GSBR: GEOMETRIC SUPPORT BEHAVIOR REGULARIZATION FOR ONLINE RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Regularizing online reinforcement learning (RL) toward a pretrained base policy can improve sample efficiency and learning stability, but regularization that is too restrictive limits how much the policy can improve. Base policies - typically diffusion or flow-matching models trained by imitation learning on large, diverse datasets - capture many behaviors that are reasonable but suboptimal. Existing regularizers using such base policies are too restrictive in one of two ways: behavioral cloning regularizers pull the actor toward the full base-policy distribution, penalizing it for discarding suboptimal modes, while hard support constraints forbid improvements that lie just outside the base policy's support. We propose Geometric Support Behavior Regularization (GSBR), which penalizes actions by their distance from the base policy's support. Actions within the support incur no penalty, so the actor can concentrate on high-value modes; outside the support, where value estimates are less reliable, the penalty grows smoothly with distance while still permitting improvement when the value gained outweighs the penalty. GSBR estimates this distance from a finite set of base-policy samples, so it requires only sample access to the base policy. Across 19 tasks on four robotic benchmarks, GSBR raises the average success rate from 33% for the base policy to 95%, achieving the highest average success rate and throughput among all compared methods, with about 23% higher normalized throughput than the next-best RL method.

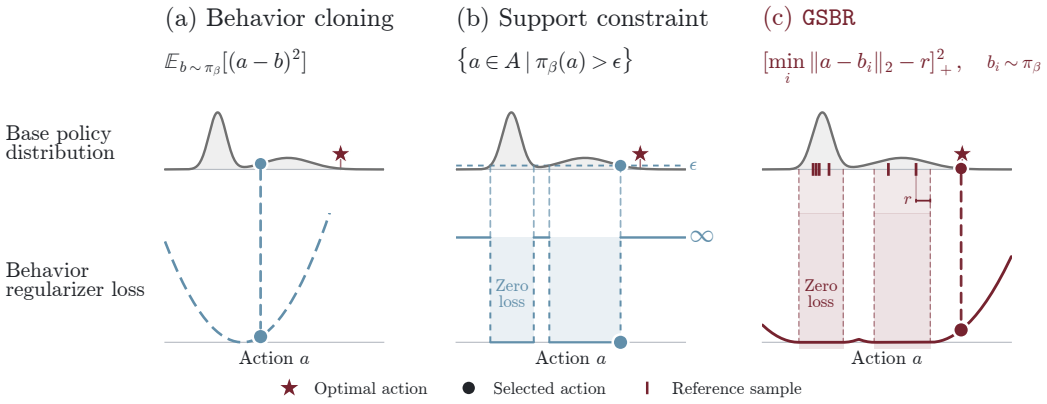

Figure 1: (a) Behavior cloning regularizer matches the base policy distribution, not allowing the RL actor to drop poor modes. (b) Support constraint allows the actor to select the best behavior in the base policy distribution, but excludes improvements beyond it. (c) **GSBR** applies distance-based regularization on deviation outside the base policy support, allowing the actor to discard suboptimal behaviors and guide exploration beyond the prior.

1 INTRODUCTION

Online reinforcement learning (RL) can be costly when every interaction happens on a physical system, so practical methods often rely on a behavior prior that focuses exploration on plausible actions. In robotics, such priors are now available at scale: vision-language-action (VLA) and world-action models (WAMs) learn diverse behaviors from large demonstration datasets (Kim et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2024; Black et al., 2024; Ye et al., 2026), but often lack the robustness and throughput required for reliable deployment (Physical Intelligence et al., 2025; Xu et al., 2026). RL fine-tuning can close this gap by updating a policy that is regularized toward the pretrained base model. This raises a general question for RL: *how should a behavior regularizer preserve the base policy's guidance while giving the actor enough freedom to improve on it?*

Improving upon base policies typically has two requirements: mode selection and local refinement. Consider a scene with two objects and the task is to grasp a specified target. Foundation models trained on large-scale data often produce diverse behaviors - and such base policies might have two action modes, one to grasp each object. The actor should be updated to discard the mode that reaches for the wrong object; we call this *mode selection*. Further, if the base policy can only reach the right object slowly, the actor should increase the action magnitude to move faster. Such gains in speed or precision may require actions outside - although not far from, the base policy's support; we call this "local refinement". However, existing behavior constraints can hinder one of the requirements and restrict policy improvement. Action regression (Fujimoto & Gu, 2021; Park et al., 2025b) and distribution matching regularizer (Li & Levine, 2026; Fang et al., 2025) tie the actor to the base policy's full action distribution, penalizing the actor even for discarding poor modes(Figure 2(a)). Support constraints (Kumar et al., 2019; Wagenmaker et al., 2025; Li et al., 2025) remove this penalty and permit mode selection, but can be too restrictive and often exclude optimal actions outside the base policy's support (Figure 2(b)).

Our key insight is to allow both mode selection and local refinement by introducing a geometric relaxation of the support constraint. We introduce Geometric Support Behavior Regularization (`GSBR`), an RL algorithm that estimates the support of the base policy using finite samples, and regularize actions differently based on whether they are inside or outside the support. Actions inside the estimated support incur no penalty, allowing the actor to discard poor modes and concentrate on better ones without preserving their relative frequencies. Outside the support, `GSBR` replaces the infinite penalty of a hard constraint with a finite one that grows with distance, reflecting that value estimates become less reliable far from the base policy. Because the support of a learned base policy is intractable to compute, we approximate it by the union of neighborhoods around actions sampled from the base policy, with a radius that adapts to the observed samples' distances from each other. `GSBR` provides a practical sample-based geometric support regularizer fitting for the diverse distribution of large-scale robot foundation policies.

Our contributions are (1) a geometric behavior regularizer that permits both mode selection and local refinement in RL finetuning, (2) a sample-based approximation of multi-modal action distribution with an adaptive radius, with error bounds and a guarantee that the error vanishes as the number of reference samples grows, (3) an evaluation across 19 tasks drawn from four benchmarks where `GSBR`raises the average success rate from 33% for the base policy to 95%, achieving the highest average success rate and throughput among all compared methods, with about 23% higher normalized throughput than the next-best RL method. Further analysis shows that our approximation closely matches the true support region and the effect of the sample count and support radius.

2 Related Work

Sample-efficient online RL. Learning from costly environment interaction is a central challenge in robotic RL. Off-policy actor–critic methods address this through experience reuse, high update-to-data ratios, and critic regularization to improve data efficiency (Haarnoja et al., 2018; Chen et al., 2021; Hiraoka et al., 2022). Prior data and demonstrations can also help by initializing policies and values, supplementing replay buffer, and guiding exploration (Ball et al., 2023; Nakamoto et al., 2023; Hu et al., 2023). These algorithmic advances enable practical systems to learn contact-rich manipulation within hours of robot interaction (Luo et al., 2024; 2025; Dong et al., 2026a). `GSBR` **is complementary to these advances**: it further improves sample efficiency through a behavior regularizer that uses a frozen base policy as a prior, reducing unnecessary exploration while allowing policy improvement beyond the prior's support.

RL with generative models, diffusion and flow policies. Diffusion and flow policies, including VLAs, represent multimodal behavior, but they require iterative action generation and complicates RL optimization. One family updates the generative model itself, through policy gradients, critic gradients, or denoising supervision (Ren et al., 2024; Wang et al., 2023; Li & Levine, 2026; Lu et al., 2025), or through distillation and advantage conditioning (Park et al., 2025b; Chen et al.,

2025; Mark et al., 2024; Physical Intelligence et al., 2025). Another keeps the model frozen and trains a lightweight actor that corrects its actions (Silver et al., 2018; Johannink et al., 2019; Ankile et al., 2024; Xiao et al., 2025; Xu et al., 2026), steers its latent noise (Wagenmaker et al., 2025), or selects among sampled candidates (Dong et al., 2026b;a; Sun & Song, 2026), sometimes distilling the result back into the base model (Xiao et al., 2025; Intelligence et al., 2026). GSBR is most similar to the second family; methods within it differ mainly in how the actor is tied to the base policy, which we compare next.

Behavior constraints for RL. Behavior constraints guide policy improvement using available experience or a pretrained policy, but differ in how they restrict mode selection and departures from prior behavior.

Distribution matching and action regression. KL regularization penalizes reallocating probability toward higher-value modes (Wu et al., 2019; Fang et al., 2025; Li & Levine, 2026), and regression toward sampled actions favors their mean (Fujimoto & Gu, 2021; Tarasov et al., 2023). FQL and RLT (Park et al., 2025b; Xu et al., 2026) alleviate this mode collapsing problem by explicitly conditioning on noise or reference-action, but their pairwise regularization still ties the actor to the modes of randomly drawn inputs. In contrast, GSBR allows the actor to focus on promising modes and dropping poor behaviors (mode selection) without retaining a particular action pairing.

Support constraints and approximations. Ideal support constraints allow the actor to choose freely among actions with non-negligible density under the behavior prior. It permit mode selection but block local refinement.BEAR, SPOT, and PRDC (Kumar et al., 2019; Wu et al., 2022; Ran et al., 2023) constrain the actor to dataset actions via sample-based MMD or a density estimator. However, they require access to the dataset and do not leverage a pretrained foundation model's broader behavioral repertoire. DSRL (Wagenmaker et al., 2025) act in the base policy's latent-noise space, while best-of-N methods select the highest-value action among sampled actions (Nakamoto et al., 2024; Li et al., 2025; Hansen-Estruch et al., 2023; Li et al., 2025). Both enable mode selection, but remain suboptimal when local refinement is needed. These limitations motivate GSBR, which constructs an unpenalized support region from the base policy distribution and applies a finite geometric penalty beyond it, allowing online exploration to depart from the prior's support.

