Efficient Diffusion Policy Fine Tuning with Latent Noise Representation Bridging

Yuanchu Liang
yuanchu.liang@anu.edu.au
The Australian National University.
Maincode.

Yue Yang*
yue@maincode.com
Maincode.

An-Chi He
anchi@maincode.com
Maincode.

Hanna Kurniawati*
hanna.kurniawati@anu.edu.au
The Australian National University.

Abstract

Pre-trained diffusion and flow policies have emerged as powerful backbones for robotics, yet adapting them to novel data via reinforcement learning (RL) remains computationally expensive, sample inefficient and lacks theoretical guarantees. Prior approaches that fine-tune the initial noise bypass costly backpropagation through time but struggle with the "noise aliasing" problem: there does not exist a natural link between the noise space value and the action space value. This paper resolves these challenges by introducing the Latent Noise Linear Markov Decision Process (LNL-MDP). We derive the Bridge Equation for LNL-MDPs which eliminates the noise aliasing problem by aligning values with a shared representation between the noise and the action space. Furthermore, our LNL-MDP fine tuning framework falls under the FineTuneRL theory with near-optimal guarantee. Leveraging these insights, we propose Latent Representation Bridging for Diffusion (LARBRID), an algorithm that efficiently learns the representation to optimize the initial distribution that steers the pre-trained model to act optimally in the environment. It yields up to 2-3$\times$ performance increases on challenging control problems across a diverse suite of robotic locomotion and manipulation tasks. Code can be found at: `github.com/Maincode-Research/LARBRID.git`.

1 Introduction

Pre-trained generative models with diffusion [22, 50] and flow-based methods [30] via behavior cloning on large scale datasets have become the central backbone of robotic foundation models [8, 32, 53, 13, 27]. However, these models often fail when dealing with tasks that are not covered in the training dataset. Recently, fine tuning pre-trained models with reinforcement learning (RL) has drawn much attention in the RL and robotics community. It enables learning from online experiences to adapt to the given tasks. Meanwhile, pre-trained models serve as a good prior for RL algorithms, as training them from scratch on complex tasks is notoriously difficult. Although many existing works focus on the integration of diffusion or flow frameworks with traditional RL pipelines [21, 42, 38, 56], they often encounter heavy computations and lack theoretical justifications on the number of extra fine tuning samples needed to achieve near optimality.

While some early RL theory provides guarantees in tabular cases [5, 6, 23], there is a recent surge of interest in finding provably efficient RL structures for commonly encountered large or possibly continuous state and action spaces [25, 1, 55, 24, 14, 17, 36]. From these theoretical advances, linear Markov Decision Processes (MDPs) [25, 1, 55] have been successfully applied by subsequent works

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

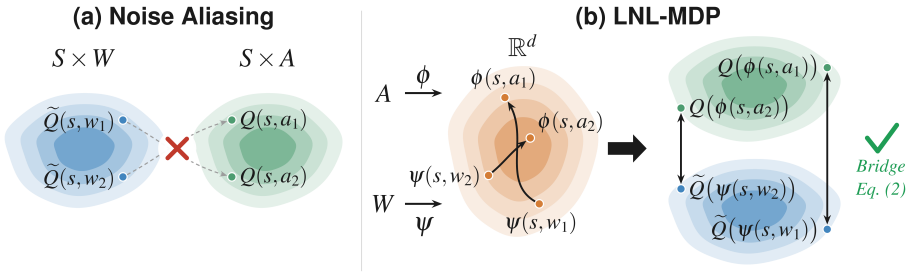

Figure 1: (a) Noise aliasing causes ambiguous associations between the action values Q and noise values $\widetilde{Q}$. (b) LNL-MDP automatically solves the aliasing problem by mapping the state space S and action space A into a shared representation space via ϕ and ψ, from which the Q values are automatically aligned.

to boost sample efficiencies in standard RL problems [44, 64, 43, 49]. Yet, little is known about whether the sample efficiency still holds analytically and practically for fine tuning pre-trained models on complex long horizon robotics problems.

This work provides affirmative answers to both the theoretical and empirical aspects of utilizing linear MDPs for fine tuning pre-trained robotics generative models. We mainly consider fine tuning only the initial distribution of pre-trained diffusion/flow models [56, 18]. Such methods can work for any model whose generative process is deterministic once the initial noise is sampled, e.g. DDIM [50] and flow [30]. The benefit with this method is that the weights of the pre-trained model are frozen during fine tuning and the model can be viewed as a black box with only API access. Gradients pass straight to the initial noise network and hence avoid heavy back propagation through time. The key challenge of this setting is the **noise aliasing problem** [56], multiple noises can produce the same action, or similar noises can produce different actions, which means that there is no natural association between the Q-values of the actions and the Q-values of the noises. The problem becomes worse in offline RL as out-of-distribution errors often require multiple (> 10) Q-networks being created independently [2, 56], amplifying the noise aliasing problem.

We address these issues analytically by endowing the environment MDP with linear structures such that components of the MDP (e.g. transition, reward, Q-values) can be represented by a single representation. Importantly, a pre-trained diffusion/flow policy operating over this linear MDP induces a Latent Noise Linear MDP (LNL-MDP) over the noise space. The corresponding latent representation is obtained by pulling back the environment representation through the pre-trained generative policy, so state-action and state-noise pairs share a common representation. This shared representation yields the Bridge Equation, which directly aligns action-space and noise-space Q-values without requiring separate critic networks, completely removes the noise-aliasing problem; see Figure 1. As a result, learning a single representation is sufficient to evaluate and optimize policies in both the original MDP and its LNL-MDP counterpart. Since much of the information about the environment is encoded into the representation, it is powerful enough to break the sub-optimality of pre-trained models and assist in finding better solutions.

By treating the pre-trained model as a virtual dataset, we can embed the LNL-MDP fine tuning problem under the theoretically well understood FineTuneRL framework [57] and inherit its near-optimal sample complexity bound. Our practical algorithm *Latent Representation Bridging for Diffusion* (LARBRID) learns the shared representations from scratch and successfully transfers the theoretical justifications to empirically observed sample efficiencies and demonstrates up to 2-3$\times$ performance increase compared to many state-of-the-art baselines across a wide range of robotic control tasks.

2 Related Work

Behavior Cloning and Pre-training of Diffusion Policies. The success of training generative models on internet scale datasets for languages and visions has inspired the robotics and control community to pursue similar directions known as behavior cloning (BC) [4, 45, 9, 63, 10, 27], where robotic agents directly learn from offline demonstration datasets. Breakthroughs in diffusion [22] and flow [30] models have inspired many subsequent BC works to generate robotic controls with them [13, 8, 53]. Although many of these works demonstrate the increasing capabilities of robots handling

diverse tasks, they often fail when encountering situations that are not well covered by the training dataset. *Our work aims to fine tune pre-trained models and demonstrate improved empirical results.*

Diffusion Fine Tuning with RL. This is a vast domain with works mainly focusing on directly optimizing the model parameters to act optimally in given environments [12, 16, 21, 48, 26, 39, 42, 38, 56]. For example, [42] proposes to optimize the first few steps of the diffusion process using PPO [47] and [39] aims to optimize the diffusion model by matching the score with the Q-function. Unlike these methods, we focus on fine tuning only the initial distribution of the diffusion or flow process, bypassing the need to perform often unstable and computationally expensive backpropagation through time [56]. Unlike [56], *we rigorously build our algorithm on the linear MDP [25] framework and show theoretical and empirical evidence of sample efficiency for fine tuning diffusion policies.*

RL under linear MDPs. Early studies of linear structures in sequential decision processes are carried out in the bandit literature [29]. Recently, linear MDPs are proposed [25, 1, 55] and shown to achieve sample efficient regret bounds. [57] later showed a near-optimal rate in the number of extra online interactions needed to find near-optimal policies while having access to an offline dataset. While the theoretical algorithms are not directly implementable, works such as [44, 64, 43, 49] use different approximation schemes to bypass the intractability and demonstrate the practicality of linear MDPs on robotic control tasks. However, *none of these works focus on demonstrating the effectiveness of linear MDPs for fine tuning generative models. It is even unclear whether the linear MDP structure can be properly defined. Our work provides positive answers to these doubts.*

3 Preliminaries

Markov Decision Processes. We consider an episodic Markov decision process [40, 57], $M = (\mathcal{S}, \mathcal{A}, \{P_h\}_{h=1}^H, \{r_h\}_{h=1}^H, H, s_1)$, with maximum horizon H, a fixed [1] initial state s_1, unknown state transition probability $\{P_h\}_{h=1}^H$ and unknown reward distribution $\{r\}_{h=1}^H$ supported on $[0,1]$. A policy is written as $\pi = (\pi_h)_{h=1}^H$. For any policy $\pi : \mathcal{S} \times [H] \to \Delta_{\mathcal{A}}$, we define the Q-value as, $Q_h^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{h'=h}^H r_{h'}(s_{h'}, a_{h'}) \mid s_h = s, a_h = a\right]$, and the corresponding V-value as $V_h^\pi(s) = \mathbb{E}_{a\sim\pi}[Q_h^\pi(s,a)]$. Let $V_0^\pi := V_1^\pi(s_1)$ denote the value of the policy π. Then the optimal policy is the one that maximizes the value $V_0^\star = V_0^{\pi^\star}$. The following definition characterizes the notion of linear MDP,

Definition 1 (Linear MDP [25, 57])**.** *There exists a known feature map $\phi^\star : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ such that $\|\phi_\star(s,a)\|_2 \le 1$ for all (s,a), and for each stage $h \in [H]$ there exist a signed vector-valued measure $\nu_h : S \to \mathbb{R}^d$ and an unknown reward vector $\theta_h^r \in \mathbb{R}^d$ satisfying $\|\nu_h(\mathcal{S})\|_2 \le \sqrt{d}$, and $\|\theta_h\|_2 \le \sqrt{d}$, such that*

$$P_h(s' \mid s,a) = \langle \phi^\star(s,a), \nu_h(s') \rangle, \qquad \mathbb{E}[r_h(s,a)] = \langle \phi^\star(s,a), \theta_h^r \rangle.$$

As shown in [25], the advantage of this linear structure is that the Q-value can be represented linearly with the feature map ϕ as,

$$Q_h^\pi(s,a) = \langle \phi^\star(s,a), \zeta_h^\pi \rangle \tag{1}$$

for some learnable weights $\{\zeta_h^\pi \in \mathbb{R}^d\}_{h=1}^H$. Linear MDP can model a wide range of non-linear dynamics when considering infinite dimensional cases (e.g. kernel embeddings), see Appendix A.

Diffusion and Flow Models. Diffusion and flow models gradually transform an easy to sample initial distribution (e.g. standard Gaussian) to a complex goal distribution $q(x_0)$ [22, 30]. Here we brief the popular Denoising Diffusion Probabilistic Model (DDPMs). Its central transform process is governed by $p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T}$, where the joint distribution can be decomposed to $p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^T p_\theta^{(t)}(x_{t-1} \mid x_t)$. The forward process samples $x \sim q(x_0)$ and gradually adds noise $\epsilon_t \sim \mathcal{N}(0, I)$ to the sample for T steps. The reverse denoising process trains a network $\epsilon_\theta^{(t)}$ to predict the noise added at time t and obtain a denoised sample via $x_{t-1} = \alpha_t(x_t - \beta_t \epsilon_\theta^{(t)}(x_t)) + \sigma_t \epsilon_t$ for coefficients $\alpha_t, \beta_t, \sigma_t$. DDPM sampling process can be formulated as stochastic differential equations [3], however, when $\sigma_t = 0$, the process becomes deterministic and boils down to the Denoising Diffusion Implicit Model (DDIM) [50]. Flow models share tight connections with diffusion, but are rooted in ordinary differential equations [30, 51]. Importantly, DDIM and flow generate deterministically once the initial noise is sampled.

