LASER: Latent Space Adjoint Matching for Support-Constrained Entropy-Regularized Offline RL

Songyuan Zhang[1] **Oswin So**[1] **Eric Yang Yu**[1] **Matthew Cleaveland**[2]
Peter Crowley-Dolen[2] **Chuchu Fan**[1]
[1]MIT [2]MIT Lincoln Laboratory
{szhang21,oswinso,eyyu,chuchu}@mit.edu
{matthew.cleaveland,Peter.Crowley-Dolen}@ll.mit.edu

Abstract

While offline reinforcement learning (RL) enables policy optimization from static datasets without costly online interaction, it remains bottlenecked by the risk of executing out-of-distribution (OOD) actions. Recent approaches mitigate this by learning a behavior-cloning policy through flow matching and then performing RL within its constrained latent space. However, naïvely optimizing the latent policy can easily cause the policy to collapse into a brittle mode or exploit sharp artifacts of the learned critic. In this work, we find that entropy regularization is essential in latent-space RL for addressing these challenges. We introduce LASER, a novel offline RL algorithm that applies latent-space adjoint matching to achieve entropy-regularized latent-space RL with expressive flow policies while avoiding backpropagation through time. Through comprehensive experiments on 40 challenging OGBench tasks with varying dataset qualities, we show that LASER achieves state-of-the-art performance. Notably, LASER uses fixed method-specific hyperparameters across all tasks and outperforms the evaluated baselines, including those with task- and dataset-specific tuning, which highlights the robust applicability of LASER. Project website: https://mit-realm.github.io/laser/.

1 Introduction

Offline reinforcement learning (RL) aims to learn policies that improve upon the behavior policy using only a static dataset generated by that behavior policy, without further environment interaction [1]. This creates a central tension: improvement requires deviating from the behavior policy while choosing actions where the dataset provides reliable supervision. For a fixed state, actions outside the dataset support may receive spuriously high predicted values, which can be exploited by policy optimization to yield out-of-distribution (OOD) actions that perform poorly when deployed [1].

A major line of work regularizes a statistical distance between the learned and behavior policies to mitigate OOD actions [2, 3, 4, 5, 6]. However, depending on the distance used, these methods can be either too restrictive and produce an overly conservative policy (e.g., KL divergence), or not restrictive enough and still produce OOD actions (e.g., Wasserstein distance, MMD distance) [7].

This has motivated a more direct alternative: enforce **support constraints** on the policy extraction process to guarantee that the actions produced lie within the support of the action distribution of the behavior policy [8, 7]. In particular, these methods learn a flow-based generative model using flow matching [9, 10, 11] to imitate the behavior policy, resulting in a compact latent space that maps to the behavior policy. Assuming the decoder maps the latent prior's support into the behavior policy's support, any latent policy restricted to that support produces in-support actions, while allowing their

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

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

probabilities to differ from those of the behavior policy. Policy optimization performed on this latent space, either via gradient descent (as in [7]) or SAC [12] (as in [8]), yields a policy that satisfies support constraints by construction.

In this work, we argue that support constraints are only part of the story. Once policy optimization is restricted to the support, the central difficulty becomes policy extraction: how should one reliably select high-value actions from the support without collapsing to brittle modes or exploiting sharp artifacts of the learned critic? Even when actions remain in support, the critic may be sharp, irregular, or locally overestimated, and a low-entropy extraction procedure can collapse onto brittle modes.

We study **entropy regularization** as an explicit objective for support-constrained flow policy optimization. While entropy regularization is well studied in online RL for its role in improving exploration [13, 12] and robustness [13, 14], its role in offline RL, and in particular in support-constrained policy extraction, has not been isolated and made explicit. We argue that entropy regularization is also crucial in offline RL, not only in discouraging premature collapse of the policy to local minima, but also in smoothing the policy optimization landscape by regularizing the critic [15].

To this end, we propose **L**atent Space **A**djoint Matching for **S**upport-Constrained **E**ntropy-**R**egularized Offline RL (**LASER**), a framework for entropy-regularized support-constrained policy extraction with latent-space flow policies. LASER builds on the latent support parameterization used in recent flow-based offline RL methods [7, 8], but replaces implicit or architecture-dependent regularization with an explicit entropy-regularized policy improvement objective. The main technical challenge is that computing and optimizing the entropy of a flow policy in latent space is nontrivial. We address this using **adjoint matching** [16], which enables entropy-regularized optimization of expressive flow policies *in latent space* while preserving the support constraints induced by the learned latent space.

We make the following contributions:

1. We identify entropy regularization as a central ingredient for stable support-constrained policy extraction with flow-matching policies in offline RL.
2. We reinterpret recent support-constrained flow policy methods, including DSRL and ReFORM, through the lens of entropy regularization.
3. We introduce LASER, an algorithm for entropy-regularized offline policy improvement with latent-space flow policies that preserves dataset-support constraints while enabling smoother and more reliable policy extraction.
4. Extensive experiments demonstrate that LASER, *using a fixed set of method-specific hyperparameters*, achieves the strongest performance profiles among the evaluated flow-policy baselines.

2 Preliminaries

2.1 Offline RL

We use $\Delta(\mathcal{X})$ to denote the set of probability distributions over $\mathcal{X}$, and gray notation for placeholder variables. A Markov Decision Process (MDP) is defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \rho_0, P, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution $\rho_0 \in \Delta(\mathcal{S})$, transition dynamics function $P(s'|s,a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and discount factor $\gamma \in [0,1]$. Given a dataset of trajectories $\mathcal{D}$ generated by some behavior policy $\pi_\beta(a|s) : \mathcal{S} \to \Delta(\mathcal{A})$, offline RL aims to learn a policy $\pi_{\mathsf{A}}(a|s) : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return $R(\pi_{\mathsf{A}}) = \mathbb{E}_{\tau\sim\rho^{\pi_{\mathsf{A}}}(\tau)}\left[\sum_{h=0}^{H}\gamma^h r(s_h,a_h)\right]$, where $\rho^{\pi_{\mathsf{A}}}(\tau) = \rho_0(s_0)\pi_{\mathsf{A}}(a_0|s_0)P(s_1|s_0,a_0)\cdots\pi_{\mathsf{A}}(a_H|s_H)$. Note that sampling trajectories during training is not allowed in the offline RL setting.

The OOD Problem. A primary challenge in offline RL is handling OOD actions [1]. Standard actor-critic frameworks evaluate a policy π_{A} by estimating its state-action value function (Q-function) $Q(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, representing the expected discounted return:

$$Q^{\pi_{\mathsf{A}}}(s,a) = \mathbb{E}\left[\sum_{h=0}^{H}\gamma^h r(s_h,a_h) \;\middle|\; s_0 = s, a_0 = a, a_h \sim \pi_{\mathsf{A}}(s_h), \forall h \geq 1\right]. \tag{1}$$

To learn $Q^{\pi_{\mathsf{A}}}$, fitted Q-evaluation is typically used to minimize the temporal difference (TD) error:

$$L(\phi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D},a'\sim\pi_{\mathsf{A}}(s')}\left[\left(r(s,a) + \gamma Q_{\hat{\phi}}^{\pi_{\mathsf{A}}}(s',a') - Q_{\phi}^{\pi_{\mathsf{A}}}(s,a)\right)^2\right], \tag{2}$$

where $Q_{\hat{\phi}}^{\pi_{\mathsf{A}}}$ is a target network often updated via Polyak averaging [17]. However, because the dataset $\mathcal{D}$ is fixed, querying the target network with unfamiliar actions $a' \sim \pi_{\mathsf{A}}(s')$ can yield unreliable and potentially large Q-values. Consequently, the policy may erroneously optimize toward these OOD actions [1]. To address this issue, the learned policy should satisfy the **support constraint** [5]:

$$\operatorname{supp}(\pi_{\mathsf{A}}(\cdot|s)) \subseteq \operatorname{supp}(\pi_\beta(\cdot|s)), \quad \forall s, \tag{3}$$

where $\operatorname{supp}(p) \coloneqq \{x \mid p(x) > 0\}$ denotes the support of a distribution p. To achieve this, many prior offline RL algorithms regularize the statistical distance $D(\pi_{\mathsf{A}} \parallel \pi_\beta)$ between the learned policy π_{A} and the behavior policy π_β, resulting in the following objective for policy updates:

$$\min_{\pi_{\mathsf{A}}} \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_{\mathsf{A}}(s)}\left[-Q_{\phi}^{\pi_{\mathsf{A}}}(s,a) + \alpha D(\pi_{\mathsf{A}} \parallel \pi_\beta)\right], \tag{4}$$

where α is a hyperparameter for the regularization weight. However, depending on the statistical distance D, these approaches may fail to prevent OOD actions or may limit the improvement of π_{A} over the behavior policy π_β [7]. The weight α also requires task- and dataset-specific tuning [1, 18].

2.2 Mitigating OOD Actions with Latent Space RL

To mitigate the limitations of statistical distance regularization, recent works propose learning policies in a *latent space* $\mathcal{L}$ [19, 8, 7]. Specifically, a **decoder** $\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}}(z) : \mathcal{L}\times\mathcal{S}\to\mathcal{A}$ is learned that maps samples $z \sim q_{\mathsf{L}}$ drawn from the latent-space prior distribution $q_{\mathsf{L}}(z) \in \Delta(\mathcal{L})$ to the dataset distribution $\pi_\beta(\cdot|s)$, i.e., the pushforward of the prior q_{L} through the decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$ approximates the behavior policy $\pi_\beta(\cdot|s)$. Note that the decoder is *deterministic* for a given state and latent input, and the stochasticity of the action-space policy comes from sampling the latent input. Then, instead of learning a policy $\pi_{\mathsf{A}}(\cdot|s)$ in the action space $\mathcal{A}$, we can learn a **latent-space policy** $\pi_{\mathsf{L}}(z|s) : \mathcal{S}\to\Delta(\mathcal{L})$ which induces a full policy in the action space via the pushforward through the decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$, i.e.,

$$\pi_{\mathsf{A}}(\cdot|s) \coloneqq (\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}})_\sharp \pi_{\mathsf{L}}(\cdot|s), \tag{5}$$

where $(\cdot)_\sharp$ is the pushforward operator. This parameterization (5) transforms the difficult action-space support constraint (3) under the complex behavior policy π_β into a simple latent-space constraint:

$$\operatorname{supp}(\pi_{\mathsf{L}}(\cdot|s)) \subseteq \operatorname{supp}(q_{\mathsf{L}}). \tag{6}$$

Since q_{L} is chosen to have simple support (e.g., a uniform distribution on a d-ball [7]), this constraint is much easier to enforce, thereby decoupling OOD action avoidance from policy optimization.

We can thus define the resulting policy optimization problem in the latent space $\mathcal{L}$ induced by the decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$. For a Q-function $Q_{\phi}^{\pi_{\mathsf{A}}}$ in the action space (1), this induces a corresponding **latent-space Q-function** $Q_{\phi}^{\pi_{\mathsf{L}}}(s,z) : \mathcal{S}\times\mathcal{L}\to\mathbb{R}$ as the pullback of $Q_{\phi}^{\pi_{\mathsf{A}}}$ through the decoder, i.e.,

$$Q_{\phi}^{\pi_{\mathsf{L}}}(s,z) = Q_{\phi}^{\pi_{\mathsf{A}}}(s, \mu_{\theta_{\mathsf{L}\to\mathsf{A}}}(z;s)). \tag{7}$$

As long as the latent-space support constraint (6) is satisfied (e.g., with $\tanh$ squashing, clipping or projection), *any* off-policy optimization algorithm, such as DDPG or TD3 in [19], SAC in [8], or even plain gradient ascent in [7], can then be used to optimize the latent policy π_{L} *without* needing to worry about OOD actions.

Learning expressive decoders with flow matching. A key component of latent-space RL is the decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$. A poorly learned decoder may output actions outside the dataset support and defeat the purpose of latent-space RL, or be too simple to capture the behavior policy and limit the performance of the final policy. Recent works use flow matching [9, 10, 11] to learn $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$ due to its ability to model complex multimodal behavior distributions [8, 7]. Flow matching learns a time-dependent velocity field $v_{\theta_{\mathsf{L}\to\mathsf{A}}}(z,t;s) : \mathcal{L}\times[0,1]\times\mathcal{S}\to\mathcal{L}$, parameterized by $\theta_{\mathsf{L}\to\mathsf{A}}$, that transports an easy-to-sample latent distribution $q_{\mathsf{L}} \in \Delta(\mathcal{L})$ (e.g., standard Gaussian $\mathcal{N}(0, I^d)$) to the pushforward action distribution $(\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}})_\sharp q_{\mathsf{L}} \approx \pi_\beta(\cdot|s)$ over the time interval $t \in [0,1]$, using the flow-matching loss:

$$L_{\mathsf{L}\to\mathsf{A}}(\theta_{\mathsf{L}\to\mathsf{A}}) = \mathbb{E}_{(s,a)\sim\mathcal{D},z\sim q_{\mathsf{L}},t\sim\mathcal{U}[0,1]}\left[\|v_{\theta_{\mathsf{L}\to\mathsf{A}}}(x_t,t;s) - (a-z)\|^2\right], \quad x_t = (1-t)z + ta. \tag{8}$$

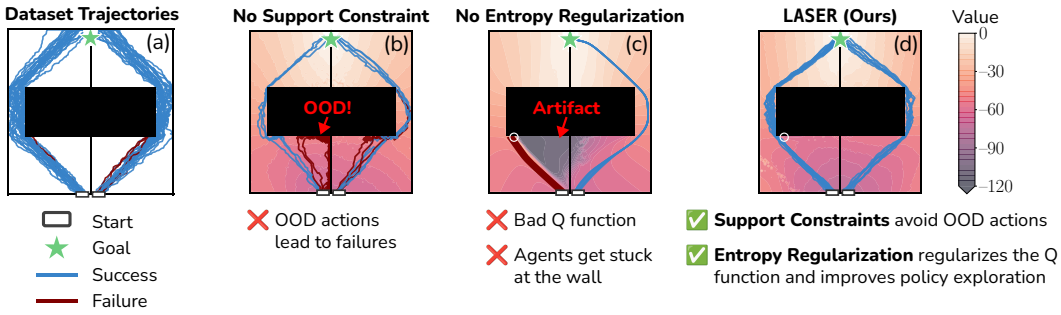

Figure 1: **Both support constraints and entropy regularization are essential.** Without support constraints, the policy generates OOD actions that have unreliable Q-values and thus fail. Without entropy regularization, the Q-function exhibits strange artifacts and the agents get stuck at the wall, resulting in failures on the left side. By combining support constraints and entropy regularization, LASER avoids OOD actions, regularizes the Q-function, and encourages policy exploration, resulting in a successful policy.

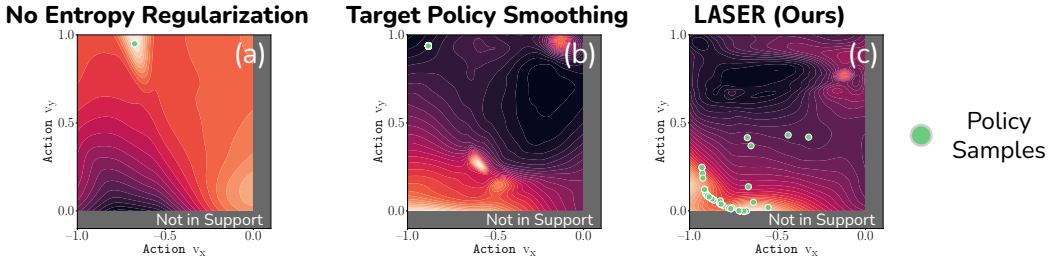

Figure 2: **Q-function and policy are poorly learned without entropy regularization.** We plot the learned Q-functions (lighter colors denote higher value) at the state marked by the white circle in Figure 1c. Without entropy regularization, the policy becomes nearly deterministic and clusters at a *false* maximum which causes the agents to move up and get stuck at the wall (Figure 1c). Adding noise to the policy in TD learning (target policy smoothing [20]) fixes the Q-function landscape, but the policy remains nearly deterministic and is suboptimal. With entropy regularization, the policy samples spread out more and yield a better regularized Q-function, enabling the agents to move around the wall and reach the target (Figure 1d).

