi_z} = F^\top B$ and $Q_z = F^\top z$. Both the *vanilla* FB [58] and MCFB [26] result in a proportion of error scaling values.

creating a consistent prediction bias across the system, which is further worsened by the suboptimal action rollout. The CQL-style regularizer in MCFB partially mitigates the issue, shifting the distributions of M^{π_z} and Q_z toward more plausible ranges. However, this correction is incomplete, and significant estimation bias persists, indicating the need for a more effective regularization strategy. In comparison, our method generates distributions for M^{π_z} - and Q_z -value that more closely match theoretical expectations, and can also result in a high-quality policy. The challenges in estimating M^{π_z} - and Q_z -value can be primarily attributed to the limited supervisory signal in the vanilla FB learning objective (Eq. (4)), which fails to constrain the scale of the learned representations adequately. Additionally, modeling successor measures and zero-shot policies across all possible task vectors z introduces substantial complexity, necessitating models with high expressive capacity. While the marginal gains from MCFB’s regularizer suggest its potential utility, a more effective formulation appears necessary. These observations are further corroborated by the results from our method, which integrates a refined OOD regularization strategy and employs a more expressive model architecture. As a result, the distributions of M^{π_z} and Q_z values are primarily confined to a reasonable range. Importantly, as illustrated in Figure 2, these improvements are not attributable to increased model capacity alone: simply scaling up the original MLP-based FB networks does not yield measurable gains. In contrast, the architectural modifications designed in this work lead to a marked improvement in performance, underscoring the influence of model expressivity on the effectiveness of zero-shot learning.

4 Behavior-Regularized Zero-Shot RL with Expressivity Enhancement

Motivated by these findings, we propose BREEZE, a framework designed to mitigate the OOD and expressivity issues in FB-based methods. We begin by imposing a behavioral regularizer on the offline zero-shot RL problem, followed by the transformation of the learning process into an in-sample weighted regression paradigm, which can naturally integrate powerful generative models to enhance policy capability. Recognizing the importance of expressivity for learning arbitrary tasks, we describe our practical architecture for stronger representation modeling.

4.1 Behavior-regularized Optimization

The main OOD extrapolation issue in the vanilla FB learning losses \mathcal{L}_{FB} and \mathcal{L}_F in Eq. (4) and Eq. (5) lies in the policy-generated actions $\hat{a}_{t+1} \sim \pi_z(s_{t+1})$ in $F(s_{t+1}, \pi_z(s_{t+1}), z)$ during offline learning. Since π_z is trained to maximize $F(s, a, z)^\top z$, it may produce actions that appear optimal under the learned representation but are actually OOD and overestimated, leading to extrapolation errors. Our solution to this is to introduce two designs: 1) stabilize the $F(s_{t+1}, \pi_z(s_{t+1}), z)$ value estimate in the \mathcal{L}_{FB} ; 2) ensure that π_z is better regularized by the dataset samples during policy optimization.

Behavior-regularized representation guidance. To stabilize the learning of representation learning, we consider a regularized version of \mathcal{L}_F in Eq. (5) by introducing the task-conditioned state-value function V with respect to policy π , given by:

$$V_{\pi_z}(s, z) := \max_{\substack{a \in A \\ \text{s.t. } \mu(a|s) > 0}} F(s, a, z)^\top z, \quad (6)$$

where z is the task vector and $\mu(a|s)$ is the behavior policy in the dataset \mathcal{D} . Compared with the traditional formulation of the state-value function, this optimization problem aims to learn an optimal task-conditioned V -function solely from dataset samples without explicitly estimating $\mu(a|s)$. It can be solved through a class of value objectives according to different regularization

formulations [67, 18, 30]. In this work, we leverage the commonly used expectile regression as in IQL [30]:

$$\mathcal{L}_{V_{\pi_z}} = \mathbb{E}_{(s,a) \sim \mathcal{D}, z \sim \mathcal{Z}} [L_2^\tau (F(s, a, z)^\top z - V_{\pi_z}(s, z))], \quad (7)$$

where $L_2^\tau(u) = |\tau - \mathbb{I}(u < 0)|u^2$, with $\tau > 0.5$ serving as the expectile parameter. By minimizing the above equations, we can obtain a well-regularized and nearly optimal state-value function V_{π_z} for different tasks. Given the reward function r_z , associated with task z , we have the following modified policy evaluation operator \mathcal{T}^π given by

$$(\mathcal{T}^\pi Q_z)(s, a, z) := \mathbb{E}_{s' \sim \mathcal{P}(s'|s, a)} [r_z(s') + \gamma V_{\pi_z}(s', z)], \quad (8)$$

This formulation of \mathcal{T}^π implicitly introduces regularization to the Q -function. Following above, we modified \mathcal{L}_F in Eq. (5) as following $\mathcal{L}_{F\text{-reg}}$:

$$\mathcal{L}_{F\text{-reg}} = \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[(F(s, a, z)^\top z - B(s')^\top \mathbb{E}_D[B B^\top]^{-1} z - \gamma V_{\pi_z}(s', z))^2 \right]. \quad (9)$$

By substituting the potentially unstable target Q -approximation $F(s_{t+1}, \pi_z(s_{t+1}), z)^\top z$ with a more well-behaved state-value function V_{π_z} , we can greatly stabilize the learning of representation, as well as respond to optimality demands simultaneously. In comparison with the direct constraint on unseen value approximation, the behavioral regularization guidance ensures maximum preservation of the representation structure, while being flexible for tuning the degree of conservatism.

Behavior-regularized policy extraction. Having introduced regularization for the value function, we now turn to policy learning, which prior work suggests can be equally or even more critical for final performance [43]. A good policy should effectively leverage high-quality behavior and further generalize near data distribution. To ensure that the policy is well regularized by the dataset samples, we replace the policy objective in Eq. (3) with the following behavior-regularized optimization problem:

$$\max_{\pi_z} \mathbb{E}_{a \sim \pi_z(\cdot|s)} [F(s, a, z)^\top z - V_{\pi_z}(s, z)] \quad \text{s.t. } \int_a \pi_z(s) da = 1, \forall s \in \mathcal{S}, \quad D_{\text{KL}}(\pi_z \| \mu) \leq \epsilon, \quad (10)$$

where the term $F(s, a, z)^\top z - V_{\pi_z}(s, z)$ acts as an advantage function maximization, which is equivalent to maximizing the Q -function. The KL constraint anchors π_z to the behavior policy μ , thereby preventing distributional shift in the offline setting. Given the optimal representation F and V_{π_z} , we can obtain the closed-form solution for the constrained optimization objective Eq. (10) by deriving the Lagrangian objective with respect to the policy and setting its derivative to zero, as shown in Proposition 1 below:

Proposition 1. *The solution to the constrained optimization problem in Eq. (10) yields an optimal policy of the form (See Appendix A for proof):*

$$\pi_z^*(s) \propto \mu(a|s) \exp (\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))), \quad (11)$$

where $1/\alpha$ is the Lagrangian multiplier for the KL constraint.

α acts as a temperature that controls the balance between the regularization strength of the behavioral policy and the optimization of the value functions. This closed-form solution elegantly combines the distribution of the behavior policy with the optimal value functions, encouraging the policy to favor high-advantage actions while ensuring all actions remain within the support of the dataset, thus balancing performance with OOD avoidance. With this stable guarantee, we now aim to enhance the policy generalization capability in the next section.

4.2 Policy Extraction via Task-Conditioned Diffusion Model

To compute the solution of the weighted behavior cloning objective in Eq. (11) in practice, a key challenge arises: *how to efficiently model and sample from potentially highly complex and diverse distributions distilled from a re-weighted behavior policy?*

Direct weighted behavior cloning with a Gaussian policy often fails to capture the complex, multi-modal policy distributions [62, 20, 7], which is a demand for arbitrary task learning. This motivates the use of diffusion models [48, 22, 37], known for their capacity to learn complex distributions through

iterative denoising. Rather than relying on guidance-based techniques that introduce a separate, time-dependent term to steer the sampling process, we draw inspiration from recent advances in weighted regression for diffusion models [7, 20, 27, 72]. Building on the theoretical foundation of prior work [72], we formalize in Proposition 2 how the optimal policy π_z^* can be extracted using a diffusion model trained with a weighted regression objective:

Proposition 2 (Task-conditioned diffusion policy extraction via weighted regression). *The extraction of optimal policy π_z^* in Eq. (11) can be achieved by (i) minimizing the weighted regression loss defined as:*

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}([0, T]), \epsilon \sim \mathcal{N}(0, I), (s, a) \sim \mathcal{D}} \left[\exp \left(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z)) \right) \|\epsilon - \epsilon_{\theta, z}(a_t, s, z, t)\|_2^2 \right], \quad (12)$$

with expectations taken over $t \sim \mathcal{U}([0, T])$, $\epsilon \sim \mathcal{N}(0, I)$, and $(s, a) \sim \mathcal{D}$, where $a_t = \alpha_t a + \sigma_t \epsilon$ follows the forward process $\mathcal{N}(a_t | \alpha_t a, \sigma_t^2 I)$ parameterized by noise schedules α_t, σ_t and $z \sim \mathbb{R}^d$ denotes the corresponding task; and (ii) sampling from π_z^* by solving the corresponding diffusion ODEs/SDEs with the learned $\epsilon_{\theta, z}$. (See Appendix A for the detailed proof.)

With the above objective, we can avoid the need for learning the additional time-dependent guidance term and thereby reduce the complexity, while getting a stable policy.

For action selection, we employ a rejection sampling mechanism to boost policy performance. Specifically, we first sample K candidate actions $\{a^{(1)}, \dots, a^{(K)}\} \sim \pi_z(s)$ through the policy rollout. We then evaluate each candidate using the Q -function (approximated by $F(s, a, z)^\top z$) and select the action with the highest value:

$$a^* \triangleq \arg \max_{a \in \{a^{(1)}, \dots, a^{(K)}\} \sim \pi_z(s)} F(s, a, z)^\top z. \quad (13)$$

This two-stage approach ensures the policy balances both conservatism and optimism. The diffusion model generates diverse, in-distribution candidates, while the selection step identifies the action with the highest expected return. This combination of expressive generative modeling and value-based selection is crucial for achieving robust, high-performance zero-shot generalization.

4.3 Expressivity Enhancement for Representation Modeling

The effectiveness of any policy—particularly a highly expressive diffusion policy—in leveraging learned action weights hinges on accurate value estimation. Biased value estimates can degrade policy learning to simple imitation, failing to redistribute probability mass toward superior actions. To fully exploit the representational capacity of our diffusion policy, the underlying value representations must precisely capture complex task and dynamic relationships. We thus introduce enhanced network architectures for the forward (F) and backward (B) representations.

As shown in Section 3, we empirically identify a pair of attention-based architectures that improve zero-shot performance over the original FB implementations. Below, we detail the design of our networks, illustrated in Figure 3.

Forward Network. The forward network encodes state-task and state-action pairs using two separate linear encoders, following Touati et al. [Touati et al. [58]]. The state-task encoder captures task-conditioned dynamics, while the state-action encoder extracts agent behavior patterns. These two feature sets correspond to semantically distinct concepts in the MDP. To accurately approximate the measure $M(s, a, s', z)$ in Eq. 3, the representation F must integrate information from both behavioral and task contexts, capturing their interdependencies effectively. We therefore model the two encoded features as a length-2 embedding sequence and process them through self-attention blocks. This allows bidirectional feature refinement between task conditions and agent behaviors. The resulting representation is projected onto a d -dimensional space via linear layers.