[1] the initial state can be probabilistic in our empirical studies

Problem Setup. BC often augments diffusion or flow models with an extra state input s and trains the diffusion or flow model $\Phi(s, w)$ with $w \sim \mathcal{N}(0, I)$ to predict actions $a \in A$ in the dataset $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$. Our problem setup requires the sampling process of Φ to be deterministic once the input state and noise are given (e.g. DDIM or flow) and keep Φ frozen throughout the fine-tuning process. Let M denote the environment MDP Φ operates in, because of its deterministic generative process after initial noise, sampling noises w from the noise space $W \subseteq \mathbb{R}^{\dim(\mathcal{A})}$ is enough to steer the reverse denoising process with $\Phi(s, w) = a$. This induces the latent noise MDP,

Definition 2 (Latent Noise MDP [56])**.** *Given the environment MDP M and a pre-trained model Φ with weights frozen, define the latent MDP $\widetilde{M} = (\mathcal{S}, W, \{\widetilde{P}_h\}_{h=1}^H, \{\widetilde{r}_h\}_{h=1}^H, H, s_1)$ by*

$$\widetilde{P}_h(s' \mid s, w) := P_h(s' \mid s, \Phi(s, w)), \qquad \widetilde{r}_h(s, w) := r_h(s, \Phi(s, w)).$$

Our goal is to find an optimal noise policy $\pi^\star_{\mathrm{dp}} : S \to \Delta_W$ in this induced latent MDP. Such a general problem structure encounters the noise aliasing problem stated earlier and does not leverage sample efficient structures that bring extra theoretical and empirical benefits. These issues motivate the LNL-MDP formulation and the LARBRID algorithms we will next introduce.

4 Latent Representation Bridging for Diffusion Fine Tuning

This section introduces the novel LNL-MDP framework by endowing a linear structure on the environment MDP M. The Bridge Equation solves the noise aliasing problem by aligning Q-values between the action and noise spaces together via a shared representation. Our practical algorithm use an existing representational learning algorithm to learn the shared representation for which the rest of the RL fine tune pipeline is conditioned on.

4.1 Latent Noise Linear MDP

When the environment MDP M is linear (Definition.1), then the latent noise MDP $\tilde{M}$ is also linear. The following proposition establishes the Latent Noise Linear MDP (LNL-MDP).

Proposition 1 (LNL-MDP)**.** *When the original MDP M that Φ acts in satisfies the linear MDP definition given in 1. For any $w \in W$, define the pulled back latent feature $\psi^\star(s, w) := \phi^\star(s, \Phi(s, w))$, then $\widetilde{M}$ is also a linear MDP, called the Latent Noise Linear MDP (LNL-MDP), specifically, its transition and reward function can be written as,*

$$\widetilde{P}(s' \mid s, w) = \langle \psi^\star(s, w), \nu(s') \rangle, \ \mathbb{E}[\widetilde{r}(s, w)] = \langle \psi^\star(s, w), \theta_h^r \rangle$$

Proof. See Appendix C for detailed proof. □

Noise Aliasing Removal By mapping state-action and state-noise pairs to the same representation space and conditioning the Q-value on this space, we can analytically solve the noise aliasing problem. Denote the Q-value in $\widetilde{M}$ as $\widetilde{Q}^{\pi_{\mathrm{dp}}}$. For any $w \sim \pi_{\mathrm{dp}}(\cdot \mid s)$,

$$Q^\pi(s, \Phi(s, w)) = \langle \phi^\star(s, \Phi(s, w)), \zeta_h^\pi \rangle = \langle \psi^\star(s, w), \zeta_h^\pi \rangle = \widetilde{Q}^{\pi_{\mathrm{dp}}}(s, w) \tag{2}$$

Since the Q values in both M and $\widetilde{M}$ are represented by their corresponding feature $\phi^\star$ and $\psi^\star$ due to 1, but $\psi^\star$ is just a pulled back feature from $\phi^\star$, knowing $\phi^\star$ is enough to associate the noise values with the action values. Furthermore, the same Q function works for both M and $\widetilde{M}$, we no longer need to refit a separate noise Q-network against the actor Q-network as done in [56]. We hence call (2) the "Bridge" Equation and drop the superscript π in Q^π. We emphasize that LNL-MDPs and the Bridge Equation are independent of the choice of RL fine tuning pipelines, making it adaptable to different algorithms and settings, as we show soon in Section 6.

Representation Learning LNL-MDP shifts the noise aliasing problem to a representation learning problem as in practice one needs to learn $\phi^\star$ from scratch. We choose diffusion spectral representation [49] to learn $\phi^\star$, see Appendix. B. When the transition model of the LNL-MDP is parameterized as an energy-based model (EBM), the feature $\phi^\star$ can be approximated by a fully connected neural network with 1 hidden layer, $\phi_{\theta_1}(s, a) = \sigma(W_2^\top \sigma(W_1^T \xi(s, a)))$, where σ denotes any activation function

Algorithm 1 Latent Representation Bridging for Diffusion (LARBRID)

Require: Pre-trained Φ, offline data D_{off}, total steps T, parameters θ, ζ, ρ and target entropy $\bar{H}$.

1: Initialize $D = \{\}$ or $D = D_{\text{off}}$ and collect warm-up rollouts.
2: **for** $t = 1, \ldots, T$ **do**
3: Sample (s, a, s', r) from D.
4: Update ϕ_{θ_1} via the score matching objective (3).
5: Update ψ_{θ_2}: $\min_{\theta_2} \mathbb{E}_D \left[||\psi_{\theta_2}(s, w) - \phi_{\theta_1}(s, \Phi(s, w))||\right]$.
6: Update Q_ζ via the TD error defined in (4).
7: Update $\pi_{\text{dp},\rho}$: $\min_\rho \mathbb{E}_{D,w\sim\pi_{\text{dp},\rho}}[\alpha \log \pi_{\text{dp},\rho}(w \mid s) - Q(\psi_{\theta_2}(s, w))]$.
8: Update α: $\min \alpha \mathbb{E}_{D,w\sim\pi_{\text{dp},\rho}}[-\alpha \log \pi_{\text{dp},\rho}(w \mid s) - \alpha\bar{H}]$.
9: **if** online **then**
10: Sample noise $w \sim \pi_{\text{dp},\rho}(\cdot \mid s)$ and obtain $a = \Phi(s, w)$.
11: Execute a and obtain s' and r.
12: Add $D = D \cup (s, a, w, s', r)$.
13: **end if**
14: **end for**

and the Q-value can be represented as $Q_\zeta(s, a) = \langle \phi_{\theta_1}(s, a), \zeta \rangle$ because of (1). Here the score components ξ and ν of the EBM can be learned via diffusion score matching with noise schedule β,

$$l_{\text{diff}}(\xi, \nu) = \min_{\xi,\nu} \mathbb{E}_\beta \mathbb{E}(s, a, \tilde{s}', s') \left[||\tilde{s}' + \beta\xi(s, a)^\top \nabla\nu(s') - \sqrt{1-\beta}s'||\right] \tag{3}$$

Remark 1. *There are multiple empirical works [44, 64, 43] that admit the linear MDP structure. We choose the diffusion spectral representation since it admits universal approximation property with efficient learning.*

4.2 Latent Representation Bridging for Diffusion (LARBRID)

We are now ready to introduce our practical algorithm, Latent Representation Bridging for Diffusion (LARBRID) (Algorithm 1). We present LARBRID with the full generality in mind, but the algorithm can be simplified in different situations. At the beginning, we learn the feature ϕ_{θ_1} from the diffusion score matching objective given in (3). In the full general case when we do not have noise action recorded in the offline dataset and Φ can be a black box generator (e.g. API access) with no gradient passing through, we may need to match the noise feature $\psi_{\theta_2}(s, w)$ against the original feature $\phi_{\theta_1}(s, \Phi(s, w))$ with a mean squared error (MSE) objective.

Since we aim to find an optimal noise distribution π_{dp}, we naturally parameterize $\pi_{\text{dp},\rho}$ as Gaussian policies with parameter ρ and employ Soft Actor Critic (SAC) [20] to solve the exploration and exploitation problem. A caveat is that due to the non-linearity created by the entropy term in SAC, the complex optimization pipelines, we parameterize Q to have extra nonlinearities to saturate these factors, $Q_\zeta(s, a) = \zeta_2^\top \sigma(\zeta_1^\top \phi_{\theta_1}(s, a)])$ where now ζ denotes the combined weights $\zeta = \{\zeta_1, \zeta_2\}$. We then use the following TD error to fit Q_ζ against the target network $\bar{Q}_{\zeta'}$

$$\min_\zeta \mathbb{E}_{D,w'\sim\pi_{\text{dp},\rho}(\cdot|s')} \left[(Q_\zeta(\phi_{\theta_1}(s, a)) - r - \gamma(\bar{Q}_{\zeta'}(\phi_{\theta_1}(s', a')) - \alpha \log(\pi_{\text{dp},\rho}(w' \mid s'))))^2\right], \tag{4}$$

where ζ' is the parameter updated from ζ using Polyak averaging. To compute this loss (line 6 of LARBRID), we need to sample the next action a' by firstly sampling $w' \sim \pi_{\text{dp},\rho}(\cdot \mid s')$ and diffuse the noise sample via Φ to obtain $a' = \Phi(s', w')$. The expectation with respect to D means the data (s, a, s', r) is taken from the replay buffer. The actor is then updated by maximizing Q_ζ in line 7 and we also update α the temperature term according to SAC. Details of these two objectives are presented in Algorithm 1. The Bridge Equation is very flexible and allows us to work in any level of MDP that is the most convenient, meaning LARBRID can be simplified in different situations.

ψ-only Simplifications. By operating only in $\widetilde{M}$, one can replace ϕ_{θ_1} with ψ_{θ_2} in LARBRID. This has the benefit of skipping the feature matching MSE step in line 5 of LARBRID, which saves massive computation when sampling from the pre-trained model is expensive. However, such form can only handle online datasets in which each transition contains the augmented noise actions w.

ϕ-only Simplifications. Implicitly, LARBRID can work directly in the environment MDP M by only learning ϕ_{θ_1}. This has the benefit of bypassing offline feature out-of-distribution errors. The

downside is that one may need to perform backpropagation through time to optimize the pre-trained model parameters. This however can be mitigated by the modification below.

4.3 Offline RL Fine Tuning with LARBRID-I

Offline RL has a notorious data out-of-distribution error where the next action predicted by the current policy may fall outside of the available dataset. [2] proposes to use multiple critic networks (>10) for learning the Q function, creating unnecessary computation burdens [56] for RL fine tuning. Instead we exploit the Bridge Equation and the ϕ-only simplification of LARBRID. Since the Bridge Equation says the Q-values between the noise and action space are aligned, we propose to directly optimize the policy π in the action space A. Indeed, our theory says fine tuning policies in the noise or the action space are simply different sides of the same coin. Specifically, given a noise policy $\pi_W \in \Delta_w$, its pushforward via the pre-trained diffusion policy is the action level policy $\pi_A = (\Phi)_{\#}\pi_W$. Conversely, if LARBRID-I fine tuned action policy is supported on $\mathrm{Im}(\Phi)$ and Φ admits a right inverse G such that $\Phi \circ G = 1$, then $\pi_W = (G)_{\#}\pi_A$, that is fine tuning at action level also induces fine tuning at the noise level. In practice, it is difficult to enforce G to exist, however, this insight motivates us to modify LARBRID based on FQL [38], an efficient offline RL algorithm that eliminates OOD error by distilling from the BC trained flow model. We call this modification LARBRID-I ('I' for implicit) and present it in Algorithm 3. It fine tunes the BC distilled action policy from FQL using learned representation ϕ using Eq. 3, avoids back propagation through time and brings massive empirical gains.

