DRIFTOPD: SEQUENCE-LEVEL REVERSE-KL DISTILLATION FOR ONE-STEP VLA POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models increasingly rely on action experts that generate short action chunks under receding-horizon control. While chunk-level training is convenient across robot embodiments, it optimizes local action likelihood without explicitly accounting for long-horizon task success. Sequence-level reinforcement learning can address this limitation, but typically requires policy rollouts and closed-loop interaction, which are costly for real-robot manipulation. We introduce **DriftOPD**, a teacher-free, rollout-free framework for sequence-level on-policy distillation of continuous VLA action experts. We show that the sequence-level reverse Kullback–Leibler (KL) divergence decomposes into a chunk-level reverse-KL term and a future-potential term that captures the long-horizon effect of the current action. DriftOPD optimizes these two terms using a one-step drifting objective and a Q-function critic learned from offline demonstrations, respectively, enabling sequence-level optimization with only offline data and one-step action generation. Across multiple VLA architectures in simulation and real-world manipulation, DriftOPD generally outperforms existing one-step distillation baselines while achieving task success performance comparable to multi-step teacher policies. These results demonstrate that long-horizon behavior can be effectively distilled into one-step VLA action experts without online interaction or a separate teacher. Code and the project page are provided in the supplementary material as `code` and `project`, with demo videos.

1 INTRODUCTION

Vision-Language-Action (VLA) models have recently emerged as a promising paradigm for general-purpose robotic manipulation (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2025a). As a robot manipulation system, a VLA leverages the reasoning and planning knowledge of Vision-Language Models (VLMs), corresponding to System 2, together with the intuitive behavior generation of a flow-based action expert, corresponding to System 1 (Black et al., 2025b; Shi et al., 2025). This VLA architecture is designed to reflect the state (e.g., environment and proprioception) that change according to actions, and VLAs perform tasks through closed-loop execution (Brohan et al., 2023; Bjorck et al., 2025). Since replanning every low-level action at each control step is computationally inefficient, VLAs commonly adopt a receding-horizon control scheme, repeatedly predicting short action chunks from updated observations (Zhao et al., 2023; Kim et al., 2025).

Action chunks provide a convenient common training unit for action experts across multiple embodiments (Black et al., 2025a; Bjorck et al., 2025). However, under this chunk-level training paradigm, VLAs suffer from two problems (Figure 1a). First, because VLAs are trained on demonstration states at train time, these VLAs suffer from a *test-time distribution mismatch* and fail to generalize to model-generated states (Ross et al., 2011; Mehta et al., 2025). Second, within an action sequence composed of action chunks, the policy naïvely generates only locally likely action chunks at each step without considering the *final task success* (Zhang et al., 2025). On the other hand, while recent reinforcement learning methods improve VLA training using various forms of sequence-level feedback (Lu et al., 2025; Li et al., 2026a; 2025), these approaches heavily rely on *policy-generated action sequences* to obtain such supervision. They must execute the current policy, evaluate the resulting sequences, and often recollect new rollouts as the policy changes. This dependence on closed-loop interaction is particularly costly for real-robot manipulation (Zhu et al., 2026).

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.

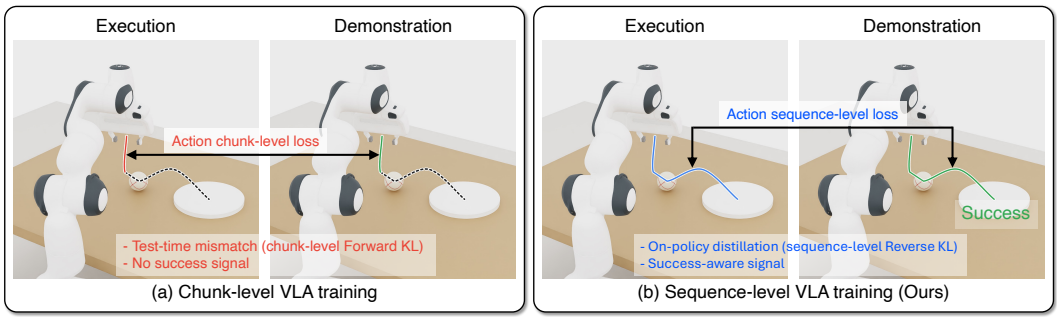


Figure 1: Chunk-level VLA training versus our sequence-level formulation

To this end, we explore a scalable and simple sequence-level objective function for action-expert training (Figure 1b). Currently, due to the multi-step sampling of diffusion/flow action experts, research on the trajectory-level reverse Kullback-Leibler divergence (KLD) for continuous generative policies has remained at the level of *chunk*-level ODE trajectories (Li et al., 2026b; Fang et al., 2026). In this work, we formulate *sequence*-level on-policy distillation (OPD) through the reverse KLD over the action sequence leading to success. Our formulated sequence-level OPD consists of a *chunk-level reverse KLD term* and a *future-potential term* for the current action chunk, similar to MiniLLM (Gu et al., 2024). However, unlike MiniLLM, the clean logits of continuous action chunks required for the future-potential term remain intractable. Therefore, the sequence-level OPD formulation itself still relies on rollout data, is bounded by the performance of the teacher model, and incurs multi-step sampling latency.

To address these challenges, we propose **DriftOPD**, a teacher-free sequence-level OPD method for one-step distillation. DriftOPD calculates the chunk-level reverse KLD term as a one-step distribution-matching drifting objective (Deng et al., 2026), while estimating the future-potential term with a critic-based neural network. In this way, DriftOPD achieves one-step inference while learning directly from human demonstrations, without a separate teacher model. In practice, we adapt VGAS (Xu et al., 2026), which provides a demonstration-trained Q-function critic, to achieve fully rollout-free sequence-level OPD. Consequently, because DriftOPD uses only demonstration data, it can be readily applied across multiple embodiments, while its generated-sample-based sequence-level OPD avoids the test-time mismatch. Moreover, one-step policy generation through the drifting objective reduces robot manipulation latency.

We apply DriftOPD to a diverse set of VLA models spanning different architectures, fine-tuning strategies, and action-expert parameterizations (Black et al., 2025a; Bjorck et al., 2025; Shukor et al., 2025; Yang et al., 2026a). We distill their multi-step sampling capability into one-step policy generation at the action-sequence level. Our experiments cover simulation and real-world execution across a wide range of robot manipulation tasks (Nasiriany et al., 2026; Chen et al., 2026a; Liu et al., 2023a; Fei et al., 2025). Experiments show that DriftOPD outperforms existing one-step distillation methods for the VLA action expert, achieving task success rates comparable to those of multi-step teachers. Further analysis shows that DriftOPD provides more stable training in data-scarce VLA settings, remains effective across diverse VLA configurations (e.g., architecture, fine-tuning strategy, and parameterization), and enables faster execution through one-step policy generation, with negligible loss of action diversity.