Residual reinforcement learning. Residual policies allow local refinement, but are often bounded by a prescribed scale (Silver et al., 2018; Johannink et al., 2019; Ankile et al., 2024). BCQ, EXPO and EXPO-FT (Fujimoto et al., 2019; Dong et al., 2026b;a) combine bounded corrections with value-based candidate selection to have both mode selection and local refinement. However, they only allow local refinement up to a *fixed magnitude* set as a hyperparameter, bounding the maximum allowed deviation without any guidance from the base policy. They can be interpreted as a implicit support constraint that admits an enlarged support region by a *fixed radius*. DICE-RL replaces the fixed magnitude by using a selectively dropped out squared-distance regularizer on the edit policy (Sun & Song, 2026). While this method can be viewed as an implicit support constraint with a geometric penalty for deviation, it creates constant optimization pressure that pulls even small edits towards each base policy sample. In contrast, GSBR explicitly approximates the support region for the base policy distribution, permitting in-distribution action to have zero behavior penalty, enabling free mode selection and refinement within the support region while allowing departures at a finite, distance-dependent cost.

3 PRELIMINARIES

Reinforcement learning. We consider a discounted Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma)$ with states $s \in \mathcal{S}$, actions $a \in \mathcal{A} \subseteq \mathbb{R}^d$, transition distribution $P(\cdot \mid s, a)$, reward $r(s, a)$, initial-state distribution ρ, and discount factor $\gamma \in [0, 1)$. The goal is to learn a policy π_θ that maximizes the expected discounted return

$$J(\theta) = \mathbb{E}_{\rho, P, \pi_\theta}[\textstyle\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]. \tag{1}$$

Online RL with a pretrained behavior prior. In designing RL algorithms to interact with the MDP environment and optimize for the return, we leverage a behavior prior to guide exploration and training. We study online policy improvement with a frozen pretrained *base policy* π_β and an offline transition dataset $\mathcal{D}_{\text{off}}$ of *task-specific transitions*, e.g., demonstrations or base-policy rollouts. The

learner collects additional experience through environment interaction and trains on both offline and online transitions. The base policy stays frozen and is accessed only through samples $b \sim \pi_\beta(\cdot \mid s)$; we require no access to its parameters or gradients, so π_β can be any generative model.

Challenges. Three properties make this setting difficult: (1) π_β is typically multimodal and its modes can differ widely in value, which behaviors are worth keeping depends on the task. (2) the base action support is intractable: we can draw samples but cannot enumerate all plausible actions (e.g., too expensive to enumerate for every state). (3) the critic is trained on limited data, its estimates are less reliable when the input is out-of-distribution; an actor that maximizes Q_ϕ without restraint will exploit these errors.

Support and optimality assumption about the base policy. Let $\pi_\beta(a \mid s)$ denote the base policy's conditional action density. We define the support of π_β at state s:

$$S_\beta(s) = \overline{\{a \in \mathcal{A} : \pi_\beta(a \mid s) > 0\}}, \tag{2}$$

and call its elements *in-distribution actions*. We assume only that the in-distribution actions include reasonable behaviors for the task, not that they include the optimal action. We further assume that improvements over π_β can be obtained in two ways: by concentrating on the most valuable supported actions regardless of how often π_β samples them, which we call *mode selection*, and by refining actions near the support, which we call *local refinement*.

Behavior-regularized actor–critic RL. Following classic actor-critic RL algorithms, we will learn two functions: actor $\pi_\theta(\cdot \mid s, \mathcal{B})$ and critic $Q^{\pi_\theta}(s, a)$. The actor $\pi_\theta(\cdot \mid s, \mathcal{B})$ conditions on the state and on a set of N reference actions $\mathcal{B} = \{b_1, \ldots, b_N\} \sim \pi_\beta(\cdot \mid s)^N$. The critic function $Q^{\pi_\theta}(s, a)$ is the expected discounted return after taking a in s and subsequently following π_θ. We learn it with classic temporal-difference learning loss,

$$\mathcal{L}_{\mathrm{TD}}(\phi) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D},\ \mathcal{B}' \sim \pi_\beta(\cdot \mid s')^N, a' \sim \pi_\theta(\cdot \mid s', \mathcal{B}')} \Big[\big(Q_\phi(s, a) - \mathrm{sg}\big(r + \gamma\, Q_{\bar{\phi}}(s', a')\big) \big)^2 \Big], \tag{3}$$

where $\mathcal{D}$ is the replay distribution over offline and online transitions, sg denotes stop-gradient, $Q_{\bar{\phi}}$ is a slowly updated target critic. The actor is trained with a behavior-regularized Q-maximization objective,

$$\mathcal{L}_{\mathrm{actor}}(\theta) = \mathbb{E}_{s \sim \mathcal{D},\ \mathcal{B} \sim \pi_\beta(\cdot \mid s)^N, a \sim \pi_\theta(\cdot \mid s, \mathcal{B})} \big[-Q_\phi(s, a) + \alpha\, \ell_{\mathrm{beh}}(a, \mathcal{B}) \big], \tag{4}$$

where ℓ_{beh} is a behavior regularization penalty and $\alpha \geq 0$ controls its strength. The choice of ℓ_{beh} determines which improvements the actor can reach.

4 Geometric Support Behavior Regularizer (GSBR)

GSBR uses the base policy to identify where exploration is promising while leaving the actor free to select and improve behaviors in that region. Section 4.1 formulates the desired behavior constraint (ℓ_{beh} in Eq. equation 4) as a geometric relaxation of support-constrained policy improvement. Section 4.2 approximates this penalty from base-policy samples and bounds the approximation error. Algorithm 1 describes the practical instantiation of our algorithm.

4.1 Geometric relaxation of support constraints

The assumption in Section 3 translates into two requirements on ℓ_{beh}. For mode selection, the penalty should be zero for in-distribution actions, so that the actor can concentrate on the most valuable modes without preserving the base policy's relative mode frequencies. For local refinement, actions outside the support should incur a finite penalty rather than being excluded, and this penalty should grow with distance: the assumption places improvements near the support, and the critic is least reliable far from it. Together, these requirements call for a region of zero penalty surrounded by a finite, distance-dependent penalty (Figure 2).

Following the definition of support region in Eq. 2, the true geometric relaxation of the support constraint is given by

$$\ell_{\mathrm{supp}}(s, a) = d(a, S_\beta)^2, \qquad d(a, \mathcal{X}) = \inf_{x \in \mathcal{X}} \|a - x\|_2. \tag{5}$$

The penalty is zero throughout S_β, leaving value maximization free to determine which supported behaviors to favor. Outside the region, it penalizes the squared distance to the nearest supported action, allowing departures when their predicted value gains justify the cost. This geometry retains guidance from the prior without imposing a hard boundary on policy improvement.

Distributional interpretation. The ideal penalty also admits a Wasserstein interpretation. Let $\mathcal{P}_2(S_\beta)$ denote the set of distributions with finite second moments supported entirely within S_β. For an actor distribution $\pi_\theta(\cdot \mid s)$ with finite second moment,

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[d(a, S_\beta)^2\right] = \inf_{\nu\in\mathcal{P}_2(S_\beta)} W_2^2(\pi_\theta(\cdot \mid s), \nu)\,. \tag{6}$$

The expected penalty is thus the squared Wasserstein distance from the actor distribution to the set of distributions supported in (S_β), that is, the minimum quadratic transport cost required to move the actor's probability mass into the support. Because ν can be any distribution on the support, the penalty constrains where the actor places its mass but not how it divides that mass among modes, which is the formal counterpart of mode selection.

4.2 Approximating support with reference-action neighborhoods

Computing ℓ_{supp} requires the support $S_\beta(s)$, which is unavailable when the base policy provides only samples. We therefore estimate the support from the reference set $\mathcal{B}$. However, the reference set alone is a poor estimate: it leaves gaps between plausible actions, and penalizing distance directly to this set would pull the actor toward individual samples. We address this sampling sparsity by introducing a support ball around each reference action, with an adaptive radius determined by the observed spaces among sampled actions (Figure 2).

Given $\mathcal{B} = \{b_1, \ldots, b_N\}$, we compute each reference actions' nearest-neighbor distance, and define the support radius as:

$$u_i := \min_{j\neq i} \|b_i - b_j\|_2, \qquad r_{\mathcal{B}} := \text{Quantile}_q\left(\{u_i\}_{i=1}^N\right)$$

The distance u_i measures the local spacing of reference actions within the same mode or between adjacent action modes; the quantile $q \in (0, 1)$ is a hyperparameter that filters out large distances of isolated action samples. The estimated support is the union of closed balls of radius $r_{\mathcal{B}}$ centered around each reference action:

$$C := \bigcup_{i=1}^{N} \overline{B}(b_i, r_{\mathcal{B}}) = \left\{a \in \mathcal{A} : \min_i \|a - b_i\|_2 \leq r_{\mathcal{B}}\right\}. \tag{7}$$

Given this estimated support region C, we approximate the geometric relaxation of the support constraint as

$$\ell_{\text{GSBR}}(a, \mathcal{B}) := d(a, C)^2 = \left[\min_i ||a - b_i||_2 - r_{\mathcal{B}}\right]_+^2 \tag{8}$$

where $[x]_+ = \max(x, 0)$. We train the actor with Eq. 4 by setting $\ell_{\text{beh}} = \ell_{\text{GSBR}}$.

Because ℓ_{GSBR} depends on $\mathcal{B}$ only through the nearest reference, an actor that observes all of $\mathcal{B}$ can move to any sampled mode at zero cost. This distinguishes `GSBR` from regularizers that tie each action to a single sampled reference action (Park et al., 2025b; Xu et al., 2026), which retain the base policy's relative mode frequencies.

Approximation Guarantee. The following theorem shows that $\ell_{\text{GSBR}} + 2Dr$ upper bounds the true loss ℓ_{supp}; their expected difference is bounded and approaches zero as the number of reference samples grows.