5 Near-Optimal Fine Tuning Guarantees for LNL-MDPs

We now theoretically show that if the representation $\psi^\star$ is obtained (e.g. via Eq. 3), the rest of the fine-tuning regret is near-optimal. For a fixed state s, write $\Phi_s : \mathcal{W} \to \mathcal{A}$ for the map $\Phi_s(w) := \Phi(s, w)$. For a distribution $\mu \in \Delta\mathcal{W}$, let $(\Phi_s)_{\#}\mu \in \Delta\mathcal{A}$ denote its pushforward through Φ_s, i.e., $((\Phi_s)_{\#}\mu)(B) := \mu(\{w \in \mathcal{W} : \Phi(s, w) \in B\})$ for every measurable set $B \subseteq \mathcal{A}$. Equivalently, for every bounded measurable function $f : \mathcal{A} \to \mathbb{R}$, $\mathbb{E}_{a\sim(\Phi_s)_{\#}\mu}[f(a)] = \mathbb{E}_{w\sim\mu}[f(\Phi(s, w))]$. The theory relies on three mild assumptions listed.

Assumption 1. *The MDP that Φ is trained on satisfies the linear MDP structure defined in 1.*

Assumption 2. *For every state $s \in \mathcal{S}$, the map $w \mapsto \Phi(s, w)$ is deterministic.*

Assumption 3 (Optimal-policy realizability through the decoder)**.** . *There exists an optimal Markov policy $\pi^\star = (\pi^\star_h)_{h=1}^H$ for the original MDP M and a Markov latent policy $\mu^\star = (\mu^\star_h)_{h=1}^H$ for the latent MDP $\widetilde{M}$ such that, for every stage $h \in [H]$ and every state s reachable at stage h under $\pi^\star$, $\pi^\star_h(\cdot \mid s) = (\Phi_s)_{\#}\mu^\star_h(\cdot \mid s)$. Equivalently, for every bounded measurable $f : \mathcal{A} \to \mathbb{R}$, $\mathbb{E}_{a\sim\pi^\star_h(\cdot|s)}[f(a)] = \mathbb{E}_{w\sim\mu^\star_h(\cdot|s)}[f(\Phi(s, w))]$.*

Assumption 3 only requires the reachability of optimal trajectories to ensure that the optimal policy is preserved when bridging M to $\widetilde{M}$. It does not require full surjectivity onto all possible trajectories.

These assumptions and the Bridge Equation work together to ensure that the optimal value and the optimal policy are preserved as shown in the proposition below,

Proposition 2 (Value preservation)**.** *Suppose Assumption 2 and 3 hold, then the optimal values of the original MDP and the latent MDP are equivalent: $V_0^\star = \widetilde{V}_0^\star$.*

Proof. See Appendix D for detailed proof □

The central insight to our theory is to view the pre-trained frozen Φ as a virtual offline dataset.

Definition 3 (Warm-up rollout dataset)**.** *Execute Φ for N_{off} episodes in the real environment. In episode i and step h, sample $w_h^i \sim \mathcal{N}(0, I)$ and $a_h^i = \Phi(s_h^i, w_h^i)$,, then observe reward r_h^i and next state s_{h+1}^i to collect, $D_{\mathrm{off}} = \{(s_h^i, w_h^i, r_h^i, s_{h+1}^i)\}_{h\in[H],\, i\in[N_{\mathrm{off}}]}$. The empirical covariate generated by D_{off} is defined as, $\Lambda^h_{\mathrm{off}} := \sum_{i=1}^{N_{\mathrm{off}}} \psi^\star(s_h^i, w_h^i)\psi^\star(s_h^i, w_h^i)^\top$.*

We show in Proposition 3 that above definition indeed provides the correct offline dataset according to the FineTuneRL framework [57].

Proposition 3 (Validity of D_{off})**.** *Under Assumptions 1 and 2, the warm-up dataset D_{off} is a valid offline dataset on the latent MDP $\widetilde{M}$. More precisely: D_{off} consists of N_{off} i.i.d. trajectories of $\widetilde{M}$ generated by the policy π_{dp};*

Proof. See Appendix E for detailed proof □

Denote the expected covariate generated by playing policy π_{dp} as $\Lambda^h := \mathbb{E}_{\pi_{\text{dp}}}\left[\psi^\star(s_h, w)\psi^\star(s_h, w)^\top\right]$. The pre-trained model when rolled out approximate this covariate and serves as a valid prior as described in the following proposition.

Definition 4 (Offline-to-online Concentrability Coefficient [57])**.** *For the latent MDP $\widetilde{M}$ with offline covariance Λ^h_{off}, define the offline-to-online concentrability coefficient, with some $\epsilon > 0$,*

$$C^h_{o2o}(\Lambda^h_{\text{off}}, \varepsilon, T) := \inf_{\pi_{\text{exp}}} \max_{\pi \in \Pi} \frac{\|\psi^\star_{\pi,h}\|^2_{(T\Lambda^h_{\pi_{\text{exp}}} + \Lambda^h_{\text{off}})^{-1}}}{(\widetilde{V}^\star_0 - \widetilde{V}^\pi_0)^2 \vee \varepsilon^2},$$

where $\psi^\star_{\pi,h} := \mathbb{E}_\pi[\psi^\star(s_h, w_h)]$, $\Lambda^h_{\pi_{\text{exp}}}$ is the stage-h exploration covariance.

In the LNL-MDP context, this coefficient additionally measures the quality of Φ. Intuitively, a good Φ should contain sufficient amount of information such that minimum amount of online interactions is needed to find an optimal policy. The previous steps complete the reduction: the frozen diffusion decoder induces the LNL-MDP, and warm-up rollouts on the diffusion decoder produce a valid offline dataset on the LNL-MDP. We can therefore apply FineTuneRL [57] directly to obtain the following bound that depends on the warm up rollout dataset.

Theorem 1 (Non-asymptotic Guarantee)**.** *Under Assumptions 1, 2, and 3, suppose that we first collect D_{off} via Definition 3, there is an algorithm (FTPedel [57]) such that the number of additional online episodes needed to obtain an ε-optimal policy satisfies*

$$T_{\text{on}}(\varepsilon) = \widetilde{O}\left(\max_{h\in[H]} \min\left\{T : C^h_{o2o}(\Lambda^h_{\text{off}}, \varepsilon, T) \le \frac{1}{\bar{\beta}}\right\} \cdot H\right),$$

where $\bar{\beta} = \widetilde{\Theta}(d^2H^2)$. Moreover, any algorithm in this setting requires at least

$$T_{\text{on}}(\varepsilon) = \widetilde{\Omega}\left(\max_{h\in[H]} \min\left\{T : C^h_{o2o}(\Lambda^h_{\text{off}}, \varepsilon, T) \le \frac{1}{\bar{\beta}}\right\} \cdot \frac{1}{H}\right)$$

online episodes to return an ε-optimal policy with constant probability.

Proof. See Appendix F for detailed proof □

Theorem 1 proves that latent diffusion-policy fine-tuning is minimax-optimal in the LNL-MDP setting up to H factors. The quality of Φ is translated into the quality of the offline data it can collect. Extending the result to incorporate feature learning and SAC style optimization is beyond the scope of this work, we expanded on this in Appendix. G. The main point here is that once ψ is known, the fine tuning problem is theoretically easy to solve. Indeed, we next show that empirically this is also true.

6 Experiments

We conduct thorough experiments to understand the capability of LARBRID through online (Section 6.1), offline (Section 6.2) and offline-to-online fine tuning settings (Section 6.3) across diverse benchmarks. We compare LARBRID against multiple state-of-the-art baselines. Further experimental details can be found in Appendix H.

6.1 Online Fine Tuning

We run online fine tuning of pre-trained DDIM experiments on two standard benchmarks, Gymnasium [54] and Robomimic [33]. We compare our method against three other recent competitive baselines, DSRL [56], DPPO [42] and IDQL [21]. Some earlier baselines such as DIPO [61] and QSM [39]

are dropped, because they aim to adapt from scratch and have been shown to be less competitive than the latest baselines. We set all methods to use the same DDIM checkpoints provided by [42] and use the same hyperparameters and public code released by these methods. LARBRID can be easily implemented on top of DSRL by conditioning DSRL on the representation space $\mathbb{R}^d$ of the LNL-MDP. DSRL runs their full algorithm for all experiments except for Transport task due to heavy DDIM sampling. For fair comparisons, we also run LARBRID in full and only apply ψ-only simplification to the Transport task. The results are averaged over 4 random seeds, with their mean and standard errors displayed in Figure 2 and Figure 3.

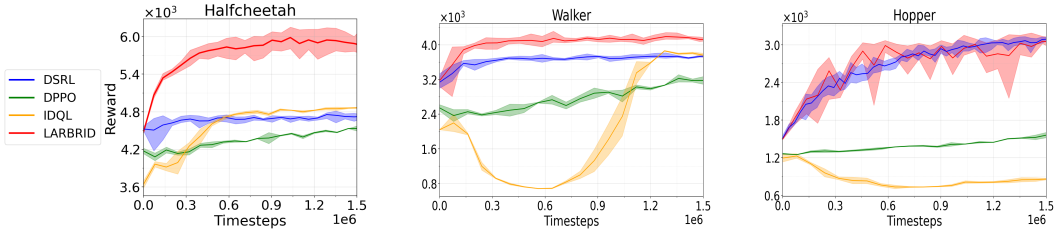

Figure 2: Online fine tuning results of LARBRID on Gymnasium [54] control tasks.

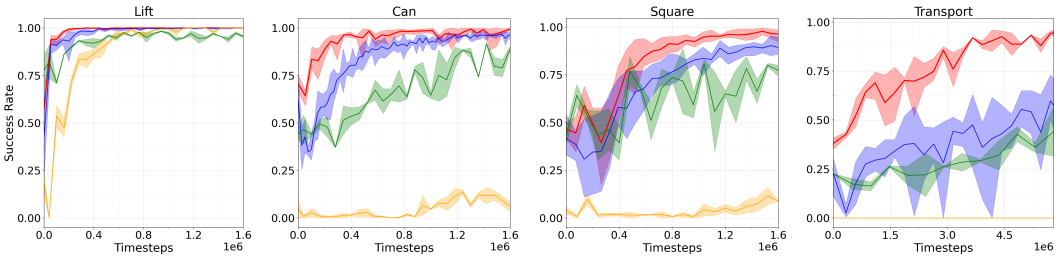

Figure 3: Online fine tuning results of LARBRID on Robomimic [33] control tasks.

Since LARBRID learns good representations fast (see Figure 5), which resolve the noise aliasing problem for free, it is able to break the sub-optimality of the pre-trained model and discover better noises to sample conditioned on the learned representation. This is shown across both the Gymnasium and Robomimic tasks. Notably for all tasks, LARBRID converges very fast with training variance at least on par with the baselines, demonstrating its sample efficiency and stability. In particular, LARBRID is able to achieve at least 21% more reward in Halfcheetah, and $\sim$70% better performance than DSRL in the Transport task, the hardest problem considered in this setting.

6.2 Offline Fine Tuning

OOD errors in offline setting is broadly discussed in Section 4.3. DSRL handles OOD issues by deploying ensembles of 10 separate Q-networks, causing heavy computational cost. Instead of resolving to offline RL tricks, we deploy LARBRID-I by directly working with the environment MDP M, bypassing the OOD errors and computational burdens of large critic ensembles. We run LARBRID-I on OGBench [37] across 10 locomotion and manipulation tasks with 4 random seeds and compare against recent offline RL algorithms, including IQL [28], REBRAC [52], IDQL [21], SRPO [11], CAC [16], FAWAC [34], FBRAC [59], IFQL [21], FQL [38] and DSRL [56]. We also include 5 manipulation tasks with visual observations and report the average performance (see Appendix H.2 for individual results). Due to high computational costs, we select 6 strong performing baselines to compare against ours for the visual input tasks. All tasks are set to their default single task variant in OGBench and each method is trained with 1M steps (500K for visual tasks). The results are displayed in Table 1 and all baseline results are copied from [56]. As also observed in the online fine tuning setting, the representation power provided by LNL-MDP allows LARBRID-I to achieve high success rates across 7 out of 10 tasks. It runs particularly well on the difficult humanoid maze problem and brings at least $1.5\times$ performance increase for the medium maze and $3\times$ performance increase in the large maze. LARBRID-I also handles visual inputs well and brings an extra 7% improvement on top of the second strongest baseline FQL.