Our main contributions are summarized as follows:

- We formulate **sequence-level on-policy-distillation for VLAs** by minimizing the reverse KLD over action-sequence distributions, decomposing it into a local action-chunk reverse-KL term and a future-potential term.
- We propose **DriftOPD** to make sequence-level OPD tractable for continuous VLA action experts. It realizes the local reverse KLD term with a one-step drifting objective and approximates the future-potential term with a demonstration-trained Q-function critic, enabling teacher-free and rollout-free one-step distillation.
- Across diverse VLA configurations in simulation and real-world manipulation, DriftOPD generally outperforms existing one-step distillation baselines while achieving task success comparable to multi-step policies. It further provides stable training in data-scarce VLA settings and faster execution than multi-step policy generation.

2 Related Work

2.1 Vision-Language-Action (VLA) Models

Vision-Language-Action (VLA) models have emerged as general-purpose robot policies that couple vision-language representations with imitation learning from large-scale robot demonstrations (Zitkovich et al., 2023; Black et al., 2025b; Bjorck et al., 2025). Early models such as RT-1 (Brohan et al., 2023) and RT-2 (Zitkovich et al., 2023) established scalable language-conditioned robot policies. Recent models such as π_0 series (Black et al., 2025a;b) use continuous action experts for diverse robot tasks. In parallel, ACT (Zhao et al., 2023) and Diffusion Policy (Chi et al., 2023) demonstrated the effectiveness of predicting short action sequences (i.e., chunk) rather than individual actions. Action chunking has since become a common training and execution unit in continuous-control VLAs (Black et al., 2025b; Kim et al., 2025; 2026), and is adopted by recent models such as π_0 series (Black et al., 2025a;b), RLDX (Kim et al., 2026), ABot (Yang et al., 2026a), and GR00T series (Bjorck et al., 2025). However, behavior-cloned policies remain vulnerable to test-time distribution shift, where the pretrained model fail to generalize to model- generated state at the test time (Ross et al., 2011; Mehta et al., 2025). Moreover, chunk-level imitation objectives provide no direct supervision for final task success (Zhang et al., 2025; Zhu et al., 2026). Recent reinforcement learning methods address these limitations through sequence-level feedback (Lu et al., 2025; Li et al., 2026a; 2025). However, obtaining such feedback from physical rollouts requires costly closed-loop interaction on real robots (Zhu et al., 2026).

2.2 One-Step Distillation of Diffusion/Flow Models

Diffusion and flow-based models (Song & Ermon, 2019; Song et al., 2021b; Liu et al., 2023b) have been widely adopted as action experts in VLA models (Black et al., 2025a; Bjorck et al., 2025), but their iterative sampling incurs substantial inference latency (Song et al., 2021a; Lu et al., 2022). To reduce the number of sampling steps, prior work has explored progressive and consistency distillation (Salimans & Ho, 2022; Song et al., 2023; Lu & Song, 2025), distribution matching (Yin et al., 2024), and more recently native one-step formulations such as consistency objective (Song et al., 2023; Song & Dhariwal, 2024; Lu & Song, 2025), MeanFlow (Geng et al., 2025; 2026), and distribution drifting (Deng et al., 2026). Similar ideas have been introduced for robotic policies, including consistency- and distribution-matching-based policy distillation (Prasad et al., 2024; Jia et al., 2024). For VLA models, Li et al. (2026c) employ self-distillation for one-step action generation, while Chen et al. (2026b) and Gao et al. (2026) directly introduce native one-step generation objectives into VLA action experts. Despite their efficiency, existing approaches mainly optimize action-chunk-level ODE-trajectory or distribution matching, without directly distilling supervision from task success, which can leave a gap between chunk-level training objectives and closed-loop test-time behavior.

2.3 On-Policy Distillation

Prevalent action-chunking VLAs can be viewed as autoregressive (AR) sequence models at the chunk level, where each action chunk is conditioned on states induced by previously executed chunks. This perspective naturally connects VLAs to knowledge distillation in large language models (LLMs), where recent work minimizes the divergence between teacher and student distributions (Gu et al., 2024; Agarwal et al., 2024). In particular, a common formulation of on-policy distillation (OPD) minimizes the sequence-level reverse Kullback-Leibler divergence (KLD), thereby evaluating the student and teacher distributions at student-visited states (Gu et al., 2024; Zhao et al., 2026b). This reverse KLD formulation alleviates the train-test distribution mismatch of supervised fine-tuning by training on student-generated sequences (Gu et al., 2024; Agarwal et al., 2024). Such on-policy supervision has been particularly effective for long-horizon generation and reasoning tasks, including mathematical reasoning (Zhao et al., 2026b;c).

Recent works extend this principle to continuous generative models, where DiffusionOPD (Li et al., 2026b) and FlowOPD (Fang et al., 2026) distill teacher knowledge along student-generated ODE *trajectories* from noise to data. However, these ODE trajectory-level OPD still distills knowledge only within each action chunk. It does not explicitly capture the *sequence* of executed chunks that determines final task success. Moreover, flow-based action experts do not directly provide clean-action

likelihoods $\pi_\theta(\cdot \mid \boldsymbol{s}_t)$ (Lipman et al., 2023), making sequence-level OPD over action chunks difficult to apply directly. VLA-OPD (Zhong et al., 2026) instead derives dense teacher-based distillation rewards on student rollouts, but requires closed-loop environment interaction and repeated teacher queries. This work therefore aims to overcome these challenges, thereby enabling task-success signals to be incorporated across action-chunk sequences.

3 Preliminary

3.1 On-Policy Distillation

On-policy distillation (OPD) minimizes the reverse Kullback-Leibler divergence (KLD) between a model distribution q_θ and a target distribution p over model-generated sequences (Gu et al., 2024):

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{c\sim\mu,\, \boldsymbol{x}\sim q_\theta(\cdot|c)} \left[\log \frac{q_\theta(\boldsymbol{x} \mid c)}{p(\boldsymbol{x} \mid c)} \right], \tag{1}$$

where c is the conditioning context and $\boldsymbol{x} = (x_1, \ldots, x_T)$ is an autoregressively generated sequence. A useful property of sequence-level reverse-KL optimization is that its gradient separates the discrepancy at the current prediction from the effect of that prediction on subsequent sequence-level discrepancy (Gu et al., 2024). This decomposition is expressed through step-wise log-density ratios,

$$r_t = \log \frac{p(z_t \mid c, \boldsymbol{z}_{<t})}{q_\theta(z_t \mid c, \boldsymbol{z}_{<t})}, \qquad R_{t+1} = \sum_{k=t+1}^{T} r_k, \tag{2}$$

where r_t represents the current-step discrepancy and R_{t+1} accumulates the discrepancy over subsequent steps. For discrete autoregressive models such as LLMs, these terms are tractable because the step-wise conditional distributions are explicitly available from model logits. In contrast, continuous generative policies generally do not provide tractable clean-sample densities, making the same formulation difficult to apply directly.

