TMRL: Diffusion Timestep-Modulated Pretraining Enables Exploration for Efficient Policy Finetuning

Matthew M. Hong[1], Jesse Zhang[1], Anusha Nagabandi[2], Abhishek Gupta[1]

[1]University of Washington [2]Amazon FAR

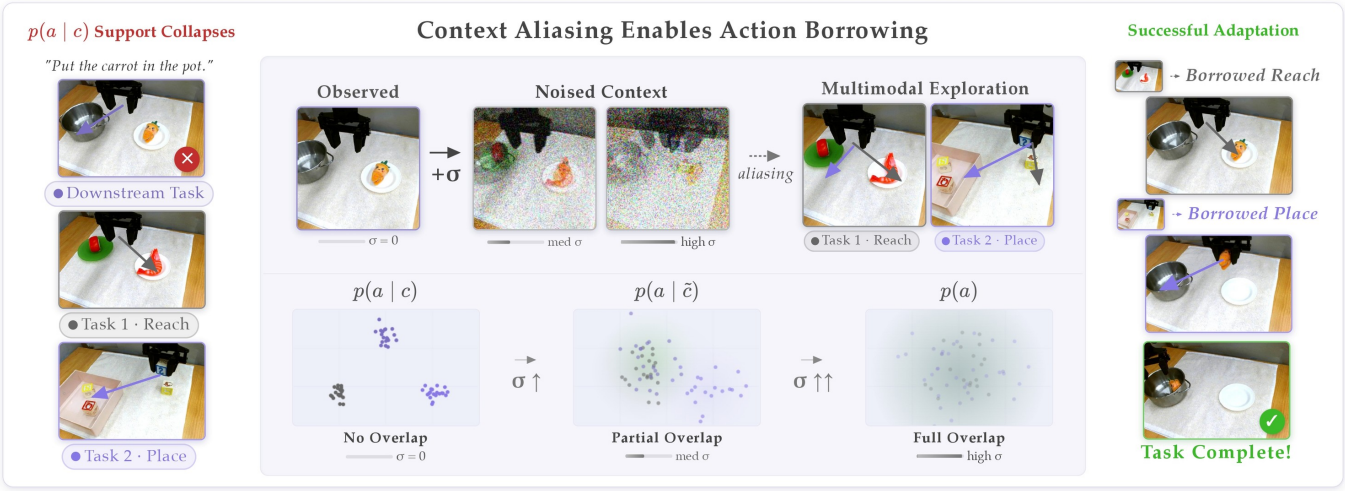

Fig. 1: TMRL bridges behavior cloning (BC) pre-training and RL fine-tuning by smoothing the conditioning of policy inputs (contexts). During pre-training, Context-Smoothed Pre-training (CSP) injects noise into contexts c, inducing a continuum from sharp imitation $p(a \mid c)$ to broader, marginal action distributions $p(a)$ via diffusion noise parameterized by σ (bottom row). This smoothing causes nearby contexts to overlap in representation space, with similar contexts (e.g., **downstream task**, **task 1** in left column) merging at lower σ than dissimilar ones (e.g., **task 2**). During RL fine-tuning, TMRL *learns* to dynamically *modulate* conditioning strength, interpolating between context-conditioned and exploratory actions for improved exploration and adaptation.

***Abstract*—Fine-tuning pre-trained robot policies with reinforcement learning (RL) often inherits the bottlenecks introduced by pre-training with behavioral cloning (BC), which produces narrow action distributions that lack the coverage necessary for downstream exploration. We present a unified framework that enables the exploration necessary to enable efficient robot-policy finetuning by bridging BC pre-training and RL fine-tuning. Our pre-training method, Context-Smoothed Pre-training (CSP), injects forward-diffusion noise into policy inputs, creating a continuum between precise imitation and broad action coverage. We then fine-tune pre-trained policies via Timestep-Modulated Reinforcement Learning (TMRL), which trains the agent to dynamically adjust this conditioning during fine-tuning by modulating the diffusion timestep, granting explicit control over exploration. Integrating seamlessly with arbitrary policy inputs, e.g., states, 3D point clouds, or image-based VLA policies, we show that TMRL improves RL fine-tuning sample efficiency. Notably, TMRL enables successful real-world fine-tuning on complex manipulation tasks in under one hour. Videos and code available at https://weirdlabuw.github.io/tmrl/.**

I. Introduction

A dominant paradigm for training real-world robotic policies is to first pre-train on large-scale demonstration datasets via imitation learning, and then fine-tune the resulting policy with reinforcement learning (RL) in deployment environments [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. This RL fine-tuning stage is critical for improving task precision, throughput and robustness [11, 6, 12], yet remains bottlenecked by sample efficiency due to the cost of real-world interaction. While prior work in robotics primarily focuses on improving the RL algorithms themselves, comparatively little attention has been paid to ensuring that pre-trained policies provide effective initializations for downstream RL. In this work, we propose a combined robot policy pre-training and RL fine-tuning framework that enables exploration for sample-efficient adaptation.

Standard behavior cloning (BC) trains a policy to imitate demonstrator actions [13, 14]. When demonstrations densely cover a context c (e.g., observation and task instruction), BC accurately models the conditional action distribution $p(a \mid c)$ [15, 16]. However, in sparsely covered regions and under distribution shift, BC overfits to observed data: the conditional support collapses and BC can assign near-zero probability to optimal actions. As a result, online rollouts during downstream adaptation yield little reward, causing reinforcement learning to fail to receive the learning signal needed to improve.

Building on this insight, recent work has examined the relationship between pre-training via supervised learning and RL fine-tuning [17, 18, 19, 1, 20], primarily in language domains. In robotics specifically, Wagenmaker et al. [1] shows that achieving high *action coverage*—the ability to sample actions a demonstrator would take—is *necessary* for efficient RL fine-tuning. Thus, they inject Gaussian action noise during training to increase coverage. However, there is a fundamental trade-off: random actions guarantee coverage but lack task

proficiency, while BC policies can achieve high proficiency but have limited out-of-distribution coverage. A naive solution is to collect more data, but doing so for every new setting can be costly. Instead, we ask: how can we pre-train policies that *adaptively* broaden their action distribution during RL?

Our key insight is to train policies that *interpolate* between the learned conditional $p(a \mid c)$ and marginal $p(a)$ action distributions. While $p(a \mid c)$ offers precise behavior in familiar contexts, $p(a)$ provides broad coverage needed for exploration. Interpolating between these extremes enables balancing exploitation and exploration based on the context's novelty.

We instantiate this idea through *Context-Smoothed Policies (CSPs)*, a pre-training framework that expands the support of BC policies by injecting noise into context inputs via a *forward diffusion process*. Introducing low levels of diffusion noise can alias similar contexts c [21]. Aliasing the input contexts leads to learning a broader conditional action distribution, since actions are "borrowed" across nearby contexts. As the noise increases, the policy smoothly transitions from a narrow conditional distribution $p(a \mid c)$ toward a broad marginal distribution $p(a)$. At maximum noise, c contains no information and the policy recovers $p(a)$ [22], ensuring full training distribution coverage. We pre-train CSPs across *all* noise levels, enabling smooth interpolation between these extremes. Therefore, the tradeoff between action coverage and conditioning fidelity can be selected at inference time.

Finally, having established a pre-training framework that exposes this tradeoff, the central challenge becomes how best to harness this flexible initialization during downstream RL finetuning. During RL fine-tuning, the optimal amount of context conditioning necessary—and therefore the balance between broad exploration and precise execution—can vary dramatically even within a single trajectory. To fully exploit our pre-trained CSPs, we introduce *Timestep-Modulated Reinforcement Learning* (TMRL) to dynamically adjust the diffusion timestep during RL. This mechanism provides an RL agent with an explicit control variable that modulates conditioning strength, enabling it to interpolate between conditional and marginal behaviors in the pre-trained policy and thereby explore more effectively. Ideally, a policy *learns* when to rely on precise imitation and when to broaden its action support. In practice, TMRL is simple and easy to implement, enabling steering/fine-tuning of diverse policies, from state-input diffusion policies to vision-language-action (VLA) models.

We evaluate our proposed methodology across a range of simulated and real-world robotic tasks. First, we show that CSPs alone substantially improve action coverage and achieve stronger zero-shot performance on unseen tasks over standard BC and prior pre-training approaches. TMRL then converts this coverage advantage into significantly better RL sample efficiency on manipulation and navigation tasks in simulation, outperforming state-of-the-art steering methods even when the base policy contains sparse behavioral coverage. We further show that context-smoothing extends naturally to VLA and 3D-input policies, where noising VLA embeddings and point clouds, respectively, enables broader exploration across image-input and dexterous manipulation tasks. Finally, we demonstrate that our approach scales to the real world, enabling rapid RL of manipulation behaviors within *an hour* of experiment time while the baseline achieves near-zero success.

II. Related Work

Existing research on RL fine-tuning in robotics spans a wide spectrum of topics. In this section, we review these paradigms and situate both context-smoothed policies and TMRL within them. Specifically, we differentiate our approach in (i) the design of pre-training objectives for action coverage, (ii) the explicit, dynamic tuning of exploration optimism during post-training, and (iii) the role of restricted information in inducing structured state aliasing.

Pre-training for RL Fine-tuning. Standard practice in robot learning involves pre-training policies on large-scale demonstration datasets via offline RL or BC, followed by RL fine-tuning [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 6, 12]. However, these imitation objectives learn the narrow conditional action distribution, $p(a \mid c)$. In low-density or out-of-distribution contexts, this conditional distribution overcommits to observed data, collapsing its support and assigning near-zero probability to potentially optimal actions.

Recent findings in language modeling suggest that standard imitation losses are a poor predictor of downstream fine-tuning performance [17, 18, 19, 20]. Instead, PostBC [1] demonstrates that successful RL fine-tuning of robot policies requires sufficient *action coverage* to enable RL to sample and amplify optimal behaviors. PostBC specifically fits the posterior distribution of the demonstrator's behavior using an ensemble of single-action policies, then injects Gaussian noise into the actions at low-density states to broaden the pre-training distribution by using a covariance estimate from the ensemble. While mathematically sufficient for coverage, injecting unstructured noise directly into the action space often leads to "dithering" and execution-level incoherence, as the noise is typically uncorrelated across timesteps.

Instead, we introduce *context-smoothed policies*. Rather than perturbing actions, we inject sampled diffusion noise into the *input* space of a BC policy during pre-training. By shifting the perturbation to the context, we enable an RL agent to borrow coherent action sequences from nearby contexts, the marginal, or a mix of the two via diffusion timestep interpolation, guaranteeing exploratory coverage without the dithering inherent to unstructured action-space noise.

RL Fine-tuning and Policy Steering. Once pre-trained, policies are often improved via RL fine-tuning. Some approaches apply entropy regularization or action-noise strategies in offline or fine-tuning settings [23, 24, 25], or directly fine-tune pre-trained diffusion policies [26], but still rely on standard BC policies, which may overfit and perform poorly in unseen deployment settings. More structured methods perform *steering* over a frozen pre-trained policy [27, 28, 29, 30, 7, 2, 31], training a high-level RL policy to effectively modulate the base policy's output. However, in these settings, steering is strictly limited to the support of the base policy's conditional

action distribution $p(a \mid c)$, so if $p(a \mid c)$ collapses in unseen contexts, as standard BC policies routinely do, the optimal action is unreachable regardless of how well the RL policy is trained. TMRL addresses this collapse by allowing the agent to dynamically tune its reliance on context conditioning, interpolating between $p(a \mid c)$ and the marginal distribution $p(a)$, enabling adaptive exploration beyond the base policy while remaining grounded in offline data.

Generalization via Restricted Information. Finally, context-smoothed policies align with work showing that restricting input information can improve policy generalization and robustness. Prior work demonstrated that intentionally limiting observations can lead to better robustness in control, imitation learning, and image generation [32, 33, 34, 35, 36, 37, 21, 38, 39]. For example, Stem-Ob [21] trains policies with diffusion-noised visual conditioning to improve zero-shot generalization. Most similar to ours, Classifier-Free Guidance [40] can also improve generalization by training both a conditional and a marginal policy, where the conditional policy *guides* the marginal policy, but our ablations in Section VI-E demonstrate that CSPs are a more effective pre-training procedure for RL fine-tuning on new tasks.