Table 1: Offline LARBRID-I task success results on OGBench [37].
hum: humanoidmaze, med: medium, s: single, d: double, pz: puzzle.
visual: visual manipulation tasks, averaged over 5 visual tasks.

Method	ant-large	ant-giant	hum-med	hum-large	soccer	cube-s	cube-d	scene	pz-3x3	pz-4x4	visual
IQL	48 ±9	0 ±0	32 ±7	3 ±1	3 ±2	85 ±8	1 ±1	12 ±3	2 ±1	5 ±2	42 ±4
REBRAC	91 ±10	**27** ±22	16 ±9	2 ±1	0 ±0	92 ±4	7 ±3	50 ±13	2 ±1	10 ±3	60 ±2
IDQL	0 ±0	0 ±0	1 ±1	0 ±0	0 ±1	**96** ±2	16 ±10	33 ±14	0 ±0	26 ±6	–
SRPO	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	82 ±16	0 ±0	2 ±2	0 ±0	7 ±4	–
CAC	42 ±7	0 ±0	38 ±19	1 ±1	0 ±0	80 ±30	2 ±2	50 ±40	0 ±0	1 ±1	–
FAWAC	1 ±1	0 ±0	6 ±2	0 ±0	12 ±13	81 ±9	2 ±1	18 ±8	1 ±1	0 ±0	–
FBRAC	70 ±20	0 ±1	25 ±8	0 ±1	24 ±4	83 ±13	22 ±12	46 ±10	2 ±2	5 ±1	22 ±2
IFQL	24 ±17	0 ±0	69 ±19	6 ±2	16 ±9	73 ±3	9 ±5	0 ±0	0 ±0	21 ±11	50 ±5
FQL	80 ±8	4 ±5	19 ±12	7 ±6	**39** ±6	**97** ±2	36 ±6	76 ±6	**16** ±5	11 ±3	65 ±2
DSRL	40 ±29	0 ±0	34 ±20	10 ±12	28 ±9	**93** ±14	**53** ±14	**88** ±9	0 ±0	**37** ±13	–
LARBRID-I	**95** ±0	0 ±0	**96** ±1	**35** ±15	**36** ±9	**93** ±5	**51** ±2	**89** ±5	9 ±2	16 ±1	**70** ±3

6.3 Offline to Online Fine Tuning

We also test the capabilities of both LARBRID and LARBRID-I to handle offline-to-online fine tuning tasks from OGBench[37]. To obtain the offline pre-trained policy for LARBRID, we pre-train the default BC model in FQL [38] for 1M offline steps and fine tune it with LARBRID online for another 1M steps. For LARBRID-I we continue its training after 1M offline steps with another 1M online steps. We add two specific offline to online baselines, RLPD [7] and Cal-QL [35]. The results are displayed in Table2 with other baseline results copied from [38]. See Figure 7 and 8 for additional training plots. Overall, both models combined together are able to solve all benchmark problems displayed here. LARBRID is able to learn the representative features from scratch and steer the poor performing pre-trained BC model to achieve remarkably high success across multiple tasks. It failed to solve humanoid maze, likely due to the combined factors of the BC model not covering the near-optimal trajectories well and the learned feature being not representative enough. LARBRID-I on the other hand is able to find policies to solve all tasks except Puzzle-4x4 which LARBRID performs better in comparison. Empirically, this gap between LARBRID and LARBRID-I is expected. Since ψ^* and π are learned in practice, training them in the noise space or the action space may incur different optimization landscapes, causing the results to differ especially in hard tasks like humanoid and ant soccer. The different results show the flexibility of the LNL-MDP framework, as if one space is too hard to solve in practice, one can work with the other space.

Table 2: Offline-to-online fine-tuning results of LARBRID on OGBench [37].
Left number denotes offline fine tuned success rate, right number denotes online fine tuned success rate

Method	cube-double	scene	puzzle-4x4	hum-med	ant soccer
IQL	0 ±1 → 0 ±0	14 ±11 → 10 ±9	5 ±2 → 1 ±1	21 ±13 → 16 ±8	2 ±1 → 0 ±0
REBRAC	6 ±5 → 28 ±28	55 ±10 → 100 ±0	8 ±4 → 14 ±35	16 ±20 → 1 ±1	0 ±0 → 0 ±0
Cal-QL	0 ±0 → 0 ±0	1 ±2 → 50 ±53	0 ±0 → 0 ±0	0 ±0 → 0 ±0	0 ±0 → 0 ±0
RLPD	0 ±0 → 0 ±0	0 ±0 → **100** ±0	0 ±0 → **100** ±1	0 ±0 → 0 ±8	0 ±0 → 0 ±0
IFQL	12 ±9 → 40 ±5	0 ±1 → 60 ±39	23 ±6 → 19 ±33	56 ±35 → 82 ±20	26 ±15 → 39 ±10
FQL	40 ±11 → 92 ±3	82 ±11 → **100** ±1	8 ±3 → 38 ±52	12 ±7 → 22 ±12	28 ±8 → 86 ±5
LARBRID	0 ±0 → **99** ±1	0 ±0 → **100** ±0	0 ±0 → **100** ±0	0 ±0 → 0 ±0	0 ±0 → 0 ±0
LARBRID-I	51 ±2 → **98** ±2	88 ±5 → **100** ±0	16 ±1 → 68 ±47	96 ±1 → **100** ±0	36 ±9 → **99** ±1

6.4 Fine Tuning Pre-trained Robotic Foundation Models

This section further stretches the capability of LARBRID by testing whether it can be used to fine tune robotic foundation models online. We choose to fine tune π_0 [8], a 3.3B parameter VLA model that uses a VLM-based backbone to process visual and language inputs and adopt a flow policy to output control actions. We test LARBRID on three simulated benchmarks with different dynamics. The first one is `AlohaTransferCube` [65] that requires two aloha robots to cooperate, one picks up a cube and transfers it to another with high precision. The other two tasks are from the Libero

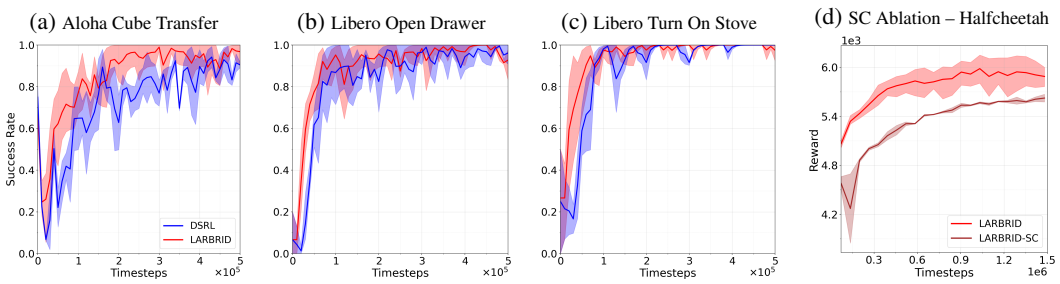

Figure 4: When fine tuning π_0 using LARBRID, its evaluated success rate converges faster than DSRL across all plots shown in (a), (b) and (c). Separate critic (SC) ablation studies carried in Halfcheetah shown in (d) confirms the benefits brought by the Bridge Equation.

[31] benchmark that requires the robot to open a drawer and turn on a stove. We compare against DSRL, the strongest online fine-tuning baseline and dropped RLPD [7] which has been shown to be dominated by DSRL. The plots are shown in Figure 4. Both LARBRID and DSRL are able to fine tune π_0 to achieve high success rate, however, LARBRID converges faster than DSRL across all tasks. In the `AlohaTransferCube` task, LARBRID is able to achieve a success rate $2\times$ higher than DSRL at early training steps (50k-80k) and improves steadily afterwards whereas DSRL's performance fluctuates more and converge to a suboptimal point.

6.5 Ablations

To isolate the effect of the Bridge Equation and demonstrate that the noise aliasing problem is indeed solved by it, we carefully design an ablation of LARBRID by creating a separate noise critic network that takes in the noise and state inputs instead of the learned representation ψ and matches this new critic against the action critic trained from the learned action-state representation ϕ. In other words, we keep the state-action representation but do not bridge it to the noise-action space such that the variant needs to re-solve the noise-aliasing problem online. We call this variant LARBRID-SC ('SC' for separate critics). We compare LARBRID-SC against LARBRID on the Halfcheetah task and the training plot is given in (d) of Figure 4. We can see that the representation from LNL-MDP still brings massive performance gains, but because LARBRID-SC now needs to handle the noise-aliasing problem, it converges much slower and to a point inferior to that of LARBRID. Additional separate critic ablation results is provided in Figure 9. Furthermore, in Appendix H.5, we also test the sensitivity of LARBRID to representation learning and matching errors by introducing weights to the representation learning loss (line 4 of Algorithm 1) and to the representation matching loss (line 5 of Algorithm 1). Overall, we found that LARBRID is quite robust to the underlying representation shifts.

7 Discussions

Limitations We suspect that when LARBRID gets stuck in local optimal features due to complex optimization landscape of hard problems like humanoid maze, the representation is not good enough for downstream fine-tuning, causing poor performances. Understanding such failure modes better can further boost the practical performances of LARBRID. It is also important to understand when a pre-trained generative model can be efficiently fine tuned by LARBRID, as this helps with providing a good prior for fine tuning. Due to the extra DDPM sampling induced by (3), our method trains slower than the method LARBRID is based on but share the same inference speed, see Figure.6. This is somewhat expected as sample efficient algorithms typically require extra computation to process more information out of each datum.

Conclusion The representation power brought by LNL-MDP allows LARBRID to handle diverse RL fine tuning settings across different tasks. By embedding the fine tuning process into the learned latent representation space, we theoretically remove the noise-aliasing problem and show a near-optimal fine tuning rate when the representation is known. Empirically, our algorithm LARBRID and LARBRID-I work well in a diverse set of tasks and achieves a performance increase of up to 2-3 $\times$ compared to other baselines. In particular, the flexibility of LNL-MDP allows one to pick an easier space to fine-tune a model when the other space is too hard. The generality of LNL-MDP means it may even benefit future RL fine tuning pipelines or problems from different domains and motivate further representation learning for RL fine tunings.

8 Acknowledgments

Yuanchu Liang is partially supported by Australia RTP scholarship and conducted this research under the AI Residency program in Maincode, Australia. Yuanchu Liang and Hanna Kurniawati are partially supported by ARC Research Hub in Intelligent Robotic Systems for Real-Time Asset Management (IH210100030). Hanna Kurniawati is partially supported by the Australasian Space Innovation Institute (fka SmartSat CRC). Yue Yang and An-Chi He are supported by Maincode.

References

[1] A. Agarwal, S. Kakade, A. Krishnamurthy, and W. Sun. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020.

[2] G. An, S. Moon, J.-H. Kim, and H. O. Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

[3] B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[5] P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

[6] M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In *International conference on machine learning*, pages 263–272. PMLR, 2017.

[7] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[8] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[9] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[11] H. Chen, C. Lu, Z. Wang, H. Su, and J. Zhu. Score regularized policy optimization through diffusion behavior. *arXiv preprint arXiv:2310.07297*, 2023.

[12] H. Chen, C. Lu, C. Ying, H. Su, and J. Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

[13] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[14] S. R. Chowdhury, A. Gopalan, and O.-A. Maillard. Reinforcement learning in parametric mdps with exponential families. In *International Conference on Artificial Intelligence and Statistics*, pages 1855–1863. PMLR, 2021.

[15] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. *Advances in neural information processing systems*, 27, 2014.

[16] Z. Ding and C. Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

[17] S. Du, S. Kakade, J. Lee, S. Lovett, G. Mahajan, W. Sun, and R. Wang. Bilinear classes: A structural framework for provable generalization in rl. In *International Conference on Machine Learning*, pages 2826–2836. PMLR, 2021.