3.2 Drifting Models

Drifting models (Deng et al., 2026) are updated by a sample-space vector field induced by the discrepancy between a target distribution and a model distribution. Given a condition $\boldsymbol{s}$, let $p(a_t \mid \boldsymbol{s})$ and $q_\theta(a_t \mid \boldsymbol{s})$ denote the target and model distributions over data samples a_t, respectively. Then, their reverse KLD is

$$D_{\mathrm{KL}}\left[q_\theta(a_t \mid \boldsymbol{s}) \,\|\, p(a_t \mid \boldsymbol{s})\right] = \mathbb{E}_{a_t\sim q_\theta} \left[\log \frac{q_\theta(a_t \mid \boldsymbol{s})}{p(a_t \mid \boldsymbol{s})} \right]. \tag{3}$$

The corresponding local descent direction in the data space is given by the difference between their score functions:

$$\boldsymbol{V}^*(a_t, \boldsymbol{s}) = \nabla_{a_t} \log p(a_t \mid \boldsymbol{s}) - \nabla_{a_t} \log q_\theta(a_t \mid \boldsymbol{s}). \tag{4}$$

When the densities are not directly available, they can be approximated from samples using kernel density estimation (KDE) (Cheng, 1995; Lee et al., 2026),

$$p_h(a_t \mid \boldsymbol{s}) = \mathbb{E}_{a_t^p\sim p(\cdot|\boldsymbol{s})}\left[K_h(a_t, a_t^p)\right], \qquad q_{\theta,h}(a_t \mid \boldsymbol{s}) = \mathbb{E}_{a_t^q\sim q_\theta(\cdot|\boldsymbol{s})}\left[K_h(a_t, a_t^q)\right]. \tag{5}$$

where $K_h(\cdot,\cdot)$ denotes a kernel function with bandwidth $h > 0$. This yields the sample-based drifting field $\boldsymbol{V}_{\theta,h}(a_t, \boldsymbol{s})$:

$$\begin{aligned} \boldsymbol{V}_{\theta,h}(a_t, \boldsymbol{s}) =& \nabla_{a_t} \log p_h(a_t \mid \boldsymbol{s}) - \nabla_{a_t} \log q_{\theta,h}(a_t \mid \boldsymbol{s}) \\ =& \nabla_{a_t} \log \mathbb{E}_{a_t^p\sim p(\cdot|\boldsymbol{s})}\left[K_h(a_t, a_t^p)\right] - \nabla_{a_t} \log \mathbb{E}_{a_t^q\sim q_\theta(\cdot|\boldsymbol{s})}\left[K_h(a_t, a_t^q)\right] \\ =& \frac{\mathbb{E}_{a_t^p\sim p(\cdot|\boldsymbol{s})}\left[\nabla_{a_t} K_h(a_t, a_t^p)\right]}{\mathbb{E}_{a_t^p\sim p(\cdot|\boldsymbol{s})}\left[K_h(a_t, a_t^p)\right]} - \frac{\mathbb{E}_{a_t^q\sim q_\theta(\cdot|\boldsymbol{s})}\left[\nabla_{a_t} K_h(a_t, a_t^q)\right]}{\mathbb{E}_{a_t^q\sim q_\theta(\cdot|\boldsymbol{s})}\left[K_h(a_t, a_t^q)\right]}. \end{aligned} \tag{6}$$

A drifting model $\boldsymbol{G}_\theta$ can then be learned by updating the current generated samples toward this field,

$$\mathcal{L}_{\mathrm{drift}}(\theta) = \mathbb{E}_{\boldsymbol{s},\, a_t}\left[\left\| \boldsymbol{G}_\theta(a_t, \boldsymbol{s}) - \mathrm{stopgrad}\left(\boldsymbol{G}_\theta(a_t, \boldsymbol{s}) + \eta \boldsymbol{V}_{\theta,h}(a_t, \boldsymbol{s})\right) \right\|_2^2 \right]. \tag{7}$$

For the full derivation, please refer to Section A.1.

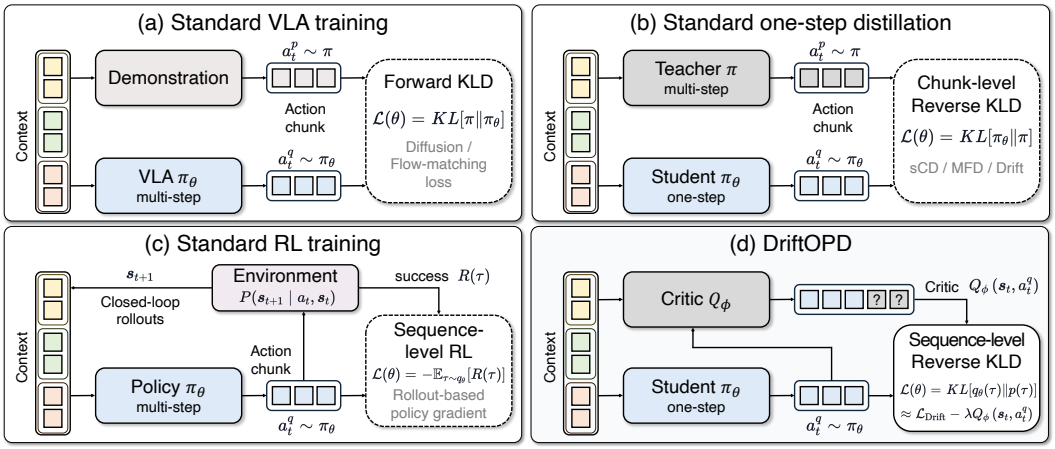


Figure 2: Comparison of VLA training paradigms. (a) Standard VLA training uses demonstration-supervised forward KLD, (b) existing one-step distillation optimizes chunk-level reverse KLD, and (c) RL relies on closed-loop rollouts, whereas (d) DriftOPD approximates sequence-level reverse KLD using a drifting objective and critic guidance without online rollouts.

4 METHOD

4.1 SEQUENCE-LEVEL REVERSE KL DIVERGENCE IN VLA

We formulate VLA learning through the reverse KLD between the distributions over action *sequences* induced by the model and target policies. This sequence-level perspective is illustrated in Figure 2. Let t index the action-chunk sequence, with $\boldsymbol{s}_t$ denoting the current VLA context (e.g., observation, instruction, and action state) and a_t the corresponding action chunk. Accordingly, let $\tau = (\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, \ldots, a_T, \boldsymbol{s}_{T+1})$ denote a trajectory composed of a sequence of action chunks and their corresponding conditioning contexts. The trajectory distributions induced by the model policy π_θ and the target policy π are respectively given by

$$q_\theta(\tau) = \rho(\boldsymbol{s}_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid \boldsymbol{s}_t) P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t), \qquad p(\tau) = \rho(\boldsymbol{s}_1) \prod_{t=1}^{T} \pi(a_t \mid \boldsymbol{s}_t) P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t), \tag{8}$$

where $\rho(\boldsymbol{s}_1)$ denotes the initial context distribution and $P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t)$ denotes the transition induced by executing a_t under the current context. Since the two trajectory distributions share the same initial context distribution and transition dynamics, their reverse KLD reduces to the cumulative policy discrepancy along model-generated trajectories (full derivation in Section A.2):

$$\mathcal{L}_{\text{RKL}}(\theta) = D_{\text{KL}}\left[q_\theta(\tau) \,\|\, p(\tau)\right] = \mathbb{E}_{\tau \sim q_\theta}\left[\log \frac{q_\theta(\tau)}{p(\tau)}\right] = \mathbb{E}_{\tau \sim q_\theta}\left[\sum_{t=1}^{T} \log \frac{\pi_\theta(a_t \mid \boldsymbol{s}_t)}{\pi(a_t \mid \boldsymbol{s}_t)}\right]. \tag{9}$$