A closely related idea is that of *randomized smoothing*, used in stochastic optimization [41], for smoothing discontinuous gradients [42], or to increase classifier robustness [43], in which random noise is applied to smooth a vector. For TMRL, we train a *context-smoothed* policy that sees conditioning (context) inputs *smoothed* by a noising diffusion forward process. This context-smoothed policy serves as a steerable base policy, which TMRL trains an RL policy over, enabling explicit and tunable control over exploration optimism.

III. The Need for Action Distribution Interpolation

We begin with a simple experiment that illustrates why we need a *pre-training* procedure that can interpolate between the marginal action distribution $p(a)$ and a conditional distribution $p(a \mid c)$ when encountering unseen tasks. In a simple 2D point-maze from the simulated OGBench benchmark [44], we train two policies to approximate $p(a)$ and $p(a \mid c)$ where c is the goal position. These policies are trained on the `pointmaze-large-navigate` from OGBench [44] and then evaluated on a larger, out-of-distribution maze with an unseen goal position. In Figure 2, we visualize samples from the two distributions and the corresponding paths taken at various starting positions, gradually approaching the goal. The agent is in yellow and the goal position in pink. Figure 2 demonstrates that when the agent is far away, $p(a \mid \bullet)$ is overfit to the training distribution and the sampled action sequences lead it *further away* from the goal. Meanwhile, $p(a)$'s action coverage is very broad, but does place more probability mass on actions that take the agent closer to the goal. As the agent gets closer to the goal, $p(a \mid c)$ becomes more useful for reaching the goal.

Takeaways for RL. In general cases where $p(a \mid c)$ is overfit, $p(a)$ represents an action distribution that is more likely to cover the optimal action distribution necessary to solve the task. However, a pre-training recipe that naïvely trains both policies directly is insufficient for two reasons: (1) it is unclear how a downstream RL algorithm can decide when to pick each distribution to solve a given downstream task; and (2) while $p(a)$ can provide greater coverage of the optimal action, it represents an overly broad distribution, which can make downstream learning slow and unsafe.

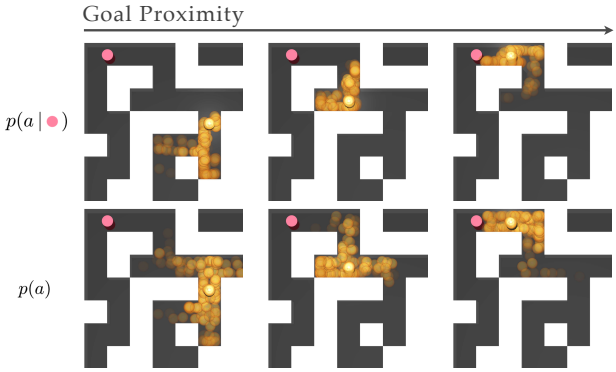

Fig. 2: For an unseen goal position, the conditional action distribution $p(a \mid \bullet)$ remains narrow, lacking sufficient support for the agent to reach the goal. In contrast, the marginal $p(a)$ provides the required coverage, though with too much action variance near the goal. In this setting, an optimal policy should more closely match $p(a)$ at the beginning of the trajectory and $p(a \mid \bullet)$ at the end. *Context-smoothed* policy pre-training enables TMRL to *interpolate* between both distributions for more effective RL fine-tuning.

To resolve both problems, we propose pre-training a policy via **context-smoothing** to interpolate between $p(a \mid c)$ and $p(a)$. Then, at inference time, we fine-tune with **TMRL** to learn *when* and *how* to move along this interpolation, dynamically adjusting coverage to explore effectively while simultaneously learning the new task. At convergence, TMRL interpolates between $p(a)$ and $p(a \mid c)$ to maximize optimal action coverage (2) and also to determine when to lean more on either distribution to solve the task (1).

IV. Timestep Modulated RL on Context-Smoothed Policies

A. Problem setting

Pre-training. We assume access to an offline dataset of near-optimal pre-training trajectories $\mathcal{D} = \{\tau_j\}$, where each trajectory $\tau_j = \{(c_i, a_i)\}_{i=1}^{T_j}$ contains context–action pairs. In this paper, we use context to abstractly define a variety of inputs to the policy, such as states, RGB images, point clouds, language instructions, and other task descriptors. More generally than maximum likelihood, we learn a policy via a supervised objective with a user-specified loss ℓ:

$$\min_{\theta} \; \mathbb{E}_{(c_t, a_t) \sim \mathcal{D}} \Big[\ell\big(\theta;\; c_t, a_t\big) \Big]. \tag{1}$$

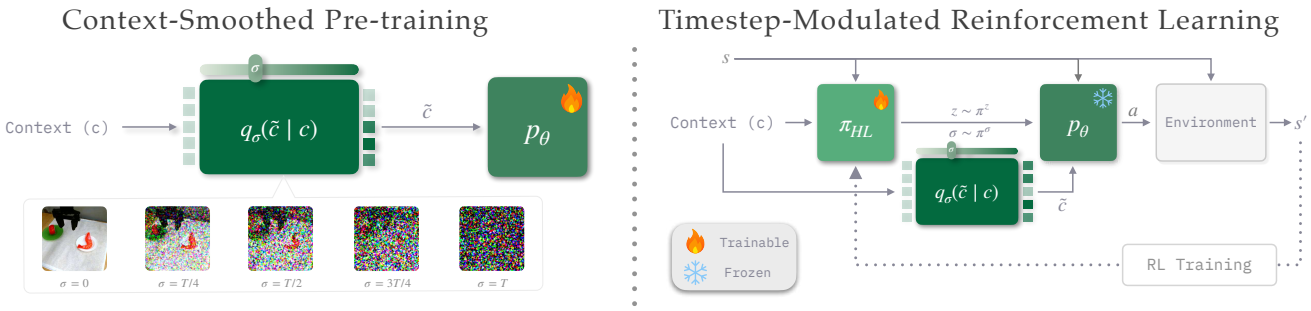

Fig. 3: **Timestep-modulated exploration via context smoothing.** (*Left*) During pre-training, a steerable policy p_θ is trained across all noise levels σ by corrupting the context c via the kernel $q_\sigma(\tilde{c} \mid c)$, producing a policy that can be queried at any conditioning strength during inference. (*Right*) During RL fine-tuning, TMRL exposes p_θ with a *context-noise dial* σ as an explicit control variable for the high-level policy π_{HL}.

This formulation subsumes imitation learning as a special case: choosing $\ell(\theta; c, a) = -\log p_\theta(a \mid c)$ recovers maximum likelihood (behavior cloning). Other choices of ℓ yield different training algorithms and model classes, e.g., denoising/score-matching objectives for score-based or diffusion policies.

In this work, we will focus on a class of generative control policies (GCPs) [16, 45, 46], where policies are represented by generative models such as diffusion/flow models and are trained to maximize a lower bound of the action likelihood.[1] Note that our pre-training and fine-tuning procedures are general and do not particularly rely on GCPs.

RL Fine-tuning. Given this pre-trained policy, our goal is to develop an efficient RL fine-tuning algorithm in novel contexts. Of particular interest in this work is the class of RL algorithms for "steering" pre-trained policies [2, 1, 47], although our formulation is more broadly applicable. In GCPs, the inference procedure allows for steerability via a latent variable z, giving $p(a \mid c, z)$. For flow/diffusion policies, this corresponds to the initial noise used during denoising [2]. Thus, the RL agent represents a high-level policy $\pi_{\text{HL}}(z \mid c)$ that selects the latent variable z to maximizing downstream reward r sum discounted by γ in conjunction with $p(a|c, z)$:

$$\max_{\pi_{\text{HL}}} \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r_t\Big], \tag{2}$$
$$\text{s.t.} \quad z_t \sim \pi_{\text{HL}}(\cdot \mid c_t), \quad a_t^{1:H} \sim p_\theta(\cdot \mid c_t, z_t).$$

Action Coverage. A fundamental bottleneck in RL fine-tuning is distribution shift: in novel contexts, the pre-trained conditional distribution $p_\theta(\cdot \mid c, z)$ often overcommits, resulting in a collapsed action distribution. Wagenmaker et al. [1] formalizes this issue through the concept of *demonstrator action coverage*. Specifically, a pre-trained policy achieves coverage with parameter $\kappa > 0$ if, $\forall(c, a)$:

$$p_\theta(a \mid c) \geq \kappa \cdot p^\beta(a \mid c), \tag{3}$$

where p^β represents the true demonstrator policy. This coverage is a strict prerequisite for successful RL fine-tuning. Since fine-tuning algorithms rely on online rollouts to update behaviors, a collapsed initialization (where $\kappa \to 0$ for unseen optimal actions) will never sample the transitions required to discover meaningful reward signal. Because standard BC fails to guarantee meaningful coverage in low-density regions, it often yields initializations incapable of downstream learning. Our method resolves this by instantiating a new class of RL fine-tuning algorithms based on "context-smoothed pre-training", which allows the RL agent to adaptively modulate the amount of action coverage needed for different contexts c.

[1]For simplicity, we will stick to the (a, c) notation, although this is naturally extended to H-step action chunks $a^{t:t+H}$ and context histories $c_{t-H:t}$.

B. *Context-smoothing*

Intuitively, context-smoothed policies make a simple change to standard policy pre-training: they inject noise into the context c while being pre-trained on the offline dataset $\mathcal{D}$. We show that doing so enables policies to adaptively expand their action coverage during evaluation and RL fine-tuning.

To define this formally - let $q_\sigma(\tilde{c} \mid c)$ be a corruption kernel that injects noise into the context, with noise scale $\sigma \geq 0$. We define the *context-smoothed* steerable policy as the mixture

$$\begin{aligned} p_{\theta,\sigma}(a \mid c, z) &:= \mathbb{E}_{\tilde{c}\sim q_\sigma(\cdot|c)}\Big[p_\theta(a \mid \tilde{c}, z)\Big] \\ &= \int p_\theta(a \mid \tilde{c}, z)\, q_\sigma(\tilde{c} \mid c)\, d\tilde{c}. \end{aligned}$$

We refer to $p_{\theta,\sigma}$ as a *context-smoothed policy*. Given a context c, action inference with a context-smoothed policy amounts to corrupting the context with $\tilde{c} \sim q_\sigma(\tilde{c} \mid c)$, and then sampling actions with the corrupted context from $p_\theta(a \mid \tilde{c}, z)$.

Intuition behind context smoothed policies: For $\sigma \downarrow 0$, $q_\sigma(\tilde{c} \mid c)$ concentrates near c, so $p_{\theta,\sigma}(\cdot \mid c, z)$ approaches the original conditional controller. As σ increases, the corrupted context $\tilde{c}$ becomes less informative, and the induced action distribution becomes a broader mixture over behaviors associated with *nearby / aliased* contexts. This controlled interpolation between the conditional distribution $p(a|c)$ and the marginal $p(a)$ creates *structured action coverage expansion*: instead of random action noise, the policy can borrow coherent action chunks from related contexts present in the dataset.

Algorithm 1 Context-smoothed pre-training.

Input: dataset $\mathcal{D} = \{(c, a^{1:H})\}$, corruption kernel q_σ, noise-level sampler $\sigma \sim \mathcal{S}$
Initialize: parameters θ of $p_\theta(a^{1:H} \mid c, z, \sigma)$ (and any latent prior over z if used)
while not converged **do**
  Sample $(c, a^{1:H}) \sim \mathcal{D}$
  Sample $\sigma \sim \mathcal{S}$ (diffusion timestep t_c in our instantiation)
  Sample $\tilde{c} \sim q_\sigma(\cdot \mid c)$ (Eq. (4))
  Update θ to minimize $\ell(\theta;\ a^{1:H}, \tilde{c}, \sigma)$ (Eq. (5))
end while

Algorithm 2 Timestep-Modulated RL (TMRL), simplified.