The corresponding BC **flow map** $\psi_{\theta_{\mathsf{L}\to\mathsf{A}}}(z,t;s) : \mathcal{L}\times[0,1]\times\mathcal{S}\to\mathcal{L}$ for the velocity field $v_{\theta_{\mathsf{L}\to\mathsf{A}}}$ can be found as the solution to the ODE with initial condition z at $t=0$ and velocity field $v_{\theta_{\mathsf{L}\to\mathsf{A}}}$, i.e.,

$$\psi_{\theta_{\mathsf{L}\to\mathsf{A}}}(z,t;s) = z + \int_0^t v_{\theta_{\mathsf{L}\to\mathsf{A}}}(\psi_{\theta_{\mathsf{L}\to\mathsf{A}}}(z,\tau;s),\tau;s)\,\mathrm{d}\tau. \tag{9}$$

The decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$ is then defined as the BC flow map at time $t=1$, i.e., $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}(z;s) \coloneqq \psi_{\theta_{\mathsf{L}\to\mathsf{A}}}(z,1;s)$.

3 Latent Space Adjoint Matching for Support-Constrained Entropy-Regularized Offline RL (LASER)

3.1 Entropy Regularization is Essential for Latent Space RL

Given that latent-space RL methods claim to solve the OOD problem, one might be tempted to conclude that directly performing policy optimization in the latent space with support constraints results in a strong policy. However, we find this to be false. We illustrate this in a simple 2D reach-avoid environment (see Appendix E for details). Choosing π_{L} to be a flow policy as in [7], i.e., for a base distribution q_{B} and velocity field $v_{\theta_{\mathsf{B}\to\mathsf{L}}}$, and corresponding flow map $\psi_{\theta_{\mathsf{B}\to\mathsf{L}}}$,

$$\pi_{\mathsf{L}}(\cdot|s) = (\mu^s_{\theta_{\mathsf{B}\to\mathsf{L}}})_\sharp q_{\mathsf{B}}, \quad \mu^s_{\theta_{\mathsf{B}\to\mathsf{L}}}(w) \coloneqq \psi^s_{\theta_{\mathsf{B}\to\mathsf{L}}}(w,1), \tag{10}$$

we find that naïvely training π_{L} to maximize $Q^{\pi_{\mathsf{L}}}_\phi$ with reparametrized gradients and enforcing the latent-space support constraint (6) via $\tanh$ squashing (Figure 1c) yields a policy that causes the agent to become stuck at the wall and fail to reach the target (Figure 1). Moreover, the value function

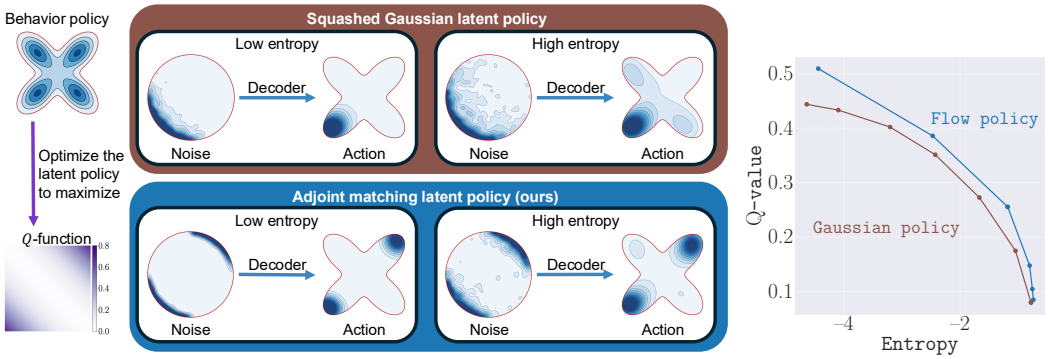

Figure 3: **Flow latent policy + adjoint matching Pareto-dominates a squashed Gaussian latent policy.** For every entropy and Q-value achieved by the Gaussian latent policy, there exists an operating point achievable by the flow latent policy that has both higher entropy and higher Q-value.

is poorly learned and exhibits artifacts. We explore this further by visualizing the learned Q-function and policy π_{A} at the state right before the agent gets stuck at the wall (Figure 2a). Surprisingly, the learned policy is maximizing the learned Q-function and is not stuck in a local optimum despite the policy failing to reach the target. Using target policy smoothing (TPS) [20] to smooth the learned Q-function by adding noise to the target action a' in the TD learning target (2) fixes the Q-function landscape (Figure 2b), but the resulting policy is now suboptimal. This is a problem of the policy having insufficient exploration[1], which can be mitigated by increasing policy entropy [12].

This motivates the use of an **explicit entropy regularization term** to obtain a policy with higher entropy, which not only helps with policy optimization via exploration, but also obviates the need for TPS in TD learning and yields a better regularized Q-function. Namely, for weight $\alpha > 0$, we consider the following entropy-regularized objective to train the latent-space policy π_{L}:

$$L_{\mathsf{B}\to\mathsf{L}}(\theta_{\mathsf{B}\to\mathsf{L}}) = -\mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{w\sim q_{\mathsf{B}}}\left[Q_\phi^{\pi_{\mathsf{L}}}(s, \mu_{\theta_{\mathsf{B}\to\mathsf{L}}}(w; s))\right] + \alpha\mathcal{H}(\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}(\cdot; s))\right]. \tag{11}$$

The entropy term can also be optimized using reparametrized gradients by computing the log probability under the flow map $\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}$ using the continuity equation [9] to obtain

$$\mathcal{H}(\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}(\cdot; s)) = \mathbb{E}_{w\sim q_{\mathsf{B}}}\left[-\log q_{\mathsf{B}}(w) + \int_0^1 \nabla_{x_t} \cdot v_{\theta_{\mathsf{B}\to\mathsf{L}}}(x_t, t; s)\,\mathrm{d}t\right], \quad x_t = \psi_{\theta_{\mathsf{B}\to\mathsf{L}}}(w, t; s). \tag{12}$$

We discuss a more efficient method that does not require backpropagation through the flow map in the next subsection. Adding explicit entropy regularization improves both the Q-function and the policy (Figure 2c), resulting in a policy that successfully guides the agent around the wall to the target in the simple illustrative environment (Figure 1d).

Remark 3.1. Examining recent latent-space RL methods, we find that their policy extraction does include an entropy regularization term despite its importance not being made explicit in their papers. This is done either explicitly, as in DSRL [8], which uses a Gaussian latent policy and an SAC-style objective [12], or implicitly as in ReFORM [7], where the geometric reflection term prevents latent collapse (see Appendix C.1). However, using an expressive flow policy for the latent-space policy and explicitly considering entropy regularization enables our method to empirically Pareto-dominate Gaussian policies (Figure 3) and maximize the Q-function better than other implicit entropy regularization schemes (Appendix C.1). A comparison of methods with no, implicit, and explicit entropy regularization across 20 tasks in more complex manipulation environments further supports the benefit of explicit regularization (Appendix F.5).

Why use a flow policy for π_{L}? While a Gaussian latent-space policy significantly simplifies entropy-regularized optimization, we motivate our use of a flow model with two arguments. (1) A Gaussian π_{L} produces unimodal latents, which restricts the ability of the full policy in action space to capture multimodality [7]. (2) While high-entropy policies facilitate robust Q-function learning and help avoid local minima, they often yield lower Q-values. By being more expressive, flow policies empirically Pareto-dominate Gaussian policies on the performance-entropy tradeoff (Figure 3).

[1]Here, exploration refers to sampling diverse in-support actions during policy optimization and critic updates.

3.2 Efficient Entropy-Regularized Policy Optimization via Adjoint Matching

Efficient entropy estimation for expressive policies remains challenging [21, 22]. While (11) can be used as-is to train the latent policy π_{L} with reparametrized gradients, computing the gradient for this flow parameterization requires backpropagating through the flow map $\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}$ (10) and the entropy (12), both of which are solutions to ODEs and require backpropagation through time (BPTT), which is computationally expensive and prone to numerical instability [23, 24, 25].

To circumvent these computational bottlenecks, we adopt an alternative approach. Instead of optimizing the loss directly via BPTT, we derive the optimal latent-space policy $\pi_{\mathsf{L}}{}^\star(\cdot|s) = (\mu^s_{\theta_{\mathsf{B}\to\mathsf{L}}})^\star_\sharp q_{\mathsf{B}}$ that minimizes the entropy-regularized policy loss (11) in the typical case where the support of the latent-space prior distribution q_{L} has finite Lebesgue measure (e.g., a d-ball).

Lemma 3.2. *Fix a state $s \in \mathcal{S}$ and parameters of the critic and the decoder. Suppose $\mathrm{supp}(q_L)$ has finite positive Lebesgue measure, and the Q-function $Q^{\pi_L}_\phi(s,\cdot)$ is measurable and has a finite upper bound on $\mathrm{supp}(q_L)$. Then, among latent densities supported on $\mathrm{supp}(q_L)$, the optimal distribution $\pi_L{}^*$ that minimizes the loss* (11) *is given by the following Q-tilted distribution:*

$$\pi_L{}^\star(\cdot|s) \propto q_L^{\mathrm{unif}}(\cdot)\exp\left(Q^{\pi_L}_\phi(s,\cdot)/\alpha\right), \tag{13}$$

where $q_L^{\mathrm{unif}} = \mathcal{U}(\mathrm{supp}(q_L))$ is the uniform distribution on the support of q_L.

This optimal latent distribution has the Gibbs form used in maximum-entropy RL [12, 21], here restricted to the latent support. See Appendix B for the proof. Note that this is a statewise, fixed-critic optimum and it does not establish convergence of the joint actor-critic updates. With Lemma 3.2, we can optimize (11) by learning to sample from the optimal distribution $\pi_{\mathsf{L}}{}^\star$, which can be done efficiently *without BPTT* using **adjoint matching** [16].

Namely, suppose both π_{L} and $q_{\mathsf{L}}^{\mathrm{unif}}$ are flow models with a common base distribution q_{B}, i.e., (10) holds for π_{L}, and for flow map $\psi^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}$ generated by velocity field $v^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}$,

$$q_{\mathsf{L}}^{\mathrm{unif}} = (\mu^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}})_\sharp q_{\mathsf{B}}, \quad \mu^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}(w) = \psi^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}(w,1), \tag{14}$$

and let $p_v(\{w_t\}_{t\in[0,1]}) \in \Delta(\mathcal{B}^{[0,1]})$ denote the trajectory law of the following memoryless stochastic differential equation (SDE) with control v and $w_0 \sim q_{\mathsf{B}}$:

$$\mathrm{d}w_t = (2\,v(w_t,t;s) - w_t/t)\,dt + \sigma_t\,\mathrm{d}B_t, \tag{15}$$

where $\sigma_t = \sigma_0\sqrt{2(1-t)/t}$ is the memoryless noise schedule and B_t is standard Brownian motion. We fix the state s and suppress it in p_v. For the Gaussian base $q_{\mathsf{B}} = \mathcal{N}(0,\sigma_0^2 I)$ used here, the reference SDE with $v = v^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}$ shares the time marginals of the reference flow ODE in the exact flow-matching formulation, including the terminal marginal $q_{\mathsf{L}}^{\mathrm{unif}}$. Moreover, consider the tilted distribution over *SDE trajectories* $p_{v^\star}$, defined as

$$p_{v^\star}(\{w_t\}_{t\in[0,1]}) \propto p_{v^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}}(\{w_t\}_{t\in[0,1]})\exp\left(Q^{\pi_{\mathsf{L}}}_\phi(s,w_1)/\alpha\right), \tag{16}$$

where (13) now corresponds to the marginal of $p_{v^\star}$ at $t=1$. Then, $p_{v^\star}$ can be learned without BPTT by repeatedly optimizing a regression loss using a "special estimate" of $v^\star$ as the regression target. Specifically, the optimal control $v^\star$ realizing (16) solves the following stochastic optimal control (SOC) problem with quadratic control cost and terminal cost given by the negative Q-function [16]:

$$\min_v \mathbb{E}_{s\sim\mathcal{D},\{w_t\}}\left[\int_0^1 \frac{2}{\sigma_t^2}\left\|v(w_t,t;s) - v^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}(w_t,t)\right\|^2 dt - Q^{\pi_{\mathsf{L}}}_\phi(s,w_1)/\alpha\right]. \tag{17}$$

Then, given a rollout $\{w_t\}_{t\in[0,1]}$ sampled with control v, we can construct the following estimate $\hat{v}^\star$ of the optimal control $v^\star$:

$$\hat{v}^\star(w_t,t;s) = v^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}(w_t,t) - \sigma_t^2\tilde{g}_t/2, \tag{18}$$

where $\tilde{g}_t$ is the "lean" adjoint state computed by integrating the reverse ODE using $\{w_t\}_{t\in[0,1]}$

$$d\tilde{g}_t = -\left(\nabla_{w_t}\left[2v^{\mathrm{base}}_{\mathsf{B}\to\mathsf{L}}(w_t,t) - w_t/t\right]\right)^\top \tilde{g}_t dt, \quad \tilde{g}_1 = -\nabla_{w_1}Q^{\pi_{\mathsf{L}}}_\phi(s,w_1)/\alpha. \tag{19}$$

In implementation, we substitute the target critic $Q^{\pi_{\mathsf{L}}}_{\hat{\phi}}$ for $Q^{\pi_{\mathsf{L}}}_\phi$ in these policy-improvement expressions (Appendix D). Using the estimate $\hat{v}^\star$ as the regression target in the following regression loss then

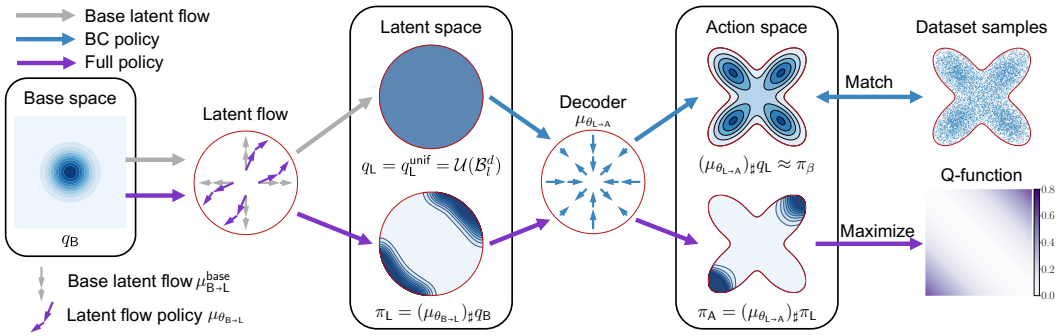

Figure 4: **LASER Algorithm.** (→) We train a flow-matching decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$ that maps samples from the uniform distribution in the d-ball $q_\mathsf{L} = q_\mathsf{L}^{\text{unif}} = \mathcal{U}(\mathcal{B}_l^d)$ in *latent space* to the dataset actions in *action space*. (→) We then learn a base latent flow $\mu^{\text{base}}_{\mathsf{B}\to\mathsf{L}}$ that maps the prior distribution q_B in the base space to the uniform distribution in the latent space $q_\mathsf{L}^{\text{unif}}$. (→) The base latent flow is then fine-tuned via adjoint matching, which results in a latent flow policy $\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}$ that generates the final action distribution $\pi_\mathsf{A} = (\mu_{\theta_{\mathsf{L}\to\mathsf{A}}})_\sharp(\mu_{\theta_{\mathsf{B}\to\mathsf{L}}})_\sharp q_\mathsf{B}$.

results in a unique fixed point at the optimal control $v^\star$ [16], where we replace the current velocity field v with the trainable velocity field $v_{\theta_{\mathsf{B}\to\mathsf{L}}}$ to be optimized:

$$L_{\text{AM}}(\theta_{\mathsf{B}\to\mathsf{L}}) = \mathbb{E}_{s\sim\mathcal{D},\{w_t\}}\left[\frac{1}{2}\int_0^1 \left\| \frac{2}{\sigma_t}\left(v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_t,t;s) - \hat{v}^\star(w_t,t;s)\right)\right\|^2 dt\right]. \tag{20}$$

Importantly, optimizing (20) *does not require backpropagation through the flow* and only requires the solution of an ODE to compute the adjoint state $\tilde{g}_t$ for the regression target $\hat{v}^\star$, circumventing the computational and numerical stability issues of BPTT. For a more comprehensive mathematical treatment of adjoint matching, we refer readers to [16].