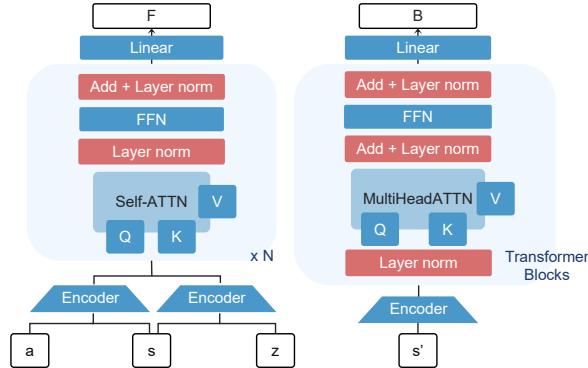


Figure 3: Architecture of BREEZE

Table 1: **IQM results on ExORL benchmark.** We report the best overall performance when all tasks perform well. Each value is averaged over 10 rollouts on 5 random seeds.

Dataset	Domain	SF-LAP	FB	VCFB	MCFB	HILP	BREEZE
RND	Walker	516 ± 97	661 ± 10	653 ± 22	659 ± 51	665 ± 33	693 ± 16
	Jaco	18 ± 18	32 ± 23	46 ± 35	41 ± 34	52 ± 21	84 ± 14
	Quadruped	330 ± 165	671 ± 14	609 ± 29	684 ± 18	674 ± 28	725 ± 23
APS	Walker	324 ± 24	517 ± 99	487 ± 75	578 ± 35	643 ± 22	637 ± 21
	Jaco	39 ± 26	22 ± 14	20 ± 18	22 ± 3	84 ± 16	132 ± 16
	Quadruped	498 ± 160	668 ± 29	664 ± 3	659 ± 50	679 ± 14	698 ± 24
PROTO	Walker	382 ± 129	650 ± 19	611 ± 94	602 ± 112	715 ± 31	663 ± 19
	Jaco	15 ± 14	21 ± 26	13 ± 12	20 ± 21	44 ± 19	74 ± 26
	Quadruped	199 ± 10	222 ± 107	185 ± 72	219 ± 135	216 ± 54	389 ± 44
DIAYN	Walker	239 ± 79	338 ± 74	268 ± 67	268 ± 97	461 ± 64	463 ± 42
	Jaco	32 ± 26	22 ± 6	24 ± 3	15 ± 1	52 ± 7	78 ± 11
	Quadruped	207 ± 168	562 ± 23	511 ± 37	643 ± 14	670 ± 4	666 ± 2

Table 2: **IQM results with 100k-dataset.** Experiments on randomly sampled 100k-transition data from each dataset on the ExORL benchmark

Dataset	Domain	FB	VCFB	MCFB	BREEZE (ours)
RND	Walker	264 ± 33	350 ± 29	287 ± 48	525 ± 13
	Jaco	7 ± 5	9 ± 2	13 ± 7	36 ± 5
	Quadruped	176 ± 123	233 ± 52	123 ± 61	474 ± 21
APS	Walker	370 ± 66	416 ± 10	389 ± 77	539 ± 15
	Jaco	21 ± 17	14 ± 13	29 ± 27	38 ± 9
	Quadruped	340 ± 29	351 ± 57	318 ± 122	556 ± 52
PROTO	Walker	415 ± 19	513 ± 31	463 ± 11	553 ± 18
	Jaco	16 ± 2	18 ± 12	12 ± 7	29 ± 12
	Quadruped	198 ± 111	106 ± 103	240 ± 134	181 ± 60
DIAYN	Walker	202 ± 94	210 ± 81	196 ± 26	330 ± 43
	Jaco	17 ± 11	18 ± 5	20 ± 26	22 ± 15
	Quadruped	295 ± 46	288 ± 48	286 ± 34	446 ± 78

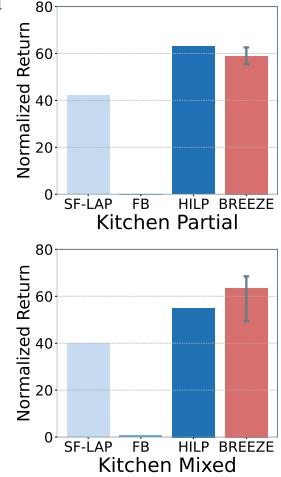


Figure 4: Normalized scores on Kitchen environment

Backward Network. As defined in Eq. 3, the backward representation B encodes state-level structural information, serving as a global embedding of the environment. Intuitively, B should employ a more complex architecture, maintaining orthogonality and enforcing F alignment. We equip B with a stack of standard transformer-based networks with multi-head attention. The final output is projected onto a d -dimensional space, consistent with the forward network.

These architecturally enhanced representations enable BREEZE to model complex relationships more accurately, leading to improved value estimates and policy performance. Further implementation details are provided in Appendix B.4.

5 Experiments

We conduct extensive evaluations of BREEZE against previous offline zero-shot RL algorithms across various challenging domains with distinct tasks. We also present the ablations on component choices and hyperparameters.

Environmental setup. Our main experiments are conducted on the ExORL benchmark [70], which provides a variety of datasets collected by several unsupervised RL algorithms [34]. We select datasets collected by 4 algorithms: *RND* [5], *APS* [38], *DIAYN* [11], and *PROTO* [69]. The experiments span 3 domains and 12 tasks (Walker: Stand, Walk, Run, Flip; Jaco: Reach Top/Bottom Left/Right; Quadruped: Stand, Walk, Run, Jump), bringing the total to 48 state-based complex tasks for performance calculation. In addition, we consider four challenging multi-stage tasks in the D4RL [14] Franka Kitchen domain [19] with two datasets (mixed and partial), which require long-horizon sequential manipulation on 4 subtasks. The overall setup involves 2 locomotion domains and 2 manipulation (goal-reaching) domains. For all goal-conditioned domains, we use the backward representation to calculate the goal features as the task vector z .

Baselines. We only consider offline algorithms that could perform a zero-shot policy generalization for our evaluation. Our baselines include: 1) *SF-LAP* [3, 4]: successor feature-based method with basic features to be *Laplacian Eigenfunction (LAP)* [64], 2) *vanilla FB* [57], 3) *VCFB* and *MCFB* [26]: FB-based offline algorithms with *CQL*-style regularizer [33] on successor measure M^π and Q -value separately for OOD issue avoidance, 4) *HILP* [44]: a state-based representation modeling algorithm keeping distance-preserving temporal structures in latent space having zero-shot capability.

Evaluation. Our experiments are designed to evaluate two key aspects: **zero-shot performance** and **robustness**. To assess the overall zero-shot capability, we experiment on the full transitions of each dataset on ExORL and D4RL Kitchen. For robustness, we first present performance curves across training time to demonstrate learning stability and then conduct experiments on a uniformly subsampled 100,000 transitions to examine performance in a limited and low-quality data coverage scenario, following the evaluation process of Jeen et al. [26]. We record the Interquartile Mean (IQM), a robust metric that reduces the influence of outliers. We evaluate the checkpoints every 10,000 updates with 10 rollouts for each experiment. On Kitchen, we multiply the returns by 25 for normalization, following Park et al. [44]. Further implementation and experimental details are provided in Appendix B.

5.1 Experimental Results

We analyze the findings of our experiments through the following key questions:

- *Does BREEZE possess zero-shot capability for novel tasks?* Following previous studies [1, 26], we record the highest aggregated mean performance across random seeds in the offline pretrain stage. Table 1 reports this score across all tasks on each domain of ExORL, averaged on 5 random seeds, with standard deviations. BREEZE achieves the best or near-best returns in most domains, demonstrating superior zero-shot generalization capability. Moreover, BREEZE can enhance *vanilla FB*'s generalization capability in long-horizon tasks. As shown in Figure 4, we draw the box with the performance reported in the previous study [44], and record BREEZE's top averaged returns across 4 random seeds. While *vanilla FB* struggles in these long-horizon tasks, BREEZE achieves significantly higher performance, suggesting its ability to better leverage the dataset for complex sequential manipulation.

More results are provided in Appendix D.2.

- *How does BREEZE quantitatively compare to baselines in terms of stability and convergence speed?* We provide the learning curves in *RND* [5] datasets as Figure 5. In locomotion domains (i.e., Quadruped and Walker), BREEZE converges faster to higher performance, with smoother curves and lower variance (as shown by the small shaded area). In the manipulation domain (i.e., Jaco), BREEZE substantially outperforms all baselines with a higher learning speed. Overall, BREEZE demonstrates faster convergence and enhanced stability across both locomotion and manipulation scenarios.
- *How does BREEZE perform facing different quality, diversity of datasets?* Table 2 shows results on 100,000-transition subsets of ExORL datasets. BREEZE maintains a pronounced advantage over FB-based baselines in this small-sample regime. Compared to explicit constraint methods (*MCFB/VCFB*), these results highlight the importance of behavior alignment across different data regimes. BREEZE also exhibits superior stability, as shown in Figure 6.

Refer to Appendix D.1 for more learning curves.

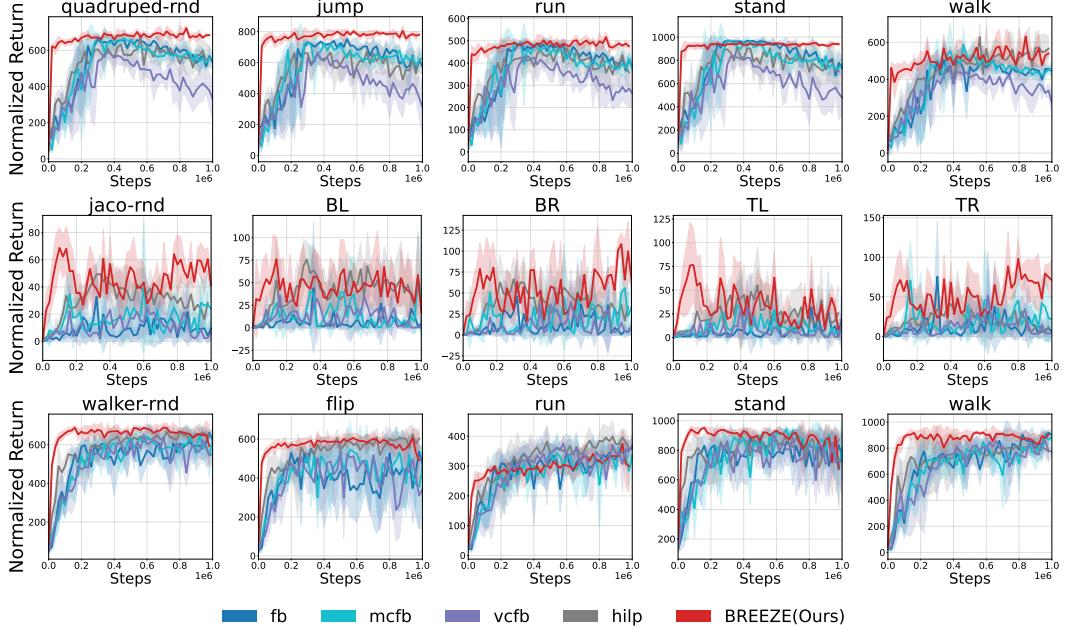


Figure 5: **Learning Curves on ExORL RND.** The solid lines represent the average return over 5 random seeds, while the shaded area denotes the standard deviation.

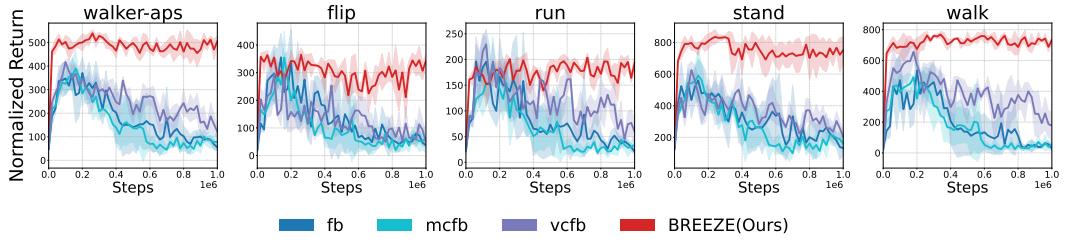


Figure 6: **Learning Curves on 100k-subsample transitions on ExORL APS Walker domain.** The solid lines represent the average return over 3 random seeds, while the shaded area denotes the standard deviation.