Theorem 4.1 (Expected approximation error of the GSBR loss)**.** *Assume the action space A is compact with diameter D, and that $\pi_\beta(a \mid s)$ has continuous density. Fix any $a \in A$, and define our reference action set $\mathcal{B}$ as above.*

Then, for any radius $r > 0$, define a $\lambda_r > 0$ as:

$$\lambda_r := \inf_{c\in S_\beta} \Pr_{b\sim\pi_\beta}\left(\|b - c\|_2 \leq r\right)$$

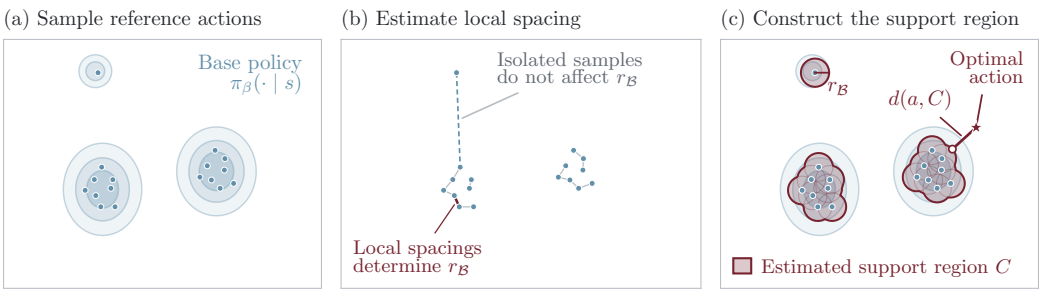

Figure 2: **GSBR**. To implement the geometric support regularizer on a generative robot policy, draw samples from $\pi_\beta(\cdot \mid s)$ and compute the nearest-neighbor distance between them. Set $r_\mathcal{B}$ to a quantile over these distances and construct the support region C as the union of balls of radius $r_\mathcal{B}$ around each sample. The behavior regularizer loss is computed as the Euclidean distance to the closest point in C.

representing the worst-case probability over our support set that a sampled action from π_β falls within distance r of a given action in S_β.

The following bounds then hold:

$$\ell_{\text{supp}}(a) \leq \ell_{\text{GSBR}}(a, \mathcal{B}) + 2Dr, \tag{9}$$

$$\mathbb{E}_\mathcal{B}[|\ell_{\text{GSBR}}(a, \mathcal{B}) - \ell_{\text{supp}}(a)|] \leq 2Dr + D^2(1 - \lambda_r)^N. \tag{10}$$

The upper bound in equation 9 also holds with $r = r_\mathcal{B}$ where $r_\mathcal{B}$ is shown above as the quantile of the nearest-neighbor distances between base policy samples. Further, with this choice of radius, $r_\mathcal{B}$ becomes arbitrarily small as N increases, and the approximation converges uniformly over the action space in expectation:

$$\mathbb{E}_\mathcal{B}\left[\sup_{a \in A} |\ell_{\text{GSBR}}(a, \mathcal{B}) - \ell_{\text{supp}}(a)|\right] \longrightarrow 0 \qquad \text{as } N \to \infty. \tag{11}$$

The full proof is in Appendix A.2. The term $D^2(1 - \lambda_r)^N$ accounts for the event that no reference lies within radius r of the in-distribution action nearest to a, and it vanishes as $N \to \infty$; The term $2Dr$ bounds the error caused by subtracting the quantile radius.

We note that for generative policies, such as flow matching, they theoretically assign nonzero likelihood to every action (Lipman et al., 2023). Instead, we can define our constraint with respect to some π_ϵ derived through filtering for actions such that $\pi_\beta(a|s) \geq \epsilon$, but we found this was not necessary in practice.

4.3 Practical Implementation

Algorithm 1 summarizes `GSBR`. Training alternates between environment interaction and gradient updates. Training always sample from a replay buffer $\mathcal{D}$ that mixes online and offline transitions. The learned actor receives the state and a set of reference action $\mathcal{B}$ as inputs. The algorithm has hyperparameters: N - number of sample actions, q - quantile that affects the size of the radius, and α - that controls the strength of the behavior regularizer.

5 Experimental Results

In this section, we empirically evaluate the performance of `GSBR` against previous online RL methods that leverage a frozen base policy across benchmarks covering both state and pixel inputs with MLP and VLA base policies. Throughout our experiments, we answer the following questions: (1) Is `GSBR` an effective behavior regularizer for online RL, (2) how sensitive is `GSBR` to the choice of support radius and number of samples, and (3) how well does `GSBR` estimate the true support region?

5.1 Experimental Setup

Benchmarks. We evaluate all methods on 19 tasks across four benchmarks, each having different properties that change may change the distribution of base policies. In **RoboMimic** (Mandlekar et al., 2022), we select four challenging tasks: `can-mh`, `square-mh`, `transport-mh`,

Algorithm 1 Online RL with `GSBR`

Require: Frozen base policy π_β, offline buffer $\mathcal{D}_{\text{off}}$.

Initialize actor π_θ, critic Q_ϕ, and online buffer $\mathcal{D}_{\text{on}}$.
for environment steps **do**
  Sample $\mathcal{B} \sim \pi_\beta(\cdot \mid s)^N$ and execute $a \sim \pi_\theta(\cdot \mid s, \mathcal{B})$; store transition in $\mathcal{D}_{\text{on}}$.
  for each gradient update **do**
    Sample a mini-batch from $\mathcal{D}_{\text{off}}$ and $\mathcal{D}_{\text{on}}$.
    Update Q_ϕ using TD loss.
    Sample $\mathcal{B} = \{b_i\}_{i=1}^N \sim \pi_\beta(\cdot \mid s)^N$ and $\widetilde{a} \sim \pi_\theta(\cdot \mid s, \mathcal{B})$.
    $r_{\mathcal{B}} \leftarrow \text{Quantile}_q\Big(\{\min_{j \neq i} \|b_i - b_j\|_2\}_{i=1}^N\Big)$.
    Update θ to minimize $-Q_\phi(s, \widetilde{a}) + \alpha \left[\min_i \|\widetilde{a} - b_i\|_2 - r_{\mathcal{B}}\right]_+^2$.
  end for
end for

and `tool-hang-ph`. Each task contains demonstrations collected by human operators of various proficiency levels and captures challenges like precise alignment and bimanual coordination. We test five variants of the `cube-triple-task{1-5}` in **OGBench** (Park et al., 2025a). These long-horizon tasks contain a play-style exploration dataset that may lead to a base policy with many action modes and test the RL actor's ability to drop bad actions while exploring better ones. The recently released benchmark **OCBench** (Anonymous, 2026) provides a more realistic dataset that emulates the statistical properties of human demonstrations. We select `block-lite-double-task{1-2}` and `chamber-lite-easy-task{1-3}`. In these three benchmarks, the base policy is an MLP behavior cloning policy trained on only the task-specific dataset, and both the base and RL policies receive state information from the simulator. To test each method on more realistic settings, we also experiment on **LIBERO** (Liu et al., 2023) with the fine-tuned $\pi_{0.5}$-LIBERO base policy using pixel inputs. We hand-picked five tasks where the base VLA leaves room for improvement: `task{14, 29, 54, 60, 64}`. On these image-based tasks, the RL network trains a CNN from scratch. Since the $\pi_{0.5}$ policy is slow to sample from, we cache a set of reference actions and sample from these instead of sampling from the policy at each training step.

Baselines. We compare `GSBR` with seven online RL baselines. **TD3+BC** (Fujimoto & Gu, 2021) trains a state-conditioned actor with a Q-maximization objective and a squared regression penalty to base policy samples. For expressive base policies, TD3+BC regularizes the action to the mean of the reference action distribution. **RLT** (Xu et al., 2026) conditions a Gaussian chunk actor on a single sampled base policy chunk and penalizes deviation from that specific reference. While the conditioning alleviates the mode averaging problem, RLT cannot drop action modes. **Residual Policy** (Silver et al., 2018; Johannink et al., 2019) scales and adds a learned correction to a sampled base policy action, so its deviation from the base policy proposal is limited. We follow the formulation of Dong et al. (2026a) and learn an action chunked residual policy conditioned on the base policy input. **BFN** (Li et al., 2025) does not learn an explicit actor, instead formulating it as the base policy sample that maximizes the Q value. This is analogous to a hard support constraint since the implicit actor cannot generate actions outside the base policy's support. **DSRL** (Wagenmaker et al., 2025) learns to steer the frozen base policy through its latent noise rather than directly producing corrected actions, also limiting the possible actions to the support of the base policy. We use the DSRL-NA variant for the state-based tasks and the DSRL-SAC variant for LIBERO. **EXPO-FT** (Dong et al., 2026a) combines bounded residual edit policy with best-of-N sampling at test-time to achieve mode selection. For a fair comparison, we freeze the base policy. **DICE-RL** (Sun & Song, 2026) learns latent-conditioned residual corrections to base policy samples, applies selective BC regularization on the residual policy, and applies best-of-N sampling.

All methods use the same offline data, online interaction budget, network size, and architecture. They also share training settings, including the update-to-data ratio, action-chunk horizon, discount factor, and critic configuration. For each task, we evaluate each policy for 50 episodes and report the mean and standard error of the mean (SEM) across three seeds.

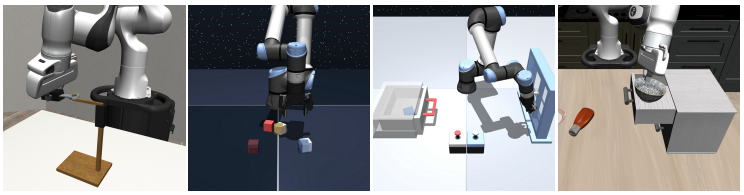

Figure 3: **Benchmarks**. We evaluate our method across the four benchmarks above. From left to right, the tasks are RoboMimic `tool-hang`, OGBench `cube-triple-task4`, OCBench `chamber-lite-easy-task3`, and LIBERO `task29`.