3.3 Algorithm

We now present **L**atent Space **A**djoint Matching for **S**upport-Constrained **E**ntropy-**R**egularized Offline RL (**LASER**). LASER combines the aforementioned insights to perform latent-space RL with an expressive flow decoder and efficient entropy-regularized policy optimization via adjoint matching.

- Following [7], we choose the latent prior $q_\mathsf{L} = \mathcal{U}(\mathcal{B}_l^d)$ to be the uniform distribution on the ball $\mathcal{B}_l^d$ of radius l in $\mathbb{R}^d$, which has finite Lebesgue measure and satisfies the conditions of Lemma 3.2.
- Adjoint matching requires $q_\mathsf{L}^{\text{unif}}$, the uniform distribution over the support of q_L, to also be a flow model. While an analytical vector field can be derived since $q_\mathsf{L}^{\text{unif}} = q_\mathsf{L}$ is the uniform distribution over the ball, it involves special functions with no easy closed-form expression (see Appendix C.2). Instead, we train a flow model via flow matching to match $q_\mathsf{L}^{\text{unif}}$ with the same base distribution q_B as π_L to simplify the implementation.

We now summarize LASER (Figure 4), which consists of four components trained jointly: **(1. Flow-matching decoder)** We train a state-conditioned flow decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$ using flow matching to map samples from the uniform distribution on the d-ball q_L to dataset actions, i.e., behavior cloning (Section 2.2). **(2. Flow matching for $\mathbf{q_\mathsf{L}^{unif}}$)** We train a flow model $\mu^{\text{base}}_{\mathsf{B}\to\mathsf{L}}$ to match the uniform distribution $q_\mathsf{L}^{\text{unif}}$ in latent space from q_B to use in adjoint matching. **(3. Critic learning)** We learn a critic $Q_\phi^{\pi_\mathsf{A}}$ using the standard TD learning objective (2). **(4. Entropy-regularized policy optimization)** Finally, we perform entropy-regularized policy optimization in latent space to maximize the latent-space Q-function $Q_\phi^{\pi_\mathsf{L}}$ (7) using adjoint matching (Section 3.2).[2] We provide practical implementation details and a high-level algorithm for LASER in Appendix D.

[2] In practice, we project the numerical latent flow output onto $\mathcal{B}_l^d$ before decoding. This support correction approximates the density-based formulation (Appendix D).

4 Related Work

We discuss the most closely related work in this section and defer a broader discussion of offline RL and generative policies to Appendix A.

Distribution shift in offline RL. A central challenge in offline RL is distribution shift: learned value functions can overestimate actions outside the dataset support, causing policy improvement to select out-of-distribution actions with poor true value. Prior methods address this issue by regularizing the policy toward the behavior policy with statistical distances, learning pessimistic value functions, or explicitly restricting policy improvement to supported actions [5, 26, 27, 28, 4]. We follow the support-constrained perspective, using an expressive flow policy to enforce support by construction.

Expressive generative policies for offline RL. Diffusion [29, 30] and flow-matching [9, 11, 10] models have recently been used as expressive alternatives to deterministic or Gaussian policies, especially in robotic imitation learning where action distributions are often multimodal [31, 32]. These policies have also been adapted to offline RL, where the goal is to improve beyond the behavior policy using a learned critic. Existing methods differ in how they use the critic: some use critic values to select, reweight, or clone high-value actions [33, 34, 35, 36], while others use critic gradients to directly improve the generated actions [37, 32, 23]. Most closely related to our optimization mechanism, [25] fine-tunes flow policies using critic gradients under a KL-regularized reward objective. Unlike these methods, which rely on statistical regularization to control distribution shift, LASER enforces support constraints through its latent action parameterization.

Support-constrained policy extraction with latent-space RL. Recent methods avoid OOD actions by learning a latent action space whose decoder maps into the support of the behavior policy, then performing policy optimization in this latent space [19, 8, 7]. Our method builds on this idea, but revisits the optimization problem that remains once support has been controlled. In particular, we show that entropy regularization is important for preventing policy collapse and stabilizing policy extraction within the supported action set.

Entropy regularization. Entropy regularization is a classical tool for smoothing policy optimization, improving robustness, and encouraging exploration [13, 12, 15], with extensions to expressive policies [21, 22]. In offline RL, though some works have noted its benefits for exploration during online fine-tuning [37, 38], it receives much less attention, perhaps due to the implicit entropy regularization effect from behavior policy regularization techniques. Our perspective is different: because support constraints are enforced by the policy parameterization, entropy regularization can be used for its standalone role in policy extraction.

5 Experiments

We conduct experiments to answer the following research questions. **(Q1):** How does LASER perform against other offline RL algorithms with flow policies? **(Q2):** How sensitive is LASER to hyperparameters? **(Q3):** How does adjoint matching compare with BPTT? **(Q4):** Does the latent space decouple entropy regularization from behavior regularization? Details on implementation, baselines, environments, and additional results and ablations are in Appendix F.

5.1 Setup

Environments. We evaluate our method on the OGBench offline RL benchmark [18]. The main evaluation suite spans four environments, covering both locomotion and manipulation domains, with five tasks per environment. Each task is evaluated on two datasets (CLEAN and NOISY), yielding a total of $4 \times 5 \times 2 = 40$ evaluation settings. We additionally evaluate harder tasks in Appendix F.5.

Baselines. Since recent studies have shown that flow-based policies outperform traditional alternatives [23, 39, 24, 25], we benchmark LASER against state-of-the-art flow-based methods. These include IFQL [34, 23], which combines a BC flow policy with an IQL critic and best-of-N action selection; FQL [23], which applies Wasserstein distance regularization; and QAM [25], which uses adjoint matching to approximate the Q-value-weighted BC action distribution, alongside its extension QAM-E [25] that learns an additional residual policy. We also compare against latent-space RL methods, specifically DSRL [8] with a Gaussian latent policy, and ReFORM [7] with a reflected flow latent policy. The method-specific hyperparameters of LASER, ReFORM, and IFQL remain fixed across all evaluated

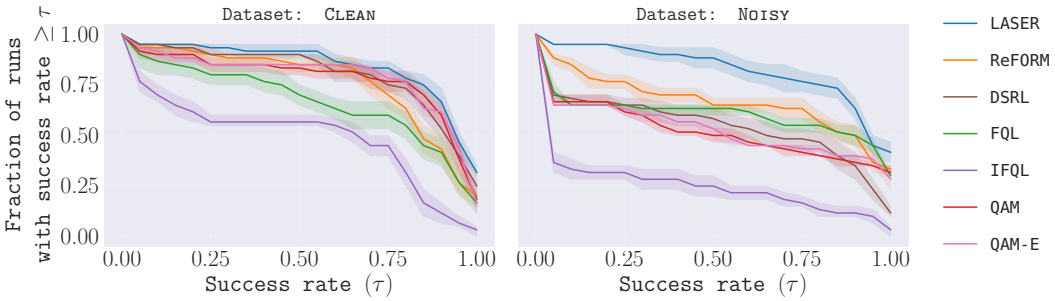

Figure 5: **Performance profile plots for the main evaluation suite with different datasets.** For a given success rate τ (x-axis), we plot the fraction of runs with a success rate $\geq \tau$ [40]. Shading shows pointwise 95% bootstrap confidence intervals, computed by resampling training seeds within each task 2000 times. LASER attains the strongest empirical performance profile on both datasets.

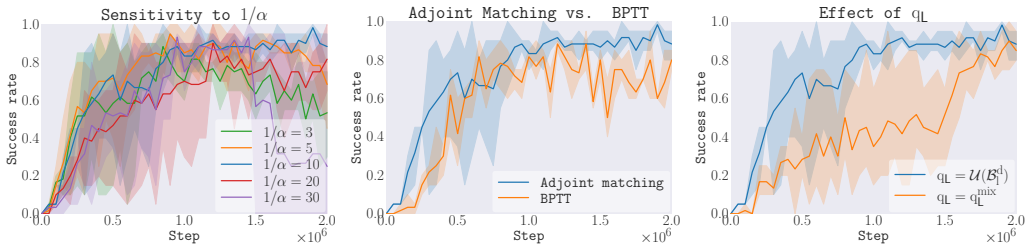

Figure 6: **Ablation studies in** cube-double-play-singletask-task2-v0**.** Moderate values of the inverse temperature $1/\alpha \in [5, 20]$ yield comparable results. Adjoint matching achieves better and more stable results than BPTT. Using a different q_L yields comparable final success rates in this task, consistent with the proposed separation between entropy regularization and behavior regularization. Lines show mean success rates, and shading spans the minimum and maximum across runs.

tasks, while those of the other baselines vary by environment and dataset (Table 3). All methods share the environment-dependent action chunk lengths and pessimistic backup weights (Table 2).

5.2 Results

(Q1): LASER achieves state-of-the-art performance using a single set of method-specific hyperparameters. In Figure 5, we present the performance profiles [40] aggregating the 40 main evaluation settings across both the CLEAN and NOISY datasets. The profiles show that LASER outperforms all evaluated baselines in aggregate. It achieves at least 95% of the best mean success rate in $34/40$ settings and a mean success rate of 100% in $13/40$ settings. A paired statistical comparison supports the aggregate advantage (global $p = 1.35 \times 10^{-3}$), with details in Appendix F.5. Notably, on the NOISY datasets, LASER and ReFORM achieve the highest mean success rates, underscoring the advantage of enforcing support constraints, which does not restrict policy improvement. Furthermore, LASER significantly improves upon ReFORM by using explicit entropy regularization and avoiding the static-flow behavior (Appendices C.1 and F.5).

(Q2): The key hyperparameter of LASER is the inverse temperature. Using a constant set of method-specific hyperparameters for LASER across all tasks demonstrates its robustness to different tasks and datasets. Nevertheless, we analyze the sensitivity of LASER to the inverse temperature $1/\alpha$ (Figure 6). Values of $1/\alpha \in [5, 20]$ yield comparable results. At $1/\alpha = 3$, performance drops on both datasets, whereas $1/\alpha = 30$ degrades performance on CLEAN data but remains competitive on NOISY data (Table 6). We find that $1/\alpha = 10$ yields stable results for all environments. We also compare LASER with Q-only optimization and TPS in this ablation and find that LASER with $1/\alpha \in [5, 20]$ outperforms both of them (Table 6). In addition, we analyze sensitivity to σ_0 and find that LASER is robust to this hyperparameter (Appendix F.5).

(Q3): Adjoint matching achieves better results and is faster than BPTT using reparametrized gradients. A key advantage of adjoint matching is that it circumvents the need to perform BPTT

on (12), which is computationally expensive and numerically unstable. We validate this empirically and observe that adjoint matching yields more stable training and stronger performance compared to BPTT (Figure 6) while training $3.6\times$ faster ($123\,\mathrm{it/s}$ vs $34\,\mathrm{it/s}$).

(Q4): The latent space successfully decouples entropy regularization from behavior regularization. To understand the improvements of LASER, we change q_{L} to a distribution with the same support but a different density, which changes the behavior regularization effect but keeps the same latent-space entropy regularization (see Appendix F.5). Both runs retain the uniform entropy reference on the same latent ball. Changing the decoder prior can still alter the learned latent-to-action mapping and learning dynamics, while leaving the entropy regularizer unchanged. We observe that changing the decoder prior yields comparable final success rates in this task, consistent with the proposed separation between latent-space entropy regularization and behavior regularization (Figure 6).

6 Conclusion

We introduced LASER, an entropy-regularized approach to support-constrained flow policy optimization for offline RL. Our key insight is that support constraints and entropy regularization solve different problems: the former prevent OOD actions, while the latter stabilizes policy extraction within the learned dataset support. By combining a flow-based decoder with latent-space adjoint matching, LASER effectively helps prevent mode collapse, smooths the optimization landscape, and avoids local minima, demonstrating state-of-the-art performance on the OGBench suite with fixed method-specific hyperparameters.

Limitations and Future Work. We identify several promising avenues for extension. First, although adjoint matching circumvents the need for BPTT, solving the ODEs needed to sample from the flow model during training remains computationally intensive for deep neural ODEs. This computational cost could be reduced through distillation [24]. Second, by relying on the decoder to enforce support constraints, LASER inherits OOD errors made by the decoder itself. Integrating behavior cloning methods with stricter support constraints, diagnosing when the BC model generates OOD errors, or applying a pre-trained robust BC model could mitigate this. In addition, LASER uses a simple value-function learning method and an actor-critic architecture similar to those in [23]. Both could be improved using ideas from [41, 42, 43, 39].

Broader Impacts. Learning policies from existing datasets may reduce the cost and risks of online data collection in robotics. However, support constraints alone do not guarantee safe behavior, as datasets may contain unsafe actions and learned decoders can make errors. Deployment under distribution shift could therefore cause physical harm. Safety validation and monitoring remain necessary before deploying these policies on physical systems.

Acknowledgements

This material is based upon work supported by the Under Secretary of War for Research and Engineering under Air Force Contract No. FA8702-15-D-0001 or FA8702-25-D-B002. In addition, Zhang, So, and Fan are supported by the National Science Foundation (NSF) CAREER Award #CCF-2238030. This work was also supported in part by a grant from Amazon. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of War for Research and Engineering.

© 2026 Massachusetts Institute of Technology.

Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

References

[1] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[2] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

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

[4] Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported trust region optimization for offline reinforcement learning. In *International Conference on Machine Learning*, pages 23829–23851. PMLR, 2023.

[5] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

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

[7] Songyuan Zhang, Oswin So, H M Sabbir Ahmad, Eric Yang Yu, Matthew Cleaveland, Mitchell Black, and Chuchu Fan. ReFORM: Reflected flows for on-support offline RL via noise manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[8] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[9] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[10] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[11] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

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

[13] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[14] Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *International Conference on Learning Representations*, 2022.

[15] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pages 151–160. PMLR, 2019.

[16] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

[17] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[18] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025.