[18] L. Eyring, S. Karthik, A. Dosovitskiy, N. Ruiz, and Z. Akata. Noise hypernetworks: Amortizing test-time compute in diffusion models. *arXiv preprint arXiv:2508.09968*, 2025.

[19] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[21] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023.

[22] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[23] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

[24] C. Jin, Q. Liu, and S. Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms. *Advances in neural information processing systems*, 34:13406–13418, 2021.

[25] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on learning theory*, pages 2137–2143. PMLR, 2020.

[26] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

[27] K. Kawaharazuka, J. Oh, J. Yamada, I. Posner, and Y. Zhu. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 2025.

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

[29] T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[30] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[31] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[32] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[33] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[34] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[35] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

[36] R. Ouhamma, D. Basu, and O. Maillard. Bilinear exponential family of mdps: frequentist regret bound with tractable exploration & planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9336–9344, 2023.

[37] S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

[38] S. Park, Q. Li, and S. Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025.

[39] M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[40] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[41] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[42] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. In *arXiv preprint arXiv:2409.00588*, 2024.

[43] T. Ren, T. Zhang, L. Lee, J. E. Gonzalez, D. Schuurmans, and B. Dai. Spectral decomposition representation for reinforcement learning. *arXiv preprint arXiv:2208.09515*, 2022.

[44] T. Ren, T. Zhang, C. Szepesvári, and B. Dai. A free lunch from the noise: Provable and practical exploration for representation learning. In *Uncertainty in Artificial Intelligence*, pages 1686–1696. PMLR, 2022.

[45] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[46] H. Sasaki and A. Hyvärinen. Neural-kernelized conditional density estimation. *arXiv preprint arXiv:1806.01754*, 2018.

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

[48] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[49] D. Shribak, C.-X. Gao, Y. Li, C. Xiao, and B. Dai. Diffusion spectral representation for reinforcement learning. *Advances in Neural Information Processing Systems*, 37:110028–110056, 2024.

[50] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

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

[52] D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023.

[53] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[54] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[55] M. Uehara, X. Zhang, and W. Sun. Representation learning for online and offline rl in low-rank mdps. *arXiv preprint arXiv:2110.04652*, 2021.

[56] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine. Steering your diffusion policy with latent space reinforcement learning. *Conference on Robot Learning (CoRL)*, 2025.

[57] A. Wagenmaker and A. Pacchiano. Leveraging offline data in online reinforcement learning. In *International Conference on Machine Learning*, pages 35300–35338. PMLR, 2023.

[58] M. J. Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

[59] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

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

[61] L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

[62] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[63] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. Ieee, 2018.

[64] T. Zhang, T. Ren, M. Yang, J. Gonzalez, D. Schuurmans, and B. Dai. Making linear mdps practical via contrastive representation learning. In *International Conference on Machine Learning*, pages 26447–26466. PMLR, 2022.

[65] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A Linear MDPs for Non-Linear Dynamics

Linear MDPs impose a linear decomposition on the transition and reward functions of a MDP. This may seem restrictive in the beginning, however, representations can be made arbitrarily non-linear and indeed many non-linear dynamics can be captured by this model when considering infinite dimensional spaces. We here provide a simple example of how this can be done. Consider the stochastic system $f : S \times A \to \mathbb{R}$,

$$s' = f(s, a) + \epsilon, \text{ for } \epsilon \sim \mathcal{N}(0, \sigma^2),$$

which can be equivalently written as,

$$P(s' \mid s, a) \propto \exp\left(-\frac{||s' - f(s, a)||_2^2}{2\sigma^2}\right).$$

Note there is no restrictions on the form f might take, a linear system can be written as $f(s, a) = As + Ba$ for some design matrices A, B. Using the reproducing property of Gaussian kernels, we can equivalently write the density above as,

$$P(s' \mid s, a) = \left\langle k(f(s, a), \cdot), (2\pi\sigma^2)^{d/2} k(s', \cdot) \right\rangle_{\mathcal{H}},$$

Here $k(x, y) = \exp \frac{-||x-y||^2}{2\sigma^2}$ is the Gaussian kernel with bandwidth σ and $\mathcal{H}$ is the reproducing Hilbert space associated with the kernel [58]. And as one can see the associated density satisfies the linear decomposition. A even more general example that uses EBMs instead of Gaussian to obtain the linear decomposition is given below.

B Diffusion Spectral Representation

Here we give a brief overview of the diffusion spectral representation [49], the core method we use to learn the representation needed for LARBRID. It relies on parameterising the transition function as an energy based model (EBM), which admits linear decomposition with respect to the Gaussian kernel and whose score components can be learned via score matching. Specifically, when the transition function $P(\cdot)$ is parameterized as an EBM,

$$P(s' \mid s, a) = \exp(\xi(s, a)^\top \nu(s') - \log Z(s, a)), Z(s, a) = \int \exp(\xi(s, a)^\top \nu(s')) \mathrm{d}s' \tag{5}$$

where $\xi(\cdot)$ and $\nu(\cdot)$ together compute the score function of the EBM and $\log Z(\cdot)$ is the log partition function. Applying random Fourier feature [41, 15], the spectral decomposition of the transition function admits,

$$P(s' \mid s, a) = \langle \phi_w(s, a), \mu_w(s') \rangle_{\mathcal{N}(w)} \tag{6}$$

such form can model arbitrary transition densities [46]. With a slight abuse of notation we also use w to denote the noises used in this decomposition, specifically for $w \sim \mathcal{N}(0, I)$, we have

$$\phi_w(s, a) = \exp(-iw^\top \xi(s, a)) \exp(||\xi(s, a)||^2/2 - \log Z(s, a)) \tag{7}$$

$$\mu_w(s) = \exp(-iw^\top \nu(s')) \exp(||\nu(s)||^2/2) \tag{8}$$

and the log partition function can be linearly represented as $Z(s, a) = \langle \exp(-iw^\top \xi(s, a) + ||\xi(s, a)||^2/2), u \rangle_{\mathcal{N}(w)}$. Hence knowing $\nu(\cdot)$ and $\xi(\cdot)$ are enough to obtain $\phi_w(\cdot)$ and $\mu_w(\cdot)$. But these features can be learned via diffusion score matching using the following objective,

$$l_{\text{diff}}(\xi, \nu) = \min_{\xi,\nu} \mathbb{E}_\beta \mathbb{E}(s, a, \tilde{s}', s') \left[||\tilde{s}' + \beta \xi(s, a)^\top \nabla \nu(s') - \sqrt{1 - \beta} s'|| \right] \tag{9}$$

where $\tilde{s}'$ is the noise corrupted next state and β is the noise schedule in standard DDPM style diffusion. Substituting $\phi_w(s, a)$ into (1), we obtain,

$$Q(s, a) = \langle \exp(-iw^T \xi(s, a) - \log \langle \exp(-iw^\top \xi(s, a)), u \rangle_{\mathcal{N}(w)}, \zeta_h^\pi \rangle) \rangle_{\mathcal{N}(w)}. \tag{10}$$

That is $\phi^\star(s, a)$ can be represented by a fully connected neural network with 1 hidden layer, $\phi_{\theta_1}(s, a) = \sigma(W_2^\top \sigma(W_1^T \xi(s, a)))$ and the Q-value can be represented as $Q_\zeta(s, a) = \langle \phi_{\theta_1}(s, a), \zeta \rangle$. We refer readers to [49] for more details.

C Proof for Proposition 1

Proof. Fix any stage $h \in [H]$, state $s \in \mathcal{S}$, latent code $w \in \mathbb{R}^{d_w}$, and next state $s' \in \mathcal{S}$. By Definition 2,

$$\widetilde{P}_h(s' \mid s, w) = P_h(s' \mid s, \Phi(s, w)).$$

Applying Assumption 1 to the action $a = \Phi(s, w)$ gives

$$P_h(s' \mid s, \Phi(s, w)) = \langle \phi_\star(s, \Phi(s, w)), \nu_h^\star(s') \rangle.$$

By the definition of $\psi^\star$,

$$\langle \phi^\star(s, \Phi(s, w)), \nu_h^\star(s') \rangle = \langle \psi^\star(s, w), \nu_h^\star(s') \rangle.$$

Hence

$$\widetilde{P}_h(s' \mid s, w) = \langle \psi^\star(s, w), \nu_h^*(s') \rangle.$$

Similarly,

$$\mathbb{E}[\widetilde{r}_h(s, w)] = \langle \psi^\star(s, w), \theta_h^r \rangle. = \langle \phi^\star(s, \Phi(s, w)), \theta_h^r \rangle = \mathbb{E}[r_h(s, \Phi(s, w))]$$

Finally, since $\Phi(w; s) \in \mathcal{A}$ and $\|\phi^\star(s, a)\|_2 \leq 1$ for all (s, a),

$$\|\psi^\star(s, w)\|_2 = \|\phi^\star(s, \Phi(s, w))\|_2 \leq 1.$$

Therefore $\widetilde{M}$ satisfies the linear-MDP definition with feature map $\psi^\star$, ν_h^* and parameters $\{\theta_h^r\}_{h=1}^H$. □

D Proof for Proposition 2

Proof. Let

$$\bar{r}_h(s,a) := \mathbb{E}[r_h(s,a)]$$

denote the expected reward in the original MDP, and define the latent expected reward by

$$\widetilde{\bar{r}}_h(s,w) := \bar{r}_h(s,\Phi(s,w)).$$

By the definition of the latent MDP,

$$\widetilde{P}_h(\cdot \mid s,w) = P_h(\cdot \mid s,\Phi(s,w)).$$

We first prove that

$$\widetilde{V}_0^\star \le V_0^\star.$$

Let $\mu = (\mu_h)_{h=1}^H$ be any latent Markov policy. Define its induced action policy π^μ in the original MDP by

$$\pi_h^\mu(\cdot \mid s) := (\Phi_s)_\# \mu_h(\cdot \mid s).$$

Equivalently, for every bounded measurable function $f : \mathcal{A} \to \mathbb{R}$,

$$\mathbb{E}_{a\sim\pi_h^\mu(\cdot|s)}[f(a)] = \mathbb{E}_{w\sim\mu_h(\cdot|s)}[f(\Phi(s,w))].$$

We claim that, for all $h \in [H+1]$ and all $s \in \mathcal{S}$,

$$\widetilde{V}_h^\mu(s) = V_h^{\pi^\mu}(s),$$

and, for all $h \in [H]$, $s \in \mathcal{S}$, and $w \in \mathcal{W}$,

$$\widetilde{Q}_h^\mu(s,w) = Q_h^{\pi^\mu}(s,\Phi(s,w)).$$

We prove this by backward induction on h. At the terminal stage $H+1$, both value functions are zero by convention.

Assume the claim holds at stage $h+1$. Then, for any s, w,

$$\begin{aligned}
\widetilde{Q}_h^\mu(s,w) &= \widetilde{\bar{r}}_h(s,w) + \mathbb{E}_{s'\sim\widetilde{P}_h(\cdot|s,w)}[\widetilde{V}_{h+1}^\mu(s')] \\
&= \bar{r}_h(s,\Phi(s,w)) + \mathbb{E}_{s'\sim P_h(\cdot|s,\Phi(s,w))}[V_{h+1}^{\pi^\mu}(s')] \\
&= Q_h^{\pi^\mu}(s,\Phi(s,w)).
\end{aligned}$$

Note this is essentially the Bridge Equation (2). The second equality uses the definition of the latent MDP and the induction hypothesis.