Dataset	w/o FB Enhancement	w/o Diffusion	BREEZE
Walker-RND	646±18	707±13	693
Jaco-RND	80±24	62±7	84
Quadruped-RND	685±13	530±33	725
Walker-APS	614±48	587±53	637
Jaco-APS	82±13	45±50	132
Quadruped-APS	655±20	568±19	698

Table 3: Ablation on necessity of FB enhancement and diffusion policy (3 seeds).

5.2 Ablation Studies

Hyperparameter choices. We ablate two key hyperparameters in our behavior-constrained algorithm: the expectile value τ and the temperature α . As shown in Figure 7 and Figure 8 (Quadruped domain, RND dataset, 3 random seeds), performance improves near monotonically with τ and peaks at $\alpha = 0.05$, and similarly to other domains. This indicates that, while in-sample conservatism is crucial, the algorithm benefits from more aggressive, value-weighted optimization. Consequently, we select $\tau = 0.99$ and $\alpha = 0.05$ as defaults in ExORL, optimally balancing conservatism with performance for downstream tasks.

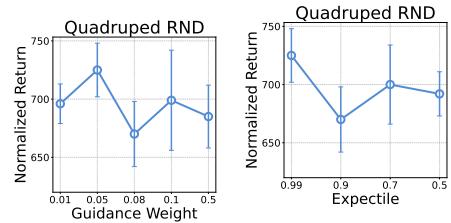


Figure 7: Ablation on the expectile factor.

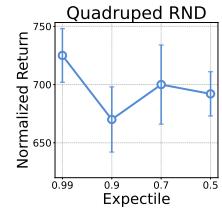


Figure 8: Ablation on the expectile factor.

Necessity of each component. We ablate our two core innovations—an enhanced representation model and a diffusion policy—to assess their necessity in RND and APS datasets. BREEZE is ablated using: a gaussian policy variant (*w/o Diffusion*) and the baseline architecture variant (*w/o FB Enhancement*). We process transitions with data normalization for the diffusion policy, as in a commonly used preprocessing step in previous diffusion-based methods. We cancel this process and return to the default policy setting to have a reasonable comparison of the components’ contributions. As shown in Table 3, each component somehow individually slightly boosts performance, but their synergy creates a leap in zero-shot capability. This demonstrates a critical interdependence: the design of BREEZE is a logical combination to handle the system issue.

6 Related Works

Unsupervised zero-shot RL. Recent advancements in unsupervised zero-shot RL [34, 44, 13] mainly build upon the utilization of successor representation (SR) [10], with two main branches: successor feature (SF) [3, 4] and forward-backward representation (FB) [57, 58]. Both approaches require decomposing and linearizing reward-aware dynamics to enable zero-shot inference for arbitrary downstream rewards. Other approaches relax the linearity assumption by modeling SR as full distributions using generative models [25, 55, 63, 12]. Among all, FB has emerged as a pivotal framework, factorizing successor measures into forward and backward components and avoiding the potential collapse issue in SF at the same time. Extensions like MCFB/VCFB [26] build on offline RL, enforcing explicit value constraints to mitigate the over-estimation issues that arise from updating on OOD rollouts. FB-CPR [56] introduces a regularization of the Q -estimation by imitating pre-collected datasets in an online setting. FB-AWARE [6] introduces an autoregressive mechanism to handle the over-linearization problem in vanilla FB. Unlike SR-based approaches, HILP [44] structures the latent space so that distances correspond to transition times between states, enabling hierarchical, goal-conditioned, and zero-shot policy learning. Meanwhile, functional-encoder methods [13, 23] directly encode task information by pretraining on manually designed reward functions, but are constrained by the need for extensive reward engineering and fine-tuning.

Offline RL. Numerous studies address offline RL, in which learning is restricted to a fixed dataset without online interaction, leading to a distributional shift. The earliest approaches use policy constraints, enforcing the learned policy to stay close to the behavior policy [17, 32, 15, 35] or the distribution coverage to stay near the dataset [36, 8]. Another established line of work employs value regularization, which directly penalizes the value estimates of OOD actions [33, 31, 42, 68, 66]. Instead of an explicit constraint, in-sample learning methods [30, 67, 65, 61] improve stability by learning values and policies only from state-action pairs within the dataset.

Moving beyond conventional Gaussian policies, recent studies have highlighted the importance of modeling multimodal action distributions [62, 20, 7]. Diffusion models [48, 22, 37], known for their strong multi-modal modeling capability, have been well studied in imitation and trajectory modeling [24, 73, 9, 41, 52], and have recently been integrated into offline RL frameworks. Some approaches use the value function as energy guidance for sampling [39, 40]. Other works, such as SfBC [7], IDQL [20], and EDP [27], extract a diffusion policy via weighted regression under the in-sample learning framework.

7 Conclusion

In this work, we present BREEZE, a novel framework that mitigates the improper scaling issue in existing FB-based zero-shot RL methods. Specifically, BREEZE tackles two critical limitations: offline extrapolation errors and constrained expressivity. Our solution integrates behavior-regularized value estimation with a task-conditioned diffusion policy, enabling stable in-sample learning while capturing complex, multimodal action distributions. Coupled with expressive attention-based representations, BREEZE more accurately models dynamics and value functions. Extensive experiments demonstrate that BREEZE consistently outperforms existing methods across various benchmarks, underscoring the importance of calibrated regularization and sufficient model capacity for zero-shot generalization. While BREEZE demonstrates improved robustness, its primary limitation is the increased computational cost from diffusion-based sampling, which is a common trade-off for utilizing high-performance generative policies. Further discussion is provided in Appendix E.

Acknowledgment

This work is supported by the Wuxi Research Institute of Applied Technologies, Tsinghua University, under Grant 20242001120, and funded by Horizon Robotics, AsiaInfo, and the Xiongan AI Institute.

References

- [1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.
- [2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [3] André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017.
- [4] Diana Borsa, André Barreto, John Quan, Daniel Jaymin Mankowitz, Rémi Munos, H. V. Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. In *The Seventh International Conference on Learning Representations*, 2019.
- [5] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *ArXiv*, abs/1810.12894, 2018.
- [6] Edoardo Cetin, Ahmed Touati, and Yann Ollivier. Finer behavioral foundation models via auto-regressive features and advantage weighting. *Reinforcement Learning Conference*, abs/2412.04368, 2025.
- [7] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.
- [8] Peng Cheng, Xianyuan Zhan, Zhihao Wu, Wenjia Zhang, Shoucheng Song, Han Wang, Youfang Lin, and Li Jiang. Look beneath the surface: Exploiting fundamental symmetry for sample-efficient offline rl. In *Advances in Neural Information Processing Systems*, 2023.
- [9] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- [10] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.
- [11] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *The Seventh International Conference on Learning Representations*, 2019.
- [12] Jesse Farbrother, Matteo Pirotta, Andrea Tirinzoni, Rémi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. In *International Conference on Machine Learning*, 2025.
- [13] Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Unsupervised zero-shot reinforcement learning via functional reward encodings. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.
- [14] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.
- [15] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

- [16] 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.
- [17] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2018.
- [18] Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023.
- [19] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019.
- [20] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023.
- [21] Longxiang He, Li Shen, Junbo Tan, and Xueqian Wang. Aligniql: Policy alignment in implicit q-learning through constrained optimization, 2024.
- [22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.
- [23] Tyler Ingebrand, Amy Zhang, and Ufuk Topcu. Zero-shot reinforcement learning via function encoders. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024.
- [24] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research.
- [25] Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: generative temporal difference learning for infinite-horizon prediction. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.
- [26] Scott Jeen, Tom Bewley, and Jonathan M. Cullen. Zero-shot reinforcement learning from low quality data. In *Advances in Neural Information Processing Systems*, 2024.
- [27] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.
- [28] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Annual Conference on Neural Information Processing Systems*, 2021.
- [29] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2022. doi: 10.1109/TITS.2021.3054625.
- [30] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *International Conference on Learning Representations*, 2021.
- [31] Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2021.
- [32] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.
- [33] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

- [34] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: unsupervised reinforcement learning benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021.
- [35] Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, and Ya-Qin Zhang. Proto: Iterative policy regularized offline-to-online reinforcement learning, 2023.
- [36] Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. When data geometry meets deep function: Generalizing offline reinforcement learning. In *International Conference on Learning Representations*, 2023.
- [37] Sungbin Lim, Eunbi Yoon, Taehyun Byun, Taewon Kang, Seungwoo Kim, Kyungjae Lee, and Sungjoon Choi. Score-based generative modeling through stochastic evolution equations in hilbert spaces. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.
- [38] Hao Liu and P. Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, 2021.
- [39] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, 2023.
- [40] Liyuan Mao, Haoran Xu, Weinan Zhang, Xianyuan Zhan, and Amy Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.
- [41] Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning, ICML 2023*, Proceedings of Machine Learning Research, 2023.
- [42] Haoyi Niu, Shubham Sharma, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming HU, and Xianyuan Zhan. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [43] Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *Advances in Neural Information Processing Systems*, 37:79029–79056, 2024.
- [44] Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *International Conference on Machine Learning*, pages 39737–39761. PMLR, 2024.
- [45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.
- [46] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [47] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.
- [48] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 2015.

- [49] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.
- [50] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.
- [51] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998. doi: 10.1109/TNN.1998.712192.
- [52] Tianyi Tan, Yinan Zheng, Ruiming Liang, Zexu Wang, Kexin Zheng, Jinliang Zheng, Jianxiong Li, Xianyuan Zhan, and Jingjing Liu. Flow matching-based autonomous driving planning with advanced interactive behavior modeling. *arXiv preprint arXiv:2510.11083*, 2025.
- [53] Chen Tang, Ben Abatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: a survey of real-world successes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.
- [54] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018.
- [55] Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition. In *International Conference on Machine Learning*, 2022.
- [56] Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *International Conference on Representation Learning*, 2025.
- [57] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.
- [58] Ahmed Touati, Jérémie Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023.
- [59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [60] Shuhe Wang, Shengyu Zhang, Jie Zhang, Runyi Hu, Xiaoya Li, Tianwei Zhang, Jiwei Li, Fei Wu, Guoyin Wang, and Eduard Hovy. Reinforcement learning enhanced llms: A survey, 2025.
- [61] Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. In *Advances in Neural Information Processing Systems*, 2023.
- [62] Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.
- [63] Harley Wiltzer, Jesse Farebrother, Arthur Gretton, Yunhao Tang, André Barreto, Will Dabney, Marc G. Bellemare, and Mark Rowland. A distributional analogue to the successor representation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.
- [64] Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. In *The Seventh International Conference on Learning Representations*, 2019.
- [65] Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

- [66] 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, pages 8753–8760, 2022.
- [67] Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations*, 2023.
- [68] Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. *Advances in neural information processing systems*, 35:23851–23866, 2022.
- [69] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, 2021.
- [70] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don’t change the algorithm, change the data: Exploratory data for offline reinforcement learning, 2022.
- [71] 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, pages 4680–4688, 2022.
- [72] Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *The Twelfth International Conference on Learning Representations, ICLR 2024*, 2024.
- [73] Yinan Zheng, Ruiming Liang, Kexin ZHENG, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *The Thirteenth International Conference on Learning Representations*, 2025.

A Theoretical Proofs

A.1 Proof of Proposition 1

Proposition 1. *The closed-form optimal policy for Eq. (10) follows the form:*

$$\pi_z^*(s) \propto \mu(a|s) \exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))), \quad (14)$$

where μ denotes the behavior distribution of the dataset.