[19] Wenxuan Zhou, Sujay Bajracharya, and David Held. PLAS: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pages 1719–1735. PMLR, 2021.

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

[21] Safa Messaoud, Billel Mokeddem, Zhenghai Xue, Linsey Pang, Bo An, Haipeng Chen, and Sanjay Chawla. S^2AC: Energy-based reinforcement learning with stein soft actor critic. In *International Conference on Learning Representations (ICLR)*, 2024.

[22] Safa Messaoud, Skander Charni, Elaa Bouazza, Ali Pourghasemi Fatideh, and Halima Bensmail. Particles don't care about Z: Towards scaling entropy estimation of unnormalized densities. In *Forty-third International Conference on Machine Learning*, 2026.

[23] Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. In *International Conference on Machine Learning (ICML)*, 2025.

[24] Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kianté Brantley, and Wen Sun. Scaling offline RL via efficient and expressive shortcut models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[25] Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *The Fourteenth International Conference on Learning Representations*, 2026.

[26] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[27] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

[28] Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:31278–31291, 2022.

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

[30] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[31] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[32] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[34] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[35] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

[36] Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR 2025)*, 2025.

[37] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via Q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

[38] Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. *arXiv preprint arXiv:2507.07986*, 2025.

[39] Bhavya Agrawalla, Michal Nauman, Khush Agrawal, and Aviral Kumar. floq: Training critics via flow-matching for scaling compute in value-based RL. *arXiv preprint arXiv:2509.06863*, 2025.

[40] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34:29304–29320, 2021.

[41] Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported value regularization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:40587–40609, 2023.

[42] Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme Q-learning: Maxent RL without entropy. In *International Conference on Learning Representations*, 2023.

[43] Tenglong Liu, Yang Li, Yixing Lan, Hao Gao, Wei Pan, and Xin Xu. Adaptive advantage-guided policy regularization for offline reinforcement learning. In *International Conference on Machine Learning*, pages 31406–31424. PMLR, 2024.

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

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

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

[47] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

[48] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative Q-learning for offline reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[49] Zhepeng Cen, Zuxin Liu, Zitong Wang, Yihang Yao, Henry Lam, and Ding Zhao. Learning from sparse offline datasets via conservative density estimation. In *The Twelfth International Conference on Learning Representations*, 2024.

[50] Qingjun Wang, Hongtu Zhou, Hang Yu, Junqiao Zhao, Yanping Zhao, Chen Ye, Ziqiao Wang, and Guang Chen. Beyond penalization: Diffusion-based out-of-distribution detection and selective regularization in offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[51] Jing Zhang, Chi Zhang, Wenjia Wang, and Bingyi Jing. Constrained policy optimization with explicit behavior density for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:5616–5630, 2023.

[52] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023.

[53] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[54] Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. In *The Twelfth International Conference on Learning Representations*, 2024.

[55] Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[56] Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B Schön, and Per Mattsson. Entropy-regularized diffusion policy with Q-ensembles for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:98871–98897, 2024.

[57] Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3116–3123, 2024.

[58] Franki Nguimatsia Tiofack, Théotime Le Hellard, Fabian Schramm, Nicolas Perrin-Gilbert, and Justin Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[59] Jongseong Chae, Jongeui Park, Yongjae Shin, Gyeongmin Kim, Seungyul Han, and Youngchul Sung. Flow actor-critic for offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[60] Xiaoyuan Cheng, Haoyu Wang, Wenxuan Yuan, Ziyan Wang, Zonghao Chen, Li Zeng, and Zhuo Sun. Fisher decorator: Refining flow policy via a local transport map. *arXiv preprint arXiv:2604.17919*, 2026.

[61] He Zhang, Ying Sun, and Hui Xiong. Goldenstart: Q-guided priors and entropy control for distilling flow policies. In *The Fourteenth International Conference on Learning Representations*, 2026.

[62] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[63] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[64] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pages 22825–22855. PMLR, 2023.

[65] Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

[66] Haoran Xu, Kaiwen Hu, Somayeh Sojoudi, and Amy Zhang. Value gradient flow: Behavior-regularized RL without regularization. In *The Fourteenth International Conference on Learning Representations*, 2026.

[67] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and MCMC. In *International Conference on Machine Learning*, pages 8489–8510. PMLR, 2023.

[68] Kamyar Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? Estimating uncertainties for offline RL through ensembles, and why their independence matters. *Advances in Neural Information Processing Systems*, 35:18267–18281, 2022.

[69] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[70] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[71] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

A Extended Related Work

Mitigating distribution shift in offline RL. A central challenge in offline RL is mitigating distribution shift: learned value functions can overestimate actions that are not in the dataset, causing the policy to select these out-of-distribution actions. Classical approaches address this issue by either regularizing the policy to remain close to the behavior policy with statistical distances (e.g., KL divergence [2, 3, 6, 44, 45, 46, 47, 27], MMD distance [5] or Wasserstein distance [6]), or regularizing the value function to be pessimistic on out-of-distribution actions [26, 27, 48, 41, 49, 50]. Another family of methods addresses the OOD problem by penalizing actions that fall outside the support of the behavior policy [5, 28, 4, 51]. However, these methods rely on heuristic sample-based proxies [5] or Gaussian density approximations [28, 4, 51] which can fail to capture expressive multimodal behavior policies.

Expressive generative policies for offline RL. Diffusion [29, 30] and flow-matching [9, 11, 10] models have recently been used as expressive alternatives to deterministic or Gaussian policies, especially in robotic imitation learning where action distributions are often multimodal [31, 52, 53, 32]. These policies have also been adapted to offline RL, where the goal is to improve beyond the behavior policy using a learned critic. However, because flow and diffusion policies generate actions through an iterative sampling process, critic-based optimization is more challenging than for standard Gaussian actors. Existing reward-optimization methods for generative policies can be broadly divided by how they use the learned critic: *critic-evaluation methods*, which use only scalar values $Q(s, a)$ to select or reweight actions, and *critic-gradient methods*, which use $\nabla_a Q(s, a)$ to directly improve generated actions.

(i) *Critic-evaluation* methods include sampling-based methods that imitate the behavior policy and use the critic to sample from the reward-tilted distribution either at test time [33, 34] or during training, when the sampled actions serve as imitation targets [54]. These methods often reliably improve the quality of the extracted policy at greater computational cost and depend on the quality of the behavior policy. Weighted behavioral cloning methods also fall under this category. These methods use the critic to assign per-sample weights in the diffusion or flow-matching loss, selectively cloning high-value actions in the dataset [35, 37, 36], but generally do not optimize the learned policy towards the reward-tilted distribution [25].

(ii) Some *critic-gradient* methods combine a behavior-cloning term with a term that increases the critic-estimated value. These methods require backpropagation through the iterative sampling process [32, 55, 56, 57], which can be computationally expensive and unstable [23, 24, 25]. Other methods learn a distilled policy [23, 58, 59, 60, 61] at the expense of policy expressiveness. Another line of work uses guidance [62, 63] during the iterative sampling process to favor high-value actions [64, 65, 66], but these methods rely on approximations that do not have theoretical guarantees [67, 25]. Most closely related to our optimization mechanism, [25] provides an efficient way to fine-tune flow-based policies using critic gradients and a KL-regularized reward objective via adjoint matching [16]. However, unlike critic-evaluation methods where samples from the learned policy are either directly cloned (sampling-based methods) or selectively cloned (weighted BC methods), these methods rely on statistical distance regularization to avoid the OOD problem (e.g., L2 / Wasserstein distance [23] or KL regularization [25]). A key challenge is that the regularization weight is a hyperparameter that significantly affects the performance and needs to be tuned for each task and dataset [18, 23].

Our method falls under the *critic-gradient* category as we also use adjoint matching to optimize the policy with critic gradients under KL regularization. However, to address the OOD problem, LASER relies on the latent space to enforce the support constraints by construction instead of using KL regularization toward the behavior policy, as in [25], which can be too restrictive in cases where optimal actions have very low probability under the behavior policy [5].

Support constraints in offline RL. Instead of addressing the OOD problem with regularization, another line of work proposes to address the OOD problem by learning a bounded latent space that maps to the support of the behavior policy, then performing policy extraction in this latent space to restrict the support of the learned policy by construction [19, 8, 7]. Our work follows this support-constrained perspective, but revisits the optimization problem that remains after support has been controlled. Namely, we find that entropy regularization is essential to prevent policy collapse, and leverage adjoint matching [16] to perform entropy-regularized policy extraction in latent space

with an *expressive* flow policy that achieves a better performance-entropy tradeoff than Gaussian policies.

Entropy-regularized policy extraction. The role of entropy regularization has been well identified in *online* RL as a way to improve exploration [13, 12] and policy robustness [13, 14] and to smooth the optimization landscape [15]. In *offline* RL, while some works note the benefits of entropy regularization for exploration [37, 38], the importance of entropy regularization has been less emphasized, perhaps due to the implicit entropy regularization effect from various behavior policy regularization techniques. In our setting, the learned decoder and latent support constraint handle support control, so no separate behavior-policy regularization term is needed. This makes the role of entropy regularization in policy extraction explicit.

B Proof of Lemma 3.2

For a Lebesgue-measurable set $X \subseteq \mathbb{R}^d$, let $\lambda(X)$ denote its Lebesgue measure. We first prove the following technical lemma to express entropy as a constant minus a KL divergence.

Lemma B.1. *Fix a state $s \in \mathcal{S}$. For brevity, we omit the state dependence in the policy. Let $\mathcal{L}_q \coloneqq \mathrm{supp}(q_{\mathsf{L}}) \subseteq \mathcal{L}$ denote the support of q_{L}. Suppose $\mathcal{L}_q$ has finite positive Lebesgue measure, i.e., $0 < \lambda(\mathcal{L}_q) < \infty$, and π_{L} has a density and satisfies the latent-space support constraints* (6)*, i.e.,*

$$\mathrm{supp}(\pi_{\mathsf{L}}) \subseteq \mathrm{supp}(q_{\mathsf{L}}). \tag{21}$$

Let $q_{\mathsf{L}}^{\mathrm{unif}}$ be the uniform distribution over $\mathrm{supp}(q_{\mathsf{L}})$, i.e.,

$$q_{\mathsf{L}}^{\mathrm{unif}}(z) = \begin{cases} 1/\lambda(\mathcal{L}_q) & z \in \mathcal{L}_q \\ 0 & \textit{otherwise} \end{cases}. \tag{22}$$

Then, the entropy of π_{L} can be expressed as

$$\mathcal{H}(\pi_{\mathsf{L}}) = -D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\mathrm{unif}}) + \log \lambda(\mathcal{L}_q), \tag{23}$$

where the second term in (23) *is a constant independent of π_{L}.*

Proof. Expanding the KL divergence,

$$\begin{aligned} D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\mathrm{unif}}) &= \int_{\mathbb{R}^d} \pi_{\mathsf{L}}(z) \log \frac{\pi_{\mathsf{L}}(z)}{q_{\mathsf{L}}^{\mathrm{unif}}(z)} \, \mathrm{d}z, && (24) \\ &= \int_{\mathcal{L}_q} \pi_{\mathsf{L}}(z) \log \pi_{\mathsf{L}}(z) \, \mathrm{d}z - \int_{\mathcal{L}_q} q_{\mathsf{L}}^{\mathrm{unif}}(z) \log q_{\mathsf{L}}^{\mathrm{unif}}(z) \, \mathrm{d}z, && (25) \\ &= -\mathcal{H}(\pi_{\mathsf{L}}) - \int_{\mathcal{L}_q} q_{\mathsf{L}}^{\mathrm{unif}}(z) \log \left(\frac{1}{\lambda(\mathcal{L}_q)} \right) \mathrm{d}z, && (26) \\ &= -\mathcal{H}(\pi_{\mathsf{L}}) + \log \lambda(\mathcal{L}_q). && (27) \end{aligned}$$

Rearranging the above equation yields the desired result. □

We are now ready to prove Lemma 3.2.

Proof. Fix a state $s \in \mathcal{S}$. For brevity, we omit the state dependence in the policy. We write $Q^{\pi_{\mathsf{L}}}(z)$ for $Q_\phi^{\pi_{\mathsf{L}}}(s, z)$, held fixed with respect to the candidate policy. Restating the entropy-regularized optimization problem (11) in terms of the policy π_{L} directly, we wish to solve the following optimization problem over densities supported on $\mathcal{L}_q$

$$\min_{\pi_{\mathsf{L}}} -\mathbb{E}_{z \sim \pi_{\mathsf{L}}} \left[Q^{\pi_{\mathsf{L}}}(z) \right] - \alpha \mathcal{H}(\pi_{\mathsf{L}}). \tag{28}$$

By Lemma B.1, the entropy of π_{L} can be expressed as

$$\mathcal{H}(\pi_{\mathsf{L}}) = -D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\mathrm{unif}}) + \log \lambda(\mathcal{L}_q). \tag{29}$$

Substituting this into (28) and dropping the constant term, we now have a KL-regularized optimization problem

$$\min_{\pi_{\mathsf{L}}} -\mathbb{E}_{z \sim \pi_{\mathsf{L}}} \left[Q^{\pi_{\mathsf{L}}}(z) \right] + \alpha D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\mathrm{unif}}). \tag{30}$$

Now, define the partition function Z as

$$Z \coloneqq \int_{\mathbb{R}^d} q_{\mathsf{L}}^{\text{unif}}(z) \exp\left(\frac{1}{\alpha} Q^{\pi_{\mathsf{L}}}(z)\right) \mathrm{d}z. \tag{31}$$

Since $Q^{\pi_{\mathsf{L}}}$ is measurable and bounded from above, there exists an $m \in \mathbb{R}$ such that $Q^{\pi_{\mathsf{L}}}(z) \leq m$ for all $z \in \mathcal{L}_q$. Hence,

$$0 < \exp\left(\frac{1}{\alpha} Q^{\pi_{\mathsf{L}}}(z)\right) \leq \exp\left(\frac{1}{\alpha} m\right) < \infty, \tag{32}$$

and thus

$$0 < Z \leq \exp\left(\frac{1}{\alpha} m\right) < \infty, \tag{33}$$

so $\pi_{\mathsf{L}}^\star$, defined as

$$\pi_{\mathsf{L}}^\star(z) \coloneqq \frac{1}{Z} q_{\mathsf{L}}^{\text{unif}}(z) \exp\left(\frac{1}{\alpha} Q^{\pi_{\mathsf{L}}}(z)\right), \tag{34}$$

is well-defined. Since $D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\text{unif}}) = +\infty$ for any $\pi_{\mathsf{L}} \not\ll q_{\mathsf{L}}^{\text{unif}}$, it is sufficient to consider policies π_{L} that are absolutely continuous with respect to $q_{\mathsf{L}}^{\text{unif}}$. Note that for $z \in \mathcal{L}_q$,

$$\frac{\pi_{\mathsf{L}}{}^\star(z)}{q_{\mathsf{L}}^{\text{unif}}(z)} = \frac{1}{Z} \exp\left(\frac{1}{\alpha} Q^{\pi_{\mathsf{L}}}(z)\right), \qquad \log\left(\frac{\pi_{\mathsf{L}}{}^\star(z)}{q_{\mathsf{L}}^{\text{unif}}(z)}\right) = -\log Z + \frac{1}{\alpha} Q^{\pi_{\mathsf{L}}}(z). \tag{35}$$