To expose the sequential structure of this objective, we define $J_t(\boldsymbol{s}_t)$ as the reverse-KL cost-to-go from sequence index t. Its Bellman recursion is

$$J_t(\boldsymbol{s}_t) = \underbrace{D_{\text{KL}}\left[\pi_\theta(\cdot \mid \boldsymbol{s}_t) \,\|\, \pi(\cdot \mid \boldsymbol{s}_t)\right]}_{\mathcal{L}_{\text{Local}}} + \underbrace{\mathbb{E}_{\substack{a_t \sim \pi_\theta(\cdot \mid \boldsymbol{s}_t) \\ \boldsymbol{s}_{t+1} \sim P(\cdot \mid \boldsymbol{s}_t, a_t)}}\left[J_{t+1}(\boldsymbol{s}_{t+1})\right]}_{\mathcal{L}_{\text{Future}}}. \tag{10}$$

Here, $\mathcal{L}_{\text{Local}}$ captures the discrepancy between the model and target distributions over the current action chunk, whereas $\mathcal{L}_{\text{Future}}$ captures how the current action chunk affects the distributional discrepancy over subsequent action chunks through the induced future contexts. Thus, sequence-level reverse-KL in VLAs accounts for both the current action-distribution discrepancy and its effect on future behavior, as in language-model OPD (Gu et al., 2024).

4.2 SINGLE-POSITIVE LOCAL TERM: DRIFTING FIELD

The local term $\mathcal{L}_{\text{Local}}$ in Eq. (10) requires minimizing the reverse KLD between the model and target action-chunk distributions, $D_{\text{KL}}[\pi_\theta(\cdot \mid \boldsymbol{s}_t) \,\|\, \pi(\cdot \mid \boldsymbol{s}_t)]$. However, unlike autoregressive LLMs,

VLA action experts do not provide explicit densities $\pi_\theta(\cdot \mid \boldsymbol{s}_t)$ over action chunks. We therefore optimize a kernel-smoothed local reverse KLD using the sample-space drifting field (Section 3.2). Let $K_h(a, a') = \kappa_h(a - a')$ be a normalized, nonnegative, differentiable density with bandwidth h, and define the kernel-smoothed distributions as

$$\pi_h(a \mid \boldsymbol{s}_t) = \mathbb{E}_{a^p \sim \pi(\cdot|\boldsymbol{s}_t)} \left[K_h(a, a^p)\right], \qquad \pi_{\theta,h}(a \mid \boldsymbol{s}_t) = \mathbb{E}_{a^q \sim \pi_\theta(\cdot|\boldsymbol{s}_t)} \left[K_h(a, a^q)\right]. \tag{11}$$

The corresponding smoothed local objective is

$$\mathcal{L}^h_{\text{Local}}(\theta; \boldsymbol{s}_t) = D_{\text{KL}} \left[\pi_{\theta,h}(\cdot \mid \boldsymbol{s}_t) \,\|\, \pi_h(\cdot \mid \boldsymbol{s}_t)\right]. \tag{12}$$

Its drifting field is defined by the difference between the target and model scores,

$$\boldsymbol{V}_{\theta,h}(a, \boldsymbol{s}_t) = \nabla_a \log \pi_h(a \mid \boldsymbol{s}_t) - \nabla_a \log \pi_{\theta,h}(a \mid \boldsymbol{s}_t). \tag{13}$$

Proposition 1 (Local reverse KL as drifting)**.** *Let $a_t^q = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$ and $\widetilde{a}_t^q = a_t^q + \boldsymbol{\epsilon}_h$, where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{\epsilon}_h \sim \kappa_h$. Define*

$$\mathcal{L}_{\text{Drift}}(\theta; \boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{z}, \boldsymbol{\epsilon}_h} \left[\left\| \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t) - \text{stopgrad}(a_t^q + \eta\, \boldsymbol{V}_{\theta,h}(\widetilde{a}_t^q, \boldsymbol{s}_t)) \right\|_2^2\right]. \tag{14}$$

Then, for any $\eta > 0$,

$$\nabla_\theta \mathcal{L}^h_{\text{Local}}(\theta; \boldsymbol{s}_t) = \frac{1}{2\eta} \nabla_\theta \mathcal{L}_{\text{Drift}}(\theta; \boldsymbol{s}_t). \tag{15}$$

Proof. The proof and detailed derivation are provided in Section A.3. □

In our setting, the target policy is accessible only through offline demonstrations rather than repeated queries at a fixed context $\boldsymbol{s}_t$. For each demonstration context, we therefore use its single recorded action chunk a_t^p as the positive sample, following prior work (Houssaini et al., 2026; Lu et al., 2026),

$$\pi(\cdot \mid \boldsymbol{s}_t) \approx \delta_{a_t^p}(\cdot), \qquad \pi_h(a \mid \boldsymbol{s}_t) \approx K_h(a, a_t^p), \tag{16}$$

where $\delta_{a_t^p}(\cdot)$ denotes the Dirac delta at a_t^p. The drifting field then becomes

$$\boldsymbol{V}_{\theta,h}(a, \boldsymbol{s}_t) = \nabla_a \log K_h(a, a_t^p) - \frac{\mathbb{E}_{a^q \sim \pi_\theta(\cdot|\boldsymbol{s}_t)} \left[\nabla_a K_h(a, a^q)\right]}{\mathbb{E}_{a^q \sim \pi_\theta(\cdot|\boldsymbol{s}_t)} \left[K_h(a, a^q)\right]}. \tag{17}$$

Hence, the local reverse-KL term can be optimized using a single positive demonstration chunk together with samples from the current VLA policy. A detailed analysis of the single-positive drifting field is provided in Section D.3.

4.3 Action-Value Future Term: Critic Surrogate

The future term $\mathcal{L}_{\text{Future}}$ in Eq. (10) accounts for how the current action chunk affects the subsequent trajectory through future observations. Following the Bellman recursion, we define the divergence-to-go after taking a_t at $\boldsymbol{s}_t$ as

$$Q_t^\theta(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{s}_{t+1} \sim P(\cdot|\boldsymbol{s}_t, a_t)} \left[J_{t+1}(\boldsymbol{s}_{t+1})\right], \tag{18}$$

such that the future term is

$$\mathcal{L}_{\text{Future}}(\theta; \boldsymbol{s}_t) = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\boldsymbol{s}_t)} \left[Q_t^\theta(\boldsymbol{s}_t, a_t)\right]. \tag{19}$$

In the sequence-level gradient decomposition, Q_t^θ is evaluated under the current policy and treated as fixed with respect to θ when differentiating the contribution at sequence index t (Section A.2). Using $a_t = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$, the resulting future gradient contribution is

$$\boldsymbol{g}_{\text{Future}}(\theta; \boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{z}} \left[(\partial_\theta \boldsymbol{G}_\theta)^\top \nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t)\right]. \tag{20}$$

Directly evaluating Q_t^θ requires rolling out the current policy and evaluating the remaining divergence along its future trajectory. Instead, we use a frozen action-value critic $Q_\phi(\boldsymbol{s}_t, a_t)$ to approximate the unavailable action gradient,

$$\nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t) \approx -\nabla_{a_t} Q_\phi(\boldsymbol{s}_t, a_t). \tag{21}$$