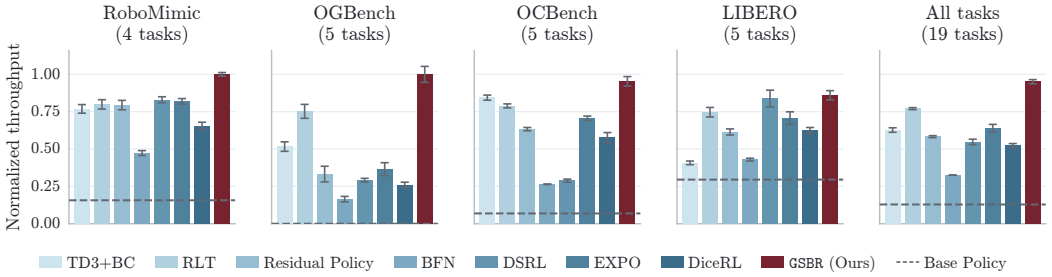

Figure 4: **Normalized throughput across benchmarks**. `GSBR` achieves the highest average normalized throughput across all 19 tasks and leads all baselines on `RoboMimic`, `OGBench`, and `OCBench`. Throughput (success rate / mean episode length) is normalized by the best method's value per task, then averaged across tasks. Error bars show SEM across seeds.

Method	RoboMimic (4 tasks)	OGBench (5 tasks)	OCBench (5 tasks)	LIBERO (5 tasks)	All tasks (19 tasks)
Base Policy	44	0	41	49	33
TD3+BC	96 ± 2	73 ± 1	98 ± 0	54 ± 2	79 ± 1
RLT	96 ± 2	90 ± 5	98 ± 0	$\mathbf{86} \pm 2$	92 ± 1
Residual Policy	94 ± 2	68 ± 10	95 ± 0	78 ± 1	83 ± 2
BFN	86 ± 2	69 ± 5	88 ± 2	66 ± 1	77 ± 1
DSRL	97 ± 1	67 ± 5	87 ± 2	81 ± 2	82 ± 2
EXPO	97 ± 1	80 ± 1	98 ± 1	82 ± 2	89 ± 1
DiceRL	95 ± 1	40 ± 1	97 ± 1	74 ± 1	75 ± 1
`GSBR` (Ours)	$\mathbf{98} \pm 0$	$\mathbf{99} \pm 1$	$\mathbf{99} \pm 0$	85 ± 1	$\mathbf{95} \pm 0$

Table 1: **Average success rate (%) across benchmarks**. Uncertainty is SEM across three seed-wise task averages. Values are rounded to the nearest integer; bold marks the highest unrounded mean in each column. Base Policy SEM is omitted because complete per-seed evidence is unavailable.

5.2 Results

Is `GSBR` an effective behavior regularizer for online RL? `GSBR` demonstrates strong empirical performance across benchmarks compared to the baselines. As shown in Figure 4 and Table 1, across 19 tasks, `GSBR` achieves 95% success rate and 0.95 normalized throughput, while the next-best baseline, RLT, achieves 92% success rate and 0.77 normalized throughput. On RoboMimic, while most methods have a similar success rate at around 95%, `GSBR` achieves the highest throughput, surpassing the next-best method, DSRL, by 20%. On OGBench, `GSBR` achieves a near-perfect success rate across all five tasks, which is 10% (absolute) higher success rate and 33% (relative) higher throughput compared to RLT, the strongest baseline. On OCBench, most methods achieve a near-perfect success rate yet `GSBR` has approximately 13% higher throughput than TD3+BC.

Further, we evaluate the performance of our behavior regularizer on LIBERO with a modern VLA as the base policy. Using $\pi_{0.5}$-LIBERO, the strongest baseline is DSRL in terms of throughput at 84 and RLT in terms of success rate at 86%. `GSBR` achieves 85% success rate and 0.86 normalized throughput, closely matching or beating the best baselines on either metric. These results demonstrate that `GSBR` provides an effective behavior regularizer that empirically outperforms baselines in success rate and throughput, indicating that our regularizer allows the maximum deviation from the base policy while providing stabilizing guidance. For additional experimental detail and results, refer to Appendix A.4.1 and A.4.2

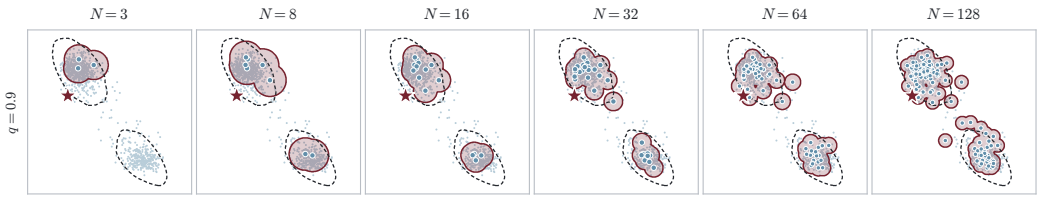

Figure 5: **Visualization of the true and estimated support region on `cube-triple`**. The estimated support region (in burgundy) closely matches the true support using only 8 samples. The RL actor learned an action just outside the support region (marked with star).

How sensitive is GSBR to the choice of support radius and number of samples? We sweep over a range of N, the number of base policy samples used to compute the behavior regularizer loss on `cube-triple` and find little effect on the performance of GSBR. The normalized throughput for $N = 8$, $N = 16$, and $N = 32$ is 1.0, 0.91, and 0.93, respectively. These results suggest that as few as eight samples from the base policy suffice for effective behavior regularization on these tasks.

We also ablate the effect of computing the support radius $r_{\mathcal{B}}$. With $r_{\mathcal{B}} = 0$, the throughput degrades significantly to only 0.66. This ablation supports the benefit of leaving neighborhoods around reference actions unpenalized, unlike in other prior works, allowing the actor to refine its actions within these neighborhoods without being pulled toward individual samples.

GSBR Ablation	Normalized TP (successes/h)
N=8	1.00 ± 0.05
N=16	0.91 ± 0.02
N=32	0.93 ± 0.02
$r_{\mathcal{B}}$	0.66 ± 0.07

Table 2: **Ablations on N and $r_{\mathcal{B}}$.** On cube triple, we sweep over N and find little effect on the performance of GSBR. We also ablated removing the support radius $r_{\mathcal{B}}$ and find that the performance degrades significantly.

How well do we approximate the true support region? Figure 5 shows the empirical support of the base policy visualized on a bimodal state in `cube-triple`. We visualize the first two action dimensions using $q = 0.9$, and the zero-penalty estimated support region C is marked in burgundy. We estimate the true support of the base policy with KDE cutoff at 95% from 5,000 base-policy samples. With only $B = 3$ references, the sampled actions cover one mode and miss the other. At $B = 8$, the zero-penalty region already captures both modes while leaving the low-density gap between them penalized. As N increases, the support radius shrinks as the local spacing between samples are tighter. On this particular state, the trained RL actor predicts an action slightly beyond the true support, which incur little loss as it is close to the support region.

5.3 Conclusions

Foundation models offer an appealing behavior prior for sample-efficient online RL, which raises the question of how to regularize an actor toward such a prior while still allowing it to improve. We introduce GSBR, a geometric support behavior regularizer that provides sufficient guidance without overconstraining policy improvement. We provide a theoretically motivated implementation that estimates the ideal loss based on base policy sample distribution.

GSBR demonstrated strong empirical performance: Across 19 tasks on four robotic benchmarks, GSBR raises the average success rate from 33% for the base policy to 95%, achieving the highest average success rate and throughput among all compared methods, with about 23% higher normalized throughput than the best prior method. GSBR still samples the base policy at test time; reducing this cost and validating GSBR on real robots are natural next steps. Future work could also explore alternative ways to construct the support (e.g., an alternative estimate for r) which could improve performance or reduce the number of hyperparameters.

AI USE STATEMENT

In this work, we used generative AI tools for coding and implementation assistance, feedback on mathematical claims, polishing writing, and help creating figures in matplotlib. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, all of our numbers are reported across 3 different seeds with SEM across seeds. We include detailed implementation details and all of our hyper parameters in appendix A.6. We will further release all the code and artifacts necessary to fully replicate our results.

REFERENCES

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement—residual RL for precise assembly, 2024. URL https://arxiv.org/abs/2407.16677.

Anonymous. OCBench: a controllable benchmark for studying behavioral cloning, 2026. URL https://openreview.net/forum?id=bTZVvwj6R6. Submitted to ICLR 2027.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 1577–1594. PMLR, 2023. URL https://proceedings.mlr.press/v202/ball23a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model, 2021. URL https://arxiv.org/abs/2101.05982.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. ConRFT: A reinforced fine-tuning method for VLA models via consistency policy, 2025. URL https://arxiv.org/abs/2502.05450.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. EXPO-FT: Sample-efficient reinforcement learning finetuning for vision-language-action models, 2026a. URL https://arxiv.org/abs/2605.25477.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. EXPO: Stable reinforcement learning with expressive policies. In *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2507.07986.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bing-Yi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2405.20555.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2106.06860.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2052–2062. PMLR, 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies, 2023. URL https://arxiv.org/abs/2304.10573.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout Q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.02034.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning, 2023. URL https://arxiv.org/abs/2311.02198.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *International Conference on Robotics and Automation*, 2019. URL https://arxiv.org/abs/1812.03201.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1906.00949.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching, 2026. URL https://arxiv.org/abs/2601.14234.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=XUks1Y96NR.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18719.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A software suite for sample-efficient robotic reinforcement learning, 2024. URL https://arxiv.org/abs/2401.16013.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 2025. doi: 10.1126/scirobotics.ads5033. URL https://arxiv.org/abs/2410.21845.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164, pp. 1678–1690. PMLR, 2022.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic RL: Offline RL and online RL fine-tuning of any class and backbone, 2024. URL https://arxiv.org/abs/2412.06685.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2303.05479.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance, 2024. URL https://arxiv.org/abs/2410.13816.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 48104–48127. PMLR, 2025b. URL https://proceedings.mlr.press/v267/park25f.html.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: A VLA that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

Yuhang Ran, Yi-Chen Li, Fuxiang Zhang, Zongzhang Zhang, and Yang Yu. Policy regularization with dataset constraint for offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 28701–28717. PMLR, 2023. URL https://proceedings.mlr.press/v202/ran23a.html.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization, 2024. URL https://arxiv.org/abs/2409.00588.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning, 2018. URL https://arxiv.org/abs/1812.06298.