For any $\pi_{\mathsf{L}} \ll q_{\mathsf{L}}^{\text{unif}}$, we have

$$\begin{aligned}
D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| \pi_{\mathsf{L}}{}^\star) &= \int_{\mathbb{R}^d} \pi_{\mathsf{L}}(z) \log \frac{\pi_{\mathsf{L}}(z)}{\pi_{\mathsf{L}}{}^\star(z)} \, \mathrm{d}z, && (36)\\
&= \int_{\mathbb{R}^d} \pi_{\mathsf{L}}(z) \log \frac{\pi_{\mathsf{L}}(z)}{q_{\mathsf{L}}^{\text{unif}}(z)} \, \mathrm{d}z - \int_{\mathbb{R}^d} \pi_{\mathsf{L}}(z) \log \frac{\pi_{\mathsf{L}}{}^\star(z)}{q_{\mathsf{L}}^{\text{unif}}(z)} \, \mathrm{d}z, && (37)\\
&= D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\text{unif}}) + \log Z - \frac{1}{\alpha} \int_{\mathbb{R}^d} \pi_{\mathsf{L}}(z) Q^{\pi_{\mathsf{L}}}(z) \, \mathrm{d}z, && (38)\\
&= D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\text{unif}}) + \log Z - \frac{1}{\alpha} \mathbb{E}_{z \sim \pi_{\mathsf{L}}}[Q^{\pi_{\mathsf{L}}}(z)]. && (39)
\end{aligned}$$

Thus, the objective can be written as the KL divergence to $\pi_{\mathsf{L}}{}^\star$ plus a constant:

$$-\mathbb{E}_{z \sim \pi_{\mathsf{L}}}[Q^{\pi_{\mathsf{L}}}(z) - \alpha D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| q_{\mathsf{L}}^{\text{unif}})] = \alpha D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| \pi_{\mathsf{L}}{}^\star) - \alpha \log Z. \tag{40}$$

Since $D_{\mathrm{KL}}(\pi_{\mathsf{L}} \| \pi_{\mathsf{L}}{}^\star) \geq 0$ for all π_{L} with equality if and only if $\pi_{\mathsf{L}} = \pi_{\mathsf{L}}{}^\star$, we conclude that $\pi_{\mathsf{L}}{}^\star$ is the unique optimal solution to the optimization problem (28). □

C More Theoretical Results

C.1 The Static Flow Issue and Implicit Entropy Regularization in ReFORM

To enforce the latent-space support constraint (6), ReFORM [7] introduces a reflection term during the integration of the latent flow policy $\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}$. The corresponding modified Euler step is defined as:

$$w_{k+1} = \mathbb{1}\{\hat{w}_{k+1} \in \mathcal{B}_l^d\} \hat{w}_{k+1} + (1 - \mathbb{1}\{\hat{w}_{k+1} \in \mathcal{B}_l^d\}) \left(\hat{w}_{k+1} - \langle v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_k, k\Delta t; s)\Delta t, n_{k+1}\rangle n_{k+1}\right), \tag{41}$$

where $\hat{w}_{k+1} = w_k + v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_k, k\Delta t; s)\Delta t$ denotes the standard Euler step, N is the total number of integration steps, $k \in \{0, \ldots, N-1\}$, $\Delta t = \frac{1}{N}$, $n_{k+1} = \frac{\hat{w}_{k+1}}{\|\hat{w}_{k+1}\|}$, and $\langle \cdot, \cdot \rangle$ represents the inner product. Integrating this modified step yields the final latent variable $z \leftarrow w_N$.

While [7] proves that this modified integration step guarantees adherence to the latent support constraint (6), it introduces a degenerate behavior we term "static flow". This occurs when the proposed step violates the boundary ($\hat{w}_{k+1} \notin \mathcal{B}_l^d$) and the flow velocity is strictly codirectional with the outward normal vector ($v_{\theta_{\mathsf{B}\to\mathsf{L}}} \uparrow\uparrow n_{k+1}$). Under these conditions, the reflection term perfectly cancels the velocity step:

$$\begin{aligned}
w_{k+1} &= \hat{w}_{k+1} - \langle v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_k, k\Delta t; s)\Delta t, n_{k+1}\rangle n_{k+1} \\
&= \left(w_k + v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_k, k\Delta t; s)\Delta t\right) - v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_k, k\Delta t; s)\Delta t = w_k.
\end{aligned} \tag{42}$$

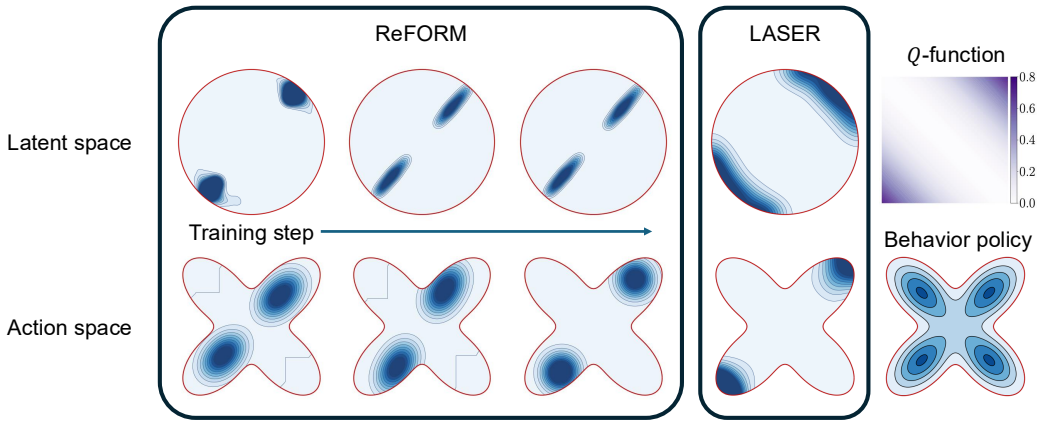

Figure 7: **ReFORM fails to maximize the Q-value because of the "static flow" issue.** For a given behavior policy and a ground-truth Q-function, the generated latent-space distribution $(\mu^s_{\theta_{\mathsf{B}\to\mathsf{L}}})_\sharp q_\mathsf{B}$ concentrates along radial directions aligned with the flow, rather than moving toward the boundary of $\mathcal{B}^d_l$. This makes the flow static, resulting in suboptimal performance compared with LASER.

Consequently, the corrected step is stationary under this alignment condition. Rather than following the flow field $v_{\theta_{\mathsf{B}\to\mathsf{L}}}$ to the boundary of $\mathcal{B}^d_l$, the pushforward distribution of $\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}$ unnaturally accumulates along the radial trajectory. To illustrate this phenomenon, we train ReFORM on the 2D toy environment introduced in Figure 3. The learned latent distribution and its resulting action distribution are visualized in Figure 7. We empirically observe that the latent mass inappropriately concentrates along two radii, resulting in a suboptimal action distribution. We also conduct additional experiments in cube-double in Appendix F.5 to measure how often the boundary correction is activated.

Interestingly, we find that this static flow artifact unintentionally serves as an implicit entropy regularizer, which partially explains ReFORM's empirical success. As demonstrated in Figure 7, the premature halting of trajectories prevents the latent distribution from collapsing to a single point, artificially maintaining the spread of the distribution.

C.2 The Ground-truth Base Velocity Field from the Base Space to the Latent Space

As mentioned in Section 3.3, there exists a ground-truth velocity field $v_{\mathsf{B}\to\mathsf{L}}$ that maps samples from $w \sim q_\mathsf{B} = \mathcal{N}(0, \sigma_0^2 I)$ to $z \sim \mathcal{U}(\mathcal{B}^d_l)$. We show here how we can find such a ground-truth velocity field and why we choose not to use it in our implementation.

Since both the base distribution $q_\mathsf{B} = \mathcal{N}(0, \sigma_0^2 I)$ and the target distribution $\mathcal{U}(\mathcal{B}^d_l)$ are spherically symmetric, the optimal transport (OT) mapping between them operates purely along the radial direction. Let $r_w = \|w\|$ denote the radius of a sample w from the base distribution, and $r_z = \|z\|$ denote the radius in the latent space. The cumulative distribution function (CDF) for the uniform distribution on a d-dimensional ball of radius l is $F_\mathcal{U}(r) = (r/l)^d$ for $r \in [0, l]$. Meanwhile, the radius of the Gaussian base distribution follows a scaled χ-distribution with d degrees of freedom, with its CDF denoted as $F_\chi(r)$.

By matching the quantiles of these radial distributions, we can construct the exact analytical mapping $\mu_{\mathsf{B}\to\mathsf{L}} : \mathbb{R}^d \to \mathcal{B}^d_l$ as follows:

$$\mu_{\mathsf{B}\to\mathsf{L}}(w) = \frac{w}{\|w\|} F_\mathcal{U}^{-1}(F_\chi(\|w\|)) = w \frac{l \cdot F_\chi(\|w\|)^{1/d}}{\|w\|}. \tag{43}$$

At the origin, we define $\mu_{\mathsf{B}\to\mathsf{L}}(0) = 0$ by continuity. However, although we have derived an exact analytical mapping $\mu_{\mathsf{B}\to\mathsf{L}}$, it is nontrivial to derive the corresponding velocity field $v_{\mathsf{B}\to\mathsf{L}}(w_t, t; s)$, i.e., the instantaneous velocity of a particle currently at location w_t at time t. Under standard optimal transport displacement interpolation, a particle travels in a straight line from its origin w to its destination $\mu_{\mathsf{B}\to\mathsf{L}}(w)$ at a constant speed, yielding the trajectory:

$$w_t = (1-t)w + t\mu_{\mathsf{B}\to\mathsf{L}}(w). \tag{44}$$

While the velocity of this specific particle is trivially $v(w) = \mu_{\mathsf{B}\to\mathsf{L}}(w) - w$, the velocity field must be expressed independently of the origin w. This requires inverting the trajectory equation to express w purely as a function of the current state w_t and time t.

Because our mapping $\mu_{\mathsf{B}\to\mathsf{L}}$ is purely radial, the vectors w and w_t share the same direction, meaning $\frac{w}{\|w\|} = \frac{w_t}{\|w_t\|}$ for $w \neq 0$. Let $r_w = \|w\|$ and $r_t = \|w_t\|$. Taking the norm of the trajectory equation yields the following relationship between the magnitudes:

$$r_t = (1-t)r_w + t \cdot l \cdot F_\chi(r_w)^{1/d}. \tag{45}$$

To compute the velocity at w_t, we must find the inverse function $g_t(r_t)$ that recovers the initial radius r_w given r_t and t. For each $t \in [0,1]$, the radial map is strictly increasing, so it has a unique inverse on its image. For $t > 0$ and $w_t \neq 0$, the exact Eulerian velocity field is:

$$v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}(w_t, t) = \frac{w_t - w}{t} = \left(\frac{\|w_t\| - g_t(\|w_t\|)}{t}\right)\frac{w_t}{\|w_t\|}. \tag{46}$$

The values at $t = 0$ and $w_t = 0$ are defined by continuity. Thus, although g_t exists, evaluating it generally requires numerical inversion. For intermediate times $0 < t < 1$, computing $g_t(\|w_t\|)$ would add a numerical solve at each velocity evaluation. We instead learn $v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}$ via flow matching to avoid these repeated solves.

D Practical Implementation of LASER

We provide a high-level summary of LASER in Algorithm 1.

Flow-matching Decoder Learning. We parameterize the velocity field $v_{\theta_{\mathsf{L}\to\mathsf{A}}}$ of the decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$ with a multilayer perceptron (MLP), and learn it via the flow-matching loss (8). The prior latent-space distribution is chosen to be a bounded uniform distribution over a d-ball with radius l, i.e., $q_{\mathsf{L}} = \mathcal{U}(\mathcal{B}_l^d)$. After training, we sample $z \sim q_{\mathsf{L}}$ and use Euler integration to get the action samples:

$$z_0 \leftarrow z, \quad z_{(k+1)\Delta t} = z_{k\Delta t} + v_{\theta_{\mathsf{L}\to\mathsf{A}}}(z_{k\Delta t}, k\Delta t; s)\Delta t, \tag{47}$$

where N is the total number of integration steps, $\Delta t = \frac{1}{N}$, $k \in \{0, \dots, N-1\}$, and the final generated action is $a \leftarrow z_1$.

Flow matching for $q^{\text{unif}}_{\mathsf{L}}$. Similarly, the velocity field $v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}$ is parameterized by an MLP and learned via the flow-matching loss (8), with prior distribution $q_{\mathsf{B}} = \mathcal{N}(0, \sigma_0^2 I)$ and the desired pushforward $q^{\text{unif}}_{\mathsf{L}} = \mathcal{U}(\mathcal{B}_l^d)$. Note that $v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}$ is state-independent.

Critic Learning. The critic is learned via the standard TD learning objective (2) with an ensemble of K critics. In some environments, we found that using a pessimistic target-value backup [68] can improve performance. In that case, the loss function becomes

$$L(\phi_j) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}, a'\sim\pi_{\mathsf{A}}(\cdot|s')}\left[\left(Q_{\phi_j}(s,a) - r - \gamma\left(Q^{\text{mean}}_{\hat{\phi}}(s',a') - \rho Q^{\text{std}}_{\hat{\phi}}(s',a')\right)\right)^2\right], \tag{48}$$

where ϕ_j denotes the parameters of the jth critic, $Q^{\text{mean}}_{\hat{\phi}}(s',a') \coloneqq \frac{1}{K}\sum_k Q_{\hat{\phi}_k}(s',a')$, $Q^{\text{std}}_{\hat{\phi}}(s',a') = \sqrt{\frac{1}{K}\sum_k \left(Q_{\hat{\phi}_k}(s',a') - Q^{\text{mean}}_{\hat{\phi}}(s',a')\right)^2}$, and ρ is the pessimistic backup weight.

Entropy-regularized Policy Optimization via Adjoint Matching. We adapt the adjoint-matching implementation [16], retaining its endpoint regularization and final noiseless reference-flow step. We discretize the forward SDE and backward adjoint ODE on $t_i = i\Delta t$, with $\Delta t = 1/N$ and $N = 10$. To avoid singular endpoint coefficients, we replace the denominator t_i by $t_i + \Delta t$ and use the regularized noise schedule $\bar{\sigma}_{t_i} = \sigma_0\sqrt{2(1-t_i+\Delta t)/(t_i+\Delta t)}$. Starting from $w_0 \sim \mathcal{N}(0, \sigma_0^2 I)$, the first $N-1$ steps use Euler integration. The final step uses the reference flow and then projects its output onto the latent ball, ensuring $w_1 \in \mathcal{B}_l^d$:

$$\begin{aligned} w_{t_{i+1}} &\leftarrow w_{t_i} + \left(2v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_{t_i}, t_i; s) - \frac{w_{t_i}}{t_i + \Delta t}\right)\Delta t + \sqrt{\Delta t}\,\bar{\sigma}_{t_i} n_i, \quad i = 0, \dots, N-2, \\ w_1 &\leftarrow \Pi_{\mathcal{B}_l^d}\left(w_{t_{N-1}} + \Delta t\, v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}(w_{t_{N-1}}, t_{N-1})\right), \end{aligned} \tag{49}$$

Algorithm 1 LASER

while not converged **do**

▷ *Train Critic*

Sample batch $\{(s, a, r, s')\} \sim \mathcal{D}$

$w \sim \mathcal{N}(0, \sigma_0^2 I_d)$

$a' = \mu^{s'}_{\theta_{\mathsf{L}\to\mathsf{A}}}(\mu^{s'}_{\theta_{\mathsf{B}\to\mathsf{L}}}(w))$

Update ϕ to minimize $L(\phi) = \mathbb{E}[\big(r(s,a) + \gamma Q^{\pi_\mathsf{A}}_{\hat{\phi}}(s', a') - Q^{\pi_\mathsf{A}}_{\phi}(s,a)\big)^2]$ (2)

▷ *Train vector field $v_{\theta_{\mathsf{L}\to\mathsf{A}}}$ in decoder $\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}$*

$x^0 \sim q_\mathsf{L}$

$x^1 \leftarrow a$

$t \sim \mathcal{U}([0,1])$

$x^t \leftarrow (1-t)x^0 + tx^1$

Update $\theta_{\mathsf{L}\to\mathsf{A}}$ to minimize $L_{\mathsf{L}\to\mathsf{A}}(\theta_{\mathsf{L}\to\mathsf{A}}) = \mathbb{E}\left[\|v_{\theta_{\mathsf{L}\to\mathsf{A}}}(x^t, t; s) - (x^1 - x^0)\|^2\right]$ (8)

▷ *Train vector field $v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}$ to match $q^{\text{unif}}_{\mathsf{L}}$*

$x^0 \sim q_\mathsf{B}$

$x^1 \sim q^{\text{unif}}_\mathsf{L}$

$t \sim \mathcal{U}([0,1])$

$x^t \leftarrow (1-t)x^0 + tx^1$

Update $v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}$ to minimize $\mathbb{E}\left[\|v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}(x^t, t) - (x^1 - x^0)\|^2\right]$

▷ *Train Latent-Space Policy using Adjoint Matching*

Set $\Delta t = 1/N$ and $t_i = i\Delta t$ for $i = 0, \dots, N$

$w_0 \sim \mathcal{N}(0, \sigma_0^2 I)$

Forward simulate using (49) to obtain $\{w_{t_i}\}_{i=1}^N$, held fixed during policy regression.