Proof. Construct the Lagrangian with dual variables λ_1 and λ_2 for the equality and KL-divergence constraints, respectively:

$$\mathcal{L}(\pi_z, \lambda_1, \lambda_2) = \mathbb{E}_{a \sim \pi_z} [F(s, a, z)^\top z - V_{\pi_z}(s, z)] - \lambda_1 \left(\int \pi_z(a|s) da - 1 \right) - \lambda_2 (D_{\text{KL}}(\pi_z \| \mu) - \epsilon). \quad (15)$$

Taking the functional derivative of \mathcal{L} with respect to $\pi_z(a|s)$ and setting it to zero yields:

$$\frac{\partial \mathcal{L}}{\partial \pi_z} = F(s, a, z)^\top z - V_{\pi_z}(s, z) - \lambda_2 (\log \pi_z(a|s) - \log \mu(a|s) + 1) - \lambda_1 = 0. \quad (16)$$

Solving above for π_z and absorbing constant terms into the normalization factor $Z(s, z)$ gives:

$$\pi_z^*(a|s) = \frac{\mu(a|s)}{Z(s, z)} \exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))). \quad (17)$$

where $\alpha = 1/\lambda_2$ and the partition function is defined as:

$$Z(s, z) = \int \mu(a|s) \exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))) da, \quad (18)$$

This ensures the policy is properly normalized. The proof is completed. \square

A.2 Proof of Proposition 2

We begin by conducting a theoretical review of diffusion models, and then demonstrate how to derive the optimal policy through task-conditioned weighted regression.

Diffusion (probabilistic) models. Diffusion models [48, 22, 37] are powerful generative models designed to learn and replicate complex distributions. Given an N -sample dataset $\mathcal{D} = \{x_0^i\}_{i=1}^N$ from an unknown distribution $q(x_0)$, diffusion models can accurately approximate $q(x_0)$ and generate new samples from the approximation by the following two processes:

- **Forward process.** The forward process gradually adds Gaussian noise on sample x_0 from time 0 to x_T over T timesteps, with transformation distribution follows:

$$q_{t0}(x_t|x_0) = \mathcal{N}(x_t|\alpha_t x_0, \sigma_t^2 I), \quad (19)$$

where α_t and σ_t are predefined noise schedules ensuring $q_T(x_T|x_0) \approx q_T(x_T) \approx \mathcal{N}(x_T|0, \tilde{\sigma}^2 I)$ for some $\tilde{\sigma} > 0$ that is independent of x_0 .

- **Backward process.** Starting from $x_T \sim \mathcal{N}(x_T|0, \tilde{\sigma}^2 I)$, diffusion models reconstruct the original data x_0 by solving diffusion ODE/SDE [49] from T to 0:

$$(\text{Diffusion ODE}) \quad dx_t = \left[f(t)x_t - \frac{1}{2}g^2(t)\nabla_{x_t} \log q_t(x_t) \right] dt, \quad (20)$$

$$(\text{Diffusion SDE}) \quad dx_t = [f(t)x_t - g^2(t)\nabla_{x_t} \log q_t(x_t)] dt + g(t)d\tilde{\omega}, \quad (21)$$

where $f(t) = \frac{d \log \alpha_t}{dt}$, $g^2(t) = \frac{d \sigma_t^2}{dt} - 2\frac{d \log \alpha_t}{dt}\sigma_t^2$ [28], and $\tilde{\omega}$ is a standard Wiener process when time flows backwards from T to 0.

Diffusion models train a neural network $\epsilon_\theta(x_t, t)$ parameterized by θ with the following objective [22, 37]:

$$\min_{\theta} \mathbb{E}_{x_t, t \sim \mathcal{U}([0, T]), \epsilon \sim \mathcal{N}(0, I)} [\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2], \quad (22)$$

where $x_0 \sim q_0(x_0)$, $x_t = \alpha_t x_0 + \sigma_t \epsilon$. The fitted θ can be used to estimate the score function $\nabla_{x_t} \log q_t(x_t)$ by $\epsilon_\theta(x_t, t) \approx -\sigma_t \nabla_{x_t} \log q_t(x_t)$ and substituted into Eq. (20-21) for x_0 generation.

Energy-guided diffusion sampling. To bias sampling toward preferred distributions, the prior energy-guidance framework [39] provides the re-weighted distribution $p_0(x_0)$:

$$p_0(x_0) \propto q_0(x_0) f(\mathcal{E}(x_0)), \quad (23)$$

where $q_0(x_0)$ is the unknown data distribution, $\mathcal{E}(x_0)$ is any form of energy function that encodes human preferences, and $f(x) \geq 0$ can be any non-negative function. To solve this sampling problem, previous approaches train a separate time-dependent classifier \mathcal{E}_t [39, 24], which is computationally expensive and may introduce additional training errors. To address this, previous work [72] introduces weighted regression method to perform exact energy guidance:

Lemma 1 (*Weighted regression as exact energy guidance*). *We can sample $x_0 \sim p_0(x_0)$ where p_0 is the weighted distribution in Eq. (23) by (i) optimizing the weighted regression loss:*

$$\min_{\theta} \mathbb{E}_{x_t, t \sim \mathcal{U}([0, T]), \epsilon \sim \mathcal{N}(0, I)} [f(\mathcal{E}(x_0)) \|\epsilon - \epsilon_\theta(x_t, t)\|_2^2], \quad (24)$$

where $x_0 \sim q_0(x_0)$, $x_t = \alpha_t x_0 + \sigma_t \epsilon$; and (ii) substituting the obtained ϵ_θ into diffusion ODEs/SDEs [49] solving process.

Task-conditioned diffusion sampling. In this work, we extend the Lemma 1 and provide the following proposition for task-specific sampling:

Proposition 2 (*Task-conditioned diffusion policy extraction via weighted regression*). *The extraction of optimal policy π_z^* in Eq. (11) can be achieved by (i) minimizing the weighted regression loss defined as:*

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}([0, T]), \epsilon \sim \mathcal{N}(0, I), (s, a) \sim \mathcal{D}} [\exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))) \|\epsilon - \epsilon_{\theta, z}(a_t, s, z, t)\|_2^2],$$

and (ii) solving diffusion ODEs/SDEs [49] by substituting $\epsilon_{\theta, z}$ in Eq. (20-21).

Proof. We aim to extract the optimal policy forming as a weighted distribution as follows:

$$\pi_z^*(a|s) \propto \mu(a|s) \exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))),$$

where $\mu(a|s)$ is the behavior distribution in dataset \mathcal{D} and $F(s, a, z)^\top z - V_{\pi_z}(s, z)$ is the task-dependent weight function. Set $p_z(a|s)$ as a task-conditioned reweighted version of the unknown data distribution $q_0(a|s)$ and substitute the task-dependent weight function into Eq. (23), we have:

$$p_z(a|s) \propto q_0(a|s) \exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))). \quad (25)$$

There exist a normalization constant $Z(s, z) = \int q_0(a|s) \exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))) da$ ensures $p_z(a|s)$ to be a valid distribution. We now aim to sample from the distribution $p_z(a|s)$.

Parameterize the denoising process with a task-conditioned model $\epsilon_{\theta, z}(a_t, s, z, t)$ and substitute in Eq. (24), we can get a modified task-conditioned weighted regression objective for the diffusion models by replacing the energy function with $F(s, a, z)^\top z - V_{\pi_z}(s, z)$:

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}([0, T]), \epsilon \sim \mathcal{N}(0, I), (s, a) \sim \mathcal{D}} [\exp(\alpha \cdot (F(s, a, z)^\top z - V_{\pi_z}(s, z))) \|\epsilon - \epsilon_{\theta, z}(a_t, s, z, t)\|_2^2]. \quad (26)$$

According to the Lemma 1, substituting $\epsilon_{\theta, z}^*$ into the diffusion ODEs/SDEs [49] generates data that sample from $p_z(a|s)$, thereby recovering $\pi_z^*(a|s)$. The proof is completed. \square

B Experimental Setups

Our experiments span two benchmarks, ExORL [70] and D4RL [14], across two locomotion domains (*Walker* and *Quadruped*) and two manipulation goal-reaching domains (*Jaco* and *Kitchen*). All experiments we conducted are in state-based tasks.

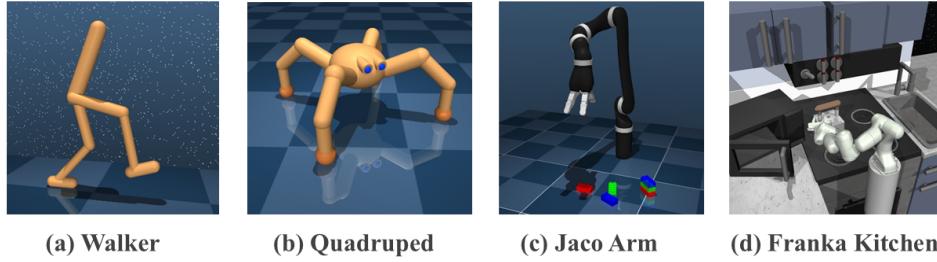


Figure 9: **Domains.** Walker, Quadruped, Jaco, and Franka Kitchen.

B.1 Datasets

ExORL [70]. ExORL consists of datasets collected by several unsupervised RL algorithms [34] on the DeepMind Control Suite [54]. We select datasets collected by four unsupervised RL algorithms: *APS* [38], *RND* [5], *PROTO* [69], and *DIAYN* [11] for each of the three domains (*Walker*, *Jaco*, and *Quadruped*). Each domain has four tasks for evaluation at test time.

D4RL [14] Kitchen. D4RL Kitchen datasets consist of robotic arm manipulating trajectories of different tasks in the Franka Kitchen [19] domain. We adopt two datasets: "kitchen-partial-v0" and "kitchen-mixed-v0".

- **Partial:** Subtasks involve the microwave, kettle, light switch, and slide cabinet. Tasks are guaranteed to be solved in a subset of the 'partial' dataset.
- **Mixed:** Subtasks involve microwave, kettle, bottom burner, and light switch. No trajectories solve any tasks completely in the 'mixed' dataset.

Dataset preprocessing. BREEZE incorporates standardized data normalization for diffusion models across all training datasets as a preprocessing step to stabilize the learning process and enhance generalization capability.

B.2 Domains

- **Walker** (Locomotion): A bipedal robot with 24-dimensional states (joint positions/velocities) and 6-dimensional actions. Test tasks include *Flip*, *Run*, *Stand*, and *Walk*. Rewards combine dense objectives: maintaining torso height (*Stand*), achieving target velocities (*Run/Walk*), or angular momentum (*Flip*).
- **Quadruped** (Locomotion): A four-legged robot with 78-dimensional states and 12-dimensional actions. Tasks include *Jump*, *Run*, *Stand*, and *Walk*, with rewards for torso stability and velocity tracking.
- **Jaco** (Manipulation/Goal-reaching): A 6-DoF robotic arm with 55-dimensional states and 6-dimensional actions. Tasks involve reaching four target positions (*Top/Bottom Left/Right*) using sparse rewards based on proximity to goals.
- **Franka Kitchen** [19] (Manipulation/Goal-reaching): A robotic manipulation environment controlling a 9-DoF Franka robot with multiple given test-time objectives. The agent must sequentially achieve four subtasks per episode, receiving sparse rewards for each subtask. States include arm joint positions, velocities, torques, and task-specific object features. Test-time goals are defined as proprioceptive states concatenated with target object states.

B.3 Baselines

We utilize the following open-source codebases for experiments on baselines. We set the batch size to 512 to ensure a fair comparison with other methods, while keeping all other hyperparameters at their default values.

- For *SF-LAP* [4] and *FB* [57], *MCFB* and *VCFB* [26], we utilized the open-source implementation available at <https://github.com/enjeeneer/zero-shot-rl>.