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive RL finetuning, 2026. URL https://arxiv.org/abs/2603.10263.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning, 2023. URL https://arxiv.org/abs/2305.09836.

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, 2025. URL https://arxiv.org/abs/2506.15799.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2208.06193.

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2202.06239.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019. URL https://arxiv.org/abs/1911.11361.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL, 2025. URL https://arxiv.org/abs/2511.00091.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL Token: Bootstrapping online RL with vision-language-action models, 2026. URL https://arxiv.org/abs/2604.23073.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

A Appendix

A.1 Practical Implementation

In this section, we discuss some of the practical implementation choices for our method.

Action chunking. Both the base policy and RL policy learn over action chunks of H consecutive steps, which vary between three and ten depending on the domain. During rollout and evaluation, we execute the entire action chunk before querying the next one.

Reference action conditioning. Following Xu et al. (2026), we condition the RL actor on N reference action samples to provide additional learning signal and guidance. During training, we apply 50% dropout to prevent the actor from copying the reference action.

Using demo data. Following Ball et al. (2023), we store separate offline and online replay buffers $\mathcal{D}_{\text{off}}$ and $\mathcal{D}_{\text{on}}$ to maintain a 50/50 sampling ratio between demo and online data.

Critic ensemble. We train an ensemble of 10 Q networks $\{Q_{\phi_k}\}_{k=1}^{K}$ and use the following pessimistic target value

$$y = r + \gamma m \left(\mu_{\bar{\phi}}(s', \widetilde{a}') - \tfrac{1}{2}\sigma_{\bar{\phi}}(s', \widetilde{a}')\right) \tag{12}$$

where $\mu_{\bar{\phi}}(s', \widetilde{a}')$ and $\sigma_{\bar{\phi}}(s', \widetilde{a}')$ are the mean and standard deviation over the 10 target Q networks and $a \sim \pi_\theta(s')$.

A.2 PROOF OF PROPOSITION 4.1

Theorem 4.1 restated: *Assume the action space A is compact with diameter D, and that $\pi_\beta(a \mid s)$ has continuous density. Fix any $a \in A$, and define our reference action set $\mathcal{B}$ as above.*

Then, for any radius $r > 0$, define a $\lambda_r > 0$ as:

$$\lambda_r := \inf_{c \in S_\beta} \Pr_{b \sim \pi_\beta} \left(\|b - c\|_2 \leq r \right)$$

representing the worst-case probability over our support set that a sampled action from π_β falls within distance r of a given action in S_β.

The following bounds then hold:

$$\ell_{\text{supp}}(a) \leq \ell_{\text{GSBR}}(a, \mathcal{B}) + 2Dr,$$
$$\mathbb{E}_{\mathcal{B}}[|\ell_{\text{GSBR}}(a, \mathcal{B}) - \ell_{\text{supp}}(a)|] \leq 2Dr + D^2(1 - \lambda_r)^N.$$

The upper bound in equation 9 also holds with $r = r_{\mathcal{B}}$ where $r_{\mathcal{B}}$ is shown above as the quantile of the nearest-neighbor distances between base policy samples. Further, with this choice of radius, $r_{\mathcal{B}}$ becomes arbitrarily small as N increases, and the approximation converges uniformly over the action space in expectation:

$$\mathbb{E}_{\mathcal{B}}\left[\sup_{a \in A} |\ell_{\text{GSBR}}(a, \mathcal{B}) - \ell_{\text{supp}}(a)| \right] \longrightarrow 0 \qquad \text{as } N \to \infty.$$

Proof. We first show that $\lambda_r > 0$. Since S_β is compact, it can be covered by finitely many balls of radius $r/2$ centered in S_β. Each has positive probability under π_β, because its center belongs to the support of π_β. For every $c \in S_\beta$, some covering ball has its center within $r/2$ of c, so that entire ball lies within distance r of c. Thus λ_r is at least the smallest probability among these finitely many covering balls, which is positive.

Now fix an action $a \in \mathcal{A}$ and write

$$\rho_a := d(a, S_\beta), \qquad X := d(a, \mathcal{B}), \qquad t := [X - r]_+.$$

Because $\rho_a = d(a, S_\beta)$ and $B \subseteq S_\beta$, we have $\rho_a \leq X$. Since $\rho_a \leq X$, $X - t = \min\{X, r\} \leq r$, and $X + t \leq 2D$, we have

$$\ell_{\text{supp}}(a) - \ell_{\text{GSBR}}(a, \mathcal{B}) = \rho_a^2 - t^2 \leq X^2 - t^2 = (X - t)(X + t) \leq 2Dr.$$

Thus, adding $\ell_{\texttt{GSBR}}(a, \mathcal{B})$ to both sides shows that the GSBR penalty plus $2Dr$ upper bounds the support penalty:

$$\ell_{\text{supp}}(a) \leq \ell_{\text{GSBR}}(a, \mathcal{B}) + 2Dr.$$

Next for the error bound, since S_β is compact, there is a closest point $b^\star \in S_\beta$ with $\|a - b^\star\|_2 = \rho_a$. A reference within distance r of $b^\star$ is also within distance $\rho_a + r$ of a, by the triangle inequality. Thus,

$$\Pr_{b \sim \pi_\beta} \left(\|b - b^\star\|_2 \leq r \right) \leq \Pr_{b \sim \pi_\beta} \left(\|b - a\|_2 \leq \rho_a + r \right)$$

And using the definition of λ_r as the lower bound, $\lambda_r \leq \Pr_{b \sim \pi_\beta} \left(\|b - a\|_2 \leq \rho_a + r \right)$

Because the references are independent of one another and of a, the probability that all N references miss this neighborhood is at most

$$\Pr_{\mathcal{B}}(X > \rho_a + r \mid a) \leq (1 - \lambda_r)^N.$$

We now consider a decomposition of the expectation by partitioning on the event that $X \leq \rho_a + r$.

On the event $X \leq \rho_a + r$, subtracting r from both sides gives $X - r \leq \rho_a$, and since $\rho_a \geq 0$, $t = [X - r]_+ \leq \rho_a$. Therefore, the support penalty is at least the GSBR penalty, so the bound from above gives

$$|\ell_{\text{GSBR}}(a, \mathcal{B}) - \ell_{\text{supp}}(a)| = \rho_a^2 - t^2 \leq 2Dr.$$

On the complementary event $X > \rho_a + r$, both penalties lie in $[0, D^2]$, so their absolute difference is at most D^2. Combining the two events gives

$$\begin{aligned}\mathbb{E}_B[|\ell_{\mathrm{GSBR}}(a, \mathcal{B}) - \ell_{\mathrm{supp}}(a)|\,] &\le 2Dr\ \Pr(X \le \rho_a + r \mid a) + D^2\ \Pr(X > \rho_a + r \mid a) \\ &\le 2Dr + D^2(1 - \lambda_r)^N.\end{aligned}$$

Finally, consider the quantile radius $r_{\mathcal{B}}$. The one-sided bound above is pointwise in r and $\mathcal{B}$, so it also holds with $r = r_{\mathcal{B}}$.

Define the largest distance from the support to the reference set by

$$h_N := \sup_{c \in S_\beta} d(c, \mathcal{B}).$$

For any $a \in A$, using the notation above with $r = r_{\mathcal{B}}$,

$$\rho_a \le X \le \rho_a + h_N, \qquad \rho_a - r_{\mathcal{B}} \le t = [X - r_{\mathcal{B}}]_+ \le \rho_a + h_N.$$

Thus $|t - \rho_a| \le \max\{h_N, r_{\mathcal{B}}\}$. Since both t and ρ_a lie in $[0, D]$, we have, for every $a \in A$,

$$|\ell_{\mathrm{GSBR}}(a, \mathcal{B}) - \ell_{\mathrm{supp}}(a)| = |t^2 - \rho_a^2| \le 2D \max\{h_N, r_{\mathcal{B}}\}.$$

To show that this error vanishes, fix any $\delta > 0$. Cover S_β by M closed balls of radius $\delta/2$ centered in S_β. Each ball has probability at least $\lambda_{\delta/2} > 0$. If every covering ball contains a reference, every support point is within distance δ of a reference. Therefore,

$$\Pr_{\mathcal{B}}(h_N > \delta) \le M(1 - \lambda_{\delta/2})^N.$$

The same cover controls the quantile radius. If $u_i > \delta$, any covering ball containing b_i cannot contain another reference, since the distance between two points in that ball is at most δ. Hence at most M references have $u_i > \delta$. For all sufficiently large N, $N - M \ge \lceil qN \rceil$, so $r_{\mathcal{B}} \le \delta$.

For all sufficiently large N, we have $r_{\mathcal{B}} \le \delta$. On the event $h_N \le \delta$, the absolute penalty error is therefore at most $2D\delta$ simultaneously for every $a \in \mathcal{A}$. On the complementary event, it is at most D^2. Taking the supremum over actions and then the expectation over reference samples gives

$$\begin{aligned}\mathbb{E}_{\mathcal{B}}\left[\sup_{a \in A} |\ell_{\mathrm{GSBR}}(a, \mathcal{B}) - \ell_{\mathrm{supp}}(a)|\right] &\le 2D\delta + D^2 \Pr_{\mathcal{B}}(h_N > \delta) \\ &\le 2D\delta + D^2 M(1 - \lambda_{\delta/2})^N.\end{aligned} \tag{13}$$

Let $\eta > 0$ be any desired error tolerance. Choose $\delta > 0$ small enough that $2D\delta < \eta/2$. For this fixed δ, $D^2 M(1 - \lambda_{\delta/2})^N \to 0$ as $N \to \infty$, thus it must be less than $\eta/2$ for all sufficiently large N. The expected uniform error is therefore less than η for all sufficiently large N. Since $\eta > 0$ is arbitrary, the expected uniform error converges to zero as $N \to \infty$, proving equation 11. □

A.3 Implementation Choices

Action chunking. For robotic control, we instantiate the action space using chunks of H consecutive control actions. Each action chunk is flattened into a vector of dimension $d = Hd_a$, where d_a is the dimension of an individual control action. In this implementation, a, b, and the references in $\mathcal{B}$ denote action chunks. Training uses H-step replay transitions (s, a, R, s'), where a concatenates the recorded control actions $a_t, \ldots, a_{t+H-1}$, $s = s_t$, $s' = s_{t+H}$ and

$$R = \sum_{h=0}^{H-1} \gamma^h r(s_{t+h}, a_{t+h}), \qquad s = s_t, \quad s' = s_{t+H}. \tag{14}$$

Reference action conditioning. We parameterize the actor as a tanh-squashed Gaussian policy conditioned on s. To reduce the burden on the actor on learning the full distribution of the base policy and provide early-training exploration signal, we adopt the reference action passthrough mechanism from Xu et al. (2026) and additionally condition the actor on all sampled reference actions $\mathcal{B}$. During training, we zero the entire reference input with probability p_{drop}. In early testing, we found that this mechanism improves initial performance without having to sample from the base policy during deployment.