$\tilde{g}_1 \leftarrow -\frac{1}{\alpha}\nabla_{w_1} Q^{\pi_\mathsf{L}}_{\hat{\phi}}(s, w_1)$

Compute $\tilde{g}_{t_i}$ for $i = N-1, \dots, 0$ using (50)

Compute detached regression targets $\hat{v}^\star_{t_i}$ for $i = 0, \dots, N-1$ using (51)

Update $v_{\theta_{\mathsf{B}\to\mathsf{L}}}$ to minimize

$$L_{\text{AM}}(\theta_{\mathsf{B}\to\mathsf{L}}) = \mathbb{E}\left[\frac{1}{2N}\sum_{i=0}^{N-1}\left\|\frac{2}{\bar{\sigma}_{t_i}}\left(v_{\theta_{\mathsf{B}\to\mathsf{L}}}(w_{t_i}, t_i; s) - \hat{v}^\star_{t_i}\right)\right\|^2\right] \quad (20)$$

return policy π_A

where $n_i \sim \mathcal{N}(0, I)$ and $\Pi_{\mathcal{B}_l^d}$ is Euclidean projection. We initialize the lean adjoint with $\tilde{g}_1 = -\nabla_{w_1} Q^{\pi_\mathsf{L}}_{\hat{\phi}}(s, w_1)/\alpha$. Writing $b_i(w) = 2v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}(w, t_{i+1}) - w/t_{i+1}$, the backward update is

$$\tilde{g}_{t_i} \leftarrow \tilde{g}_{t_{i+1}} + \Delta t\, \text{VJP}\left(\nabla_w b_i(w_{t_i}), \tilde{g}_{t_{i+1}}\right), \qquad i = N-1, \dots, 0, \quad (50)$$

where $\text{VJP}(J, g) = J^\top g$. Regression uses only $t_0, \dots, t_{N-1}$, excluding $t_N = 1$, with targets

$$\hat{v}^\star_{t_i} = \text{sg}\left(v^{\text{base}}_{\mathsf{B}\to\mathsf{L}}(w_{t_i}, t_i) - \frac{\bar{\sigma}^2_{t_i}}{2}\tilde{g}_{t_i}\right), \qquad i = 0, \dots, N-1, \quad (51)$$

where sg denotes stop-gradient. The sampled states and regression targets are held fixed when differentiating the loss in Algorithm 1, so gradients pass only through $v_{\theta_{\mathsf{B}\to\mathsf{L}}}$. To stabilize training, we clip terminal latent critic gradients before initializing $\tilde{g}_1$. For each minibatch, we compute the Euclidean norm of each sample's gradient $\nabla_{w_1} Q^{\pi_\mathsf{L}}_{\hat{\phi}}(s, w_1)$ and rescale gradients whose norms exceed the minibatch's 90th percentile so that their norms equal that threshold, while preserving their directions. We then divide all gradients by their mean norm, floored at 10^{-6}, before multiplying by $-1/\alpha$. This operation leaves environment rewards unchanged and is separate from clipping network-parameter gradients. For critic targets and evaluation, we numerically integrate the latent policy ODE and project its terminal output before decoding. Denoting the numerical ODE map by $\widehat{\mu}^s_{\theta_{\mathsf{B}\to\mathsf{L}}}$, the deployed latent policy is $(\Pi_{\mathcal{B}_l^d} \circ \widehat{\mu}^s_{\theta_{\mathsf{B}\to\mathsf{L}}})_\sharp q_\mathsf{B}$. Projection enforces latent support, but if it places positive probability on the boundary, the resulting law has a singular component and no Lebesgue density. Thus, the entropy identity (12) and the optimality characterization in Lemma 3.2

do not directly apply to this projected law. The projected implementation is a practical approximation to the density-based formulation.

E Details of the Reach-avoid Example in Figure 1

The agent starts in one of two regions at the bottom and must reach the goal at the top. Actions are two-dimensional velocities. If a proposed next state lies inside an obstacle, we set $s' = s$. The "No Entropy Regularization" baseline is constructed by replacing the adjoint matching component in LASER with plain gradient ascent, and the generated latent samples z are squashed inside the ball $\mathcal{B}_l^d$ to satisfy the support constraint. The "No Support Constraint" baseline is constructed with a QAM [25] model with a large inverse temperature τ that drives the action distribution away from the behavior policy distribution.

F Experiments

F.1 Environments

We conduct experiments on the widely used OGBench offline RL benchmark [18]. Although OGBench is designed for offline goal-conditioned RL, it provides the -singletask flag for standard offline RL. We use 4 environments in the main comparison, including one locomotion environment (antmaze) and three manipulation environments (cube-single, cube-double, and scene), each with 5 tasks. For each task, we consider 2 datasets, yielding a total of 40 tasks.

We consider the following tasks with CLEAN datasets, i.e., the demonstrations are randomly generated by an expert policy:

- antmaze-large-navigate-singletask-task{1, 2, 3, 4, 5}-v0
- cube-single-play-singletask-task{1, 2, 3, 4, 5}-v0
- cube-double-play-singletask-task{1, 2, 3, 4, 5}-v0
- scene-play-singletask-task{1, 2, 3, 4, 5}-v0

In addition, we consider the following tasks with NOISY datasets, i.e., the demonstrations are generated by a highly noisy policy:

- antmaze-large-explore-singletask-task{1, 2, 3, 4, 5}-v0
- cube-single-noisy-singletask-task{1, 2, 3, 4, 5}-v0
- cube-double-noisy-singletask-task{1, 2, 3, 4, 5}-v0
- scene-noisy-singletask-task{1, 2, 3, 4, 5}-v0

More details of the environments and datasets can be found in [18].

F.2 Baselines

In this section, we briefly introduce the details of each baseline.

FQL. FQL [23] is an algorithm that uses the statistical distance regularization objective (4), using a squared action-matching penalty that upper-bounds the squared 2-Wasserstein distance to its learned BC policy. It begins by training the decoder $\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}}$ via the flow-matching loss (8), assuming a standard normal latent distribution $q_{\mathsf{L}} = \mathcal{N}(0, I)$. Subsequently, it fine-tunes this decoder by learning a *one-step* distilled policy $\mu_{\theta'_{\mathsf{L}\to\mathsf{A}}}(z, s) : \mathcal{L} \times \mathcal{S} \to \mathcal{A}$. This policy is optimized to maximize expected return while remaining anchored to the behavior cloning (BC) decoder $\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}}$. Specifically, the one-step policy minimizes the following loss:

$$\begin{aligned} L(\theta'_{\mathsf{L}\to\mathsf{A}}) = \mathbb{E}_{s\sim D, z\sim q_{\mathsf{L}}}\left[-Q_\phi^{\mu_{\theta'_{\mathsf{L}\to\mathsf{A}}}}(s, \mu_{\theta'_{\mathsf{L}\to\mathsf{A}}}(z, s))\right] \\ + \alpha\mathbb{E}_{s\sim D, z\sim q_{\mathsf{L}}}\left[\|\mu_{\theta'_{\mathsf{L}\to\mathsf{A}}}(z, s) - \mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}}(z)\|^2\right], \end{aligned} \tag{52}$$

where the first term optimizes task performance, and the second term is an upper-bound surrogate for their squared 2-Wasserstein distance [23]. Although the one-step policy is less expressive than the original flow policy [24], training this policy to maximize the Q-value avoids the BPTT process. However, the Wasserstein distance regularization is not enough to avoid OOD actions [7].

QAM. QAM [25] is an algorithm that uses the statistical distance regularization objective (4), instantiating $D(\pi_\theta \parallel \pi_\beta)$ as the KL divergence. It begins by training the decoder $\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}}$ via the flow-matching loss (8), assuming a standard normal latent distribution $q_{\mathsf{L}} = \mathcal{N}(0, I)$. Subsequently, it fine-tunes the decoder via adjoint matching to align with the target tilted distribution $\pi_\beta(\cdot|s)\exp(\tau Q^{\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}}(s,\cdot))$. This procedure is mathematically equivalent to optimizing the following KL-divergence-regularized objective:

$$L(\theta) = -\mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\mu_{\theta_{\mathsf{L}\to\mathsf{A}}}}(s,a)\right] - \frac{1}{\tau}D_{\mathrm{KL}}(\pi_\theta(\cdot|s) \parallel \pi_\beta(\cdot|s))\right], \tag{53}$$

where $\pi_\theta(\cdot|s) = (\mu^s_{\theta'_{\mathsf{L}\to\mathsf{A}}})_\sharp q_{\mathsf{L}}$, and $\mu^s_{\theta'_{\mathsf{L}\to\mathsf{A}}}$ is the fine-tuned flow decoder. QAM avoids the BPTT process by leveraging the adjoint matching technique, similar to LASER, but the KL divergence regularization can limit the policy improvement [7].

QAM-E. QAM-E [25] extends QAM with an edit policy. After fine-tuning the decoder $\mu^s_{\theta'_{\mathsf{L}\to\mathsf{A}}}$, it learns an edit policy $\pi_\omega(a|s,a) : \mathcal{S}\times\mathcal{A}\to\Delta(\mathcal{A})$ that modifies the action generated by $\pi_\theta(\cdot|s) = (\mu^s_{\theta'_{\mathsf{L}\to\mathsf{A}}})_\sharp q_{\mathsf{L}}$ by at most σ_a to further improve performance. In practice, the edit policy π_ω is parameterized as a squashed Gaussian policy that only outputs values in $[-\sigma_a, \sigma_a]$. Empirically, QAM-E outperforms QAM [25], but it is sensitive to σ_a, and the edit policy distribution may generate OOD actions.

IFQL. Presented in [23], IFQL is the flow version of the implicit diffusion Q-learning (IDQL) algorithm [34]. Similarly to the aforementioned algorithms, it first learns the decoder $\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}}$ via the flow-matching loss (8), assuming $q_{\mathsf{L}} = \mathcal{N}(0, I)$. However, instead of learning the policy Q-function with an actor-critic structure with loss (2), it learns an approximately optimal critic via implicit Q-learning (IQL) [27] using the following losses:

$$L(\phi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[(Q_\phi(s,a) - r - \gamma V_\xi(s'))^2\right], \tag{54}$$

$$L(\xi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\kappa(Q_{\hat{\phi}}(s,a) - V_\xi(s))\right], \tag{55}$$

where $L_2^\kappa(u) = |\kappa - \mathbb{1}(u<0)|u^2$ is the expectile regression loss. We use $\kappa = 0.9$ for all tasks. The bootstrap term $\gamma V_\xi(s')$ is masked out for terminal transitions. To extract a policy from the learned Q-function and the decoder, IFQL uses best-of-N sampling with $N = 32$ candidates:

$$\begin{aligned} i^\star &= \operatorname*{arg\,max}_{i\in\{1,\dots,N\}} Q(s,a_i), \qquad a^\star = a_{i^\star}, \\ a_1,\dots,a_N &\overset{\text{i.i.d.}}{\sim} (\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}})_\sharp q_{\mathsf{L}}. \end{aligned} \tag{56}$$

IFQL satisfies the support constraint by construction, but the candidate search is inefficient in high-dimensional spaces with a complex Q-function.

DSRL. DSRL [8] is a recently proposed latent-space RL algorithm, which has several key differences compared with LASER. First, the latent distribution of the decoder is a standard Gaussian in DSRL, i.e., $q_{\mathsf{L}} = \mathcal{N}(0, I)$. Because the support of q_{L} is unbounded, steering q_{L} might generate a distribution that has high density far from the origin, where the samples are not well represented during the training of the decoder. To avoid this problem, DSRL parameterizes the latent policy π_{L} with a squashed Gaussian distribution that only generates samples within the range of $[-\sigma_z, \sigma_z]$, where σ_z is a hyperparameter that requires fine-tuning for each task. DSRL follows an SAC-style loss [12] and explicitly regularizes the entropy, which is straightforward for a Gaussian latent policy. However, because of the Gaussian parameterization, DSRL has a suboptimal entropy-performance tradeoff as shown in Figure 3.

ReFORM. ReFORM [7] is another recently proposed latent-space RL algorithm, which also uses a flow latent policy. ReFORM learns its BC decoder $\mu^s_{\theta_{\mathsf{L}\to\mathsf{A}}}$ in the same way as LASER. However, unlike LASER, ReFORM does not train the latent policy via adjoint matching, but via plain gradient descent.

Table 1: **Fixed common hyperparameters.** These hyperparameters are shared by LASER and all the baselines and are fixed in all environments.

Hyperparameter	Value
Learning rate	0.0003
Optimizer	Adam [69]
Maximum gradient norm	1 (QAM, QAM-E), 10 (others)
Target network smoothing coefficient	0.005
Discount factor γ	0.995
MLP dimensions	$[512, 512, 512, 512]$
Nonlinearity	GELU [70]
Flow integration steps	10
Flow time sampling distribution	$\mathcal{U}[0, 1]$
Minibatch size	256
Number of critics in the ensemble	8
Q-function aggregation method	mean
Actor layer normalization [71]	False
Critic layer normalization [71]	True

In addition, ReFORM does not have the explicit entropy regularization term, resulting in the following loss:

$$L_{\mathsf{B}\to\mathsf{L}}(\theta_{\mathsf{B}\to\mathsf{L}}) = -\mathbb{E}_{s\sim\mathcal{D},w\sim q_{\mathsf{B}}}\left[Q^{\pi_{\mathsf{L}}}(s, \mu_{\theta_{\mathsf{B}\to\mathsf{L}}}(w; s))\right]. \tag{57}$$

Directly using gradient descent on this loss cannot guarantee satisfaction of the latent-space support constraint (6), as the flow $\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}$ can drive the samples outside the closed ball $\mathcal{B}_l^d$. To address this issue, ReFORM modifies the Euler integration step to (41) when integrating the flow $\mu_{\theta_{\mathsf{B}\to\mathsf{L}}}$, and proves that this modified step guarantees satisfaction of the latent-space support constraint. However, as discussed in Appendix C.1, this modified step can cause "static flow" when a proposal leaves the ball and the velocity aligns with the outward normal. In our toy example, this acts as implicit entropy regularization but yields a suboptimal action distribution.