Note that Q_t^θ denotes a divergence cost, whereas Q_ϕ is a value score for which larger values indicate better actions. The critic parameters ϕ are fixed during policy optimization, and the critic is independent of the VLA policy parameterization. Accordingly, the future gradient contribution is approximated as

$$\boldsymbol{g}_{\text{Future}}(\theta; \boldsymbol{s}_t) \approx -\mathbb{E}_{\boldsymbol{z}} \left[(\partial_\theta \boldsymbol{G}_\theta)^\top \nabla_{a_t} Q_\phi(\boldsymbol{s}_t, a_t) \right]. \tag{22}$$

Equivalently, this update can be obtained from the critic surrogate

$$\mathcal{L}_{\text{Critic}}(\theta; \boldsymbol{s}_t) = -\mathbb{E}_{\boldsymbol{z}} \left[Q_\phi(\boldsymbol{s}_t, \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)) \right], \tag{23}$$

where gradients are propagated through the generated action chunk while Q_ϕ remains frozen. Thus, the critic provides a tractable surrogate for the long-horizon future gradient contribution without requiring closed-loop rollouts during policy optimization. A detailed analysis and justification of the critic surrogate, together with its error analysis, are provided in Section A.4.

4.4 Objective Function

Combining the local and future terms, we introduce **Drift On-Policy Distillation (DriftOPD)**, a unified policy optimization framework that accounts for both local action-distribution matching and long-horizon action quality. The overall framework is summarized in Figure 2. Specifically, the local reverse KLD term is realized by the drifting objective in Section 4.2, while the future gradient contribution is approximated by the frozen critic surrogate in Section 4.3. Our overall objective is

$$\mathcal{L}_{\text{DriftOPD}}(\theta) = \mathbb{E}_{(\boldsymbol{s}_t, a_t^p) \sim \mathcal{D}} \left[\mathcal{L}_{\text{Drift}}(\theta; a_t^p, \boldsymbol{s}_t) + \lambda \mathcal{L}_{\text{Critic}}(\theta; \boldsymbol{s}_t) \right], \tag{24}$$

where $\mathcal{D}$ denotes the offline demonstration dataset, and $\lambda > 0$ absorbs the constant scale η induced by the drifting step and controls the relative contribution of the future term.

Algorithm. The complete training procedure of DriftOPD is summarized in Algorithm 1. For each demonstration pair $(\boldsymbol{s}_t, a_t^p)$, we sample M action chunks $a_t^q = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$ from the current VLA policy and estimate the drifting field in Eq. (17). The resulting drifting update locally aligns the model action distribution with the demonstrated target action while retaining the model-distribution score term of reverse-KL optimization. In parallel, the frozen critic Q_ϕ provides long-horizon guidance by propagating $\nabla_a Q_\phi(\boldsymbol{s}_t, a)$ through the generated action chunk. The critic parameters ϕ remain fixed throughout policy optimization.

Algorithm 1 Drift On-Policy Distillation (DriftOPD)

Input: offline demonstrations $\mathcal{D}$, critic Q_ϕ, weight $\lambda > 0$
Output: distilled VLA policy $\boldsymbol{G}_\theta$

for $k = 1, \ldots, N_{\text{train}}$ **do**
  Sample $(\boldsymbol{s}_t, a_t^p) \sim \mathcal{D}$ and $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  Generate $a_t^q = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$
  Estimate $\widehat{\boldsymbol{V}}_{\theta,h}(a_t^q, \boldsymbol{s}_t)$ using a_t^p and policy samples
  $\mathcal{L}_{\text{Drift}} = \left\| a_t^q - \text{stopgrad}\left(a_t^q + \eta \widehat{\boldsymbol{V}}_{\theta,h} \right) \right\|_2^2$
  $\mathcal{L}_{\text{DriftOPD}} = \mathcal{L}_{\text{Drift}} - \lambda Q_\phi(\boldsymbol{s}_t, a_t^q)$
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{DriftOPD}}$ ▷ Q_ϕ is frozen
end for

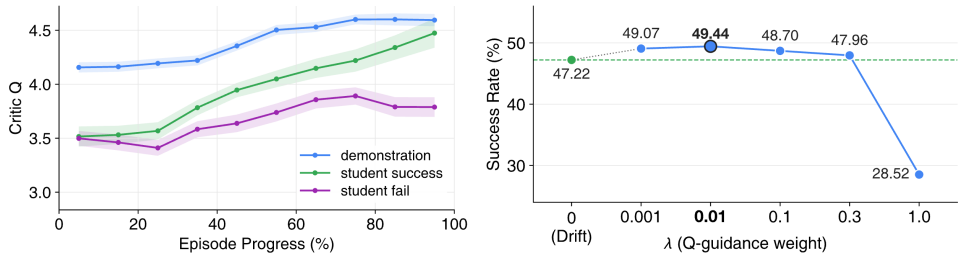
Takeaway. Consequently, the two components of DriftOPD directly correspond to the decomposition in Eq. (10). The drifting term $\mathcal{L}_{\text{Drift}}$ realizes local action-distribution matching, while the critic term $\mathcal{L}_{\text{Critic}}$ provides a tractable surrogate for the future gradient contribution.. This formulation enables on-policy distillation of VLA action experts directly from generated action chunks, without requiring explicit action densities or closed-loop rollouts during policy optimization.

5 Experiment

We experimentally evaluate and analyze the effectiveness of DriftOPD on robot manipulation tasks. We use the $\pi_{0.5}$ (Black et al., 2025a), GR00T N1.5/N1.6 (Bjorck et al., 2025), and ABot-M0 (Yang et al., 2026a) vision language action models, covering fully finetuned models, LoRA (Hu et al., 2022) finetuned models, models finetuned on selected VLM layers, and a JiT (Li & He, 2026)-style model that directly predicts clean actions. For the given embodiment-specific finetuned models, we further train each VLA using the DriftOPD objective (Eq. (24)) for only 10k steps. For the action-value critic $Q_\phi(\boldsymbol{s}_t, a_t)$ in Section 4.3, we adapt VGAS (Xu et al., 2026), a Q-function trained solely on demonstration data to avoid requiring rollout data for critic training. To compare one-step distillation in terms of performance, stability, and applicability, we use the multi-step teacher, its one-step prediction, and existing one-step distillation methods as baselines, including sCD (Lu & Song, 2025), MFD (Zhao et al., 2026a), and Drift (Zhang et al., 2026).

Table 1: Average task success rates on RoboCasa365, reported separately for each evaluation suite. Each task uses 30 episodes, totaling 540, 480, and 480 per suite.