- For *HILP* [44], we utilized the open-source implementation for state-based zero-shot RL setting, which is available at <https://github.com/seohongpark/HILP>.

B.4 Architectures

This section details the model architectures used in BREEZE.

F Network. The F network processes inputs (s, a) and (s, z) using two separate MLPs, each with two hidden layers of size 1024 and ReLU activation, projecting the inputs into a 512-dimensional feature space. The resulting embeddings are concatenated along the sequence dimension and processed through two identical processing blocks. Each block consists of a self-attention layer [59], followed by a feed-forward network with residual connections, LayerNorm [2], and dropout (rate=0.1). Specifically, the final representation is flattened and passed through two independent output heads, denoted as $F1$ and $F2$, each comprising two linear layers with a hidden dimension of 256 that map to the d -dimensional output space.

B Network. The B network is based on a standard transformer architecture with 8-head multi-head attention [59] and ReLU activation. State embeddings are processed through transformer blocks and projected to a d -dimensional space via a linear layer and further scaled to \sqrt{d} by L2 normalization. We observe that a sufficiently large B network is essential for achieving strong performance on the ExORL benchmark.

Diffusion Policy. The diffusion policy uses a residual-connected MLP as the noise predictor, following the IDQL [20] implementation. The hidden dimension is set to 1024. The task vector z is incorporated via Feature-wise Linear Modulation (FiLM) [46]: two parallel linear layers generate a scaling factor γ and a shifting factor β from z , which modulate intermediate features h as $\gamma \odot h + \beta$.

B.5 Sampling of z

Following previous studies [58, 26], we utilize a mixture of two methods for sampling the latent variable z :

- Sample z uniformly within a sphere of radius \sqrt{d} in \mathbb{R}^d .
- Derive z by setting $z = B(s)$, where $s \sim \mathcal{D}$ is randomly sampled from the data set.

In our primary experiments, we employ a balanced mix ratio of 50% between the two methods.

B.6 Value Learning

Training details. We implement the V -network with 3-layer MLPs with 512 hidden dimensions and ReLU activation functions. During training, we set the τ for expectile regression in Eq. (7) shown in Section B.7. We use clipped double value learning [16] for both the Q -value and the M -value.

Rejection sampling. To improve the zero-shot performance, we use rejection sampling rather than tuning existing parameters for greedy optimization. Rejection sampling is commonly used in diffusion policies [7, 20, 21], a stable mechanism that selects N actions from the policy to boost with the highest value. Our practical implementation of rejection sampling is in two phases:

- **Training phase:** To compute the FB loss (Eq. 4), we first sample a full transition batch, which contains states, actions, next states, and next actions. A hyperparameter ρ_a , referred to as the dataset mixture ratio, determines the proportion of next actions drawn directly from the dataset. The remaining fraction is replaced by actions sampled from the policy π_z : we generate K_{train} candidate actions $a_{t+1}^{(1)}, \dots, a_{t+1}^{(K_{\text{train}})}$ via the diffusion policy and select the optimal next action according to:

$$a_{t+1}^* = \arg \max_{a_{t+1}^{(i)}} F(s_{t+1}, a_{t+1}^{(i)}, z)^\top z. \quad (27)$$

- **Evaluation phase:** We sample K_{eval} candidate actions during evaluation, and select the optimal action according to the Q -value.

Details of practical ρ_a , K_{train} and K_{eval} is provided in Section B.7.

B.7 Hyperparameter

This section provides the detailed hyperparameter setup. In our experiments, the model architecture and basic algorithm hyperparameters remain unchanged, as detailed in Table 4. Domain-specific hyperparameters are detailed in Table 5 and Table 6.

Table 4: General hyperparameters used for BREEZE

	Hyperparameter	Value
General hyperparameters	Optimizer	AdamW
	Representation learning rate	1e-4
	V network learning rate	3e-4
	Policy learning rate	1e-4
	Discount factor γ	0.98
	Learning steps	1,000,000
	Mini-batch	512
	Representation soft update factor λ	0.01
	Policy soft update factor λ	0.001
	Latent dimension d	50
	z mixing ratio	0.5
	z inference steps	10,000
	Regularization weight for orthogonality loss w_1	10
	Beta schedule	vp
	Exponential advantages clip	($-\infty$, 100]
Architecture	F preprocessor hidden dimension	1024
	F preprocessor hidden layers	2
	F preprocessor output dimension	512
	F preprocessor activation function	relu
	F attention blocks	2
	F linear layer hidden dimension	256
	F dropout	0.1
	B nhead	8
	B encoder layers	2
	B decoder layers	2
	B d model	256
	B dropout	0.1
	B feedforward dimension	2048
	B activation function	relu
	V hidden dimension	512
	V hidden layers	2
	V activation function	relu
	Policy MLP blocks	3
	Policy hidden dimension	1024
	Policy activation function	mish
	Policy time embedding	learned

C Pseudo-Code

Algorithm 1 provides the pseudocode. Our implementation uses PyTorch [45], with all experiments conducted on a single NVIDIA A6000 GPU. Peak memory usage under 23 GB.

Table 5: Domain-specific hyperparameters for BREEZE with full dataset.

Domain-dataset	K_{train}	ρ_a	K_{eval}	expectile τ	F -reg coef. ω_q	Diffusion steps T	Temperature α
Walker-RND	9	0.1	64	0.99	0.001	5	0.05
Walker-APS	9	0.1	64	0.99	0.001	5	0.05
Walker-PROTO	9	0.1	64	0.99	0.001	5	0.05
Walker-DIAYN	9	0.1	64	0.99	0.001	5	0.05
Jaco-RND	1	0	64	0.99	0.001	10	0.05
Jaco-APS	9	0.1	64	0.99	0.0001	5	0.05
Jaco-PROTO	1	0	64	0.99	0.001	10	0.05
Jaco-DIAYN	9	0.1	64	0.99	0.0001	5	0.05
Quadruped-RND	1	0	64	0.99	0.001	10	0.05
Quadruped-APS	1	0	64	0.99	0.001	10	0.05
Quadruped-PROTO	1	0	64	0.99	0.001	10	0.05
Quadruped-DIAYN	1	0	64	0.99	0.001	10	0.05
Kitchen-mixed	1	0	16	0.7	0.001	5	0.1
Kitchen-partial	2	0.2	4	0.7	0.001	5	0.08

Table 6: Domain-specific hyperparameters for BREEZE with small sample dataset.

Domain-dataset	K_{train}	ρ_a	K_{eval}	expectile τ	F -reg coef. ω_q	Diffusion steps T	Temperature α
Walker-RND	2	0.2	32	0.99	0.0001	10	0.05
Walker-APS	2	0.2	32	0.99	0.0001	10	0.05
Walker-PROTO	2	0.2	32	0.99	0.001	10	0.05
Walker-DIAYN	8	0.2	32	0.99	0.001	10	0.05
Jaco-RND	1	0	32	0.99	0.0001	10	0.05
Jaco-APS	1	0	32	0.99	0.0001	10	0.05
Jaco-PROTO	1	0	32	0.99	0.0001	10	0.05
Jaco-DIAYN	1	0	32	0.99	0.0001	10	0.05
Quadruped-RND	2	0.2	32	0.99	0.001	10	0.05
Quadruped-APS	1	0	32	0.99	0.001	10	0.05
Quadruped-PROTO	2	0.2	32	0.99	0.001	10	0.05
Quadruped-DIAYN	2	0.2	32	0.99	0.0001	10	0.05

Algorithm 1 Behavior-Regularized Zero-shot RL (BREEZE)

Require: Latent dimension d , mini-batch b , gradient steps N , orthogonality coefficient w_1 , value regularization coefficient w_q .

- 1: Normalize dataset \mathcal{D} ; Initialize networks F_ϕ , B_ψ , V_ω , diffusion actor π_z with network ϵ_θ
- 2: **for** $n = 1, \dots, N$ **do**
- 3: Sample a mini-batch of transitions $\{(s_t, a_t, s_{t+1}, a_{t+1})_i\}_{i \in [b]} \subset \mathcal{D}$
- 4: Sample a mini-batch of $\{z_i\}_{i \in [b]} \sim \mathbb{R}^d$
- 5: Collect $\{a'_{t+1}\}_{i \in [b]}$ by mixture of data batch and π_z outputs
- 6: **// Representation learning:**
- 7: Compute $\mathcal{L}_{\text{FB}}(\phi, \psi)$ and $\mathcal{L}_{F\text{-reg}}(\phi)$ using Eq. (4), Eq. (9)
- 8: Compute orthogonality regularization loss $\mathcal{L}_{\text{ortho}}$
- 9:
$$\mathcal{L}_{\text{ortho}}(\psi) = \frac{1}{|b|^2} \sum_{i,j \in [b]} [(B_\psi(s_i)^\top B_\psi(s'_j))^2 - \|B_\psi(s_i)\|_2^2 - \|B_\psi(s'_j)\|_2^2]$$
- 10: Update F_ϕ and B_ψ by $\mathcal{L}_{\text{FB}} + w_1 \cdot \mathcal{L}_{\text{ortho}} + w_q \cdot \mathcal{L}_{F\text{-reg}}$
- 11: **// Value function learning:**
- 12: Update V_ω using Eq. (7)
- 13: **// Diffusion policy learning:**
- 14: Update ϵ_θ using Eq. (12)
- 15: **end for**

D Additional Results

This section provides a comprehensive showcase of all experimental results.

- Section D.1 provides the learning curve comparisons.

- Section D.2 reports the detailed results on ExORL benchmark.
- Section D.3 presents the visualizations of our empirically evaluated value distributions.

D.1 Learning Curves

This section presents the zero-shot learning curves on the ExORL benchmark. Results on the full dataset and the 100k small-sample dataset are shown in Figure 10-12 and Figure 13-15, respectively. The solid lines represent the mean IQM returns over 5 random seeds, and the shaded regions correspond to the standard deviation.

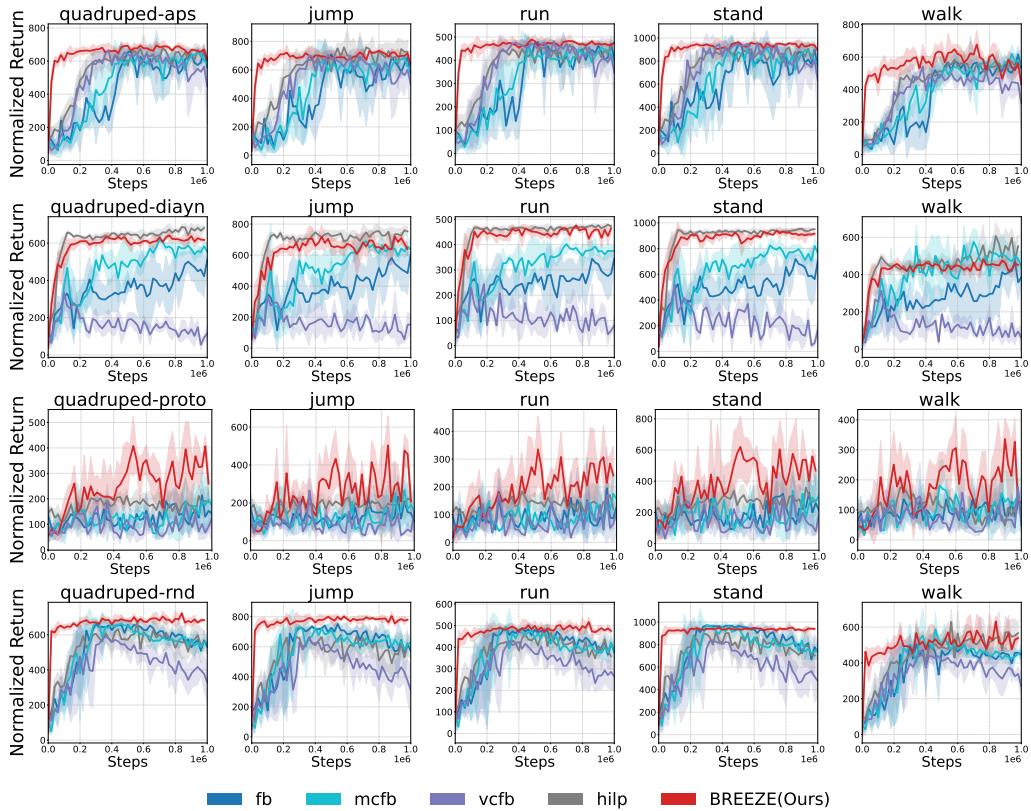


Figure 10: Curves of zero-shot performance on Quadruped domain.