Critic ensembles. To stabilize Q-learning, we train an ensemble of $K = 10$ Q-functions $\{Q_{\phi_k}\}_{k=1}^{K}$. We compute the mean and standard deviation of their target predictions as

$$\mu_{\bar{\phi}}(s', a') = \frac{1}{K}\sum_{k=1}^{K} Q_{\bar{\phi}_k}(s', a'), \quad \sigma_{\bar{\phi}}(s', a') = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(Q_{\bar{\phi}_k}(s', a') - \mu_{\bar{\phi}}(s', a')\right)^2}. \tag{15}$$

For TD learning, we use the pessimistic target and the critic loss is

$$\mathcal{L}_Q(\phi_k) = \mathbb{E}_{\substack{(s,a,R,s')\sim\mathcal{D} \\ \mathcal{B}'\sim\pi_\beta(\cdot|s')^N \\ a'\sim\pi_\theta(\cdot|s',\mathcal{B}')}}\left[\left[Q_{\phi_k}(s,a) - R - \gamma^H m\left(\mu_{\bar{\phi}}(s', a') - \tfrac{1}{2}\sigma_{\bar{\phi}}(s', a')\right)\right]^2\right]. \tag{16}$$

For the actor update, we simply take the mean of the Q-values.

Buffer sampling ratio. We maintain separate offline and online replay buffers, $\mathcal{D}_{\text{off}}$ and $\mathcal{D}_{\text{on}}$. Following the symmetric replay sampling strategy of RLPD (Ball et al., 2023), each training minibatch contains equal numbers of H-step sequences from the two buffers. This retains access to prior experience as new interactions accumulate.

Rollouts and deployment. During a rollout, we sample reference chunks and obtain an action chunk from the actor without reference dropout. During deployment, we can optionally skip querying the base policy and run the actor with reference actions set to zero.

A.4 Training details

We describe the shared training protocol for the state-based benchmarks, followed by method-specific settings and the configuration used for LIBERO. Unless specified otherwise, the state-based methods use the protocol below. Our training and evaluations were done on either A500 GPUs, or TPU V5E chips. Common training settings are described in A.6.

A.4.1 State-based benchmarks

Base policies and datasets. For each training seed, we use a seed-matched flow-matching behavior cloning policy trained offline for 10^6 training steps. The base policy produces action chunks and remains frozen during online RL. On RoboMimic and OGBench Cube Triple, it uses a four-layer MLP with 512 units per layer and 16 Euler integration steps. On the OCBench Block and Chamber tasks, it uses a residual MLP with four 1024-unit layers and 10 Euler integration steps. We clip sampled base-policy actions to the action bounds.

For RoboMimic, we use the low-dimensional v1.5 datasets: multi-human (MH) demonstrations for Can, Square, and Transport, and proficient-human (PH) demonstrations for Tool Hang due to it not providing a MH dataset. For OGBench, we use the cube triple play dataset. For OCBench, we use the type 51 dataset shards and normalize each action dimension using its 1st and 99th percentiles.

Replay and online training. The RL components receive no offline pretraining. We initialize online replay with 50 episodes collected by the base policy, then train with minibatches of 256 sequences, split equally between the offline dataset and online replay. We use two gradient updates per environment step, except for the DICE-RL where we take the max performance between a UTD 2 and the effective UTD used for their results in robosuite. We do not filter transitions by success and use the environment rewards. RoboMimic supplies a binary success reward, and episodes terminate on success; Cube Triple uses the OGBench reward.

A.4.2 LIBERO

Tasks and observations. We evaluate on LIBERO-90 tasks 14, 29, 54, 60, and 64, with an episode limit of 400 environment steps and action chunks of length $H = 10$. The frozen base policy is $\pi_{0.5}$ with the `pi05_libero` configuration, which receives images from both cameras and proprioception. The RL actor and critic use CNNs with a 64×64 image and an eight-dimensional state as input.

Training and evaluation. LIBERO training uses online data only, without demonstrations. Base-policy reference chunks are cached and reused. Each unsuccessful chunk receives reward -1, and a successful chunk receives reward 0. The discount factor is 0.975 per chunk, and TD targets are not clipped. We train for 1,000 trajectories with 20 gradient updates per action chunk.

Method-specific settings. For `GSBR`, we draw $N = 8$ reference chunks from the cache and do not apply reference dropout. We use a separate radius for each of the seven action dimensions due to the chunk horizon, with each radius shared across the chunk's time steps.

LIBERO regularizer. For chunks with $H = 10$ and $d_a = 7$, we sample $N = 8$ references $\mathcal{B} = \{b_i\}_{i=1}^{N}$. Define

$$\rho_k(x, y) := \left(\frac{1}{H} \sum_{t=1}^{H} (x_{t,k} - y_{t,k})^2 \right)^{1/2}, \tag{17}$$

$$j_i \in \arg\min_{j \neq i} \|b_i - b_j\|_F^2, \qquad r_{\mathcal{B},k} := \mathrm{Quantile}_q\big(\{\rho_k(b_i, b_{j_i})\}_{i=1}^{N}\big), \tag{18}$$

$$\ell_{\mathrm{GSBR}}^{\mathrm{LIBERO}}(a, \mathcal{B}) := \min_i \frac{1}{d_a} \sum_{k=1}^{d_a} [\rho_k(a, b_i) - r_{\mathcal{B},k}]_+^2. \tag{19}$$

We compute quantiles using linear interpolation. Nearest neighbors are selected using full chunks, while each action dimension has its own radius. The minimum selects one reference for all dimensions; the penalty vanishes when all dimension-wise distances to that reference lie within their radii. We use this loss in the actor objective, sweep α and q.

A.5 Full results

Our full per task results are shown in table 3, and 4, and visualized in figure 6.

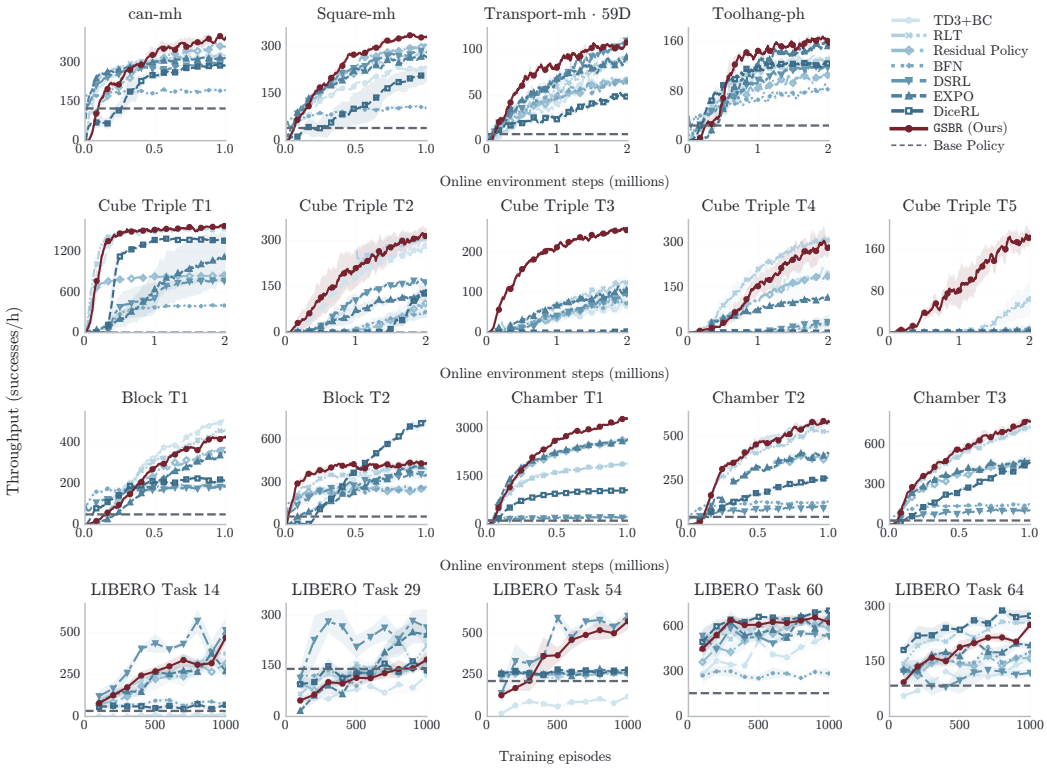

Figure 6: **Absolute throughput (successes/h) over training steps**

A.6 Hyperparameters

We include all hyper parameter details for our method and what we sweep over in table 6, the baseline sweeps in table 7, and the training setup shared parameters in table A.6.

A.7 Sensitivity Analysis

We visualize the hyper parameter grid for our method in figure 7. Our method across most tasks under a specific quantile performs similar across different magnitudes of behavior constraint weights.