Input: pretrained context-smoothed controller $p_\theta(\cdot \mid c, z, \sigma)$; initialize replay buffer $\mathcal{B}$
Initialize high-level actor $\pi_{\text{HL}}(z, \sigma \mid s)$ and critic(s)
for each training iteration **do**
  Observe s_t and set $c_t \leftarrow c(s_t)$
  Sample $(z_t, \sigma_t) \sim \pi_{\text{HL}}(\cdot \mid s_t)$, $\tilde{c}_t \leftarrow c_t$ smoothed with σ_t
  Sample and execute action chunk $a_t^{1:H} \sim p_\theta(\cdot \mid \tilde{c}_t, z_t, \sigma_t)$
  Store transitions in $\mathcal{B}$ and update critics with off-policy RL
  Update π_{HL} using critic gradient (details in App. Section C)
end for

a) Implementation of corruption kernel $q_\sigma(\tilde{c} \mid c)$: We define the corruption kernel q_σ via an iterative forward-noising process over contexts, much like the forward process of a diffusion model. Let $c_0 \equiv c$ and choose a variance schedule $\{\beta_t\}_{t=1}^{T_c}$ with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i$. The forward process implies the closed-form marginal: $q(c_t \mid c_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, c_0,\ (1-\bar{\alpha}_t)I\big)$. We take the corruption kernel at noise level t_c to be

$$q_\sigma(\tilde{c} \mid c) \equiv q_{t_c}(\tilde{c} \mid c) = \mathcal{N}\big(\sqrt{\bar{\alpha}_{t_c}}\, c,\ (1 - \bar{\alpha}_{t_c})I\big). \tag{4}$$

Since the forward corruption process is entirely determined by how many time-steps it is run forward, the noise-level σ for $q_\sigma(\tilde{c} \mid c)$ can be parameterized entirely by controlling the **diffusion timestep** $t_c \in \{0, \dots, T_c\}$, hence the moniker timestep modulated RL. We use σ to refer to corruption kernels q_σ generally and t_c for our specific instantiation with diffusion noise.

C. Pre-training a context-smoothed policy

Training a context-smoothed policy requires learning action predictors conditional on corrupted context - $p_\theta(a \mid \tilde{c}, z)$. Towards realizability, we train a policy class that can be queried at arbitrary context-noise levels by providing σ (or t_c) as an explicit input to the policy - $p_\theta(a \mid \tilde{c}, z, \sigma)$. Concretely, we introduce a policy - $p_\theta(a \mid \tilde{c}, z, \sigma)$, and train it so that, for each training pair $(c, a) \sim \mathcal{D}$, we maximize action likelihood while noising the context with a small modification to Equation (1):

$$\min_\theta\ \mathbb{E}_{(c,a)\sim\mathcal{D}}\ \mathbb{E}_{\substack{\sigma\sim\mathcal{S} \\ \tilde{c}\sim q_\sigma(\cdot|c)}}\Big[\ell\big(\theta;\ a, \tilde{c}, \sigma\big)\Big], \tag{5}$$

where $\mathcal{S}$ is a distribution over noise levels and ℓ is any supervised learning objective suitable for the chosen steerable policy class. In our specific instantiation using GCPs as p_θ, ℓ is the denoising score-matching or flow-matching loss [48, 49, 50] and σ corresponds to the diffusion noise timestep used to corrupt $\tilde{c}$, t_c. Notably, this is *not* a much more complicated procedure than standard imitation learning; it merely introduces *controlled context smoothing* during pre-training, similar to data augmentation, and conditioning p_θ on the amount of smoothing σ. As we show next, this context-smoothed pre-trained policy can result in a massive improvement for efficient RL fine-tuning. See Algorithm 1 for pseudocode.

D. RL fine-tuning with timestep-modulated context smoothing

Compared to standard pre-training, training a context-smoothed policy $p_\theta(a^{1:H} \mid c, z, \sigma)$ naturally provides an avenue to expand the exploration space during RL fine-tuning. For example, in *steering algorithms* [2], which RL fine-tune by training a high-level RL policy to select the latent noise z to initialize action selection from a GCP (see Section IV-A for a refresher), the high-level policy can now select the steering vector z *and* the context-smoothing noise level σ applied to c:

$$(z_t, \sigma_t) \sim \pi_{\text{HL}}(\cdot \mid s_t),\ \tilde{c}_t \sim q_{\sigma_t}(\cdot \mid c(s_t)),\ a_t \sim p_\theta(\cdot \mid \tilde{c}_t, z_t, \sigma_t).$$

With a frozen context smoothed base policy, we optimize $\pi_{\text{HL}}(z, \sigma \mid s)$ using any RL method (e.g., off-policy actor-critic [51]) to maximize the expected return in Equation (2). Intuitively, σ provides an exploration–exploitation, action coverage (Eq. (3)) dial: large σ aliases contexts (broader borrowing, better exploration, more coverage), while small σ sharpens conditioning (better exploitation, lower action coverage) once progress is found. See Algorithm 2 for pseudocode.

V. Theoretical Analysis: Benefits of Context Smoothing

In this section, we provide theoretical evidence and empirical analysis to better understand the gains from timestep-modulated, context-smoothed policies for RL fine-tuning.

a) Empirical Analysis: We first empirically validate that increasing σ (or t_c) yields smooth, meaningful behavioral changes. We show in Figure 4 that as corruption noise increases, the distribution of actions at a given context broadens, thereby increasing overlap with nearby contexts. Taken to the extreme, this shows interpolation between the conditional $p(a \mid c)$ and marginal $p(a)$ action distribution.

b) Theoretical Analysis: Our theoretical analysis formalizes how, under Gaussian context-smoothing, smoothing mitigates the coverage collapse of standard BC. Informally, we make two claims: **(i)** increasing context noise increases action distribution overlap across different contexts, and **(ii)** closer contexts overlap more than farther contexts. Referring back to Equation (3), these claims together show that at an individual context c, if there is a different context c' which increases demonstrator action coverage at c, then increasing context corruption σ increases overlap with the higher action coverage distribution, $p(\cdot \mid c')$.

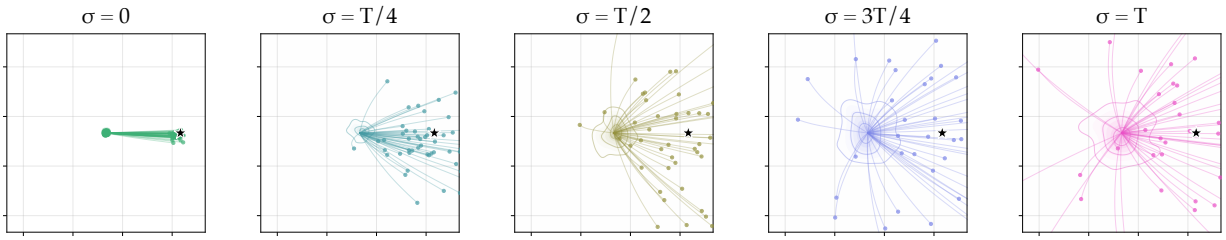

Fig. 4: **Empirical validation of context smoothing.** We train a context-smoothed diffusion policy $p(x, y \mid \tilde{\theta}, \sigma)$ with Equation (5) to produce 2D points on a unit circle conditioned on $c = \theta$, where $\sigma \in 0, 1, \ldots, T$ denotes diffusion scheduler timesteps. Each panel shows $p(x, y \mid \tilde{\theta} = 0, \sigma)$ at a fixed level of context noise σ; the shaded region and contours visualize the spread of the conditioning distribution over context inputs at each σ. As $\sigma \to 0$, conditioning sharpens — the contours collapse to a point — and outputs concentrate around the unit circle at $\theta = 0$; as σ increases, the contours widen, reflecting an induced action distribution that becomes broader and overlaps more across nearby contexts.

c) Setup (Gaussian context smoothing): For a conditional action distribution $p(\cdot \mid c)$ with $c \in \mathbb{R}^d$, consider its Gaussian context-smoothed version:

$$p_\sigma(\cdot \mid c) \;\coloneqq\; \mathbb{E}_{w \sim \mathcal{N}(0, I_d)} \big[p(\cdot \mid c + \sigma w) \big]. \tag{6}$$

Let $\mathrm{TV}(\cdot, \cdot)$ denote total variation distance, and define overlap $\mathrm{Ov}(P, Q) \coloneqq 1 - \mathrm{TV}(P, Q)$. We make the following statements about $p_\sigma(\cdot \mid c)$.

Theorem 1 (Smoothing increases overlap and makes the policy Lipschitz in context)**.** *For any two contexts $c, c' \in \mathbb{R}^d$,*

$$\mathrm{TV}\big(p_\sigma(\cdot \mid c), p_\sigma(\cdot \mid c')\big) \;\le\; \frac{\mathbb{E}\|w\|}{\sigma} \, \|c - c'\|. \tag{7}$$

Equivalently,

$$\mathrm{Ov}\big(p_\sigma(\cdot \mid c), p_\sigma(\cdot \mid c')\big) \;\ge\; 1 - \frac{\mathbb{E}\|w\|}{\sigma} \, \|c - c'\|. \tag{8}$$

Interpretation. Theorem 1 formalizes two central components of maximizing action coverage for RL fine-tuning:

- **More noising ⇒ broader coverage.** For fixed (c, c'), increasing σ decreases the upper bound in Equation (7) and increases the lower bound in Equation (8). For example, if c' is a state from the demonstration data and c is a novel, out-of-distribution state, increasing σ forces the policy at c to overlap with the demonstrator's behavior at c', thereby *increasing demonstrator action coverage*.
- **Closer contexts ⇒ targeted, coherent coverage.** For fixed σ, the bound scales linearly with $\|c - c'\|$: nearby contexts yield larger guaranteed overlap than distant contexts. This ensures that the *coverage expansion is not simply unstructured noise, but structured aliasing*: the policy borrows action chunks from related contexts.

See proof in Appendix Section A. The next corollary now uses Theorem 1 to demonstrate that if two contexts exhibit overlap *before smoothing*, they will exhibit even greater overlap *after smoothing*:

Corollary 1.1 (Sufficiently noisy smoothing ensures increased coverage relative to the base, unsmoothed policy)**.** *Fix c, c' and let $\Delta = c - c'$. Assume the pair (c, c') satisfies a (local) identifiability lower bound:*

$$\mathrm{TV}\big(p(\cdot \mid c), p(\cdot \mid c')\big) \;\ge\; m\|\Delta\| \tag{9}$$

for some $m > 0$. If

$$\sigma \;\ge\; \frac{\mathbb{E}\|w\|}{m}, \tag{10}$$

then smoothing reduces distinguishability (TV) and hence increases overlap:

$$\mathrm{Ov}\big(p_\sigma(\cdot \mid c), p_\sigma(\cdot \mid c')\big) \;\ge\; \mathrm{Ov}\big(p(\cdot \mid c), p(\cdot \mid c')\big). \tag{11}$$

Interpretation. Standard BC can create disjoint, narrow action supports for highly identifiable contexts (identifiability Equation (9)). Once σ exceeds the threshold in Equation (10), the smoothing-induced aliasing dominates in Equation (11). **This demonstrates that smoothing increases action coverage relative to a base BC policy**, mitigating the risk of zero-probability optimal actions during RL fine-tuning. A complete proof appears in Appendix Section B.

Takeaway for RL Fine-tuning. We've shown that context smoothing provably increases overlap relative to the original conditional policy: for any context c, if $\exists\, c'$ with better demonstrator action coverage, this increased overlap directly translates into a higher coverage parameter κ for c in Equation (3). These results allow the RL policy to adaptively change coverage for each c by treating σ as an adaptive coverage dial: large σ guarantees broad action coverage by pulling the distribution toward the marginal $p(a)$, while small σ preserves precise conditioning for exploitation.

Unlike unstructured noise, which expands coverage at the cost of execution coherence, context-smoothed policies expand coverage using coherent actions from nearby contexts.