Now integrate both sides over $w \sim \mu_h(\cdot \mid s)$:

$$\begin{aligned}
\widetilde{V}_h^\mu(s) &= \mathbb{E}_{w\sim\mu_h(\cdot|s)}[\widetilde{Q}_h^\mu(s,w)] \\
&= \mathbb{E}_{w\sim\mu_h(\cdot|s)}[Q_h^{\pi^\mu}(s,\Phi(s,w))] \\
&= \mathbb{E}_{a\sim\pi_h^\mu(\cdot|s)}[Q_h^{\pi^\mu}(s,a)] \\
&= V_h^{\pi^\mu}(s).
\end{aligned}$$

Thus the induction is complete. In particular,

$$\widetilde{V}_0^\mu = V_0^{\pi^\mu} \le V_0^\star.$$

Since this holds for every latent policy μ,

$$\widetilde{V}_0^\star = \sup_\mu \widetilde{V}_0^\mu \le V_0^\star.$$

We now prove the reverse inequality:

$$\widetilde{V}_0^\star \ge V_0^\star.$$

By Assumption 3, there exists an optimal Markov policy $\pi^\star$ for the original MDP and a latent Markov policy $\mu^\star$ such that, for every stage h and every state s reachable under $\pi^\star$,

$$\pi_h^\star(\cdot \mid s) = (\Phi_s)_\# \mu_h^\star(\cdot \mid s).$$

Therefore, when starting from the initial state s_1, the original rollout under $\pi^\star$ and the latent rollout under $\mu^\star$, after decoding $a_h = \Phi(w_h; s_h)$, induce the same distribution over state-action-reward trajectories. Equivalently, by the same backward-induction argument above applied on the reachable trajectory support,

$$\widetilde{V}_0^{\mu^\star} = V_0^{\pi^\star} = V_0^\star.$$

Since $\widetilde{V}_0^\star$ is the supremum over all latent policies,

$$\widetilde{V}_0^\star \geq \widetilde{V}_0^{\mu^\star} = V_0^\star.$$

Combining the two inequalities gives

$$\widetilde{V}_0^\star = V_0^\star.$$

□

E Proof for Proposition 3

Proof. We prove the two claims separately.

1. validity of the dataset: Consider one transition tuple $(s_h^i, w_h^i, r_h^i, s_{h+1}^i)$ collected during the warm-up phase. By construction, at state s_h^i we first sample a noise vector w_h^i according to the Gaussian prior used by Φ. The environment action executed is

$$a_h^i = \Phi(s_h^i, w_h^i).$$

By Definition 2, the latent transition kernel is

$$\widetilde{P}_h(\cdot \mid s_h^i, w_h^i) = P_h(\cdot \mid s_h^i, \Phi(s_h^i, w_h^i)).$$

Therefore the observed next state satisfies

$$s_{h+1}^i \sim \widetilde{P}_h(\cdot \mid s_h^i, w_h^i),$$

and the observed reward satisfies

$$r_h^i = \widetilde{r}_h(s_h^i, w_h^i).$$

Hence each collected tuple is a valid latent transition. Since different episodes are generated by independent rollouts, the trajectories are i.i.d.

2. covariance concentration: Fix a step h. Define

$$X_i := \psi^\star(s_h^i, w_h^i)\psi^\star(s_h^i, w_h^i)^\top - \Lambda^h.$$

Then $\{X_i\}_{i=1}^{N_{\text{off}}}$ are i.i.d. zero-mean random matrices.

Since $\|\psi^\star(s, w)\|_2 \le 1$,

$$\left\|\psi^\star(s_h^i, w_h^i)\psi^\star(s_h^i, w_h^i)^\top\right\|_{\text{op}} = \|\psi^\star(s_h^i, w_h^i)\|_2^2 \le 1.$$

Also, by Jensen's inequality,

$$\|\Lambda^h\|_{\text{op}} = \left\|\mathbb{E}[\psi^\star(\psi^\star)^\top]\right\|_{\text{op}} \le \mathbb{E}\left\|\psi^\star(\psi^\star)^\top\right\|_{\text{op}} \le 1.$$

Therefore

$$\|X_i\|_{\text{op}} \le 2.$$

Let

$$S_h := \sum_{i=1}^{N_{\text{off}}} X_i = \Lambda_{\text{off}}^h - N_{\text{off}}\Lambda^h.$$

A standard matrix Bernstein inequality implies that for any $t > 0$,

$$\mathbb{P}(\|S_h\|_{\text{op}} \ge t) \le 2d \exp\left(-\frac{t^2/2}{N_{\text{off}} + 2t/3}\right),$$

up to absolute constants. Choosing

$$t = C\sqrt{N_{\text{off}}\, d \log(dH/\delta)}$$

for a sufficiently large universal constant C, and applying a union bound over $h \in [H]$, yields that with probability at least $1 - \delta$,

$$\left\|\Lambda_{\text{off}}^h - N_{\text{off}}\Lambda^h\right\|_{\text{op}} \le C\sqrt{N_{\text{off}}\, d \log(dH/\delta)} \qquad \text{for all } h \in [H].$$

This proves the claim. □

F Proof for Theorem 1

Proof. We verify the assumptions required by FineTuneRL. By Proposition 1, the latent problem $\widetilde{M}$ is itself a linear MDP with feature map $\psi^\star$ and dimension d. By Proposition 3, the warm-up phase yields a valid offline dataset D_{off} on $\widetilde{M}$, and its empirical covariance is exactly the quantity required in the FineTuneRL formulation. By Proposition 2, under Assumption 3, an ε-optimal policy in the latent MDP is also an ε-optimal policy in the original MDP.

Therefore the fine-tuning problem is exactly a FineTuneRL problem on $\widetilde{M}$. Applying the FineTuneRL upper-bound theorem gives

$$T_{\text{on}}(\varepsilon) = \widetilde{O}\left(\max_{h\in[H]} \min\left\{T:\, C^h_{o2o}(\Lambda^h_{\text{off}}, \varepsilon, T) \le \frac{1}{\bar{\beta}}\right\} \cdot H\right),$$

and applying the corresponding lower-bound theorem yields

$$T_{\text{on}}(\varepsilon) = \widetilde{\Omega}\left(\max_{h\in[H]} \min\left\{T:\, C^h_{o2o}(\Lambda^h_{\text{off}}, \varepsilon, T) \le \frac{1}{\bar{\beta}}\right\} \cdot \frac{1}{H}\right).$$

□

F.1 FTPedel Algorithm Used in Theorem 1

For completeness, we restate an informal version of FTPedel from FineTuneRL [57] adapted to the latent noise MDP $\widetilde{M}_\theta$.

Algorithm 2 FTPedel on the Latent Noise MDP (Informal)

Require: D_{off}, policy class Π_{lsm}, target accuracy ϵ, confidence level δ

1: Construct the empirical offline covariance Λ^h_{off} for each stage h.
2: Select an exploration policy π_{exp} according to the offline-to-online concentrability criterion.
3: Collect T additional online episodes using π_{exp}.
4: Combine the offline dataset and online samples.
5: Estimate the linear value-function parameters over the latent feature space.
6: Return a policy $\widehat{\pi}$ that is ϵ-optimal in $\widetilde{M}$.

Remark 2. *In addition to proving the validity of dataset in Proposition .3, it also proves a concentration bound between Λ^h and Λ^h_{off}. A consequence is that in the asymptotic limit, Theorem. 1 can be stated in terms of the true covariate generated by Φ, i.e., Λ^h. This statement reflects the prior information provided by the pre-trained model, however, it is less practical, as in general we do not have infinite interactions with M.*

G Theory to Real Gap

G.1 Bridge to the practical learned-feature implementation

Theorem 1 proves a minimax guarantee in the idealized setting where the structural feature map $\phi^\star$ is known. LARBRID relaxes this assumption by also learning $\phi^\star$. Instead of directly using the exact pulled-back feature

$$\psi^\star(s,w) = \phi^\star(s, \Phi(s,w)),$$

it learns an action-space feature map and a latent-space feature map and aligns them during training.

Let $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ denote the learned action-space feature map used in practice, and let $\tilde{\phi} : \mathcal{S} \times W \to \mathbb{R}^d$ denote the learned latent-space feature map. Define again the ideal pulled-back feature $\bar{\phi}(s,w) := \phi^\star(s, \Phi(s,w))$.

Our implementation cannot directly optimize through $\bar{\phi}_\theta(s,w)$, because the decoder is frozen and treated as a black box during optimization. Instead, it learns $\tilde{\phi}(s,w)$ and aligns it to the decoded action-space feature $\phi(s, \Phi_\theta(s,w))$. The next proposition shows that the latent representation error is controlled by two quantities: the decoded action-feature approximation error and the latent feature-matching error.

Let ρ be any distribution over $\Delta_{S\times W}$. In practice, ρ may be taken as the actor-induced state-latent distribution, since feature matching is enforced on actor-sampled latent codes.

Definition 5 (Representation losses)**.** *For any distribution ρ over (s,w), define*

$$\mathcal{L}_{\mathrm{act}}(\rho) := \mathbb{E}_{(s,w)\sim\rho}\Big[\|\phi(s,\Phi(s,w)) - \phi^\star(s,\Phi(s,w))\|_2^2\Big],$$

and

$$\mathcal{L}_{\mathrm{match}}(\rho) := \mathbb{E}_{(s,w)\sim\rho}\Big[\|\tilde{\phi}(s,w) - \phi(s,\Phi(s,w))\|_2^2\Big].$$

Proposition 4 (In-distribution representation error bound)**.** *For any distribution ρ,*

$$\mathbb{E}_{(s,w)\sim\rho}\Big[\|\tilde{\phi}(s,w) - \bar{\phi}(s,w)\|_2^2\Big] \le 2\,\mathcal{L}_{\mathrm{match}}(\rho) + 2\,\mathcal{L}_{\mathrm{act}}(\rho).$$

Proof. Fix any (s,w). By definition,

$$\bar{\phi}(s,w) = \phi^\star(s,\Phi(s,w)).$$

Hence

$$\tilde{\phi}(s,w) - \bar{\phi}(s,w) = \Big(\tilde{\phi}(s,w) - \phi(s,\Phi(s,w))\Big) + \Big(\phi(s,\Phi(s,w)) - \phi^\star(s,\Phi(s,w))\Big).$$

Let

$$x := \tilde{\phi}(s,w) - \phi(s,\Phi(s,w)), \qquad y := \phi(s,\Phi(s,w)) - \phi^\star(s,\Phi(s,w)).$$

Then

$$\tilde{\phi}(s,w) - \bar{\phi}(s,w) = x + y.$$

Using

$$\|x+y\|_2^2 = \|x\|_2^2 + 2\langle x,y\rangle + \|y\|_2^2 \le 2\|x\|_2^2 + 2\|y\|_2^2,$$

we obtain

$$\|\tilde{\phi}(s,w) - \bar{\phi}(s,w)\|_2^2 \le 2\|\tilde{\phi}(s,w) - \phi(s,\Phi(s,w))\|_2^2 + 2\|\phi(s,\Phi(s,w)) - \phi^\star(s,\Phi(s,w))\|_2^2.$$

Taking expectation over $(s,w)\sim\rho$ yields

$$\mathbb{E}_{(s,w)\sim\rho}\Big[\|\tilde{\phi}(s,w) - \bar{\phi}(s,w)\|_2^2\Big] \le 2\,\mathcal{L}_{\mathrm{match}}(\rho) + 2\,\mathcal{L}_{\mathrm{act}}(\rho).$$

□

Corollary 1 (Root-mean-square representation error)**.** *Under the same conditions as Proposition 4,*

$$\Big(\mathbb{E}_{(s,w)\sim\rho}\Big[\|\tilde{\phi}(s,w) - \bar{\phi}(s,w)\|_2^2\Big]\Big)^{1/2} \le \sqrt{2\,\mathcal{L}_{\mathrm{match}}(\rho) + 2\,\mathcal{L}_{\mathrm{act}}(\rho)},$$

and

$$\mathbb{E}_{(s,w)\sim\rho}\Big[\|\tilde{\phi}(s,w) - \bar{\phi}(s,w)\|_2\Big] \le \sqrt{2\,\mathcal{L}_{\mathrm{match}}(\rho) + 2\,\mathcal{L}_{\mathrm{act}}(\rho)}.$$