Task	BC	TD3+BC	RLT	Residual policy	BFN	DSRL	EXPO-FT	DICE-RL	GSBR (ours)
RoboMimic									
can-mh	73±1	**99**±0	**99**±0	97±1	94±2	**99**±0	**99**±0	**99**±0	98±1
square-mh	37±1	96±1	96±1	98±0	78±3	98±0	98±0	96±2	**99**±0
transport-mh	15±2	88±7	90±6	83±7	81±4	92±2	93±1	84±3	**96**±1
tool-hang-ph	50±2	97±0	97±0	97±2	90±2	96±2	95±1	**98**±0	98±0
OGBench cube-triple									
Task 1	0±0	**100**±0	**100**±0	**100**±0	**100**±0	**100**±0	98±1	**100**±0	**100**±0
Task 2	0±0	98±0	**99**±0	62±31	80±9	98±0	96±2	89±5	**99**±0
Task 3	0±0	72±7	92±1	80±16	82±4	85±1	96±2	6±3	**100**±0
Task 4	0±0	91±5	**100**±0	98±1	66±5	46±23	97±1	1±1	98±1
Task 5	0±0	0±0	58±29	0±0	15±15	4±2	8±4	0±0	**95**±1
OCBench Lite									
Block 1	33±5	95±1	95±1	91±1	92±1	89±1	**97**±1	93±2	**97**±1
Block 2	36±2	**99**±0	98±0	90±4	90±2	95±2	94±1	98±0	98±0
Chamber 1	72±3	**100**±0	**100**±0	**100**±0	97±0	96±2	**100**±0	**100**±0	**100**±0
Chamber 2	36±5	**99**±0	96±2	96±2	76±4	72±7	98±1	94±1	98±1
Chamber 3	26±3	98±1	98±1	97±1	84±5	80±4	98±1	99±0	**100**±0
LIBERO									
Task 14	16	8±4	80±2	74±4	30±4	**87**±2	70±5	30±2	86±2
Task 29	52	46±8	65±7	59±5	49±2	70±7	**73**±4	51±4	60±6
Task 54	84	56±3	95±1	93±0	96±1	96±2	94±1	**97**±0	94±2
Task 60	54	94±3	98±0	96±3	80±4	**100**±0	98±0	**100**±0	98±1
Task 64	40	66±6	87±3	66±3	70±2	51±2	76±2	**90**±3	84±2

Table 3: **Final success rate (%) across all tasks**. Values are mean ± SEM.

Task	BC	TD3+BC	RLT	Residual policy	BFN	DSRL	EXPO-FT	DICE-RL	GSBR (ours)
RoboMimic									
can-mh	122±1	329±6	339±8	389±21	198±10	308±7	317±5	289±6	**416**±16
square-mh	40±1	247±6	273±13	325±6	113±7	297±6	273±3	226±20	**348**±8
transport-mh	7±1	80±11	86±11	65±11	63±5	104±5	90±2	52±2	**105**±2
tool-hang-ph	24±2	133±3	127±5	112±10	80±7	121±12	143±10	128±6	**165**±3
OGBench cube-triple									
Task 1	0±0	1569±18	1566±14	838±7	405±14	798±18	1065±371	1396±10	**1582**±8
Task 2	0±0	266±3	**329**±32	91±47	60±13	172±3	120±10	134±28	328±28
Task 3	0±0	64±12	109±7	90±35	63±5	85±3	117±14	2±1	**262**±14
Task 4	0±0	184±61	334±16	175±10	33±4	32±20	110±1	0±0	**344**±57
Task 5	0±0	0±0	66±36	0±0	7±7	1±0	3±2	0±0	**174**±15
OCBench Lite									
Block 1	32±6	**293**±8	273±4	228±11	133±5	137±7	229±15	152±13	279±10
Block 2	37±2	327±17	340±42	184±37	185±19	243±25	264±3	**426**±42	364±24
Chamber 1	80±5	1302±33	1269±13	1817±18	161±6	157±9	1805±18	695±3	**2311**±18
Chamber 2	28±4	**406**±22	316±47	252±26	78±8	73±10	300±2	180±8	388±35
Chamber 3	20±2	464±20	460±47	286±22	88±6	80±5	317±39	324±11	**522**±32
LIBERO									
Task 14	33	17±9	375±30	314±42	70±13	**514**±72	270±42	69±6	470±39
Task 29	141	122±28	209±38	163±19	131±11	**263**±55	242±25	136±16	168±28
Task 54	213	119±9	263±6	256±5	281±6	**604**±43	263±11	281±4	572±51
Task 60	156	548±60	596±15	594±44	287±34	537±12	672±27	**709**±19	630±38
Task 64	84	156±17	255±15	158±11	175±9	118±6	193±7	**274**±16	249±11

Table 4: **Absolute throughput (successes/hour) across all tasks and methods** assuming 10 Hz for RoboMimic, OGBench, and OCBench, and 20 Hz for LIBERO. Values are mean ± SEM.

Benchmark	Tasks	Online Env Steps	UTD	H	γ	TD clip
RoboMimic	`can`, `square`	1M	2	5	0.998	$[0, 1]$
RoboMimic	`transport`, `tool-hang`	2M	2	5	0.998	$[0, 1]$
OGBench	`cube-triple-T[1-5]`	2M	2	5	0.99	None
OCBench Lite	`block-double-T[1-2]` `chamber-easy-T[1-3]`	1M	2	3	0.99	$[0, 1]$
LIBERO	`Task-{14, 29, 54, 60, 64}`	1,000 episodes	2	10	0.975 per chunk	None

Table 5: Training settings. H is the action chunk length; UTD and discount γ is per environment step.

Task	BC weight α	Quantile q
RoboMimic		
can-mh	$\{\mathbf{0.1}, 0.3, 1, 3, 10\}$	$\{0.1, 0.3, \mathbf{0.5}, 0.7, 0.9\}$
square-mh	$\{\mathbf{3}, 10, 30\}$	$\{0.7, \mathbf{0.9}\}$
transport-mh	$\{0.1, 0.3, 1, \mathbf{3}, 10, 30\}$	$\{\mathbf{0.1}, 0.3, 0.7, 0.9\}$
tool-hang-ph	$\{3, \mathbf{10}, 30\}$	$\{0.1, 0.3, 0.5, 0.7, \mathbf{0.9}\}$
OGBench		
cube 1	$\{1, \mathbf{3}, 10\}$	$\{0.5, \mathbf{0.7}, 0.9\}$
cube 2	$\{3, 10, \mathbf{30}, 100\}$	$\{0.1, 0.3, 0.5, \mathbf{0.7}, 0.9\}$
cube 3	$\{30, 100, 300, \mathbf{1000}, 3000, 10000\}$	$\{0.7, \mathbf{0.9}\}$
cube 4	$\{3, 10, \mathbf{30}, 100, 300, 1000\}$	$\{0.7, \mathbf{0.9}\}$
cube 5	$\{100, 300, \mathbf{1000}\}$	$\{0.1, 0.3, 0.5, \mathbf{0.7}, 0.9\}$
OCBench Lite		
block 1	$\{0.1, \mathbf{0.3}, 1, 3, 10, 30\}$	$\{0.1, 0.3, 0.5, 0.7, \mathbf{0.9}\}$
block 2	$\{0.1, 0.3, 1, 3, \mathbf{10}\}$	$\{0.1, \mathbf{0.3}, 0.5, 0.7, 0.9\}$
chamber 1	$\{\mathbf{0.01}, 0.03, 0.1\}$	$\{\mathbf{0.1}, 0.3, 0.5, 0.7, 0.9\}$
chamber 2	$\{0.01, 0.03, \mathbf{0.1}, 0.3, 1, 3\}$	$\{\mathbf{0.1}, 0.3, 0.5, 0.7, 0.9\}$
chamber 3	$\{0.01, 0.03, \mathbf{0.1}, 0.3, 1\}$	$\{0.1, 0.3, \mathbf{0.5}, 0.7, 0.9\}$
LIBERO		
LIBERO 14	$\{30, \mathbf{100}, 300\}$	$\{\mathbf{0.1}, 0.5, 0.9\}$
LIBERO 29	$\{10, 30, \mathbf{100}, 300, 1000\}$	$\{0.1, \mathbf{0.5}, 0.9\}$
LIBERO 54	$\{30, \mathbf{100}, 300\}$	$\{\mathbf{0.1}, 0.5, 0.9\}$
LIBERO 60	$\{30, \mathbf{100}, 300\}$	$\{\mathbf{0.1}, 0.5, 0.9\}$
LIBERO 64	$\{100, \mathbf{300}, 1000, 3000\}$	$\{\mathbf{0.1}, 0.5, 0.9\}$

Table 6: `GSBR` Hyperparameter sweep range.