Model	Action prediction	Fine-tuning strategy	Teacher steps	Suite	Teacher	1-step				
						Teacher	sCD	MFD	Drift	DriftOPD
$\pi_{0.5}$	Flow matching	Full FT	10	`atomic_seen` (18)	39.6	31.30	28.33	34.44	32.22	**38.33**
				`composite_seen` (16)	7.1	0.83	1.04	2.50	5.00	**5.42**
				`composite_unseen` (16)	1.2	0.83	0.83	0.83	1.04	**1.67**
$\pi_{0.5}$	Flow matching	LoRA ($r = 16$)	10	`atomic_seen` (18)	39.6	31.30	29.26	28.33	34.07	**39.26**
				`composite_seen` (16)	7.1	0.83	1.67	2.71	3.12	**5.00**
				`composite_unseen` (16)	1.2	0.83	0.83	1.25	**1.46**	**1.46**
GR00T N1.5	Flow matching	Action expert	4	`atomic_seen` (18)	50.7	42.96	39.63	43.52	43.89	**45.37**
				`composite_seen` (16)	14.8	7.71	4.17	**8.54**	8.12	**8.54**
				`composite_unseen` (16)	2.7	2.50	1.67	2.71	2.29	**2.92**
GR00T N1.6	Flow matching	Action head + top-4 VLM layers	4	`atomic_seen` (18)	51.1	44.81	**51.67**	32.96	47.22	49.44
				`composite_seen` (16)	9.4	2.71	3.54	0.83	3.96	**5.00**
				`composite_unseen` (16)	1.7	0.83	0.62	**1.04**	0.62	**1.04**



(a) Success signal from critic Q (b) Effect of critic loss on success rate

Figure 3: Analysis of critic guidance for sequence-level VLA policy optimization on RoboCasa365 with GR00T N1.6. (a) Critic Q over episode progress for demonstrations, successful rollouts, and failed student rollouts. (b) Task success rate versus λ on the `atomic_seen` suite.

5.1 Simulation Experiments

Our simulation benchmarks include RoboCasa365 (Nasiriany et al., 2026) for large-scale household manipulation and RoboTwin 2.0 (Chen et al., 2026a) for bimanual manipulation under diverse domain variations. We additionally evaluate on LIBERO (Liu et al., 2023a) and LIBERO-Plus (Fei et al., 2025) in Section C. All evaluations follow the official publicly released protocols for reproducibility. Further implementation details are provided in Section B.

Main results. As shown in Table 1 for RoboCasa365 and Table 2 for RoboTwin 2.0, DriftOPD achieves task success rates that are mostly superior or comparable to sCD, MFD, and Drift. In several suites, it even surpasses the multi-step teacher despite using only one-step inference. Meanwhile, sCD and MFD sometimes underperform the one-step teacher prediction. We hypothesize that limited embodiment-specific data for VLAs exacerbate the difficulty of low-noise estimation, where the target distribution becomes sharply concentrated and score estimation is known to be unstable (Bortoli, 2022; De Bortoli et al., 2024; Kim et al., 2021). This instability is particularly evident in the training collapse of MFD on ABot-M0, which directly predicts clean actions. In contrast, the single-positive drifting objectives of Drift and DriftOPD in Eq. (7) may mitigate this issue through multi-bandwidth kernel density estimation (Deng et al., 2026). DriftOPD further complements this robust chunk-level objective with sequence-level critic guidance, which accounts for the effect of the action chunk on future task progress and generally improves over chunk-level drifting distillation.

Effect of critic guidance. To assess whether the critic provides a meaningful success-aware signal and its sensitivity to critic weighting, we analyze trajectory-level critic values and sweep λ. As shown in Figure 3a, the critic clearly separates successful from failed trajectories, supporting its use for sequence-level optimization. In Figure 3b, a broad range of λ values (0.001-0.3) consistently improves success over the drifting-only objective ($\lambda = 0$). An excessively large weight of 1.0 causes performance collapse, likely because RMS normalization fixes the scale of the multi-bandwidth drifting loss (Deng et al., 2026). These two results support that the critic provides useful sequence-level guidance without requiring delicate tuning.

GPU time to success. Unlike image or video generation, reducing action generation to a single inference step does not necessarily translate into faster task completion. A one-step policy may reduce success rate or require more policy calls because of noisier action sequences. We therefore measure

Table 2: Average task success rates on RoboTwin 2.0, grouped by task horizon. Each task is evaluated over 30 episodes under paired environment settings. `short`, `medium`, and `long` horizons correspond to average trajectory lengths of < 150, 150-279, and ≥ 280 steps, respectively.

Model	Action objective	Fine-tuning strategy	Teacher steps	Suite	Teacher	1-step				
						Teacher	sCD	MFD	Drift	DriftOPD
$\pi_{0.5}$-LoRA	Flow matching	LoRA ($r = 16$)	10	`short` (18)	60.74	61.30	64.81	54.44	66.30	**67.78**
				`medium` (21)	67.30	56.35	61.11	48.25	61.90	**64.13**
				`long` (11)	55.15	40.61	36.97	16.36	**48.48**	46.67
ABot-M0	Direct clean action	Joint FT (VLM + action expert)	10	`short` (18)	70.19	70.37	67.41	2.41	**69.07**	64.63
				`medium` (21)	73.97	67.78	66.83	4.13	65.71	**68.10**
				`long` (11)	36.97	36.97	30.61	0.91	35.76	**38.79**

Figure 4: GPU time to success: policy calls $\times$ per-call latency (idle B200, batch 1) (Bars: total GPU time per success, including failed episodes; Diamonds (parenthesized): mean over successful episodes).

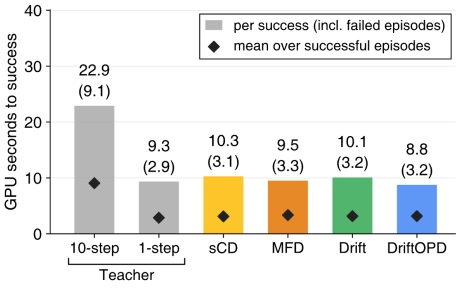


Table 3: Real-world evaluation in paired environments on unimanual pick-and-place and bimanual handover tasks. (parentheses: partial success rate (e.g., picking the object but failing to place or handover it)).

Task	4-step	1-step				
	Teacher	Teacher	sCD	MFD	Drift	DriftOPD
`Unimanual (right arm)`						
Easy	93.3	40.0	0.0 (60.0)	0.0 (0.0)	86.7	**93.3**
Moderate	73.3	46.7	6.7 (20.0)	0.0 (0.0)	**60.0**	**60.0**
Hard	66.7	13.3	0.0 (0.0)	0.0 (0.0)	60.0	**66.7**
Challenging	46.7	0.0	0.0 (0.0)	0.0 (0.0)	**13.3**	**13.3**
Avg	70.0	25.0	1.7	0.0	55.0	**58.3**
`Bimanual`						
`Handover (LR)`	73.3	60.0	0.0 (0.0)	0.0 (0.0)	53.3	**80.0**
`Handover (RL)`	86.7	53.3	0.0 (26.7)	0.0 (20.0)	40.0	**73.3**
Avg	80.0	56.7	0.0	0.0	46.7	**76.7**

GPU time to success, defined as the number of policy calls multiplied by the per-call latency, on 10 randomly selected RoboCasa365 `atomic_seen` tasks with $\pi_{0.5}$ (Figure 4). DriftOPD is $2.60\times$ faster than the 10-step teacher, thereby reducing GPU time to task success.