Proof. The first inequality follows by taking square roots on both sides of Proposition 4. The second follows from Jensen's inequality:

$$\mathbb{E}\big[\|\tilde{\phi} - \bar{\phi}\|_2\big] \le \sqrt{\mathbb{E}\big[\|\tilde{\phi} - \bar{\phi}\|_2^2\big]},$$

combined with Proposition 4. □

Assumption 4 (Uniform control of component errors on a support set)**.** *There exists a set $\Omega \subseteq \mathcal{S} \times \mathbb{W}$ such that for all $(s, w) \in \Omega$,*

$$\|\phi(s, \Phi(s, w)) - \phi^\star(s, \Phi(s, w))\|_2 \le \varepsilon_{\text{act}},$$

and

$$\|\tilde{\phi}(s, w) - \phi(s, \Phi(s, w))\|_2 \le \varepsilon_{\text{match}}.$$

Corollary 2 (Uniform representation error bound)**.** *Under Assumption 4, for all $(s, w) \in \Omega$,*

$$\|\tilde{\phi}(s, w) - \bar{\phi}(s, w)\|_2 \le \varepsilon_{\text{match}} + \varepsilon_{\text{act}}.$$

Consequently,

$$\sup_{(s,w)\in\Omega} \|\tilde{\phi}(s, w) - \bar{\phi}(s, w)\|_2 \le \varepsilon_{\text{match}} + \varepsilon_{\text{act}}.$$

Proof. Fix any $(s, w) \in \Omega$. By definition,

$$\bar{\phi}(s, w) = \phi^\star(s, \Phi(s, w)).$$

Therefore

$$\tilde{\phi}(s, w) - \bar{\phi}(s, w) = \Big(\tilde{\phi}(s, w) - \phi(s, \Phi(s, w))\Big) + \Big(\phi(s, \Phi(s, w)) - \phi^\star(s, \Phi(s, w))\Big).$$

Applying the triangle inequality,

$$\begin{aligned}\|\tilde{\phi}(s, w) - \bar{\phi}(s, w)\|_2 &\le \|\tilde{\phi}(s, w) - \phi(s, \Phi(s, w))\|_2 + \|\phi(s, \Phi(s, w)) - \phi^\star(s, \Phi(s, w))\|_2 \\ &\le \varepsilon_{\text{match}} + \varepsilon_{\text{act}},\end{aligned}$$

where the last inequality uses Assumption 4. Taking the supremum over Ω yields the result. □

Remark 3 (Interpretation of the bridge result)**.** *Proposition 4 is the natural statement for the practical algorithm, because feature matching only controls the latent representation on the sampled training distribution. Corollary 2 is stronger, but it requires stronger assumptions: namely, that both the decoded action-feature error and the latent matching error are already uniformly controlled on the support set of interest. In applications, Ω may be taken to be the support of the actor-induced state-latent distribution, or any restricted region where uniform approximation quality is plausible.*

G.2 Scope of the guarantee

At the same time, our practical implementation is more general than this idealized setting. In practice, the exact structural feature map $\phi_\star$ is not known, and the exact pulled-back feature $\psi^\star(s, w) = \phi^\star(s, \Phi(s, w))$ cannot be used directly for optimization because the decoder is frozen and treated as a black box. We therefore learn an action-space feature map ϕ and a latent-space feature map $\tilde{\phi}$ and align them during training. Proposition 4 and Corollaries 1–2 make this bridge explicit: if both the decoded action-feature approximation error and the latent feature-matching error are small, then the learned latent representation remains close to the ideal pulled-back feature required by the theory.

Accordingly, the role of Theorem 1 is to justify the latent diffusion-steering *framework*, while the role of Proposition 4 and its corollaries is to show why the learned-feature implementation can be viewed as a practical approximation to that idealized framework. We emphasize, however, that the paper does *not* claim a minimax theorem for the entire end-to-end training pipeline, including learned feature matching and SAC-style optimization. Analyzing those components jointly would require additional assumptions and new optimization theory beyond the scope of the present work.

H Experiment Details

We provide general implementation details and best practices for LARBRID here. These details and recommendations are applied to all problems. Hyperparameters can be found in Table 4.

Diffusion Spectral Representation. We found it is sufficient to use the same feature dimension as the hidden dimensions across all tasks. We use 50 DDPM sampling steps when solving the diffusion score matching problem in (3). We use variance preserving noise schedule in all trainings. We initialize the noise feature ψ_{θ_2} (if applicable) with exactly the same network used for ϕ_{θ_1}. Importantly we also keep a target copy of the learned feature ϕ_{θ_1}, and update the target parameters with Polyak averaging rate of 0.005. This slow moving target anchors the Q-network target to ensure the underlying representation space is not changing so fast that may cause learning instabilities from Q value bootstrapping. To further stablise the pipeline, we also set a slightly slower learning rate for the DiffSR features than the critic and the actor.

Critics and Actors. Due to the inherent non-linearity of the soft Q-value from SAC, the potential OOD errors from the offline dataset and the complex interactions of different networks within LARBRID, we found generally setting the critic and actor to have 2 or 3 layers of hidden dimensions are important (instead of 1) to saturate these nonlinear factors during learning. We apply the clipped double Q-network trick [19] to stablise TD computations. The learning rate for both the critic and actor is set to $3e$-4.

Visual Encoders. Instead of directly working in the image space for visual inputs, we encode the input image and embed the inputs in a lower dimensional space and run LARBRID over this lower dimensional encoded space. We use slightly different encoding schemes for OGBench visual manipulation tasks and π_0 fine tuning tasks. Details of these encoders used in offline OGBench and online π_0 fine tuning are reported below.

H.1 Details of Online Fine Tuning Tasks

Initially, we collect 1500 data samples from the environment by rolling out the pre-trained DDIM. The hidden layer dimension of the critic and actor is set to 1024. We found using two hidden layers is sufficient for all the online experiments. The diffusion spectral representation losses for Robomimic tasks are shown in Figure 5. Although we do not know the true representation $\phi^\star(\cdot)$ for each experiment, the efficiently minimized losses combined with the fast converging rewards and success rates demonstrate the effectiveness of LNL-MDPs and LARBRID. Figure 5 shows that the representation learning stage converges quickly so that downstream RL training can occur efficiently, as our theory also suggests.

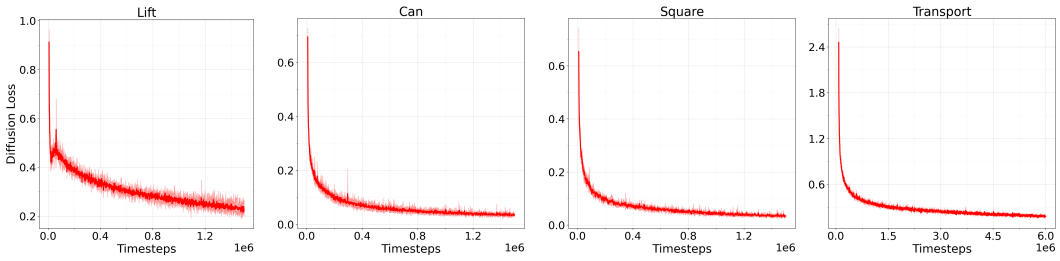

Figure 5: Spectral representation diffusion losses of LARBRID on Robomimic tasks.

H.2 Details of Offline Fine Tuning Tasks

Name Mapping. The full name of each task run in OGBench is given below.

- ant-large: `antmaze-large-navigate-singletask-v0`
- ant-giant: `antmaze-giant-navigate-singletask-v0`
- hum-med: `humanoidmaze-medium-navigate-singletask-v0`
- hum-large: `humanoidmaze-large-navigate-singletask-v0`

- soccer: `antsoccer-arena-navigate-singletask-v0`
- cube-s: `cube-single-play-singletask-v0`
- cube-d: `cube-double-play-singletask-v0`
- scene: `scene-play-singletask-v0`
- pz-3x3:`puzzle-3x3-play-singletask-v0`
- pz-4x4:`puzzle-4x4-play-singletask-v0`

Feature OOD Errors. LARBRID displayed in Algorithm 1 can theoretically work with pure offline data, we found doing so causes unstable training and fails to solve hard tasks (e.g. humanoid maze, ant soccer, puzzle 4x4) from OGBench. This is because of the feature representation OOD error. The environment MDP feature $\phi_{\theta_1}(\cdot)$ is learned on the offline dataset, which is collected from the unknown behavior policy. However, if we try to perform feature matching according to line 9 of LARBRID, we will have to generate noises from the current noise policy $\pi_{\mathrm{dp},\rho}(\cdot)$ and sample a potentially OOD action to match the features, causing wrong $\psi_{\theta_2}(\cdot)$ learned. Even worse, if the offline data does not cover the spaces well, ϕ_{θ_1} will not be representative enough for LARBRID to work. One can resort to offline RL techniques like SAC-N [2] and BRAC [60], they often come with heavy hyper-parameter tuning or large computation burden. In fact, this is how DSRL[56] solves their problem with 10 critic ensembles.

Instead, we leverage the Bridge Equation to work with only the environment MDP M, since finding an optimal actor in M is equivalent to finding an optimal noise actor in $\widetilde{M}$. This avoids the aforementioned feature OOD error, however, we will have to perform backpropagation through time to fine tune the pre-trained model. Luckily, FQL [38] has solved this problem by simultaneously learning a behavior flow policy via behavior cloning and a one-step distilled policy whose objective is to stay close to the BC policy while maximizing the critic value. LARBRID-I builds on top of this idea by adding an extra representation learning stage and condition the FQL algorithm on the learned feature space (Algorithm 3). All FQL parameters are set according to their paper with the only exception that we lower the number of hidden dimensions in the critic and actor from 4 to 3. The spectral representation learning part is set to learned $\phi_{\theta_1}(\cdot)$ with the same hyper-parameters and network architecture as before. There is no extra hyper-parameter tuning needed.

Algorithm 3 Latent Representation Bridging for Diffusion - Implicit (LARBRID-I)

Require: Offline dataset $\mathcal{D}$, total steps T, representation parameters θ_1, BC flow vector field parameters θ, one-step policy parameters ω, critic parameters ζ, discount factor γ, distillation weight α.

1: **for** $t = 1, \ldots, T$ **do**
2: Sample batch (s, a, s', r) from $\mathcal{D}$.
// Update Representation ϕ_{θ_1}:
3: $\min_{\theta_1} \mathbb{E}_\beta \mathbb{E}_{(s,a,\tilde{s}',s')} \left[||\tilde{s}' + \beta \phi_{\theta_1}(s,a)^\top \nabla \nu(s') - \sqrt{1-\beta} s'||_2 \right]$
// Update BC Flow Policy μ_θ:
4: Sample $x^0 \sim \mathcal{N}(0, I_d)$ and $t \sim \mathrm{Unif}([0,1])$.
5: Set $x^1 = a$ and interpolate $x^t = (1-t)x^0 + tx^1$.
6: $\min_\theta \mathbb{E} \left[||v_\theta(t, s, x^t) - (x^1 - x^0)||_2^2 \right]$
// Update Critic ζ:
7: Sample $z \sim \mathcal{N}(0, I_d)$ and obtain next action via one-step policy $a' = \mu_\omega(s', z)$.
8: $\min_\zeta \mathbb{E}_\mathcal{D} \left[\left(Q_\zeta(\phi_{\theta_1}(s,a)) - r - \gamma \bar{Q}_{\zeta'}(\phi_{\theta_1}(s', a')) \right)^2 \right]$
// Update Actor (One-Step Policy) ω:
9: Sample $z \sim \mathcal{N}(0, I_d)$ and obtain current action $a^\pi = \mu_\omega(s, z)$.
10: $\min_\omega \mathbb{E}_\mathcal{D} \left[-Q_\zeta(\phi_{\theta_1}(s, a^\pi)) + \alpha ||a^\pi - \mu_\theta(s, z)||_2^2 \right]$
11: **end for**

Visual Manipulation Tasks We select 5 classic manipulation tasks from the OGBench environment and test our method's capability of handling visual inputs.