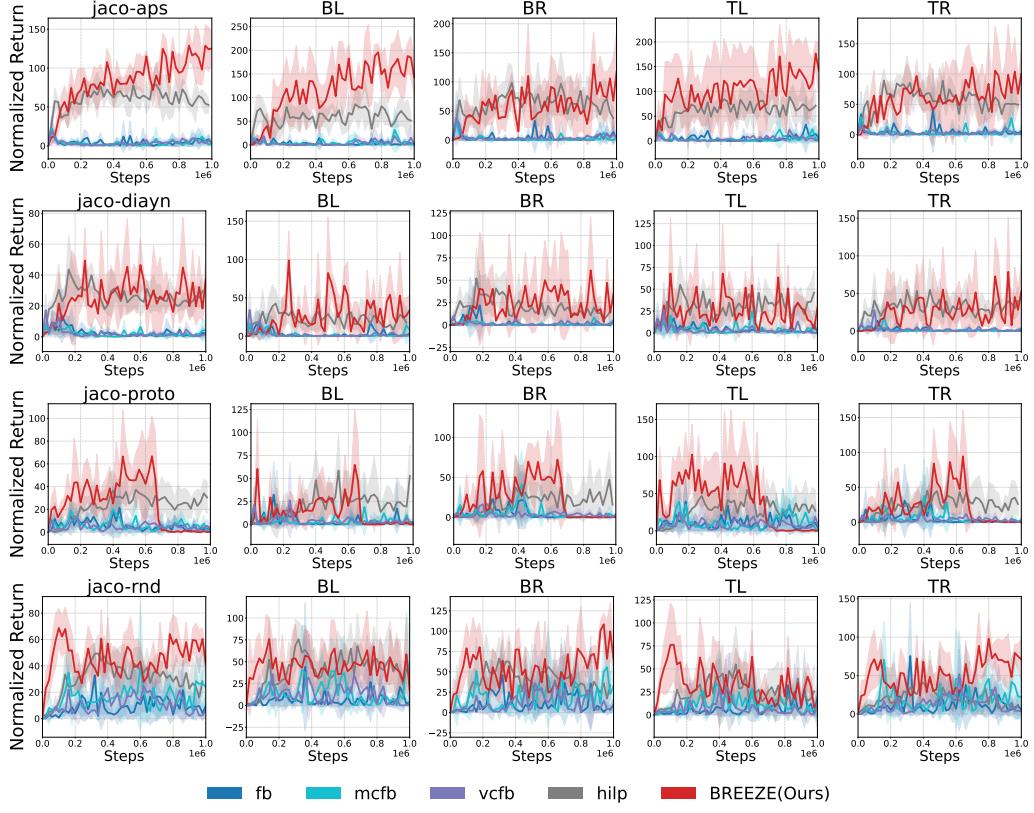


Figure 11: Curves of zero-shot performance on Jaco domain.

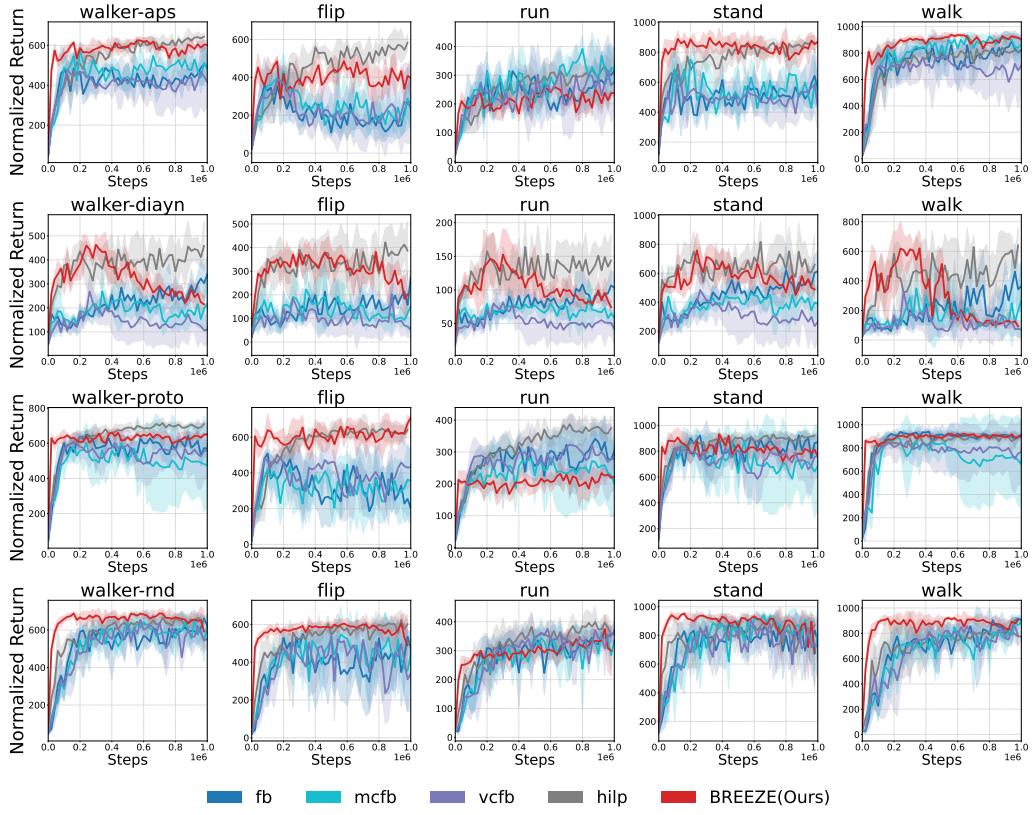


Figure 12: Curves of zero-shot performance on Walker domain.

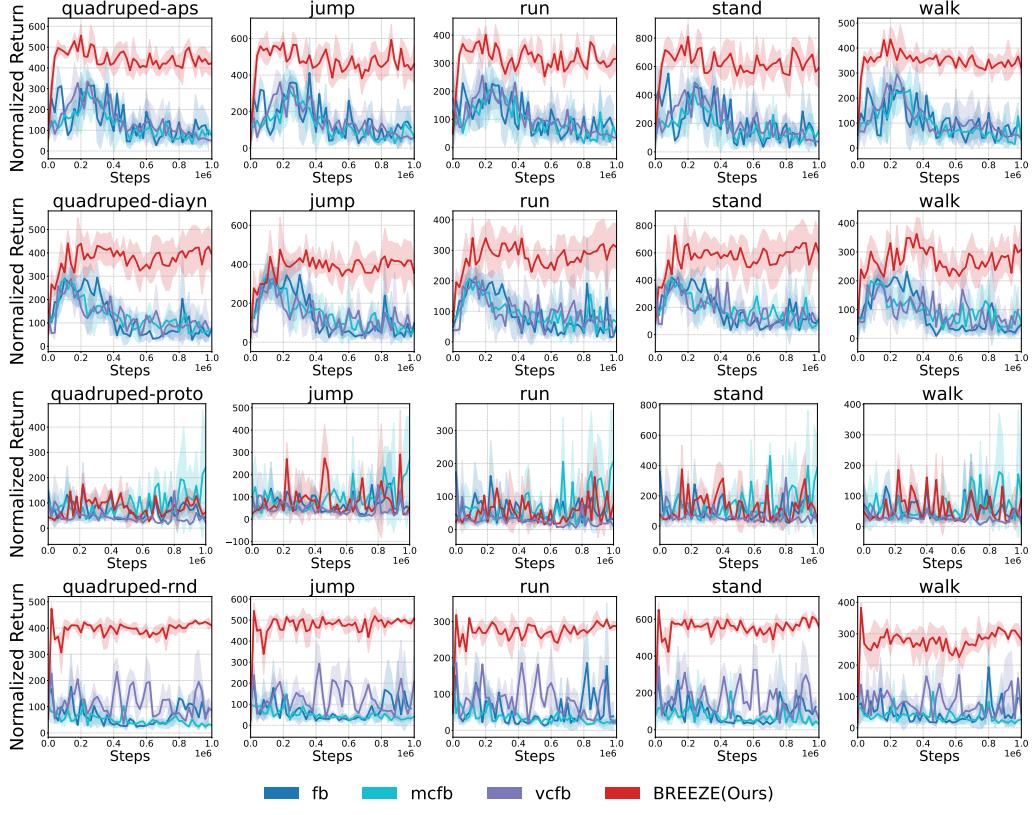


Figure 13: Curves of zero-shot performance on Quadruped domain with 100k dataset.

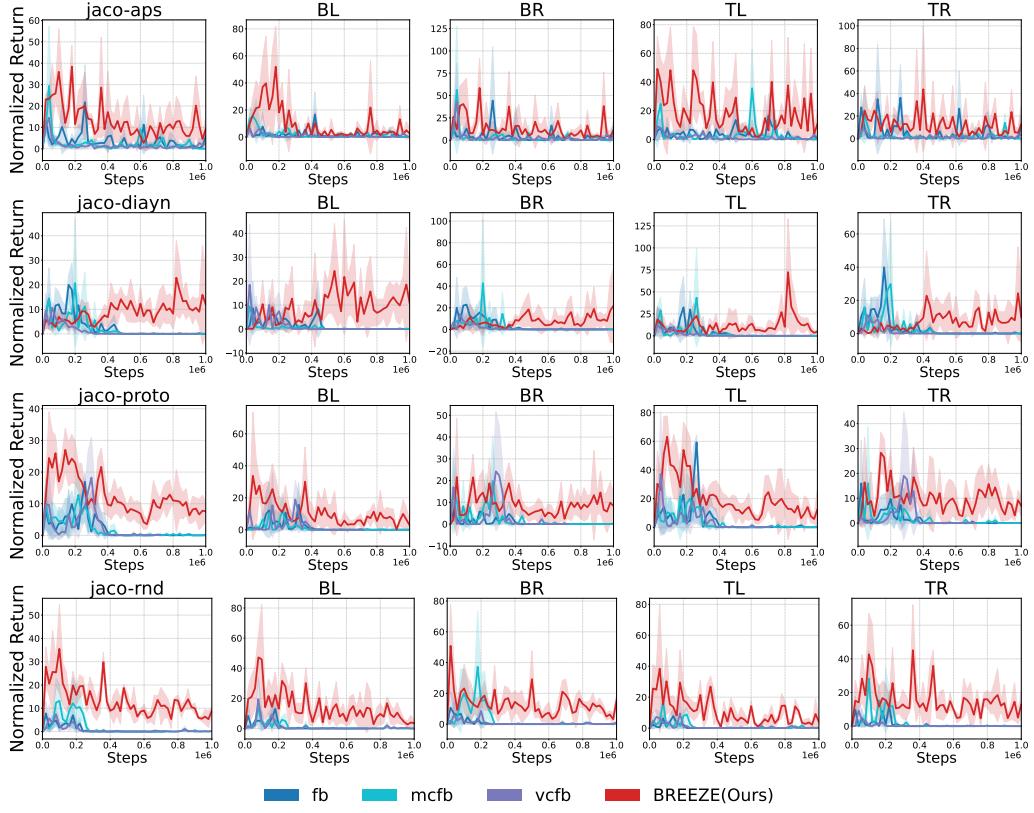


Figure 14: Curves of zero-shot performance on Jaco domain with 100k dataset.