F.3 Hyperparameters

We organize the hyperparameters into three distinct categories. **Fixed common hyperparameters** (Table 1) are shared across all algorithms and remain constant across all tasks. We do not fine-tune these, but instead adopt the standard values established in prior work [23, 25, 7]. **Environment-specific common hyperparameters** (Table 2) are also shared across algorithms but vary by environment and dataset to optimize baseline performance. Following [25], we perform a minimal sweep over two candidate values for these settings: 1 or 5 for the action chunking length, and 0 or 0.5 for the pessimistic backup weight. **Environment-specific unique hyperparameters** (Table 3) are specific to individual baselines (namely FQL, DSRL, QAM, and QAM-E) and require per-environment tuning. For these, we rely on the optimal values reported in their respective papers or tune them empirically. One notable exception is the edit scale σ_a for QAM-E. While the original authors sometimes set $\sigma_a = 0$ [25], making QAM-E identical to QAM, we enforce a minimum value of $\sigma_a = 0.01$ to ensure the algorithms remain distinct. Finally, while LASER introduces two unique hyperparameters, we keep them strictly fixed across all tasks. Specifically, we set $1/\alpha = 10$ and choose σ_0 such that $\Pr_{w\sim\mathcal{N}(0,\sigma_0^2 I)}(\|w\| \le l) = 0.999$. For LASER and ReFORM, the latent ball radius is $l = \sqrt{d}$, where d is the action or action chunk dimension [7].

F.4 Computational Resources

Most of our experiments are run on a 13th Gen Intel(R) Core(TM) i7-13700KF CPU with 64 GB of RAM and an NVIDIA GeForce RTX 4090 GPU. Table 4 reports training throughput. For a fixed iteration budget, LASER requires about 12% more training time than ReFORM and 33% more than QAM, while its throughput is comparable to QAM-E (123 vs 125 $\mathrm{it/s}$). The additional work includes computing lean adjoints and learning a latent policy alongside the decoder. Note that the lean-adjoint solve is needed only during offline training. At deployment, actions are generated by integrating the learned latent flow and then the decoder, without adjoints or critic gradients.

Table 2: **Environment-specific common hyperparameters.** These hyperparameters are shared by LASER and all baselines, but vary by environment and dataset.

Environment	Dataset	Action chunk length	Pessimistic backup weight ρ
antmaze-large-navigate	CLEAN	1	0.5
antmaze-large-explore	NOISY	1	0.5
cube-single-play	CLEAN	5	0.5
cube-single-noisy	NOISY	1	0
cube-double-play	CLEAN	5	0.5
cube-double-noisy	NOISY	1	0
scene-play	CLEAN	5	0.5
scene-noisy	NOISY	5	0.5
puzzle-4x4-play	CLEAN	5	0.5

Table 3: **Environment- and dataset-specific unique hyperparameters.** These hyperparameters are specific to individual algorithms and are tuned by environment and dataset. The method-specific hyperparameters of LASER, ReFORM, and IFQL remain fixed across environments, while their common settings follow Table 2.

Environment	Dataset	FQL (α)	DSRL (σ_z)	QAM (τ)	QAM-E (τ, σ_a)
antmaze-large-navigate	CLEAN	3.0	1.25	10.0	(1.0, 0.1)
antmaze-large-explore	NOISY	0.3	1.25	10.0	(1.0, 0.1)
cube-single-play	CLEAN	300	0.5	1.0	(1.0, 0.01)
cube-single-noisy	NOISY	30	0.5	1.0	(3.0, 0.5)
cube-double-play	CLEAN	300	1.5	1.0	(1.0, 0.01)
cube-double-noisy	NOISY	30	1.5	1.0	(1.0, 0.01)
scene-play	CLEAN	300	0.75	1.0	(1.0, 0.01)
scene-noisy	NOISY	30	0.75	1.0	(1.0, 0.01)
puzzle-4x4-play	CLEAN	1.0	1.0	30.0	(0.1, 0.9)

F.5 Additional Results

Ablation study of entropy regularization. To test the efficacy of entropy regularization beyond the simple environment introduced in Section 3.1, we extend the entropy regularization comparison to cube-single and cube-double environments with both CLEAN and NOISY datasets. NoER uses the same latent support constraint, network architectures, critic design, and other training settings as LASER, but directly maximizes Q through the latent flow using reparametrized gradients without entropy regularization. ReFORM instead uses a reflected flow with implicit entropy regularization (Appendix C.1), while LASER optimizes an explicitly entropy-regularized objective via adjoint matching. Table 5 reports the results for all five tasks in each environment with both CLEAN and NOISY datasets. We observe that both implicit (ReFORM) and explicit (LASER) entropy regularization help and significantly improve on NoER, and explicit entropy regularization (LASER) performs better than implicit entropy regularization (ReFORM).

Inverse temperature and target policy smoothing. We compare Q-only latent optimization (NoER), target policy smoothing (TPS), and LASER with different inverse temperatures on cube-double task 2 using both CLEAN and NOISY datasets. TPS [20] augments NoER by adding independent zero-mean Gaussian noise with standard deviation 0.2, clipped to $[-0.5, 0.5]$, to the target actions in the critic update. The perturbed actions are then clipped to the action bounds. As shown in Table 6, our default $1/\alpha = 10$ outperforms both NoER and TPS on both datasets. Strong regularization ($1/\alpha = 3$) reduces performance on both datasets, whereas weak regularization ($1/\alpha = 30$) performs poorly on the CLEAN dataset but remains competitive on the NOISY dataset. One possible explanation is that noisy demonstrations provide broader action coverage, reducing critic artifacts.

Uniform-mixture ablation in action space. LASER combines entropy optimization with bounded latent support. To examine a simple alternative in action space, we evaluate QAM+Mix, which broadens QAM's reference toward a uniform distribution over the action space. With probability η, each

Table 4: **Training throughput comparison.** We report the number of iterations per second (it/s) for LASER and all the baselines.

Algorithm	FQL	QAM	QAM-E	IFQL	DSRL	ReFORM	LASER
it/s	220	163	125	210	165	138	123

Table 5: **Entropy regularization ablation on 20 cube-single and cube-double tasks.** Success rates (%) are reported as mean ± standard deviation across 3 seeds, with 50 evaluation runs per seed. Bold denotes results at or above 95% of the best mean in each setting.

Task	Dataset	NoER	ReFORM	LASER
cube-single-play-singletask-task1-v0	CLEAN	$69_{\pm 27}$	$\mathbf{99}_{\pm 2}$	$\mathbf{100}_{\pm 0}$
cube-single-play-singletask-task2-v0	CLEAN	$59_{\pm 36}$	$\mathbf{96}_{\pm 3}$	$\mathbf{100}_{\pm 0}$
cube-single-play-singletask-task3-v0	CLEAN	$87_{\pm 5}$	$\mathbf{98}_{\pm 3}$	$\mathbf{100}_{\pm 0}$
cube-single-play-singletask-task4-v0	CLEAN	$51_{\pm 18}$	$87_{\pm 2}$	$\mathbf{99}_{\pm 1}$
cube-single-play-singletask-task5-v0	CLEAN	$74_{\pm 14}$	$86_{\pm 13}$	$\mathbf{93}_{\pm 5}$
cube-single-noisy-singletask-task1-v0	NOISY	$76_{\pm 6}$	$\mathbf{100}_{\pm 0}$	$\mathbf{100}_{\pm 0}$
cube-single-noisy-singletask-task2-v0	NOISY	$86_{\pm 3}$	$\mathbf{100}_{\pm 0}$	$\mathbf{100}_{\pm 0}$
cube-single-noisy-singletask-task3-v0	NOISY	$91_{\pm 7}$	$\mathbf{100}_{\pm 0}$	$\mathbf{100}_{\pm 0}$
cube-single-noisy-singletask-task4-v0	NOISY	$45_{\pm 15}$	$\mathbf{100}_{\pm 0}$	$\mathbf{100}_{\pm 0}$
cube-single-noisy-singletask-task5-v0	NOISY	$59_{\pm 7}$	$\mathbf{100}_{\pm 0}$	$\mathbf{100}_{\pm 0}$
cube-double-play-singletask-task1-v0	CLEAN	$73_{\pm 29}$	$91_{\pm 7}$	$\mathbf{97}_{\pm 1}$
cube-double-play-singletask-task2-v0	CLEAN	$55_{\pm 7}$	$83_{\pm 7}$	$\mathbf{89}_{\pm 1}$
cube-double-play-singletask-task3-v0	CLEAN	$63_{\pm 12}$	$73_{\pm 7}$	$\mathbf{81}_{\pm 7}$
cube-double-play-singletask-task4-v0	CLEAN	$29_{\pm 10}$	$22_{\pm 4}$	$\mathbf{49}_{\pm 12}$
cube-double-play-singletask-task5-v0	CLEAN	$36_{\pm 21}$	$54_{\pm 13}$	$\mathbf{70}_{\pm 6}$
cube-double-noisy-singletask-task1-v0	NOISY	$61_{\pm 15}$	$80_{\pm 3}$	$\mathbf{92}_{\pm 2}$
cube-double-noisy-singletask-task2-v0	NOISY	$51_{\pm 12}$	$29_{\pm 14}$	$\mathbf{81}_{\pm 10}$
cube-double-noisy-singletask-task3-v0	NOISY	$27_{\pm 5}$	$29_{\pm 3}$	$\mathbf{61}_{\pm 12}$
cube-double-noisy-singletask-task4-v0	NOISY	$50_{\pm 3}$	$9_{\pm 2}$	$\mathbf{65}_{\pm 5}$
cube-double-noisy-singletask-task5-v0	NOISY	$27_{\pm 6}$	$11_{\pm 6}$	$\mathbf{38}_{\pm 13}$

Table 6: **Comparison of NoER, TPS, and LASER with different inverse temperatures on cube-double task 2.** Success rates (%) are reported as mean ± standard deviation across 3 seeds, with 50 evaluation runs per seed. Bold denotes results at or above 95% of the best mean for each dataset.

Dataset	NoER	TPS	LASER ($1/\alpha$)				
			$1/\alpha = 3$	$1/\alpha = 5$	$1/\alpha = 10$	$1/\alpha = 20$	$1/\alpha = 30$
CLEAN	$55_{\pm 7}$	$49_{\pm 24}$	$50_{\pm 14}$	$78_{\pm 19}$	$\mathbf{89}_{\pm 1}$	$79_{\pm 12}$	$30_{\pm 42}$
NOISY	$51_{\pm 12}$	$61_{\pm 17}$	$31_{\pm 19}$	$72_{\pm 15}$	$\mathbf{81}_{\pm 10}$	$\mathbf{83}_{\pm 18}$	$\mathbf{83}_{\pm 19}$

behavior-cloning target is replaced by an action sampled uniformly within $\mathcal{A}$. Adjoint matching then uses the reference flow learned from these modified targets. In the ideal KL-regularized formulation, $\eta = 0$ recovers QAM, while $\eta = 1$ yields action-space maximum-entropy optimization. Thus, η controls how strongly the reference distribution is broadened toward uniform. On cube-double-play-singletask-task2-v0 with the CLEAN dataset, both tested mixing probabilities ($\eta = 0.03$ and $\eta = 0.1$) reduce QAM's mean success rate (Table 7). The results show that this naïve approach does not perform well.

Table 7: **Uniform-mixture ablation in action space** on cube-double task 2 with the CLEAN dataset. Success rates (%) are reported as mean ± standard deviation across 3 seeds, with 50 evaluation runs per seed. Bold denotes results at or above 95% of the best mean.

QAM	QAM+Mix ($\eta = 0.03$)	QAM+Mix ($\eta = 0.1$)	LASER
$83_{\pm 11}$	$38_{\pm 10}$	$9_{\pm 6}$	$\mathbf{89}_{\pm 1}$

Evaluation on harder tasks. We compare LASER with all six baselines on the five puzzle-4x4 tasks using the CLEAN (100M) dataset. Every run uses two million updates, and we evaluate the final checkpoint for each method and seed. The action chunk length is 5, giving a 25-dimensional action chunk, and the pessimistic backup weight is $\rho = 0.5$. Baseline-specific settings are listed in Table 3. As shown in Table 8, LASER still achieves high performance on these more challenging tasks.

Table 8: **Results on the five puzzle-4x4 tasks with the CLEAN dataset (100M).** Success rates (%) are reported as mean ± standard deviation across 3 seeds, with 50 evaluation runs per seed. Bold denotes results at or above 95% of the best mean for each task.

Task	FQL	QAM	QAM-E	IFQL	DSRL	ReFORM	LASER
1	$5_{\pm 4}$	$2_{\pm 2}$	$63_{\pm 14}$	$50_{\pm 13}$	$65_{\pm 24}$	$86_{\pm 6}$	$\mathbf{97}_{\pm 3}$
2	$13_{\pm 9}$	$0_{\pm 0}$	$74_{\pm 24}$	$2_{\pm 0}$	$17_{\pm 7}$	$6_{\pm 7}$	$\mathbf{87}_{\pm 2}$
3	$5_{\pm 7}$	$3_{\pm 2}$	$18_{\pm 12}$	$\mathbf{87}_{\pm 1}$	$63_{\pm 41}$	$81_{\pm 7}$	$\mathbf{84}_{\pm 7}$
4	$1_{\pm 1}$	$0_{\pm 0}$	$33_{\pm 15}$	$32_{\pm 4}$	$31_{\pm 21}$	$33_{\pm 25}$	$\mathbf{60}_{\pm 3}$
5	$0_{\pm 0}$	$1_{\pm 1}$	$53_{\pm 26}$	$2_{\pm 3}$	$14_{\pm 14}$	$6_{\pm 3}$	$\mathbf{69}_{\pm 18}$

Boundary-hit diagnostic for ReFORM. We evaluate the final ReFORM checkpoints on all five cube-double tasks with both CLEAN and NOISY datasets. At each policy call, we record the fraction of $N = 10$ latent Euler proposals satisfying $\|\hat{w}_{k+1}\| > l$, before applying the reflection correction in (41). We average these rates over policy calls within each episode, then over episodes and seeds. Table 9 reports the results. The high rates (97.4%–99.0%) show that boundary correction is frequently active, consistent with the proposed static-flow mechanism in Appendix C.1.

Table 9: **Boundary-hit rates (%) for ReFORM on cube-double.** Results are averaged across 3 seeds, with 50 evaluation episodes per seed and 10 latent flow steps per policy call.

Dataset	Task 1	Task 2	Task 3	Task 4	Task 5
CLEAN	98.2	99.0	98.9	97.6	98.0
NOISY	97.4	97.7	97.9	97.5	97.9

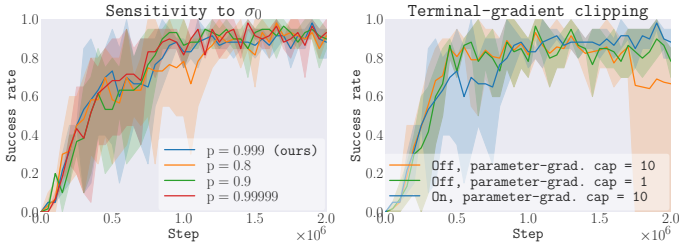

Figure 8: **Additional ablation studies.** LASER demonstrates robustness to the choice of σ_0. Furthermore, applying terminal-gradient clipping improves training stability. Reducing the parameter-gradient cap from 10 to 1 yields a comparable stabilizing effect. Lines show mean success rates, and shading spans the minimum and maximum across runs.