Similar to [49], we use the CNN autoencoder structure in [62] to learn a latent 1D embedding for each image by training the autoencoder with a standard image reconstruction loss. To handle

partial observations, we concatenate 3 image latent embeddings and use such concatenated vector as observation inputs to LARBRID. We run a warm up stage for the autoencoder after the initial rollout so that the DDPM can obtain meaningful inputs at the beginning. The rest of the training pipeline is kept the same.

Due to the high computational costs, we select five strongest performing baselines listed in [38] to compare against.

Table 3: Visual Manipulation Tasks from OGBench

Task	IQL	ReBRAC	FBRAC	IFQL	FQL	LARBRID-I
visual-cube-single-play-singletask-task1-v0	70 ± 12	$\mathbf{83 \pm 6}$	55 ± 8	49 ± 7	$\mathbf{81 \pm 12}$	$\mathbf{83 \pm 8}$
visual-cube-double-play-singletask-task1-v0	$\mathbf{34 \pm 23}$	4 ± 4	6 ± 2	8 ± 6	21 ± 11	$\mathbf{32 \pm 11}$
visual-scene-play-singletask-task1-v0	$\mathbf{97 \pm 2}$	$\mathbf{98 \pm 4}$	46 ± 4	86 ± 10	$\mathbf{98 \pm 3}$	$\mathbf{97 \pm 3}$
visual-puzzle-3x3-play-singletask-task1-v0	7 ± 15	88 ± 4	7 ± 2	$\mathbf{100 \pm 0}$	94 ± 1	87 ± 7
visual-puzzle-4x4-play-singletask-task1-v0	0 ± 0	26 ± 6	0 ± 0	8 ± 15	33 ± 6	$\mathbf{49 \pm 16}$

Computation Times. We report the training time of LARBRID-I against FQL on a single NVIDIA 4090 in Figure 6. LARBRID-I runs $2.5\times$ slower when compared to FQL due to the extra DDPM sampling (50 sampling steps) induced when performing diffusion score matching during learning. During inference, the same actor is queried for both methods and hence they share the same inference time.

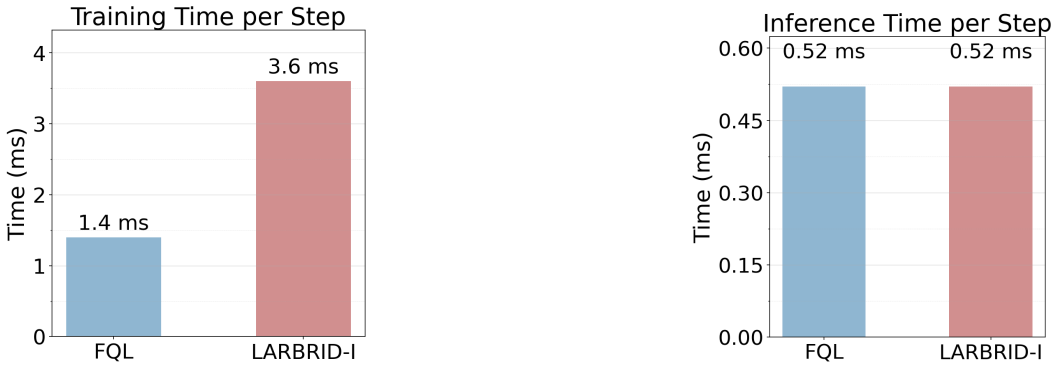

Figure 6: Training and inference time comparisons between LARBRID-I and FQL.

H.3 Details of Offline-to-online Fine Tuning Tasks

The training evaluation plots of LARBRID and LARBRID-I are displayed in Figure 8 and 7 respectively. LARBRID uses the pre-trained BC model from FQL (and offline dataset) and fine tune it online with 1M steps. LARBRID-I is trained offline first with 1M steps and then online fine tuned for another 1M steps.

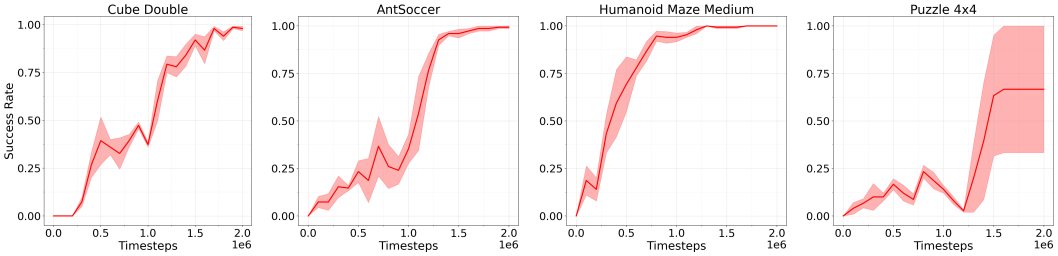

Figure 7: Training evaluation plots of LARBRID-I. The first 1M steps is pure offline training and 1M - 2M steps are trained online.

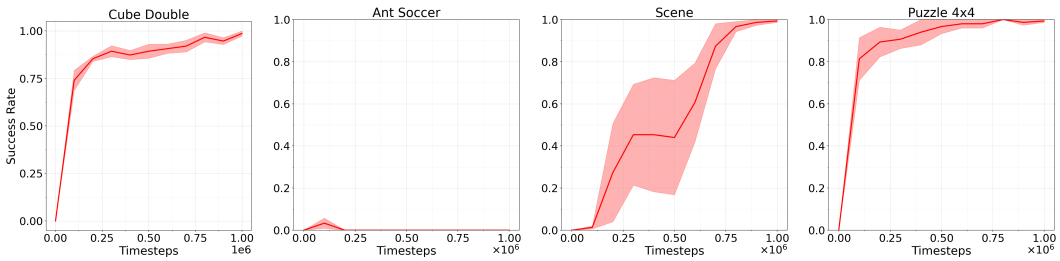

Figure 8: Training evaluation plots of LARBRID. It trains the BC model from FQL with 1M online steps.

H.4 Details of Fine Tuning Foundation Models

We use the public checkpoint at `s3://openpi-assets/checkpoints/pi0libero` for the Libero tasks and the checkpoint at `s3://openpi-assets/checkpoints/pi0alohasim` for the `AlohaTransferCube` task. Both Libero tasks are pulled from the Libero-90 pool with task ID 20 for 'turn on stove' and task ID 7 for 'open drawer'. We use the language instruction 'open the top drawer of the cabinet' for task 7 and the instruction 'turn on the stove' for task 20. To ensure fairness in comparison, we follow the same design as DSRL for the shared components. We train LARBRID on a single step noise and during execution we repeat the same sample noise across all action chunks to reduce the challenge created from high dimensional action chunks. Meanwhile, we use 10 critics with mean aggregation to stabilize the TD errors, however, we only use a single DDPM to learn the representations. Unlike the visual manipulation tasks in OGBench, here we only use a CNN encoder to encode the image observation to a lower dimensional latent space. This image embedding is concatenated with the robot proprioceptive state inputs and fed to the DDPM loss. Hence, the image encoder is trained using the DDPM loss. To reduce the computation burden, we deploy LARBRID-ψ instead of LARBRID to learn representations straightaway from the noise space. We found it is important to use spatial softmax and group normalisation for the image encoder output, as better task relevant information is kept.

H.5 More Ablation Studies

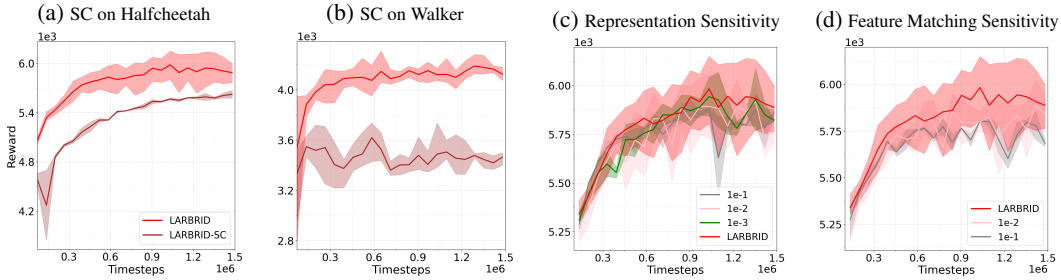

Figure 9: Ablations on the effect of the Bridge Equation using LARBRID-SC are shown in (a) and (b). Sensitivity tests over representation loss and feature matching loss are shown in (c) and (d) respectively

The separate critic ablation studies mentioned in the main text are shown in (a) and (b) of Figure 9. Additionally, we further perform a sensitivity analysis on the learned representation ϕ from LARBRID. We add $\{0.1, 0.01, 0.001, 1\}$ weights to the representation loss Eq. 3 while keeping everything else the same as LARBRID. Note 1 loss weight refers to the original LARBRID. All four variants produce roughly the same training curves as LARBRID as shown in Figure 9. We also perform sensitivity analysis on the MSE feature matching loss (line 5 of LARBRID) across $\{0.1, 0.01, 1\}$ loss weightings. As shown in Figure 9, there is no obvious drop in performance across all tests, hinting that our algorithm is quite robust to the fluctuations and errors from the learned representation and feature matching. LARBRID does not need to work with the restrictive settings studied in the theoretical analysis.

I Hyperparameters

We summarize the hyperparameters used in the table below. Since our algorithm is general, we found sticking to the same parameters for the shared components with the strong baselines works well for LARBRID and LARBRID-I. We list specific hyperparameters for each experiment below. If we do not mention the specific experiment, it means the hyperparameter works across all experiments. We use A_d to denote the action dimension.

Table 4: Hyperparameters

Hyperparameters	**Value**
representation learning rates	1e-4
other learning rates	3e-4
Target update rate (τ)	5e-3
Target entropy $\bar{H}$ (online)	0
Target entropy $\bar{H}$ (π_0)	$A_d/2$
Critic and actor #. of hidden layers	2
Critic and actor hidden layer dimension	512
ϕ_{θ_1} #. of hidden layers	2
ϕ_{θ_1} hidden dimension	512
ψ_{θ_2} #. of hidden layers	2
ψ_{θ_2} hidden dimension	512
Feature dimension (d)	512
#. of critics (online + offline)	2
#. of critics (π_0)	10
Batch size	256
DDPM Sampling Steps	50
Flow Steps (OGBench + π_0)	10
CNN Layers (visual tasks only)	(32, 32, 32, 32)
Image Embed Dimension	128
Action Chunk Size (online)	4 (8 for transport)
Action Chunk Size (offline)	1
Action Chunk Size (π_0)	50
Gradient Steps per Update (online + π_0)	20
Gradient Steps per Update (offline)	1
Image Observation Size (visual)	64×64
Discount Factor (online)	0.99
Discount Factor (π_0)	0.999
Q Aggregation (online + offline)	Min
Q Aggregation (π_0)	Mean

For LARBRID-I, the hyperparameters of the BC flow model, discount factors, and the behavior regularization coefficient α follow exactly the same as those recommended by [38]. Meanwhile, we found using the same dimension for the feature, critic and actors work well for fine tuning π_0, but we use the same hyperparameters as DSRL with image embed dimension 50 and layer dimension 128 across feature and hidden layers for fairness of comparisons.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our theoretical results and empirical results directly reflect this.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: It is discussed in the Discussion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide full proofs and further materials in the appendix to backup everything stated in the paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All algorithms are detailed in the main content or in the appendix. Best practices and hyperparameters are all given in the appendix. We will also release our code after publication.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release our code after publication, not during the review stage.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All hyperparameters, experiment settings are reported in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Each experiment is run with 4 random seeds and the variances and means are reported and compared with other baselines.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, we state the resources and time needed in the Experiment and Appendix section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have followed the code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: No societal work is performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: no such risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: References and citations are given.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Pseudoscope is given in full with detailed explanations.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: no such work is performed.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not consider any crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: No LLM is involved with the core method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.