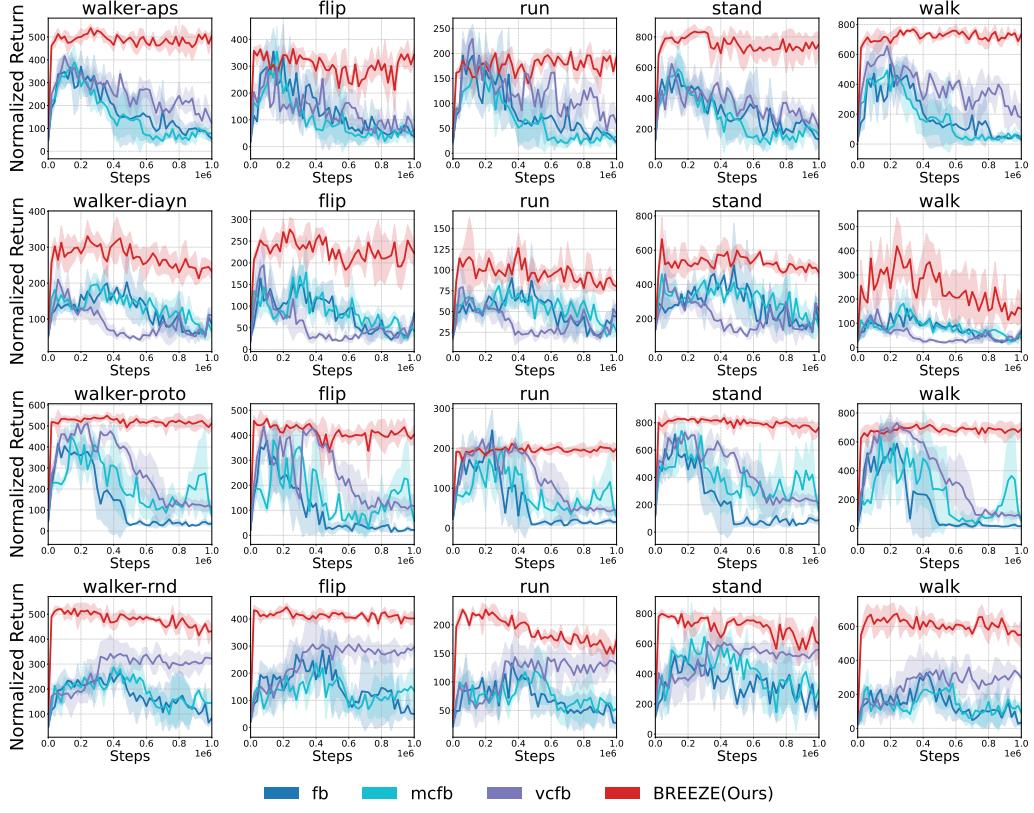


Figure 15: Curves of zero-shot performance on Walker domain with 100k dataset.

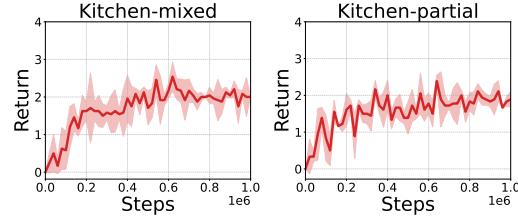


Figure 16: Curves of zero-shot performance on Kitchen domain.

D.2 Detailed Results on ExORL Benchmarks

This section presents complete experimental results on the ExORL benchmark. Table 7 summarizes performance on the full dataset, while Table 8 reports results on the 100k small-sample datasets. Evaluations were conducted every 10,000 steps during offline training. We computed the aggregate score across all tasks as the overall performance measure, and selected the highest domain-level score across 5 random seeds along with corresponding per-task results.

Table 7: Full dataset experimental results on ExORL

Dataset	Domain	Task	SF-LAP	FB	VCFB	MCFB	HILP	BREEZE
RND	Walker	walk	555 ± 221	918 ± 26	813 ± 55	842 ± 126	798 ± 29	907 ± 23
		stand	810 ± 47	834 ± 39	913 ± 29	950 ± 20	888 ± 55	938 ± 12
		run	235 ± 61	357 ± 20	376 ± 22	328 ± 20	368 ± 34	320 ± 25
		flip	465 ± 107	535 ± 29	511 ± 52	517 ± 89	608 ± 48	606 ± 22
	whole performance		516 ± 97	661 ± 10	653 ± 22	659 ± 51	665 ± 33	693 ± 16
	Jaco	reach top right	2 ± 2	75 ± 68	78 ± 63	86 ± 68	43 ± 42	108 ± 42
		reach top left	2 ± 2	4 ± 8	15 ± 16	10 ± 9	53 ± 17	68 ± 30
		reach bottom right	1 ± 0	46 ± 42	81 ± 62	56 ± 56	53 ± 41	77 ± 45
		reach bottom left	70 ± 71	4 ± 5	9 ± 10	11 ± 13	60 ± 39	84 ± 55
	whole performance		18 ± 18	32 ± 23	46 ± 35	41 ± 34	52 ± 21	84 ± 14
Quadruped	Walker	walk	206 ± 136	519 ± 35	462 ± 6	603 ± 68	629 ± 69	641 ± 64
		stand	539 ± 224	956 ± 11	883 ± 52	965 ± 5	904 ± 41	936 ± 7
		run	233 ± 136	484 ± 14	431 ± 20	486 ± 22	484 ± 20	514 ± 22
		jump	342 ± 238	722 ± 4	659 ± 53	683 ± 41	679 ± 72	808 ± 16
	whole performance		330 ± 165	671 ± 14	609 ± 26	684 ± 18	674 ± 28	725 ± 23
	Jaco	walk	272 ± 81	725 ± 118	705 ± 130	817 ± 88	802 ± 22	935 ± 23
		stand	663 ± 54	720 ± 208	567 ± 109	751 ± 65	860 ± 10	865 ± 57
		run	139 ± 25	244 ± 58	245 ± 53	316 ± 26	332 ± 26	267 ± 14
		flip	221 ± 35	380 ± 54	431 ± 78	430 ± 97	578 ± 49	482 ± 52
	whole performance		324 ± 24	517 ± 99	487 ± 75	578 ± 35	643 ± 22	637 ± 21
APS	Walker	reach top right	29 ± 22	33 ± 37	18 ± 10	28 ± 8	98 ± 44	119 ± 52
		reach top left	9 ± 15	16 ± 14	11 ± 11	15 ± 17	80 ± 28	153 ± 76
		reach bottom right	50 ± 32	33 ± 27	37 ± 46	31 ± 21	105 ± 26	70 ± 34
		reach bottom left	70 ± 72	6 ± 8	13 ± 11	15 ± 8	52 ± 20	186 ± 26
	whole performance		39 ± 26	22 ± 14	20 ± 18	22 ± 3	84 ± 16	132 ± 16
	Jaco	walk	312 ± 158	615 ± 48	541 ± 38	601 ± 90	557 ± 40	660 ± 92
		stand	796 ± 186	898 ± 57	946 ± 24	895 ± 63	934 ± 9	950 ± 12
		run	342 ± 199	471 ± 32	476 ± 10	440 ± 36	466 ± 13	472 ± 27
		jump	542 ± 126	686 ± 33	692 ± 6	699 ± 70	758 ± 21	710 ± 27
	whole performance		498 ± 160	668 ± 29	664 ± 3	659 ± 50	679 ± 14	698 ± 24
PROTO	Walker	walk	352 ± 210	889 ± 45	783 ± 165	867 ± 109	894 ± 38	923 ± 2
		stand	703 ± 155	936 ± 30	832 ± 86	845 ± 135	931 ± 22	858 ± 35
		run	202 ± 69	298 ± 27	324 ± 81	262 ± 69	387 ± 29	228 ± 26
		flip	271 ± 156	477 ± 47	503 ± 54	435 ± 143	647 ± 76	645 ± 62
	whole performance		382 ± 129	650 ± 19	611 ± 94	602 ± 112	715 ± 31	663 ± 19
	Jaco	reach top right	5 ± 9	31 ± 54	12 ± 17	37 ± 52	55 ± 23	80 ± 75
		reach top left	2 ± 2	33 ± 34	7 ± 9	14 ± 13	52 ± 28	99 ± 48
		reach bottom right	29 ± 46	19 ± 33	25 ± 36	26 ± 40	30 ± 24	91 ± 79
		reach bottom left	24 ± 41	0 ± 0	8 ± 13	2 ± 2	38 ± 33	26 ± 29
	whole performance		15 ± 14	21 ± 26	13 ± 12	20 ± 21	44 ± 19	74 ± 26
DIAYN	Walker	walk	131 ± 32	168 ± 76	136 ± 63	235 ± 144	118 ± 58	298 ± 79
		stand	293 ± 88	351 ± 263	239 ± 95	259 ± 132	329 ± 95	569 ± 55
		run	163 ± 51	125 ± 112	100 ± 45	213 ± 172	175 ± 44	283 ± 33
		jump	211 ± 77	244 ± 92	265 ± 166	169 ± 101	242 ± 70	407 ± 7
	whole performance		199 ± 10	222 ± 107	185 ± 72	219 ± 135	216 ± 54	389 ± 44
	Jaco	walk	174 ± 151	364 ± 164	329 ± 147	348 ± 174	604 ± 126	613 ± 103
		stand	558 ± 129	614 ± 146	484 ± 194	429 ± 83	696 ± 168	712 ± 66
		run	96 ± 25	103 ± 20	74 ± 26	75 ± 25	156 ± 25	152 ± 38
		flip	128 ± 24	272 ± 58	183 ± 51	222 ± 112	386 ± 76	373 ± 39
	whole performance		239 ± 79	338 ± 74	268 ± 67	268 ± 97	461 ± 64	463 ± 42
	Quadruped	reach top right	43 ± 37	17 ± 12	24 ± 16	1 ± 1	64 ± 5	76 ± 14
		reach top left	10 ± 12	18 ± 7	30 ± 17	55 ± 2	55 ± 6	58 ± 33
		reach bottom right	42 ± 42	27 ± 17	7 ± 4	1 ± 1	45 ± 26	96 ± 36
		reach bottom left	32 ± 41	28 ± 12	36 ± 12	4 ± 3	46 ± 4	82 ± 35
	whole performance		32 ± 26	22 ± 6	24 ± 3	15 ± 1	52 ± 7	78 ± 11
	Quadruped	walk	196 ± 151	475 ± 77	311 ± 47	596 ± 26	497 ± 18	482 ± 11
		stand	305 ± 270	756 ± 30	793 ± 87	865 ± 30	944 ± 4	944 ± 9
		run	139 ± 121	380 ± 22	364 ± 22	408 ± 15	469 ± 2	472 ± 5
		jump	190 ± 158	636 ± 19	576 ± 47	700 ± 18	769 ± 8	766 ± 25
	whole performance		207 ± 168	562 ± 23	511 ± 37	643 ± 14	670 ± 4	666 ± 2

Table 8: Small sample dataset experimental results on ExORL.