Method	Tasks	Hyperparameter sets
DICE-RL	can-mh	$\alpha \in \{\mathbf{3}, 10, 30, 100, 300\}$; $u \in \{\mathbf{0.3125}, 2\}$
	square-mh	$\alpha \in \{3, \mathbf{10}, 30, 100, 300\}$; $u \in \{0.3125, \mathbf{2}\}$
	transport-mh	$\alpha \in \{3, 10, 30, 100, \mathbf{300}\}$; $u \in \{\mathbf{0.3125}, 2\}$
	tool-hang-ph	$\alpha \in \{\mathbf{3}, 10, 30, 100, 300\}$; $u \in \{0.3125, \mathbf{2}\}$
	cube 1	$\alpha \in \{1, 3, \mathbf{10}, 30, 100, 300\}$; $u \in \{0.3125, \mathbf{2}\}$
	cube 2	$\alpha \in \{1, 3, 10, \mathbf{30}, 100, 300\}$; $u \in \{\mathbf{0.3125}, 2\}$
	cube 3	$\alpha \in \{1, 3, 10, 30, 100, \mathbf{300}\}$; $u \in \{\mathbf{0.3125}, 2\}$
	cube 4	$\alpha \in \{1, 3, 10, 30, 100, \mathbf{300}\}$; $u \in \{0.3125, \mathbf{2}\}$
	cube 5	$\alpha \in \{1, 3, \mathbf{10}, 30, 100, 300\}$; $u \in \{\mathbf{0.3125}\}$
	block 1	$\alpha \in \{0.3, 1, 3, 10, \mathbf{30}, 100\}$; $u \in \{\mathbf{0.3125}, 2\}$
	block 2	$\alpha \in \{0.3, \mathbf{1}, 3, 10, 30, 100\}$; $u \in \{\mathbf{0.3125}, 2\}$
	chamber 1, 3	$\alpha \in \{\mathbf{0.3}, 1, 3, 10, 30, 100\}$; $u \in \{0.3125, \mathbf{2}\}$
	chamber 2	$\alpha \in \{0.3, \mathbf{1}, 3, 10, 30, 100\}$; $u \in \{0.3125, \mathbf{2}\}$
	LIBERO 14, 29, 54	$\alpha \in \{10, 30, 100, \mathbf{300}\}$; $u \in \{\mathbf{2}\}$
	LIBERO 60, 64	$\alpha \in \{\mathbf{10}, 30, 100, 300\}$; $u \in \{\mathbf{2}\}$
EXPO-FT	can-mh; LIBERO 64	$s \in \{0.01, \mathbf{0.03}, 0.1, 0.3, 1\}$
	square-mh; cube 2	$s \in \{0.01, \mathbf{0.03}, 0.1, 0.3\}$
	transport-mh	$s \in \{\mathbf{0.01}, 0.03, 0.1, 0.3\}$
	tool-hang-ph; cube 3–5	$s \in \{0.01, 0.03, \mathbf{0.1}, 0.3\}$
	cube 1; chamber 1–2	$s \in \{0.01, 0.03, 0.1, 0.3, \mathbf{1}\}$
	block 1–2; chamber 3; LIBERO 29	$s \in \{0.01, 0.03, 0.1, \mathbf{0.3}, 1\}$
	LIBERO 14, 60	$s \in \{0.01, 0.03, \mathbf{0.1}, 0.3, 1\}$
	LIBERO 54	$s \in \{\mathbf{0.01}, 0.03, 0.1, 0.3, 1\}$
Residual	can-mh; cube 1–4; chamber 2–3	$s \in \{0.01, 0.03, 0.1, \mathbf{0.3}, 1\}$
	square-mh	$s \in \{0.01, 0.03, \mathbf{0.1}, 0.3\}$
	transport-mh; tool-hang-ph	$s \in \{0.01, \mathbf{0.03}, 0.1, 0.3\}$
	cube 5	$s \in \{0.01, 0.03, 0.1, 0.3\}$ (all tied)
	block 1; chamber 1	$s \in \{0.01, 0.03, 0.1, 0.3, \mathbf{1}\}$
	block 2; LIBERO 64	$s \in \{0.01, \mathbf{0.03}, 0.1, 0.3, 1\}$
	LIBERO 14, 29, 60	$s \in \{0.01, 0.03, \mathbf{0.1}, 0.3, 1\}$
	LIBERO 54	$s \in \{\mathbf{0.01}, 0.03, 0.1, 0.3, 1\}$
DSRL	can-mh; square-mh; block 2; chamber 1–3	$b \in \{1, 1.5, \mathbf{2}\}$
	transport-mh; cube 3, 5; LIBERO 60, 64	$b \in \{\mathbf{1}, 1.5, 2\}$
	tool-hang-ph; cube 1–2, 4; block 1; LIBERO 14, 29, 54	$b \in \{1, \mathbf{1.5}, 2\}$
RLT	can-mh	$\alpha \in \{0.03, 0.1, 0.3, \mathbf{1}, 3\}$
	square-mh; transport-mh; tool-hang-ph	$\alpha \in \{0.1, 0.3, \mathbf{1}, 3, 10, 30, 100, 300, 1000\}$
	cube 1	$\alpha \in \{1, \mathbf{3}, 10, 30, 100, 300, 1000\}$
	cube 2, 4	$\alpha \in \{1, 3, 10, \mathbf{30}, 100, 300, 1000\}$
	cube 3, 5	$\alpha \in \{1, 3, 10, 30, \mathbf{100}, 300, 1000\}$
	block 1; chamber 1–3	$\alpha \in \{\mathbf{0.1}, 0.3, 1, 3, 10, 30, 100, 300, 1000\}$
	block 2	$\alpha \in \{1, 3, \mathbf{10}, 30, 100, 300, 1000\}$
	LIBERO 14, 29, 64	$\alpha \in \{10, 30, \mathbf{100}, 300, 1000\}$
	LIBERO 54	$\alpha \in \{10, 30, 100, \mathbf{300}, 1000\}$
	LIBERO 60	$\alpha \in \{10, \mathbf{30}, 100, 300, 1000\}$
TD3+BC	can-mh	$\alpha \in \{0.03, 0.1, 0.3, \mathbf{1}, 3\}$
	square-mh; cube 1	$\alpha \in \{1, \mathbf{3}, 10, 30, 100, 300, 1000\}$
	transport-mh	$\alpha \in \{0.1, 0.3, 1, \mathbf{3}, 10, 30, 100, 300, 1000\}$
	tool-hang-ph	$\alpha \in \{0.1, 0.3, \mathbf{1}, 3, 10, 30, 100, 300, 1000\}$
	cube 2, 4	$\alpha \in \{0.1, 0.3, 1, 3, 10, \mathbf{30}, 100, 300, 1000\}$
	cube 3	$\alpha \in \{0.1, 0.3, 1, 3, 10, 30, \mathbf{100}, 300, 1000\}$
	cube 5	$\alpha \in \{0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000\}$ (all tied)
	block 1; chamber 1–3	$\alpha \in \{\mathbf{0.1}, 0.3, 1, 3, 10, 30, 100, 300, 1000\}$
	block 2	$\alpha \in \{1, 3, \mathbf{10}, 30, 100, 300, 1000\}$
	LIBERO 14, 29, 64	$\alpha \in \{10, 30, 100, \mathbf{300}, 1000, \}$
	LIBERO 54	$\alpha \in \{10, 30, 100, 300, \mathbf{1000}\}$
	LIBERO 60	$\alpha \in \{10, 30, \mathbf{100}, 300, 1000\}$

Table 7: Baseline search grids. Bold values are selected. Here s is residual scale, b is the DSRL bound/noise scale, and u is UTD.

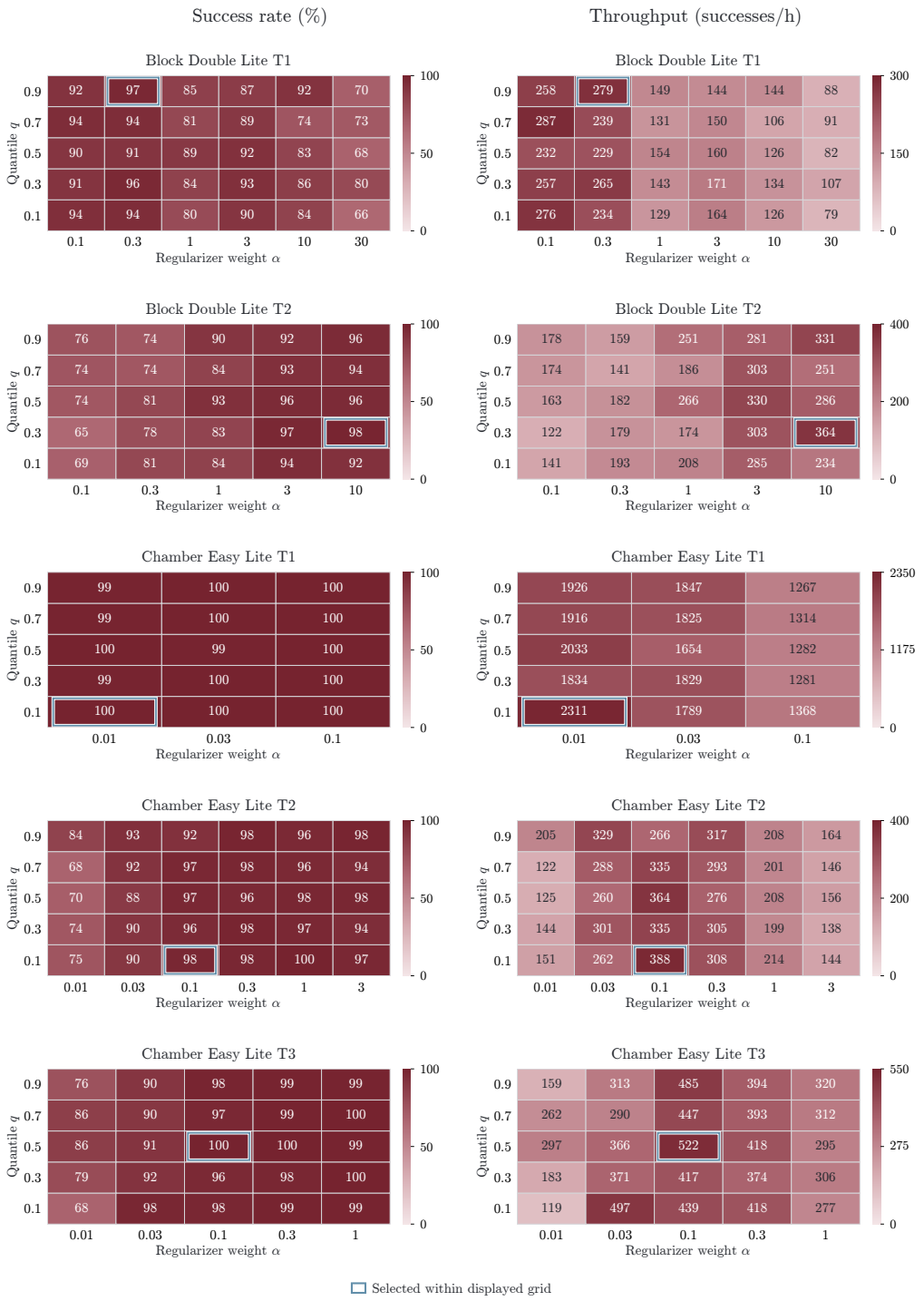

Figure 7: This figure demonstrates how the success rate, and throughput vary across different hyper parameter configs. Our method performs well across multiple magnitudes of BC weights.