Summary. In summary, we propose Timestep-Modulated Reinforcement Learning (TMRL) over *context-smoothed policies*, reframing policy pre-training for RL post-training from *unstructured* action noise into a *structured* process of context-smoothing. We first train a generative policy $p_\theta(a \mid \tilde{c}, z, \sigma)$ by injecting controlled noise into the context c via a forward diffusion process. This creates a policy that can be queried

at any noise level $\sigma \in [0, T_c]$, where σ determines the conditioning noise and guarantees coverage over the prior data. Then, during RL fine-tuning, a high-level policy $\pi_{\text{HL}}(z, \sigma \mid s)$ learns to dynamically treat σ as a coverage dial. By adaptively increasing σ, the agent aliases the current context with nearby contexts in the dataset, "borrowing" coherent behaviors to explore securely beyond the base policy's behavior distribution. Put together, our work shows the merit of rethinking the role of pre-training for finetuning, especially the responsiveness of the policy to noisy input representations.

VI. Experiments

Our experiments study the following research questions:

- **(Q1)** Does context smoothing produce better action coverage over other pre-training approaches?
- **(Q2)** Does TMRL effectively RL fine-tune policies trained across a variety of policy conditioning variables?
- **(Q3)** Does TMRL enable real-world RL on VLAs?
- **(Q4)** How does context-smoothing and TMRL compare to alternative coverage expansion mechanisms? How does TMRL learn to control context smoothing over time?

A. *(Q1) Comparing Pre-Training Action Coverage*

We first compare the effect of various pre-training procedures on action coverage by measuring success rates on unseen tasks. We compare a context-smoothed pre-training (CSP) policy against standard BC, i.e., training $p(a \mid c)$ with Eq. (1), and against PostBC [1]. Overall, **CSP outperforms both baselines in zero-shot success rate of unseen tasks**.

We demonstrate this result on two tasks from OGBench [44]: navigation (`pointmaze-giant`) and manipulation (`cube-single`). For our navigation task, we train policies on the `pointmaze-large-navigate` dataset and evaluate on the larger `pointmaze-giant` environment. The downstream environment has a larger state space, necessitating a broader action distribution. For our manipulation task, `cube-single`, we use a filtered `cube-single-play` dataset and evaluate adaptation for initial positions beyond this training dataset. See Section F for further environment details.

In Figure 7, we plot "success @ K," measuring the fraction of out-of-distribution initial states for which at least one of the K base policy rollouts succeeds. This success rate represents a direct empirical measurement of *demonstrator action coverage* from Eq. (3). On both OGBench tasks, CSP achieves higher success rates at every K with a fixed smoothing σ, with the gap most pronounced on `cube` where BC and PostBC have zero success at all K. Next, we show that CSP's broader action coverage translates to better RL performance with TMRL.

B. RL Fine-tuning Baselines

We compare TMRL against prior approaches for RL with prior data or for combining pre-training and RL fine-tuning across multiple evaluation tasks, as shown in Figure 5:

- RLPD [25]: off-policy RL that learns online while incorporating offline data as an additional buffer. It relies on Gaussian action noise for expanding action coverage.
- SPiRL [27]: a hierarchical RL algorithm that trains an RL policy over pre-trained skills learned from offline data. It uses skill sampling for expanding coverage.
- DSRL [2]: a steering algorithm that first trains a diffusion policy over action sequences with standard BC, then performs off-policy RL over the policy's noise space.
- PostBC [1]: pre-trains with additive Gaussian action noise to expand action coverage and then fine-tunes with DSRL.

C. *(Q2) RL Fine-tuning Across Varied Policy Conditioning*

Now, we examine how TMRL enables effective RL fine-tuning on difficult, unseen tasks. We consider downstream tasks that are out-of-distribution for the base policy, i.e., the training data contains no demonstrations for these downstream tasks. All results are means and standard deviations over 5 seeds.

State-Based Conditioning. As seen in the two left plots in Figure 6, we observe clear differences in how methods handle out-of-distribution generalization for the aforementioned OG bench domains from Section VI-A. RLPD achieves limited success, as it explores purely through additive Gaussian noise.

In contrast, methods that exploit action priors from the pre-training data perform better; DSRL reaches around 90% success rates in `pointmaze-giant`; however, it achieves near-0 success rates on the higher-dimensional action space task `cube-single` as it lacks any mechanism to go beyond the coverage of the base policy. PostBC performs very similarly to DSRL—while it expands action coverage during pre-training, the coverage comes from single-step Gaussian noise, causing some dithering behavior, and, unlike TMRL, cannot be adaptively controlled during RL fine-tuning. TMRL consistently outperforms these baselines, achieving an overall 101% improvement over the best-performing baseline across both tasks by *learning* to systematically expand coverage of context-smoothed policies. Moreover, while several baselines exhibit high variance across seeds, TMRL is notably more stable while still performing well.

Image-Based Tasks and VLM Embedding Conditioning: We next consider a set of image-based tasks with a large pre-trained VLA policy, π_0 [52], which uses VLM embeddings to condition a flow-based action expert. We evaluate on the LIBERO benchmark [53]. To enable TMRL we fine-tune a context-smoothed π_0 model on the `Libero-{Spatial, Object, Goal, 10}` datasets by adding noise to the VLM embeddings context c before they are input to the action expert head (see Figure 8). We then evaluate adaptation on two unseen tasks: a position-perturbed version of `Libero-Goal` task [54] and a task from the `Libero-90` task suite. We provide additional environment details in Section F.

Figure 6 shows that TMRL leverages action chunks from different tasks in the dataset to solve the `libero-goal` task more efficiently than DSRL. RLPD also learns the task because it is very short-horizon. In `libero-90`, we observe that only TMRL learns the task; TMRL leverages action sequences from other tasks, meanwhile DSRL overfits to

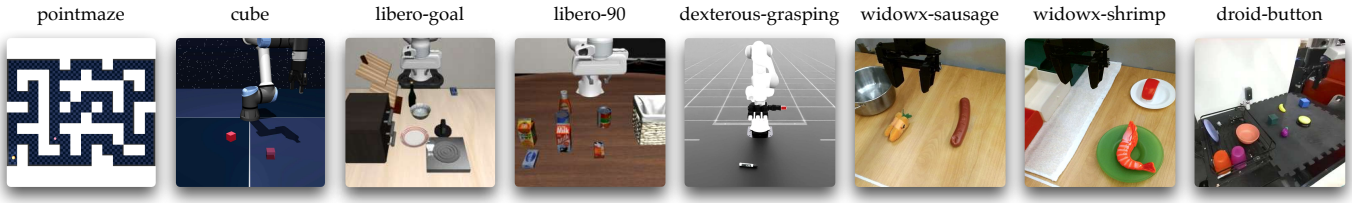

Fig. 5: **Evaluation.** We evaluate TMRL across 8 tasks spanning navigation and manipulation in simulation and the real world.

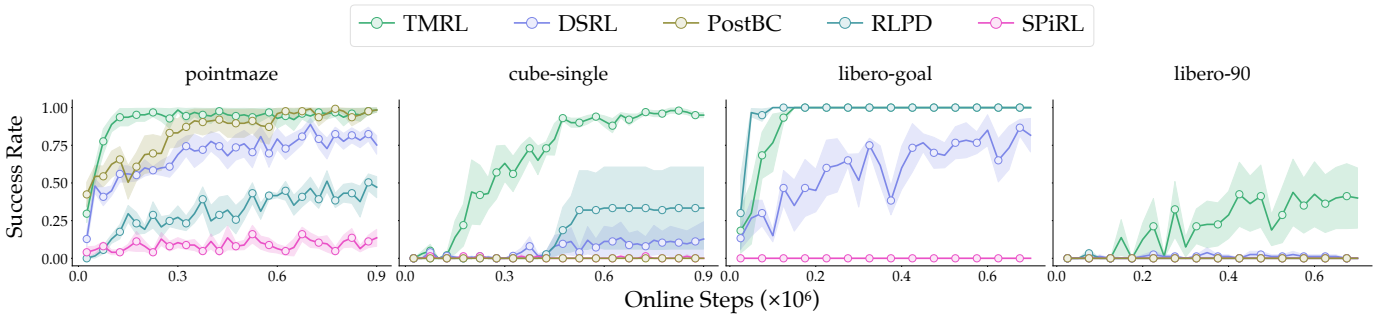

Fig. 6: **RL Success Rates for simulation tasks.** TMRL attains near 100% success rate in both OGBench tasks, outperforming the best baselines by 14% in `pointmaze-giant` and 200% in `cube-single` at final performance. In `libero-goal`, TMRL and RLPD [25] both reach 100% success. However, for the longer-horizon `libero-90` task, only TMRL explores sufficiently to achieve non-trivial success rates.

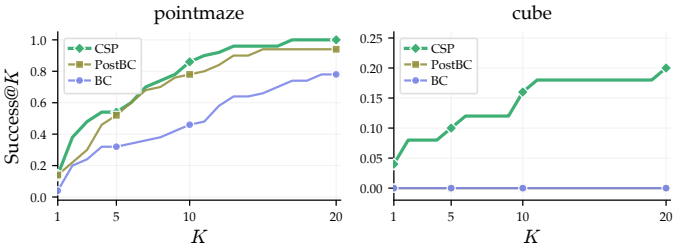

Fig. 7: **CSP unlocks better action coverage before RL fine-tuning.** We measure the Success@K for context-smoothed pre-training against standard BC and PostBC. CSP achieves greater success@K across all K on both tasks.

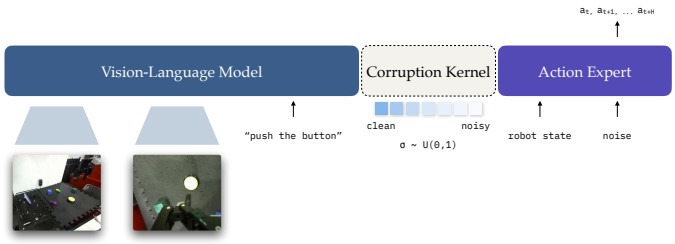

Fig. 8: Context Smoothed Pre-training for VLAs: we propose applying the corruption kernel (Eq. (4)) to VLM embeddings input to the action expert.

picking up the same object each time, resulting in an overly narrow action distribution that is insufficient to solve the task.

Pointcloud Conditioning: Next, we look at a simulated dexterous manipulation grasping setting with a LEAP hand [55] on a Franka in IsaacLab [56]. We are interested in evaluating TMRL's RL fine-tuning ability on a *pointcloud*-input policy; we examine whether context-smoothing on pointclouds enables TMRL to adapt to grasping new objects. We pre-train context-smoothed policies on three can-shaped objects and evaluate on a new marker-shaped object (as visualized in Figure 17).

While grasp strategies do not immediately transfer, we find that noised pointclouds enable TMRL to *share grasping strategies* across different objects, enabling broader exploration and, consequently, faster learning and 2.5× higher final success rates than DSRL in Figure 9.

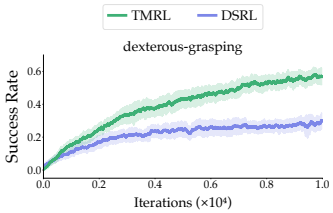

Fig. 9: TMRL enables efficient steering of 3D-input policies for dexterous grasping.

D. *(Q3): Does TMRL enable real-world RL on VLAs?*

Next, we demonstrate that TMRL enables effective adaptation of VLA policies in the real world. We evaluate on two platforms: a WidowX 250 6-DoF robot arm using the BridgeData-v2 [57] setup and dataset, and a Franka Panda 7-DoF robot arm using the DROID [58] dataset. For each, we pre-train a context-smoothed policy by fine-tuning π_0 [52] on the respective dataset using VLM embeddings as context c and the context noising objective in Equation (5). Further implementation details are provided in Section G.

We evaluate TMRL against DSRL on three tasks, `sausage-in-pot`, `shrimp-in-white-drawer`, and `press-button`, with evaluation curves shown in Figure 10.