Alternative latent prior. For the decoder-prior ablation in Figure 6, we use

$$q_{\mathsf{L}}^{\mathrm{mix}} = 0.4\,\mathcal{U}(\mathcal{B}_l^d) + 0.3\,(\Pi_{\mathcal{B}_l^d})_\sharp \mathcal{N}(0.5e_1, 0.1^2 I) + 0.3\,(\Pi_{\mathcal{B}_l^d})_\sharp \mathcal{N}(0.4e_d, 0.15^2 I),$$

where e_i is the ith coordinate vector and Gaussian samples outside the ball are projected onto its boundary. This experiment uses $d = 25$ and $l = 5$. Only the decoder's training prior changes. The latent policy's entropy reference remains $q_{\mathsf{L}}^{\mathrm{unif}} = \mathcal{U}(\mathcal{B}_l^d)$.

Ablation study of the standard deviation of the base distribution. We study the sensitivity of LASER w.r.t. the standard deviation σ_0 of q_{B}. We vary σ_0 such that the ball $\mathcal{B}_l^d$ contains probability mass p under the Gaussian distribution $\mathcal{N}(0, \sigma_0^2 I)$. We report the results in Figure 8. The results show that LASER is not sensitive to σ_0.

Ablation study of the gradient clipping technique. A known drawback of adjoint matching is its tendency to produce large gradients, which can destabilize the training process [16, 25]. In Figure 8, we compare three variants on cube-double task 2 with CLEAN data: terminal-gradient clipping disabled with a parameter-gradient cap of 10, disabled with a cap of 1, and enabled with a cap of 10 (our default). Both enabling terminal-gradient clipping and reducing the parameter-gradient cap improve stability over the first variant, with comparable performance.

Detailed results. The full results for each algorithm on the 40 main evaluation settings are provided in Table 11. Success rates are bolded for algorithms achieving at least 95% of the best performance per task. For every method and seed, we evaluate the final checkpoint. The main-suite training budgets are 5 million updates for antmaze-large-navigate, 8 million for antmaze-large-explore, 1 million for cube-single-noisy, and 2 million for the remaining settings. The training budgets are selected such that all methods converge. Each checkpoint is evaluated over 50 episodes, and we report the mean and standard deviation across 3 seeds. We observe that LASER yields the best overall performance, achieving near-perfect success rates on multiple tasks.

Statistical comparison. We pair results across the 40 main task–dataset settings, averaging each method's success rate over its 3 seeds before comparison. For each baseline j, a one-sided paired t-test evaluates $H_{0,j} : \Delta_j \le 0$, where Δ_j denotes the expected difference between LASER's and baseline j's setting-level mean success rates. To test for improvement over every baseline, we use an intersection–union test of $H_0 : \bigcup_j \{\Delta_j \le 0\}$ against $H_1 : \Delta_j > 0$ for all j. The global p-value is $\max_j p_j = 1.35 \times 10^{-3}$ (Table 10), rejecting H_0 at the 0.05 level under the paired-test assumptions. This inference treats setting-level differences as independent despite shared environments and datasets, so we interpret it as supporting evidence for aggregate performance on this suite.

Table 10: **Paired statistical comparisons on the 40 main evaluation settings.** Mean gaps are LASER minus each baseline in percentage points. One-sided p-values are computed from unrounded success rates averaged over 3 seeds per setting, with 50 evaluation runs per seed.

Baseline	FQL	QAM	QAM-E	IFQL	DSRL	ReFORM
Mean gap	20.02	18.70	17.75	47.50	16.33	11.82
p-value	9.08×10^{-6}	3.47×10^{-4}	6.44×10^{-4}	3.22×10^{-11}	1.35×10^{-3}	3.42×10^{-5}

Training curves. Figures 9 to 16 show training curves for all algorithms and tasks on both the CLEAN and NOISY datasets. Lines show means over 3 seeds, and shading spans the minimum and maximum across seeds. Each point uses 20 evaluation episodes per seed, whereas the reported table results use a separate evaluation of 50 episodes per seed.

F.6 Code

We provide code on our project website `https://mit-realm.github.io/laser/`.

Table 11: **Full results on 40 OGBench tasks.** Success rates (%) are reported as mean ± standard deviation across 3 seeds, with 50 evaluation episodes per seed. Bold denotes results at or above 95% of the best mean in each setting, following [18]. We omit the -singletask tags from the task names to save space.

Task	Dataset	FQL	QAM	QAM-E	IFQL	DSRL	ReFORM	LASER
antmaze-large-navigate-task1-v0	CLEAN	64±45	**96**±3	**93**±2	79±13	90±3	91±2	**95**±1
antmaze-large-navigate-task2-v0	CLEAN	31±43	60±42	**94**±3	65±5	81±5	78±2	**89**±2
antmaze-large-navigate-task3-v0	CLEAN	71±12	85±3	**97**±1	85±9	**94**±3	74±6	**92**±3
antmaze-large-navigate-task4-v0	CLEAN	33±41	**93**±1	**91**±1	31±43	75±14	69±4	85±1
antmaze-large-navigate-task5-v0	CLEAN	88±6	**95**±1	**93**±4	83±2	**91**±4	83±1	**91**±1
antmaze-large-explore-task1-v0	NOISY	**97**±1	0±0	0±0	50±36	0±0	91±4	**97**±2
antmaze-large-explore-task2-v0	NOISY	**97**±1	0±0	0±0	3±2	0±0	88±9	87±2
antmaze-large-explore-task3-v0	NOISY	**100**±0	0±0	0±0	**95**±1	0±0	**100**±0	**98**±3
antmaze-large-explore-task4-v0	NOISY	**89**±3	0±0	0±0	36±13	0±0	82±9	**89**±2
antmaze-large-explore-task5-v0	NOISY	**94**±6	0±0	0±0	47±34	0±0	**93**±2	**92**±3
cube-single-play-task1-v0	CLEAN	**95**±4	94±3	**95**±4	79±2	95±5	**99**±2	**100**±0
cube-single-play-task2-v0	CLEAN	**97**±1	**99**±1	**99**±2	70±5	90±7	**96**±3	**100**±0
cube-single-play-task3-v0	CLEAN	**99**±1	**100**±0	**99**±1	82±7	**97**±4	**98**±3	**100**±0
cube-single-play-task4-v0	CLEAN	90±4	91±2	85±4	82±3	**98**±2	87±2	**99**±1
cube-single-play-task5-v0	CLEAN	81±2	**89**±4	85±4	75±6	85±3	86±13	**93**±5
cube-single-noisy-task1-v0	NOISY	**100**±0	**100**±0	**100**±0	1±1	93±6	**100**±0	**100**±0
cube-single-noisy-task2-v0	NOISY	**100**±0	**100**±0	**99**±1	7±6	92±9	**100**±0	**100**±0
cube-single-noisy-task3-v0	NOISY	**100**±0	**100**±0	**100**±0	85±9	53±25	**100**±0	**100**±0
cube-single-noisy-task4-v0	NOISY	**100**±0	**100**±0	**100**±0	0±0	91±4	**100**±0	**100**±0
cube-single-noisy-task5-v0	NOISY	**100**±0	**100**±0	**99**±2	0±0	91±5	**100**±0	**100**±0
cube-double-play-task1-v0	CLEAN	78±2	**96**±2	**97**±1	19±2	**95**±1	91±7	**97**±1
cube-double-play-task2-v0	CLEAN	52±3	83±11	83±1	15±4	63±4	83±7	**89**±1
cube-double-play-task3-v0	CLEAN	40±6	69±11	75±6	9±7	63±17	73±7	**81**±7
cube-double-play-task4-v0	CLEAN	7±3	21±1	20±4	1±1	17±5	22±4	**49**±12
cube-double-play-task5-v0	CLEAN	37±10	**81**±10	73±2	8±3	73±1	54±13	70±6
cube-double-noisy-task1-v0	NOISY	53±17	89±4	**97**±2	1±1	87±5	80±3	**92**±2
cube-double-noisy-task2-v0	NOISY	5±2	74±3	**80**±4	0±0	**79**±11	29±14	**81**±10
cube-double-noisy-task3-v0	NOISY	5±2	55±5	53±6	0±0	**60**±5	29±3	**61**±12
cube-double-noisy-task4-v0	NOISY	1±1	43±13	51±2	1±1	58±18	9±2	**65**±5
cube-double-noisy-task5-v0	NOISY	2±3	35±4	30±4	0±0	**40**±11	11±6	**38**±13
scene-play-task1-v0	CLEAN	**100**±0	**100**±0	**100**±0	**99**±1	**100**±0	**100**±0	**100**±0
scene-play-task2-v0	CLEAN	**100**±0	**100**±0	**99**±1	3±2	**100**±0	**100**±0	**100**±0
scene-play-task3-v0	CLEAN	93±5	93±5	94±3	80±4	**99**±1	93±8	**100**±0
scene-play-task4-v0	CLEAN	66±13	16±19	7±4	3±1	**100**±0	60±42	58±29
scene-play-task5-v0	CLEAN	**13**±9	0±0	0±0	0±0	0±0	0±0	0±0
scene-noisy-task1-v0	NOISY	86±12	**100**±0	**100**±0	**99**±2	**100**±0	71±17	**100**±0
scene-noisy-task2-v0	NOISY	70±8	**98**±2	**98**±2	0±0	**96**±3	**97**±4	**100**±0
scene-noisy-task3-v0	NOISY	0±0	25±6	39±16	44±4	**100**±0	45±27	80±27
scene-noisy-task4-v0	NOISY	0±0	8±11	1±1	1±1	37±39	0±0	**67**±31
scene-noisy-task5-v0	NOISY	**0**±0	**0**±0	**0**±0	**0**±0	**0**±0	**0**±0	**0**±0

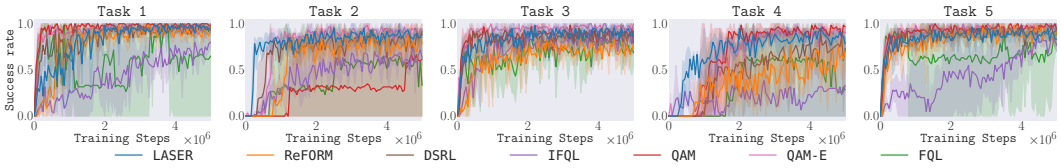

Figure 9: Training curves for the antmaze-large environment with the CLEAN dataset.

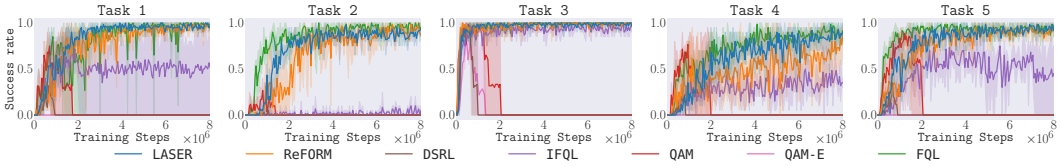

Figure 10: Training curves for the antmaze-large environment with the NOISY dataset.



Figure 11: Training curves for the cube-single environment with the CLEAN dataset.

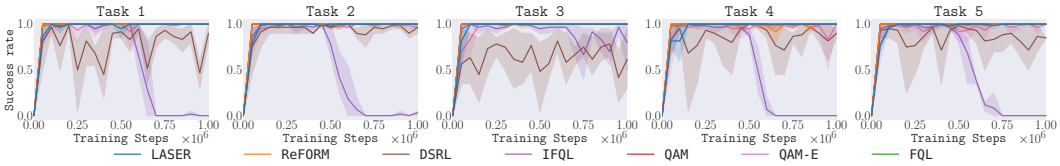

Figure 12: Training curves for the cube-single environment with the NOISY dataset.

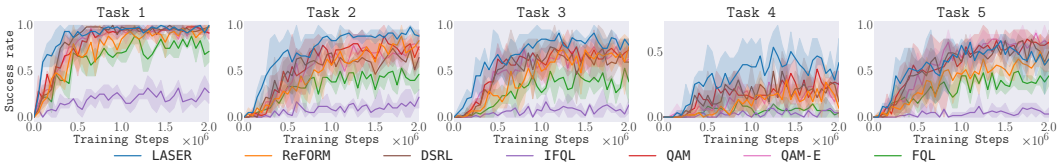

Figure 13: Training curves for the cube-double environment with the CLEAN dataset.

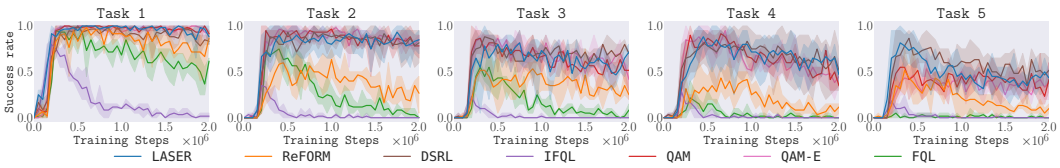

Figure 14: Training curves for the cube-double environment with the NOISY dataset.

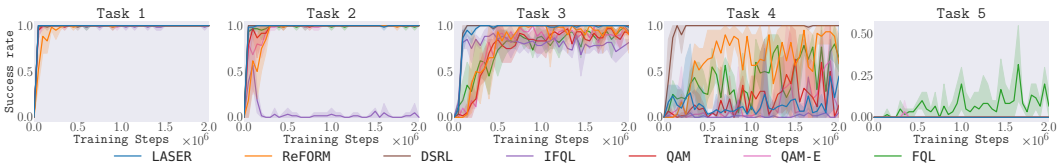

Figure 15: Training curves for the scene environment with the CLEAN dataset.

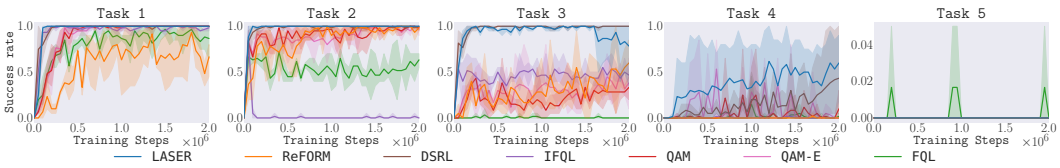

Figure 16: Training curves for the scene environment with the NOISY dataset.

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: All the claims have been justified in the paper.

   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: The limitations are discussed in Section 6.

   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: A full set of assumptions and a complete proof are provided in the appendix.

   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: We have discussed the implementation details and provided hyperparameters in the appendix. We also include code with the submission.

   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: [Yes]

   Justification: We have included code in the supplementary material along with the commands.

   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 details are provided 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: We report seed variability for the main success-rate results and define the uncertainty bands in the figure captions. Appendix F.5 also provides paired statistical comparisons and their assumptions.

   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: We have included this information in the appendix.

   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: The research conducted in the paper conforms, in every respect, with the NeurIPS 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: [Yes]

    Justification: The Broader Impacts paragraph in Section 6 discusses potential benefits for data collection in robotics, risks from unsafe data and deployment errors, and the need for safety validation and monitoring.

    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: The paper does not contain such data or models.

    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: We have cited the original papers.

    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: We provide the commands along with the code.

    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: The paper does not involve crowdsourcing or research with human subjects.

    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: The paper does not involve crowdsourcing or 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: The core method development in this research does not use LLMs as important, original, or non-standard components.

    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.