Dataset	Domain	Task	FB	VCFB	MCFB	BREEZE (ours)
RND	Walker	walk	300 ± 50	377 ± 18	225 ± 46	663 ± 41
		stand	368 ± 70	556 ± 44	587 ± 97	791 ± 21
		run	104 ± 11	144 ± 17	101 ± 18	224 ± 12
		flip	284 ± 91	319 ± 46	236 ± 44	421 ± 9
	Jaco	whole performance	264 ± 33	350 ± 29	287 ± 48	525 ± 13
		reach top right	9 ± 8	6 ± 7	28 ± 16	51 ± 10
	Quadruped	reach top left	2 ± 2	6 ± 5	1 ± 1	7 ± 3
		reach bottom right	10 ± 10	4 ± 4	19 ± 15	38 ± 22
		reach bottom left	8 ± 9	20 ± 12	1 ± 1	47 ± 22
		whole performance	7 ± 5	9 ± 2	13 ± 7	36 ± 5
APS	Walker	walk	125 ± 98	164 ± 45	94 ± 58	382 ± 15
		stand	268 ± 183	288 ± 73	159 ± 70	651 ± 37
		run	132 ± 83	186 ± 52	89 ± 49	318 ± 16
		jump	178 ± 130	292 ± 82	148 ± 70	544 ± 37
	Jaco	whole performance	176 ± 123	233 ± 52	123 ± 61	474 ± 21
		walk	535 ± 139	533 ± 114	449 ± 64	762 ± 25
	Quadruped	stand	447 ± 45	625 ± 52	562 ± 104	828 ± 19
		run	166 ± 73	216 ± 33	192 ± 50	200 ± 2
		flip	334 ± 121	293 ± 85	354 ± 95	365 ± 24
		whole performance	370 ± 66	416 ± 10	389 ± 77	539 ± 15
PROTO	Walker	reach top right	36 ± 29	7 ± 7	20 ± 24	21 ± 16
		reach top left	3 ± 2	7 ± 5	24 ± 19	21 ± 15
		reach bottom right	44 ± 60	42 ± 43	56 ± 71	58 ± 33
		reach bottom left	2 ± 3	0 ± 0	15 ± 4	51 ± 29
	Jaco	whole performance	21 ± 17	14 ± 13	29 ± 27	38 ± 9
		walk	299 ± 27	293 ± 26	255 ± 116	435 ± 49
	Quadruped	stand	479 ± 12	524 ± 141	456 ± 92	810 ± 85
		run	260 ± 10	232 ± 46	222 ± 101	401 ± 30
		jump	321 ± 115	356 ± 30	338 ± 179	577 ± 59
		whole performance	340 ± 29	351 ± 57	318 ± 122	556 ± 52
DIAYN	Walker	walk	522 ± 136	732 ± 38	685 ± 87	709 ± 27
		stand	608 ± 119	709 ± 36	637 ± 39	846 ± 22
		run	176 ± 58	215 ± 24	174 ± 49	210 ± 7
		flip	354 ± 67	394 ± 36	357 ± 43	447 ± 27
	Jaco	whole performance	415 ± 19	513 ± 31	463 ± 11	553 ± 18
		reach top right	7 ± 10	17 ± 19	6 ± 8	18 ± 13
	Quadruped	reach top left	59 ± 4	14 ± 12	19 ± 14	51 ± 22
		reach bottom right	0 ± 0	22 ± 22	12 ± 10	27 ± 19
		reach bottom left	0 ± 0	18 ± 11	12 ± 10	19 ± 12
		whole performance	16 ± 2	18 ± 12	12 ± 7	29 ± 12
Quadruped	Walker	walk	105 ± 63	22 ± 7	103 ± 85	129 ± 66
		stand	327 ± 180	196 ± 202	387 ± 185	59 ± 35
		run	169 ± 99	98 ± 110	206 ± 152	244 ± 173
		jump	193 ± 111	106 ± 103	265 ± 194	291 ± 47
	Jaco	whole performance	198 ± 111	106 ± 103	240 ± 134	181 ± 60
		walk	103 ± 79	114 ± 16	129 ± 38	418 ± 119
	Quadruped	stand	513 ± 245	460 ± 256	424 ± 27	524 ± 39
		run	87 ± 45	79 ± 22	63 ± 19	102 ± 17
		flip	106 ± 50	186 ± 44	169 ± 33	276 ± 28
		whole performance	202 ± 94	210 ± 81	196 ± 26	330 ± 43
DIAYN	Jaco	reach top right	18 ± 23	14 ± 13	29 ± 37	5 ± 34
		reach top left	32 ± 35	25 ± 9	9 ± 8	72 ± 58
		reach bottom right	15 ± 11	12 ± 9	42 ± 58	6 ± 29
		reach bottom left	5 ± 2	19 ± 24	0 ± 0	7 ± 26
	Walker	whole performance	17 ± 11	18 ± 5	20 ± 26	22 ± 15
		walk	158 ± 41	189 ± 73	190 ± 10	303 ± 69
	Quadruped	stand	430 ± 84	423 ± 46	408 ± 57	680 ± 117
		run	224 ± 41	214 ± 28	204 ± 28	341 ± 79
		jump	368 ± 83	327 ± 58	343 ± 63	459 ± 61
		whole performance	295 ± 46	288 ± 48	286 ± 34	446 ± 78

D.3 Empirically Evaluated M^{π_z} and Q_z on Other Tasks

We present empirical visualization results of M^{π_z} and Q_z distributions, shown in Figure 17-19. We trained FB, VCFB, and MCFB on the RND [5] dataset. Specifically, the trained B network was utilized for inferring the task latent vector z_{task} , and as the input to both M^{π_z} and Q_z during the evaluation phase.

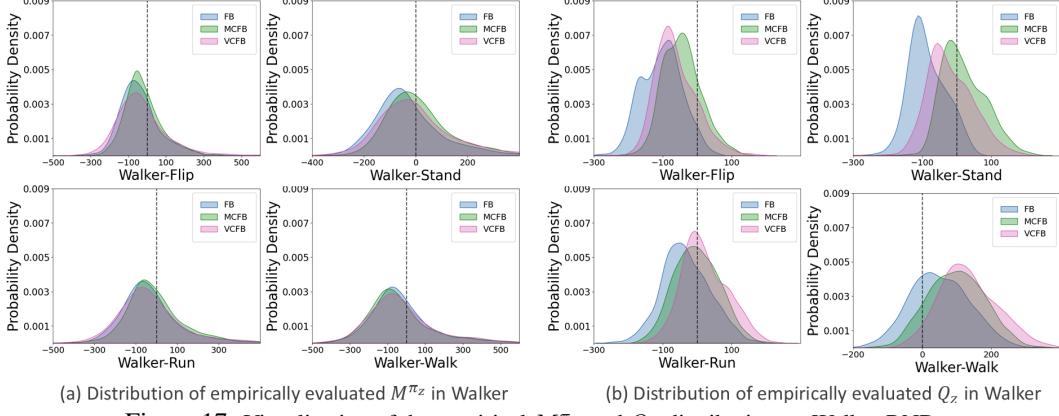


Figure 17: Visualization of the empirical M^{π_z} and Q_z distribution on Walker RND.

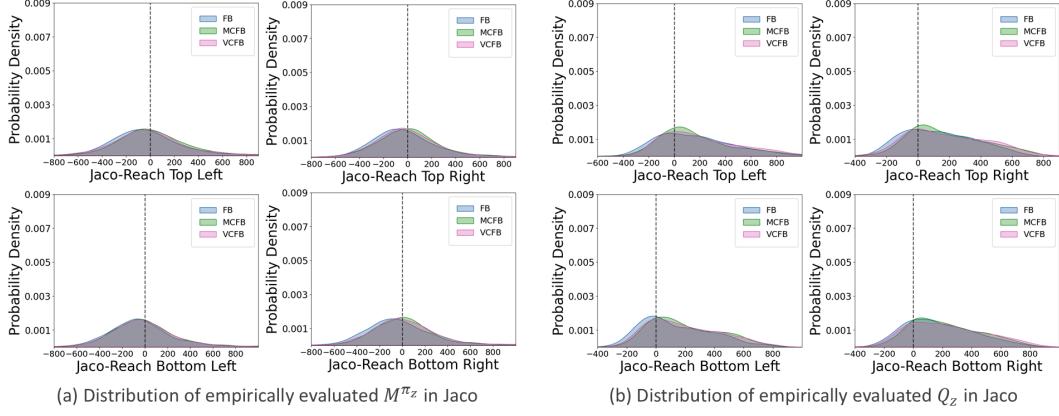


Figure 18: Visualization of the empirical M^{π_z} and Q_z distribution on Jaco RND.

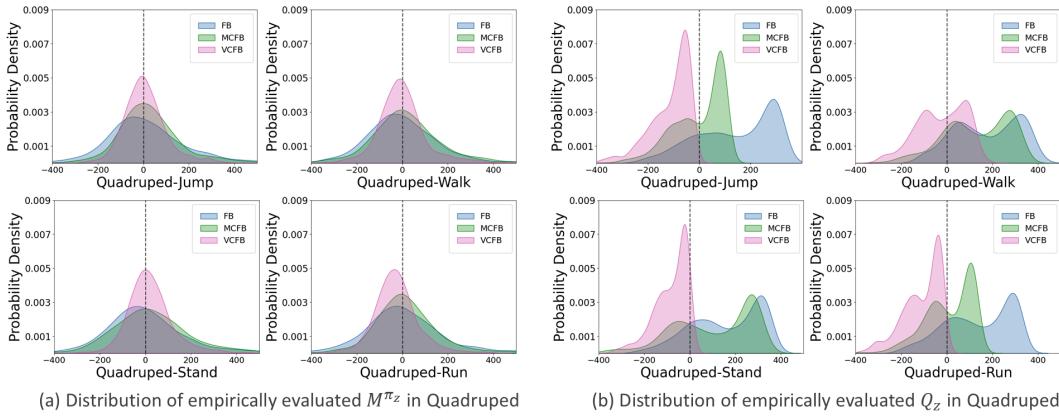


Figure 19: Visualization of the empirical M^{π_z} and Q_z distribution on Quadruped RND.

E Discussion and Limitation

E1. Hyperparameter Sensitivity and Tuning.

BREEZE introduces additional components compared to vanilla FB, resulting in more hyperparameters. While performance is generally robust, we provide the following tuning guidance:

- *Effect of components.*

Representation architectures are responsible for a high performance ceiling, as we can observe from Table 3. The diffusion policy ensures a solid performance lower bound by maintaining stable alignment with any quality dataset.

- *Greedy demand for upstream value learning.*

Training cost is mainly influenced by the action ratio ρ_a and the number of rejection sampling candidates K_{train} . While the diffusion policy provides conservatism, we find that increasing ρ_a and K_{train} further improves performance in some domains at the cost of additional computation. Near-optimal results can be achieved cost-effectively by reducing the batch size and diffusion timesteps to reallocate the computational resources to increase ρ_a and K_{train} .

E2. Computation and Performance Trade-off.

A limitation of BREEZE is its higher computational cost, attributable to the expressive representation networks and diffusion model. We consider this a reasonable trade-off given the significant improvements in stability and zero-shot performance. For a comparable and fair comparison, all reported results use a batch size of 512, and training for 1 million steps takes approximately 20 hours or less for environments with standard TD updates [50]; configurations with higher K_{train} and mixture ratios require up to 39 hours. As shown in Table 9, BREEZE typically converges within 400k steps, matching or exceeding baselines trained for 1M steps. Future work may explore framework-level optimizations, such as replacing diffusion with more efficient flow-based policies to lower costs without compromising performance.

Table 9: Computation cost v.s. performance trade-off. Aggregated scores are averaged over 5 random seeds across datasets. Experiments are conducted on a single NVIDIA A6000 GPU.

Methods	Training Steps (k)	Training Time (h)	IQM Walker	IQM Jaco	IQM Quadruped	Aggregate IQM
FB	200	0.8	480	16	303	266
	400	1.6	508	21	401	310
	1000	4.0	542	24	531	366
VCFB	200	2.4	441	18	361	273
	400	4.8	489	19	457	322
	1000	12.0	505	26	492	341
MCFB	200	2.4	473	21	347	280
	400	4.8	514	21	469	335
	1000	12.0	527	25	551	368
BREEZE (ours)	200	4.0-7.8	586	57	550	398
	400	8.0-15.6	606	75	567	416