While the pre-trained policy is unable to solve the task successfully (often reaching for the wrong object), fine-tuning with TMRL improves success rates to near-perfect levels. This is in stark contrast to DSRL [2], which performs poorly because it cannot perform tasks beyond the coverage of the base policy. We omit a PostBC comparison here because the simulation performance in Figure 6 is similar to that of DSRL and it requires 2 additional steps: 1-step Gaussian ensemble policy pre-training and data labeling, then π_0 VLA re-training.

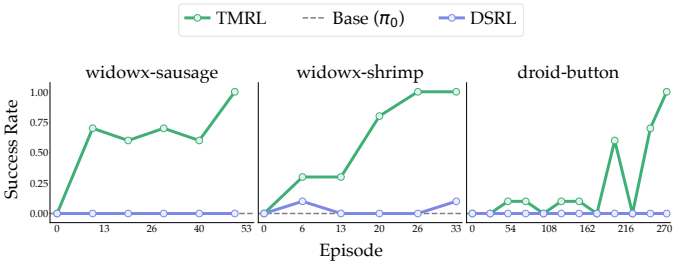

Fig. 10: TMRL enables steering of π_0 [52] across three real-world tasks, while DSRL [2] fails to learn any task.

E. *(Q4) Analysis and Ablations.*

CFG-RL Ablation: To understand the impact of the context-smoothing as compared to other means of extrapolating beyond the base policy, we compare TMRL with context-smoothed policies against those using classifier-free guidance (CFG) [40]. CFG also interpolates between the policy marginal and conditional by tuning an interpolation coefficient w on two GCPs: $\nabla_a \log p(a) + w\,(\nabla_a \log p(a \mid c) - \nabla_a \log p(a))$. We directly compare this ablation (TMRL-CFG) to timestep modulation in TMRL training the high-level steering policy to output the interpolation coefficient w rather than the timestep. As seen in Figure 11, TMRL-CFG fails because the conditioning, which still relies on $p(a \mid c)$, struggles to extrapolate to OOD contexts no matter the w. In contrast, TMRL brings OOD contexts back in-distribution via context corruption, providing coherent exploration in OOD settings as well.

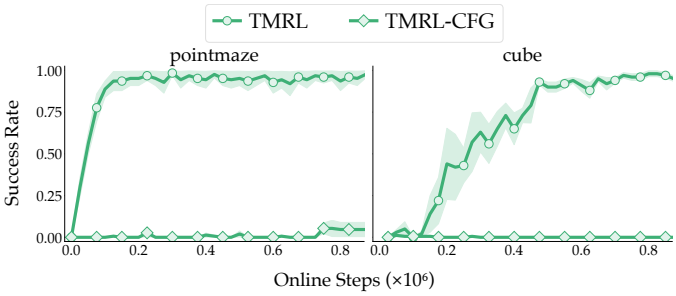

Fig. 11: Comparing adaptation with TMRL against TMRL-CFG for OGBench tasks. We find that our method substantially outperforms CFG-based interpolation.

Action Coverage Visualization: We then visualize the exploration behavior induced by CSP+TMRLvs. BC+DSRLin Fig 12. The exploration distribution of DSRL is relatively narrow, staying close to the coverage of the base policy. In contrast, the exploration behavior of TMRL is considerably broader, showing a diversity of strategies beyond the narrow base policy distribution. TMRL does not simply perform random actions, but rather aliases coherent actions from nearby states, for more informed exploration.

Fig. 12: Comparison of exploration behaviors on the task, `"pick up the butter and put it in the basket"` for RLPD (left), DSRL (center), and TMRL (right). While RLPD explores via random action noise and DSRL relies on action samples from the conditional distribution, TMRL leverages action sequences from smoothed contexts, resulting in broadened, yet coherent exploration.

Evolving Context-Smoothing Over Time: Finally, we demonstrate that TMRL learns to dynamically adjust the amount of context-smoothing (via diffusion timestep t_c in Figure 13). At the beginning of the trajectory, TMRL uses more diffusion noise so that the π_0 CSP policy can reach the sausage—by default π_0 is overfit to reaching the carrot almost every time. Near the end of the trajectory, TMRL uses less diffusion noise because once π_0 has picked up an object, it will generally attempt to correctly place it in the pot and therefore does not need as much diffusion noise to steer towards the correct behavior. We plot more examples in Appendix Fig. 18 and Fig. 19.

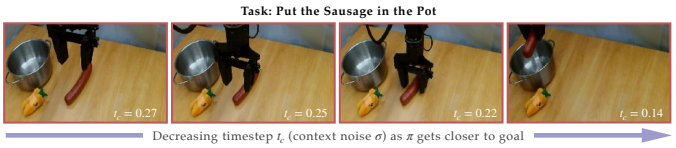

Fig. 13: A rollout with a converged TMRL policy on the WidowX where it learns to decrease the context-smoothing (t_c normalized $[0, 1]$) throughout the trajectory.

VII. Discussion and Limitations

While TMRL with context-smoothed policies can demonstrate significant performance benefits, there are many areas for improvement. Perhaps most important is safety: TMRL broadens the action distribution by aliasing actions across contexts, but this can lead to unsafe behavior. Mitigating these behaviors using a safety filter or a world model is necessary before TMRL can be used reliably. Secondly, while TMRL performs well in the real world, its sample efficiency remains impractical for many tasks of practical interest. We expect that further improvements to the steering algorithm and corruption kernel can increase TMRL's efficiency.

ACKNOWLEDGMENTS

We thank Chuning Zhu and Florian Shkurti for feedback on earlier drafts of this paper, and Dieter Fox for initial discussions during the project formulation.

This project was supported by funding from Amazon FAR through the Amazon Science Hub. Additionally, we thank the UW Hyak and Tillicum high-performance computing clusters for providing us with compute resources.

REFERENCES

[1] A. Wagenmaker, P. Dong, R. Tsao, C. Finn, and S. Levine, "Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning," *arXiv preprint arXiv:2512.16911*, 2025.

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

[3] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou, "$\pi^*_{0.6}$: A vla that learns from experience," *arXiv:2511.14759*, 2025.

[4] J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim, "Bootstrap your own skills: Learning to solve new tasks with large language model guidance," in *Conference on Robot Learning (CoRL)*, 2023.

[5] J. Zhang, K. Pertsch, J. Zhang, and J. J. Lim, "Sprint: Scalable policy pre-training via language instruction relabeling," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[6] J. Luo, C. Xu, J. Wu, and S. Levine, "Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning," *Science Robotics*, vol. 10, no. 105, p. eads5033, 2025.

[7] J. Zhang, M. Heo, Z. Liu, E. Biyik, J. J. Lim, Y. Liu, and R. Fakoor, "EXTRACT: Efficient policy learning by extracting transferrable robot skills from offline data," in *Conference on Robot Learning*, 2024.

[8] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani, "Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 3617–3624.

[9] P. Yin, T. Westenbroek, S. Bagaria, K. Huang, C.-A. Cheng, A. Kolobov, and A. Gupta, "Rapidly adapting policies to the real-world via simulation-guided fine-tuning," in *International Conference on Learning Representations (ICLR)*, 2025.

[10] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn, "Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[11] K. Lei, H. Li, D. Yu, Z. Wei, L. Guo, Z. Jiang, Z. Wang, S. Liang, and H. Xu, "Rl-100: Performant robotic manipulation with real-world reinforcement learning," *arXiv preprint arXiv: 2510.14830*, 2026.

[12] Y. Li, X. Ma, J. Xu, Y. Cui, Z. Cui, Z. Han, L. Huang, T. Kong, Y. Liu, H. Niu, W. Peng, J. Qiao, Z. Ren, H. Shi, Z. Su, J. Tian, Y. Xiao, S. Zhang, L. Zheng, H. Li, and Y. Wu, "Gr-rl: Going dexterous and precise for long-horizon robotic manipulation," *arXiv preprint arXiv:2512.01801*, 2025.

[13] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proceedings of (NeurIPS) Neural Information Processing Systems*, D. Touretzky, Ed. Morgan Kaufmann, December 1989, pp. 305 – 313.

[14] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15*, 1995.

[15] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," *Conference on Robot Learning (CoRL)*, 2021.

[16] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[17] J. M. Springer, S. Goyal, K. Wen, T. Kumar, X. Yue, S. Malladi, G. Neubig, and A. Raghunathan, "Overtrained language models are harder to fine-tune," in *Forty-second International Conference on Machine Learning*, 2025.

[18] F. Chen, A. Raventos, N. Cheng, S. Ganguli, and S. Druckmann, "Rethinking fine-tuning when scaling test-time compute: Limiting confidence improves mathematical reasoning," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[19] H. Zeng, K. Hui, H. Zhuang, Z. Qin, Z. Yue, H. Zamani, and D. Alon, "Can pre-training indicators reliably predict fine-tuning outcomes of llms?" *arXiv preprint arXiv:2504.12491*, 2025.

[20] F. Chen, A. Huang, N. Golowich, S. Malladi, A. Block, J. T. Ash, A. Krishnamurthy, and D. J. Foster, "The coverage principle: How pre-training enables post-training," in *The Fourteenth International Conference on Learning Representations*, 2026.

[21] K. Hu, Z. Rui, Y. He, Y. Liu, P. Hua, and H. Xu, "StemOB: Generalizable visual imitation learning with stem-like convergent observation through diffusion inversion,"

in *The Thirteenth International Conference on Learning Representations*, 2025.

[22] C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. Gupta, "Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets," in *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

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

[24] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1179–1191.

[25] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 23–29 Jul 2023, pp. 1577–1594.

[26] A. Ren, J. Lidard, L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, "Diffusion policy policy optimization," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 77 288–77 329.

[27] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conference on Robot Learning (CoRL)*, 2020.

[28] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning," in *International Conference on Learning Representations*, 2021.

[29] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "{OPAL}: Offline primitive discovery for accelerating offline reinforcement learning," in *International Conference on Learning Representations*, 2021.

[30] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Demonstration-guided reinforcement learning with learned skills," in *5th Conference on Robot Learning*, 2021.

[31] P. Dong, Q. Li, D. Sadigh, and C. Finn, "EXPO: Stable reinforcement learning with expressive policies," *arXiv preprint arXiv:2507.07986*, 2025.

[32] M. Tomar, R. Islam, M. E. Taylor, S. Levine, and P. Bachman, "Ignorance is bliss: Robust control via information gating," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[33] A. Goyal, R. Islam, D. Strouse, Z. Ahmed, M. Botvinick, H. Larochelle, Y. Bengio, and S. Levine, "Infobot: Transfer and exploration via the information bottleneck," *arXiv preprint arXiv:1901.10902*, 2023.

[34] K. Song, B. Chen, M. Simchowitz, Y. Du, R. Tedrake, and V. Sitzmann, "History-guided video diffusion," in *Forty-second International Conference on Machine Learning*, 2025.

[35] B. Chen, D. Martí Monsó, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann, "Diffusion forcing: Next-token prediction meets full-sequence diffusion," *Advances in Neural Information Processing Systems*, vol. 37, pp. 24 081–24 125, 2025.

[36] J. Zhang, M. Memmel, K. Kim, D. Fox, J. Thomason, F. Ramos, E. Bıyık, A. Gupta, and A. Li, "Peek: Guiding and minimal image representations for zero-shot generalization of robot manipulation policies," in *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[37] E. Zisselman, M. Mutti, S. Francis-Meretzki, E. Shafer, and A. Tamar, "Blindfolded experts generalize better: Insights from robotic manipulation and videogames," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[38] R. Mirjalili, T. Jülg, F. Walter, and W. Burgard, "Augmented reality for robots (arro): Pointing visuomotor policies towards visual robustness," *IEEE Robotics and Automation Letters*, 2026.

[39] P. Li, Y. Wu, Z. Xi, W. Li, Y. Huang, Z. Zhang, Y. Chen, J. Wang, S.-C. Zhu, T. Liu, and S. Huang, "Controlvla: Few-shot object-centric adaptation for pre-trained vision-language-action models," *CoRR*, vol. abs/2506.16211, June 2025.

[40] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[41] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 674–701, 2012.

[42] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4000–4007, 2022.