5.2 Real-World Experiments

For real-world evaluation, we use the MolmoAct 2 Research Kit[1] from I2RT Robotics. We finetune the cross-embodiment pretrained GR00T N1.6 base model on 638 teleoperated episodes comprising 366k frames, and subsequently distill the resulting policy into one-step students for 10k training steps. We evaluate four unimanual pick-and-place tasks with increasing difficulty using the right YAM robot arm, as well as two bimanual pick-and-handover-and-place tasks using two YAM robot arms. As summarized in Table 3, DriftOPD achieves the highest one-step success rates on both the unimanual and bimanual tasks. A detailed analysis of action oscillation is provided in Section D.1, while the hardware setup and evaluation protocol are described in Section B.

6 Conclusion

We introduce DriftOPD, a sequence-level one-step distillation framework for VLAs. Starting from the reverse KL divergence over action-sequence distributions, we decompose sequence-level OPD into a local action-distribution term and a future-sequence term. DriftOPD realizes the local term with a single-positive drifting objective and approximates the future gradient contribution with a demonstration-trained critic, enabling rollout-free one-step policy optimization from offline demonstrations. Across diverse VLA configurations in simulation and real-world manipulation, DriftOPD generally outperforms existing one-step distillation baselines while achieving performance comparable to multi-step policies. Our results further suggest that limited embodiment-specific data can exacerbate low-noise estimation, while the drifting objective provides stable distribution matching and the critic supplies effective sequence-level guidance. Overall, DriftOPD provides an efficient and sequence-aware approach to one-step VLA distillation without online rollouts or a separate multi-step teacher. Further discussion is provided in Section E.

[1] https://i2rt.com/products/molmoact-2-research-kit

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing, identifying related work, supporting research execution, and assisting with mathematical derivations and proofs. We did not use generative AI tools for research ideation or drafting sections of the paper, and the generation of synthetic datasets was not applicable to this work. Additionally, we used generative AI tools for monitoring computing systems and developing software for robot experiments. We reviewed all AI-assisted work. We verified AI-identified related work through a manual literature survey, checked AI-assisted mathematical derivations and proofs through independent manual derivation and verification, and verified and tested AI-generated code for correctness by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personal or sensitive data, or applications that raise significant ethical, privacy, security, or fairness concerns. The real-world robot experiments were conducted in controlled laboratory environments. We are not aware of any additional ethical concerns associated with the methods or experiments presented in this work.

REPRODUCIBILITY STATEMENT

We provide detailed theoretical and experimental specifications to support reproducibility. The derivation of the sequence-level reverse-KL objective, the drifting formulation, and the critic surrogate, including assumptions and approximation-error analyses, is provided in Section A. The complete experimental protocol is documented in Section B, including model configurations and starting checkpoints, student optimization and DriftOPD-specific hyperparameters, critic architecture and training, baseline implementations, simulation and real-world evaluation protocols, and hardware and software environments. Section C provides per-task results for all simulation benchmarks, and All evaluated benchmarks and pretrained VLA checkpoints are publicly available, and the main experiments follow their official evaluation protocols whenever applicable. We additionally provide the implementation code required to reproduce our training and evaluation as supplementary material.

AUTHOR CONTRIBUTIONS

For anonymity, we omit this section.

ACKNOWLEDGMENTS

For anonymity, we omit this section.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025a. URL https://openreview.net/forum?id=vlhoswksBO.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025b. doi: 10.15607/RSS.2025.XXI.010.

Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=MhK5aXo3gB. Expert Certification.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=itonej9GIV.

Yang Chen, Xiaoguang Ma, and Bin Zhao. Mean-flow based one-step vision-language-action. *arXiv preprint arXiv:2603.01469*, 2026b.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026.

Valentin De Bortoli, Michael Hutchinson, Peter Wirnsberger, and Arnaud Doucet. Target score matching. *arXiv preprint arXiv:2402.08667*, 2024.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Zhen Fang, Wenxuan Huang, Yu Zeng, Yiming Zhao, Shuang Chen, Kaituo Feng, Yunlong Lin, Lin Chen, Zehui Chen, Shaosheng Cao, and Feng Zhao. Flow-opd: On-policy distillation for flow matching models. *arXiv preprint arXiv:2605.08063*, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Yuxuan Gao, Shiqi Zhang, Yedong Shen, Yifan Duan, Wenhao Yu, Xin Zhang, Siyuan Cao, Jiajun Deng, and Yanyong Zhang. Driftingvla: Native one-step vision-language-action generation via per-dimension temporal drifting. *arXiv preprint arXiv:2608.29749*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uWj4s7rMnR.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30467–30476, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Wolfgang Härdle. *Smoothing techniques: with implementation in S*. Springer Science & Business Media, 2012.

Anas Houssaini, Mohamad H. Danesh, Amin Abyaneh, Scott Fujimoto, Hsiu-Chin Lin, and David Meger. Drift q-learning. *arXiv preprint arXiv:2606.00350*, 2026.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Bofang Jia, Pengxiang Ding, Can Cui, Mingyang Sun, Pengfang Qian, Siteng Huang, Zhaoxin Fan, and Donglin Wang. Score and distribution matching policy: Advanced accelerated visuomotor policies via matched distillation. *arXiv preprint arXiv:2412.09265*, 2024.

Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation. *arXiv preprint arXiv:2106.05527*, 2021.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R 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. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ZMnD6QZAE6.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.017.

Jaewoo Lee, Hyeongyu Kang, Dohyun Kim, Kyuil Sim, Woocheol Shin, Minsu Kim, Taeyoung Yun, Jeongjae Lee, Sanghyeok Choi, Tabitha Edith Lee, Jong Chul Ye, and Jinkyoo Park. Aligning few-step generative model via amortizing sample-based variational inference. In *ICML 2026 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2026. URL https://openreview.net/forum?id=5CNn617yWW.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=TQhSodCM4r.

Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, et al. Vla-rft: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*, 2025.

Quanhao Li, Junqiu Yu, Kaixun Jiang, Yujie Wei, Zhen Xing, Pandeng Li, Ruihang Chu, Shiwei Zhang, Yu Liu, and Zuxuan Wu. Diffusionopd: A unified perspective of on-policy distillation in diffusion models. *arXiv preprint arXiv:2605.15055*, 2026b.

Shaolong Li, Lichao Sun, and Yongchao Chen. One-step flow policy: Self-distillation for fast visuomotor policies. *arXiv preprint arXiv:2603.12480*, 2026c.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023a.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=XVjTT1nw5z.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *International Conference on Learning Representations*, volume 2025, pp. 50611–50649, 2025.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

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. *arXiv preprint arXiv:2505.18719*, 2025.

Susie Lu, Haonan Chen, Weirui Ye, and Yilun Du. Driftworld: Fast world modeling through drifting. *arXiv preprint arXiv:2607.15065*, 2026.

Shaunak A Mehta, Yusuf Umut Ciftci, Balamurugan Ramachandran, Somil Bansal, and Dylan P Losey. Stable-bc: Controlling covariate shift with stable behavior cloning. *IEEE Robotics and Automation Letters*, 10(2):1952–1959, 2025.

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 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=GcEIvidYSw.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tQJYKwc3n4.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Lucy Xiaoyang Shi, brian ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=lNVHg9npif.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, volume 2024, pp. 15078–15097, 2024.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 32211–32252, 2023.