[43] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

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

[45] F. Zhang and M. Gienger, "Affordance-based robot manipulation with flow matching," *arXiv preprint arXiv:2409.01083*, 2025.

[46] C. Pan, G. Anantharaman, N.-C. Huang, C. Jin, D. Pfrommer, C. Yuan, F. Permenter, G. Qu, N. Boffi, G. Shi, and M. Simchowitz, "Much ado about noising: Dispelling the myths of generative robotic control," *arXiv preprint arXiv:2512.01809*, 2025.

[47] E. Su, T. Westenbroek, A. Nagabandi, and A. Gupta, "Rfs: Reinforcement learning with residual flow steering for dexterous manipulation," in *The Fourteenth International Conference on Learning Representations*, 2026.

[48] Y. Song and S. Ermon, "Generative modeling by esti-

mating gradients of the data distribution," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[49] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *arXiv preprint arxiv:2006.11239*, 2020.

[50] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

[51] T. Haarnoja, A. Zhou, P. Abbeel, and S. 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*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1861–1870.

[52] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[53] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *arXiv preprint arXiv:2306.03310*, 2023.

[54] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun, "Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization," *[arXiv preprint arXiv:2510.03827]*, 2025.

[55] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," *Robotics: Science and Systems (RSS)*, 2023.

[56] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Muñoz, X. Yao, R. Zurbrügg, N. Rudin, L. Wawrzyniak, M. Rakhsha, A. Denzler, E. Heiden, A. Borovicka, O. Ahmed, I. Akinola, A. Anwar, M. T. Carlson, J. Y. Feng, A. Garg, R. Gasoto, L. Gulich, Y. Guo, M. Gussert, A. Hansen, M. Kulkarni, C. Li, W. Liu, V. Makoviychuk, G. Malczyk, H. Mazhar, M. Moghani, A. Murali, M. Noseworthy, A. Poddubny, N. Ratliff, W. Rehberg, C. Schwarke, R. Singh, J. L. Smith, B. Tang, R. Thaker, M. Trepte, K. V. Wyk, F. Yu, A. Millane, V. Ramasamy, R. Steiner, S. Subramanian, C. Volk, C. Chen, N. Jawale, A. V. Kuruttukulam, M. A. Lin, A. Mandlekar, K. Patzwaldt, J. Welsh, H. Zhao, F. Anes, J.-F. Lafleche, N. Moënne-Loccoz, S. Park, R. Stepinski, D. V. Gelder, C. Amevor, J. Carius, J. Chang, A. H. Chen, P. de Heras Ciechomski, G. Daviet, M. Mohajerani, J. von Muralt, V. Reutskyy, M. Sauter, S. Schirm, E. L. Shi, P. Terdiman, K. Vilella, T. Widmer, G. Yeoman, T. Chen, S. Grizan, C. Li, L. Li, C. Smith, R. Wiltz, K. Alexis, Y. Chang, D. Chu, L. J. Fan, F. Farshidian, A. Handa, S. Huang, M. Hutter, Y. Narang, S. Pouya, S. Sheng, Y. Zhu, M. Macklin, A. Moravanszky, P. Reist, Y. Guo, D. Hoeller, and G. State, "Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning," *arXiv preprint arXiv:2511.04831*, 2025.

[57] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, A. Lee, K. Fang, C. Finn, and S. Levine, "Bridgedata v2: A dataset for robot learning at scale," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 1723–1736.

[58] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn, "Droid: A large-scale in-the-wild robot manipulation dataset," 2024.

[59] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.

[60] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

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

[62] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 297–307.

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

[64] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2024.

APPENDIX

A. Extended Proof of Thm 1

Theorem 1. We restate Theorem 1 for clarity: For any two contexts $c, c' \in \mathbb{R}^d$ (Equation (7)),

$$\mathrm{TV}\big(\pi_\sigma(\cdot \mid c), \pi_\sigma(\cdot \mid c')\big) \;\le\; \frac{\mathbb{E}\|w\|}{\sigma}\,\|c - c'\|.$$

Equivalently (Equation (8)),

$$\mathrm{Ov}\big(\pi_\sigma(\cdot \mid c), \pi_\sigma(\cdot \mid c')\big) \;\ge\; 1 - \frac{\mathbb{E}\|w\|}{\sigma}\,\|c - c'\|.$$

In particular, for fixed c, c' the lower bound in Equation (8) is nondecreasing in σ, and for fixed σ it is nonincreasing in $\|c - c'\|$.

Proof: For any measurable action event $A \subseteq \mathcal{A}$, define

$$f_A(c) := p(A \mid c) \in [0, 1]. \tag{12}$$

By definition of a context-smoothed conditional action distribution p_σ in Equation (6),

$$p_\sigma(A \mid c) = \mathbb{E}[f_A(c + \sigma w)], \quad w \sim \mathcal{N}(0, I_d). \tag{13}$$

Let $g_A(c) := p_\sigma(A \mid c)$. Then using Stein's identity (Gaussian smoothing gradient identity),

$$\nabla g_A(c) = \frac{1}{\sigma}\mathbb{E}[f_A(c + \sigma w)w]. \tag{14}$$

Thus,

$$\|\nabla g_A(c)\| \le \frac{1}{\sigma}\mathbb{E}[|f_A(c + \sigma w)|\,\|w\|] \le \frac{\mathbb{E}\|w\|}{\sigma}. \tag{15}$$

The first inequality follows by Jensen's inequality and the convexity of $\|\cdot\|$, and the second from $0 \le f_A \le 1$. Hence Equation (15) implies that g_A is $(\mathbb{E}\|w\|/\sigma)$-Lipschitz continuous. Thus, for two contexts c, c' and their difference $c - c'$:

$$|p_\sigma(A \mid c) - p_\sigma(A \mid c')| = |g_A(c) - g_A(c')| \le \frac{\mathbb{E}\|w\|}{\sigma}\|c - c'\|. \tag{16}$$

Taking $\sup_A$ gives Equation (7). Then Equation (8) follows simply from the overlap identity defined in Section IV: $\mathrm{Ov}(P, Q) := 1 - \mathrm{TV}(P, Q)$. ∎

B. Extended Proof of Corollary 1.1

At a high level, Corollary 1.1 states that for two contexts c, c', smoothing increases overlap even with respect to the conditional distribution of the original policy $p(\cdot \mid c), p(\cdot \mid c')$, extending the result from Theorem 1 that smoothing increases overlap with respect to the conditional distribution of the context-smoothed policy p_σ.

Proof: Assume the identifiability lower bound from Equation (9): $\mathrm{TV}\big(p(\cdot \mid c), p(\cdot \mid c')\big) \ge m\|\Delta\|$ for some $m > 0$, $\Delta = c - c'$ Assuming $\sigma \ge \mathbb{E}\|w\|/m$ as stated in the assumption Equation (10). Then, following Equation (7) from Theorem 1, we have:

$$\begin{aligned}\mathrm{TV}\big(p_\sigma(\cdot \mid c), p_\sigma(\cdot \mid c')\big) &\le \frac{\mathbb{E}\|w\|}{\sigma}\|\Delta\| \\ &\le m\|\Delta\| \\ &\le \mathrm{TV}\big(p(\cdot \mid c_1), p(\cdot \mid c_2)\big).\end{aligned}$$

Converting $\mathrm{Ov} = 1 - \mathrm{TV}$ yields Equation (11). ∎

TABLE I: Flow policy training hyperparameters

Hyperparameter	Value
Model	
Hidden size	128
Embedding dim	256
Activation	Mish
Number of layers	4
Number of heads	4
Flow	
Training flow steps	100
Inference flow steps	10
Timestep sampling	Uniform
Training	
Batch Size	256
Optimizer	AdamW
Learning rate	$3e^{-4}$
Weight decay	$1e^{-8}$
Betas	$[0.95, 0.999]$
Epsilon	$1e^{-8}$
LR schedule	constant

C. TMRL Implementation details

Training a context-smoothed imitation learning policy. Our approach is agnostic to the choice of policy class; however, in this work we focus on generative flow and diffusion policies, motivated by their recent success in behavior cloning for robotic learning. For training our own context-smoothed policies in OGBench (not widowx or dexterous tasks), our noise prediction network uses a Diffusion Transformer (DiT) backbone adapted from Zhu et al. [22]. While typical GCPs model $p(a \mid c)$, training a context-smoothed policy differs only in sampling a context timestep and corrupting the context $\tilde{c}$ (Equation (4)). The actions a_t are embedded and fed as a sequence into the transformer. The noisy conditionings, action timestep, the conditioning timestep and other observation conditionings that are not being noised (e.g., robot proprioception) are used to condition the transformer via Adaptive Layer Normalization (AdaLN) [59]. We train the model with Equation (5)

Our corruption kernel q, uses a discrete linear scheduler with `beta_start`$= 1e^{-4}$, `beta_end`$= 0.02$ with $T = 1000$ diffusion timesteps. During training, we sample a continuous $t \sim Unif(0, 1)$ which we note is sampled independently from our action noise timestep t_a (for clarity, we denote $t_c = t$ for the context timestep). We then sample noise, $\epsilon \sim \mathcal{N}(0, I)$, convert the timestep to a discrete index, and then sample the corrupted context $c_t = \sqrt{\bar{\alpha}_t}c + \sqrt{1 - \bar{\alpha}_t}\epsilon$. The policy is trained to model the action distribution $p(a \mid c_t, t)$ at different levels of noise t conditioned on the noisy context c_t. In our experiments,

Algorithm 3 Timestep-Modulated RL (TMRL) with Context-Smoothed Pretraining.

Pre-train context-smoothed policy: Train $p_\sigma(a^{1:H} \mid c, z, \sigma)$ given context c, randomly sampled noise z (e.g., $N(0, I)$ in diffusion policies), and smoothing parameter σ (= diffusion timestep t_c) on offline data $\mathcal{D}_{\text{off}}$ with Equation (5):

$$\min_{\theta} \; \mathbb{E}_{(c,a)\sim\mathcal{D}_{\text{off}}} \; \mathbb{E}_{\substack{\sigma\sim\mathcal{S} \\ \tilde{c}\sim q_\sigma(\cdot|c)}} \Big[\ell\big(\theta;\; a, \tilde{c}, \sigma\big)\Big]$$

Initialize: replay buffer $\mathcal{B}$, smoothing-aware noise critic $Q(s, z, \sigma)$, high-level actor $\pi_{\text{HL}}(z, \sigma \mid s)$, corruption kernel q_σ

while interacting with the environment (or replaying from $\mathcal{B}$) **do**

Update Q: ▷ Update Smoothing-Aware Critic

$$\min_{Q} \mathbb{E}_{(s_t, z, r_{t:t+H}, s_{t+H}, \sigma_t)\sim\mathcal{B},\; (\sigma', z')\sim\pi_{\text{HL}}} \left[Q(s, z, \sigma) - \left(\sum_{i=t}^{t+H} \gamma^i r_t + Q(s_{t+H}, \sigma', z') \right)^2 \right]$$

Update π_{HL}: ▷ Update High-Level (Timestep-Modulating) Actor with Off-Policy RL Objective [51]

$$\max_{z,\sigma\sim\pi_{\text{HL}}} \mathbb{E}_s[Q(s, z, \sigma) - \alpha \log \pi_{\text{HL}}(\cdot \mid s)]$$

if online environment is available **then**

Sample $(z_t, \sigma_t) \sim \pi_{\text{HL}}(s_t)$

Corrupt context $\tilde{c}_t \leftarrow q_\sigma(\cdot \mid c_t)$ (Eq. (4))

Sample and execute action chunk $a^{1:H} \sim p_\theta(\cdot \mid \tilde{c}_t, z_t, \sigma_t)$

Observe $r_{t:t+H}, s_{t+H}$, and add $(s_t, z_t, r_{t:t+H}, s_{t+H}, \sigma_t)$ to $\mathcal{B}$

end if

end while

we show that context-smoothed policy training can be done from scratch or through fine-tuning on a base VLA model.

Timestep-Modulated Reinforcement Learning. Our experiments use SAC [51] or RLPD [25] as the online RL algorithm. Building on top of DSRL, we run RL over a high-level actor whose outputs are fed into the low-level GCP. Our high level actor π_{HL} outputs both the initial noise vector z as well as the noise timestep t_c. We model the actor as two independent distributions, π_{HL}^{z} and $\pi_{\text{HL}}^{t_c}$, which allows us to use different noise bounds as well as target entropies for both distributions. Following DSRL, we also train separate timestep-aware noise critics with t_c as an additional input $Q(s, a, z, t_c)$. In our experiments, we experiment with and without the timestep-aware noise critic and find no major performance differences. Our `OGBench` experiments utilize both critics while the rest of our experiments, use noise critics $Q(s, a, z, t_c)$.

TABLE II: Common TMRL hyperparameters

Hyperparameter	Value
Critic LR	$3e^{-4}$
Actor LR	$3e^{-4}$
Autotune	True
Tau	0.005
Critics	5
Actor and critic layers	3
Buffer warmup steps	500

D. Baseline Implementation Details

RLPD. Following Ball et al. [25], which we also apply to all other methods: each batch consists of 50% offline data and 50% samples from the online replay buffer; critics use layer normalization, we subsample two critics, and remove entropy backups.

SPiRL. While SPiRL [27] initially proposes to utilize an LSTM [60] as the architecture choice for the skill encoder and decoder, we adopt a more modern transformer architecture. After pre-training the skill prior, we initialize the learnable actor with the skill prior. Following Pertsch et al. [27], we minimize the reverse KL $D_{KL}(q||p)$ between the skill prior and the actor. Using the reverse KL rather than the forward KL ensures that the learned prior is mode-covering.

PostBC. Following Wagenmaker et al. [1], we first train an ensemble of MLP predictors on bootstrapped resamples of the demonstration dataset to approximate the posterior covariance of the demonstrator's actions. Each ensemble member is trained via regression on a bootstrap resample (sampled with replacement) of the dataset. The per-state posterior covariance is estimated as the empirical variance across ensemble predictions. During BC pretraining, we perturb each action target in each training batch by noise sampled from a zero-mean Gaussian with this estimated posterior covariance, scaled by a weight α. The final policy is parameterized as a diffusion model trained on these noise-perturbed action targets. At inference time, the policy is used without modification.

E. TMRL vs. CFG-RL

At their core, TMRL and CFG [40] represent independent ways to sample from distributions that control the reliance on a context c. CFG jointly trains a conditional $p(a \mid c)$ and unconditional model $p(a)$ (typically sharing parameters), and

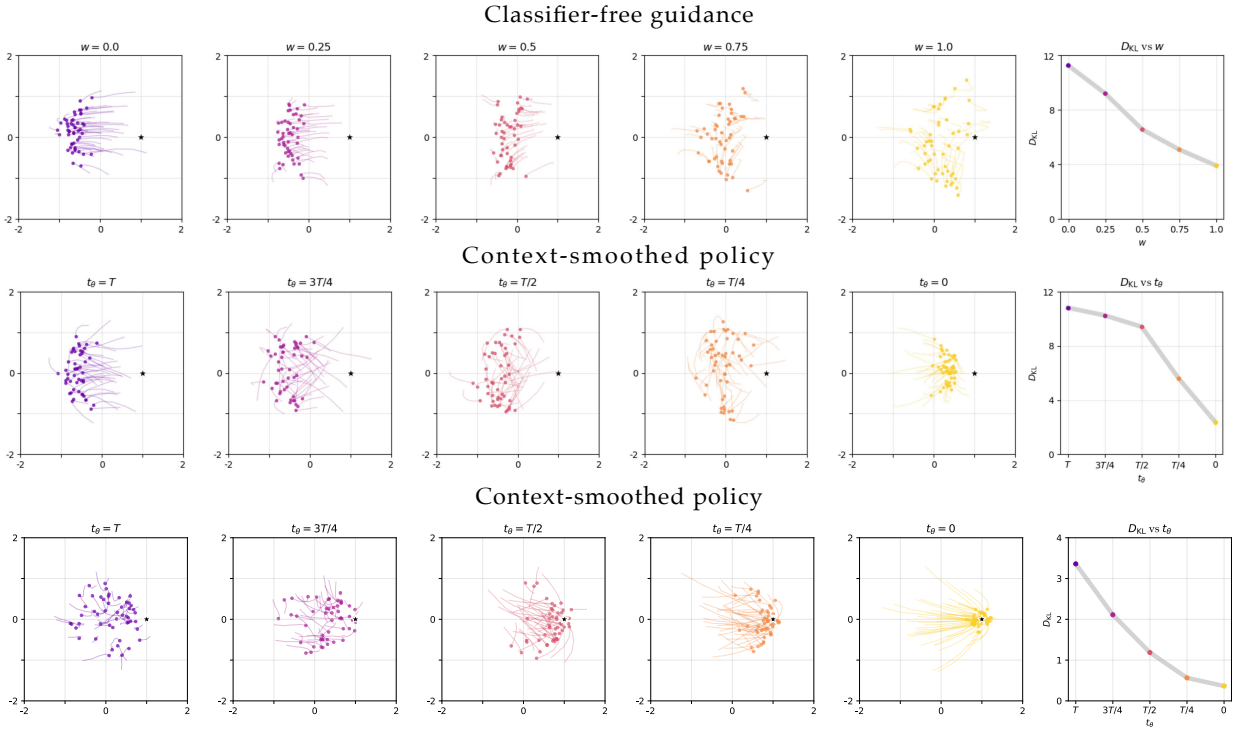

Fig. 14: **Conditioned smoothed policies enable out-of-support generalization.** (Top) In a 2D unit-circle setup (Figure 4), we restrict training to angles $\theta \in [\pi/2, 3\pi/2]$ and evaluate on an unseen angle $\theta = 0$. Unlike standard conditional diffusion, context-smoothed training (Equation (5)) successfully approximates the target Gaussian at $(\cos 0, \sin 0)$. **Noisy Conditioning Enables Marginal-to-Conditional Interpolation.** (Bottom) Angles on the unit circle define the conditioning variable θ, with corresponding 2D points $(x, y) = (\cos\theta, \sin\theta)$. The marginal distribution $p(x, y)$ consists of noisy points around the circle, while the conditional distribution $p(x, y \mid \theta)$ is a Gaussian at $N([\cos\theta, \sin\theta], I)$. By varying the conditioning noise level t_θ for $\theta = 0$, we visualize denoised samples: larger t_θ yields weaker conditioning (approaching the marginal $p(x, y)$), and smaller t_θ yields stronger conditioning (approaching $p(x, y \mid \theta)$). The rightmost plots report the KL divergence between the induced distribution at each t_θ and the target conditional $p(x, y \mid \theta = 0)$, highlighting our method's improved OOD generalization.

at inference time interpolates between them using classifier-free guidance. Concretely, the guided score is formed as $\nabla_a \log \pi(a|s) + w(\nabla_a \log \pi(a|s, o) - \nabla_a \log \pi(a|s))$. In Figure 14, we visualize a toy example comparing sampled points of an unseen conditioning θ. We observe that CFG approximates the true distribution worse than a context-smoothed policy (here, our noisy-conditioning approach). We attribute this to CFG not being trained on this out of distribution conditioning. CFG samples have higher variance and are more spread out at $w = 1.0$. Whilst, during context-smoothed policy training, the novel θ has likely been seen during training, thus the samples generates samples that more closely approximate the true distribution.

F. Details of Simulation Experiments

OGBench. For our first set of simulation experiments in OGBench [44], we utilize a flow-matching policy [61]. We use RLPD [25] as our RL algorithm, which can leverage prior data alongside online interaction data. The same offline data used to pre-train the diffusion policy is used during online learning where we employ 50/50 sampling (i.e., 50% of the data comes from the offline data and 50% of the data comes from the online replay buffer). Additionally, for all methods and experiments, we utilize 5 critics, layer norms, and remove the entropy term from the critic loss. For both evaluation environments, the agent is required to learn multiple tasks simultaneously: four for PointMaze and three for Cube. Both PointMaze and Cube employ sparse rewards: the agent receives a reward of 0 upon successful episode termination and –1 otherwise.

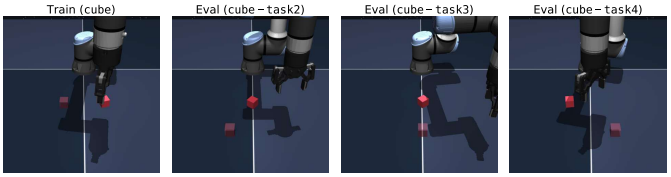

Fig. 15: **Cube environments.** We filter transitions from the `cube-single-play` dataset where the cube or the cube goal x-position is > 0.4. We evaluate on three tasks where the goal location is located beyond the filtered region.

We use the `pointmaze-large-navigate-v0` dataset for training, where each observation consists of the agent's

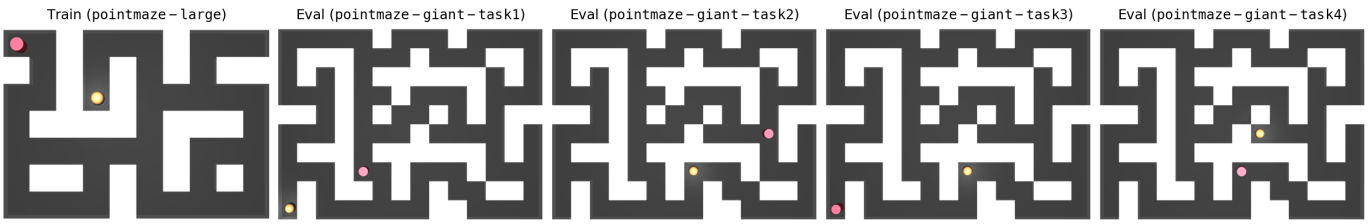

Fig. 16: **PointMaze environments.** Example train and evaluation mazes. The evaluation environment features a significantly larger maze layout. We train RL policies on four out-of-distribution goal locations simultaneously.

TABLE III: OGBench hyperparameters

Hyperparameter	Value
Action len	50 (`pointmaze`), 50 (`cube`)
Discount	0.995
Noise bound	1.0
Hidden dim	512
Action target entropy	`-action_dim`
Timestep target entropy	-1
Timestep bound	1.0
Num Envs	4

2D position concatenated with the 2D goal. This combined vector serves as the conditioning input that is subjected to noise. Evaluation is performed on the larger `pointmaze-giant-navigate` environment, as shown in Figure 16. We use the `cube-single-play` dataset, filtering out transitions where the cube's position exceeds a predefined threshold, as visualized in Figure 15. The perturbed conditioning variable is the cube's three-dimensional goal position. Evaluation is conducted on three tasks with goal locations situated in the out-of-distribution region.

Libero. For our Libero [53] experiments, we utilize the π_0-Libero checkpoint provided by Black et al. [52] at `s3://openpi-assets/checkpoints/pi0_libero` which trains on four Libero datasets: `Libero-Spatial`, `Libero-Object`, `Libero-Goal` and `Libero-10`. For TMRL, we also fine-tune π_0-base using context-smoothed policy training on the same datasets. The other baselines also use the same dataset. We evaluate on two novel tasks: task 51 from the `Libero-90` task suite and the swap perturbation for `Libero-Goal` from Zhou et al. [54].

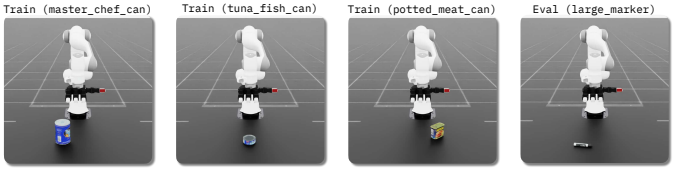

Fig. 17: **Dexterous manipulation grasping objects.** A policy is trained on three objects `master_chef_can` , `tuna_fish_can` and `potted_meat_can` and then evaluated on a novel `large_marker` object.

Dexterous grasping. Our dexterous grasping experiments are run in simulation built on IsaacLab [56], we distill an RL expert into point cloud conditioned visuomotor policies via a standard student-teacher approach [62] and train on three can-shaped objects—`master_chef_can`, `tuna_fish_can`, `potted_meat_can`—collecting 200 demonstrations for each and evaluate adaptation to the unseen `large_marker` object shown in Figure 17. We first train a diffusion policy [16] using point clouds as the noised context, we also condition on the 22-dimensional joint pose consisting of 6-dimensional arm pose and the 16-dimensional finger joint pose. We use a relative joint pose action space. After training a context-smoothed policy, we use PPO [63] as our algorithm and run TMRL over the frozen base policy.

TABLE IV: Dexterous Grasping PPO hyperparameters

Hyperparameter	Value
Number of envs	128
Batch Size	4096
Discount	0.99
GAE lambda	0.95
LR	$3e^{-4}$
Clip range	0.2
Value loss coefficient	$1e^{-4}$
Max gradient norm	1.0
Optimizer	Adam

G. Details of Real Experiments

WidowX. For our real-world WidowX experiments, we utilize the π_0 VLA model provided by Black et al. [52] at `s3://openpi-assets/checkpoints/pi0_base`. We fine-tune the model on the Bridge V2 [57] dataset which is used for DSRL and fine-tune a separate context-smoothed policy for TMRL. We perform full fine-tuning of π_0-Base with 2 Nvidia H200 GPUs, training for 20 hours. We list further training details in Table VII.

We evaluate our fine-tuned policies on two unseen tasks: `widowx-sausage` and `widowx-shrimp`. Both use maximum episode lengths of 100 steps and use sparse rewards (-1/0) rewarding the agent only when the task is completed. The `widowx-sausage` task consists of a sausage, a pot, and a distractor carrot object. The objective is to pick and place the sausage inside of the pot. We randomize all object positions within a 10 cm radius. The `widowx-shrimp` task consists of three drawers (white, red, green), a distractor sushi object on a green plate, and a shrimp object on a white plate. The objective is to pick and place the shrimp inside of the white drawer. We randomize the sushi and shrimp object positions

within a 10 cm radius as well as randomizing orientation of both objects.

DROID. For our real-world DROID experiments, we utilize the π_0 VLA model provided by Black et al. [52] at `s3://openpi-assets/checkpoints/pi0_base`. We fine-tune a context-smoothed policy on the DROID [58] dataset and utilize the the π_0 – DROID VLA model provided by Khazatsky et al. [58] at `s3://openpi-assets/checkpoints/pi0_droid` which is used for DSRL and fine-tune a separate context-smoothed policy for TMRL. We perform full fine-tuning of π_0-Base with 4 Nvidia H200 GPUs for 100k steps, which takes roughly 3 days. We list further training details in Table VII.

We follow standard practices [2, 25], such as high UTDs, layer normalizations in the critic networks, deep critic networks, and the use of multiple critics, we list additional hyperparameters in Table V. For our real-world tasks, we generally use lower action noise bounds for TMRL than DSRL as we observe that using higher action bounds for TMRL tends to generate out-of-distribution and unsafe action sequences. A possible explanation for this is that noisy contexts are more likely to be out-of-distribution for the action expert. In addition, naively sampling action distributions close to the marginal $p(a)$ results in unsafe action sequences. To mitigate this issue, we simply upper bound the maximum timestep that can be sampled by the high-level policy π_{HL} and observe that this mitigates such undesirable behaviors.

TABLE V: Finetuning π_0 Libero and Bridge hyperparameters

Hyperparameter	Value
Training	
Batch Size	32
Optimizer	cosine
EMA Decay	0.99
Decay LR	$2.5e^{-6}$
Peak LR	$5e^{-4}$
Warmup steps	$1e^{3}$
Train steps	$3e^{4}$
Flow	
Inference flow steps	10

DSRL [2] originally uses VLM embeddings from the PaliGemma VLM model as inputs to both the actor and critic. However in our experiments, we find that replacing these with image embeddings encoded by DINOv2 [64] performs comparably with far less computational overhead. We hypothesize that the advantage of VLM embeddings becomes more pronounced in multi-task settings with diverse language prompts, where language-conditioned representations can better capture task-specific structure.

TABLE VI: Hyperparameters for TMRL π_0 experiments

Hyperparameter	Libero	WidowX	DROID
Discount	0.99	0.99	0.99
Action execution len	5	5	5
Noise bound	1.0	0.25	1.0
Prefix noise bound	1.0	1.0	1.0
Hidden dim	1024	1024	512
Action target entropy	-16	-16	-16
Timestep target entropy	-0.5	-0.5	-0.5
Timestep bound	0.6	0.6	0.1
UTD	1	20	20
Num Envs	4	1	1

TABLE VII: Hyperparameters for DSRL π_0 experiments

Hyperparameter	Libero	WidowX	DROID
Discount	0.99	0.99	0.99
Action execution len	5	5	5
Noise bound	1.0	1.0	1.0
Hidden dim	1024	1024	512
Action target entropy	-16	-16	-16
UTD	1	20	20
Num Envs	4	1	1

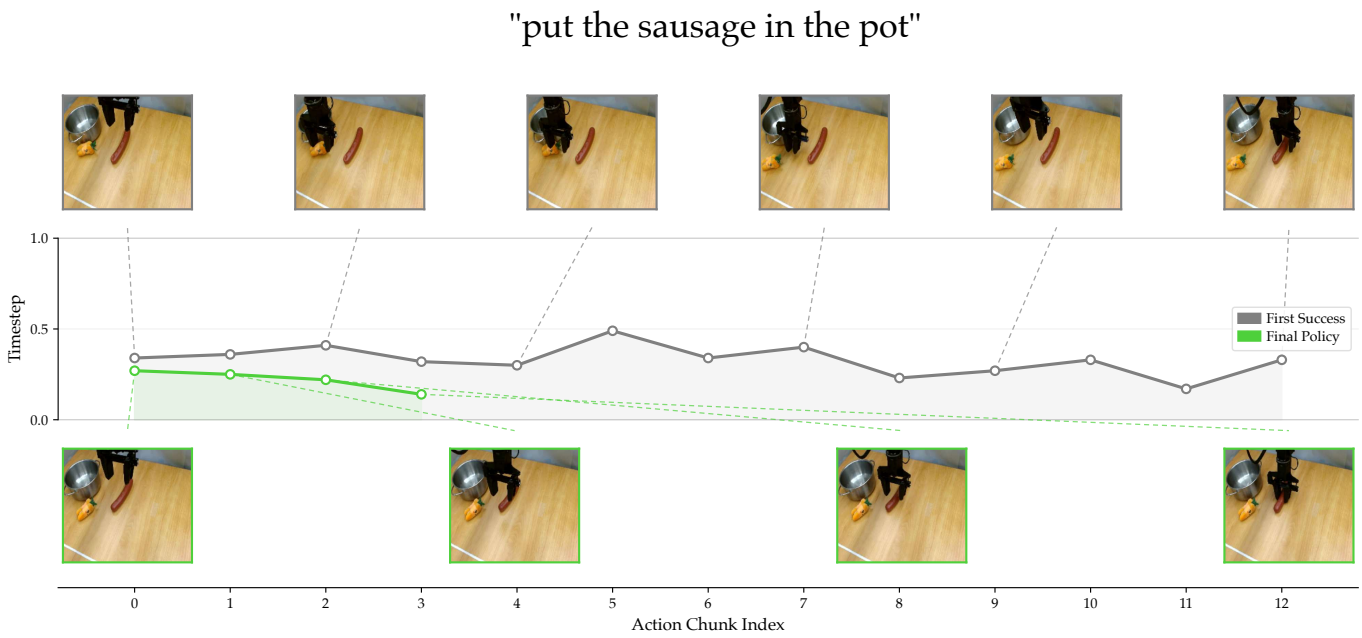

Fig. 18: **Timestep modulation over the course of a rollout.** We visualize the timesteps used by TMRL across action chunk indices for the task "put the sausage in the pot," comparing the first successful rollout to the converged final policy. The final policy uses lower timesteps toward the end of the trajectory, reflecting reliance on increasingly precise, imitation-like action distributions as the task nears completion.

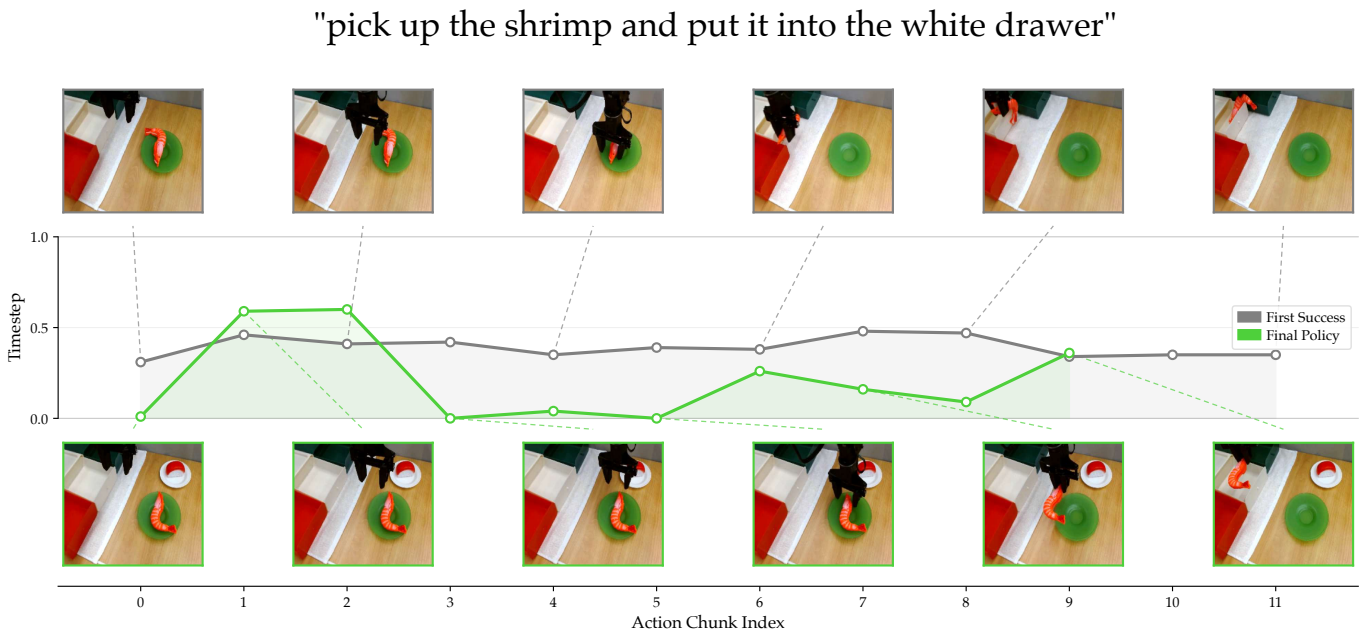

Fig. 19: **Timestep modulation over the course of a rollout.** We visualize the timesteps used by TMRL across action chunk indices for the task "pick up the shrimp and put it into the white drawer," comparing the first successful rollout to the converged final policy. The final policy uses notably lower timesteps during the manipulation-critical middle portion of the trajectory, reflecting more precise, imitation-like action distributions during the pick-and-place motion.

Fig. 20: **`Libero-Object` action distributions under context smoothing.** Each row corresponds to a distinct task in the `Libero-Object` suite. Columns show rollouts as the observation embedding is progressively smoothed from the original context (left) toward the marginal distribution (right), interpolating between the conditional and marginal policy.

Fig. 21: **`Libero-Spatial` action distributions under context smoothing.** Each row corresponds to a distinct task in the `Libero-Spatial` suite. Columns show rollouts as the observation embedding is progressively smoothed from the original context (left) toward the marginal distribution (right), interpolating between the conditional and marginal policy.

Fig. 22: **`Libero-Goal` action distributions under context smoothing.** Each row corresponds to a distinct task in the `Libero-Goal` suite. Columns show rollouts as the observation embedding is progressively smoothed from the original context (left) toward the marginal distribution (right), interpolating between the conditional and marginal policy.