Changhua Xu, En Yu, Junyu Xuan, and Jie Lu. Vgas: value-guided action-chunk selection for few-shot vision-language-action adaptation. *arXiv preprint arXiv:2602.07399*, 2026.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, Feng Xiong, Xing Wei, Zhiheng Ma, and Mu Xu. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026a.

Zemin Yang, Yaoyu He, Yiming Zhong, Yuhao Zhang, Xinge Zhu, Yao Mu, Qingqiu Huang, and Yuexin Ma. Implicit drifting policy: One-step action generation via conditional expert geometry. *arXiv preprint arXiv:2606.01098*, 2026b.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6613–6623. IEEE, 2024.

Jiahui Zhang, Ze Huang, Chun Gu, Zipei Ma, and Li Zhang. Reinforcing action policies by prophesying. *arXiv preprint arXiv:2511.20633*, 2025.

Jiawei Zhang, Mengfei Xia, Gen Li, and Yuantao Gu. Distilling drifting transformers with representation autoencoders. *arXiv preprint arXiv:2606.15553*, 2026.

An Zhao, Shengyuan Zhang, Zhongjian Sun, Yixiang Zhou, Zejian Li, Ling Yang, Tianrun Chen, and Lingyun Sun. Mean flow distillation: Robust and stable distillation for flow matching models. *arXiv preprint arXiv:2606.11155*, 2026a.

Anhao Zhao, Haoran Xin, Yingqi Fan, Junlong Tong, Wenjie Li, and Xiaoyu Shen. Decoupling kl and trajectories: A unified perspective for sft, dagger, offline rl, and opd in llm distillation. *arXiv preprint arXiv:2605.16826*, 2026b.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Forty-third International Conference on Machine Learning*, 2026c. URL https://openreview.net/forum?id=Jpxfof0EaS.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

Zhide Zhong, Haodong Yan, Junfeng Li, Junjie He, Tianran Zhang, and Haoang Li. Vla-opd: Bridging offline sft and online rl for vision-language-action models via on-policy distillation. *arXiv preprint arXiv:2603.26666*, 2026.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. Wmpo: World model-based policy optimization for vision-language-action models. In *International Conference on Learning Representations*, volume 2026, pp. 62486–62502, 2026.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R Sanketi, Grecia Salazar, Michael S Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J Joshi, Alex Irpan, brian ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=XMQgwiJ7KSX.

A Theoretical Demonstration

A.1 Drifting Field and Reverse KL

We provide a derivation connecting the drifting field in Section 3.2 to the local descent direction of the reverse KLD. Throughout this section, we fix the conditioning context $\boldsymbol{s}$ and assume that $p(a_t \mid \boldsymbol{s})$ and $q_\theta(a_t \mid \boldsymbol{s})$ are positive and continuously differentiable, with sufficient decay at the boundary to justify integration by parts.

Reverse-KL descent in sample space. For a fixed condition $\boldsymbol{s}$, consider the reverse KLD

$$\mathcal{F}(q_\theta) = D_{\mathrm{KL}}[q_\theta(a_t \mid \boldsymbol{s}) \,\|\, p(a_t \mid \boldsymbol{s})] = \int q_\theta(a_t \mid \boldsymbol{s}) \log \frac{q_\theta(a_t \mid \boldsymbol{s})}{p(a_t \mid \boldsymbol{s})}\, da_t. \tag{25}$$

To characterize a descent direction directly in the sample space, let $\boldsymbol{u}(a_t, \boldsymbol{s})$ be a smooth vector field and perturb each sample infinitesimally as

$$T_\epsilon(a_t) = a_t + \epsilon \boldsymbol{u}(a_t, \boldsymbol{s}), \qquad q_{\theta,\epsilon} = (T_\epsilon)_{\#} q_\theta, \tag{26}$$

where $(T_\epsilon)_{\#} q_\theta$ denotes the pushforward of q_θ through T_ϵ. The induced density satisfies the continuity equation

$$\left.\frac{\partial q_{\theta,\epsilon}}{\partial \epsilon}\right|_{\epsilon=0} = -\nabla_{a_t} \cdot (q_\theta(a_t \mid \boldsymbol{s}) \boldsymbol{u}(a_t, \boldsymbol{s}))\,. \tag{27}$$

Differentiating Eq. (25) along this perturbation gives

$$\begin{aligned}
\left.\frac{d}{d\epsilon}\mathcal{F}(q_{\theta,\epsilon})\right|_{\epsilon=0} &= \int \left.\frac{\partial q_{\theta,\epsilon}}{\partial \epsilon}\right|_{\epsilon=0} \left(\log \frac{q_\theta}{p} + 1\right) da_t \\
&= -\int \nabla_{a_t} \cdot (q_\theta \boldsymbol{u}) \left(\log \frac{q_\theta}{p} + 1\right) da_t \\
&= \int q_\theta\, \boldsymbol{u}^\top \nabla_{a_t} \log \frac{q_\theta}{p}\, da_t \\
&= -\mathbb{E}_{a_t \sim q_\theta(\cdot \mid \boldsymbol{s})} \left[\boldsymbol{u}(a_t, \boldsymbol{s})^\top \boldsymbol{V}^*(a_t, \boldsymbol{s})\right],
\end{aligned} \tag{28}$$

where

$$\boldsymbol{V}^*(a_t, \boldsymbol{s}) = \nabla_{a_t} \log p(a_t \mid \boldsymbol{s}) - \nabla_{a_t} \log q_\theta(a_t \mid \boldsymbol{s}). \tag{29}$$

Here, the third equality follows from integration by parts. Choosing $\boldsymbol{u} = \boldsymbol{V}^*$ yields

$$\left.\frac{d}{d\epsilon}\mathcal{F}(q_{\theta,\epsilon})\right|_{\epsilon=0} = -\mathbb{E}_{a_t \sim q_\theta(\cdot \mid \boldsymbol{s})} \left[\|\boldsymbol{V}^*(a_t, \boldsymbol{s})\|_2^2\right] \le 0. \tag{30}$$

Therefore, the difference between the target and model scores defines a local descent direction of the reverse KLD in sample space.

KDE-based drifting field. When $p(a_t \mid \boldsymbol{s})$ and $q_\theta(a_t \mid \boldsymbol{s})$ cannot be evaluated explicitly, we replace them by the kernel-smoothed densities

$$p_h(a_t \mid \boldsymbol{s}) = \mathbb{E}_{a_t^p \sim p(\cdot \mid \boldsymbol{s})}[K_h(a_t, a_t^p)], \qquad q_{\theta,h}(a_t \mid \boldsymbol{s}) = \mathbb{E}_{a_t^q \sim q_\theta(\cdot \mid \boldsymbol{s})}[K_h(a_t, a_t^q)]. \tag{31}$$

Assuming that differentiation and expectation can be interchanged, the score of the smoothed target distribution is

$$\nabla_{a_t} \log p_h(a_t \mid \boldsymbol{s}) = \frac{\mathbb{E}_{a_t^p \sim p(\cdot \mid \boldsymbol{s})}[\nabla_{a_t} K_h(a_t, a_t^p)]}{\mathbb{E}_{a_t^p \sim p(\cdot \mid \boldsymbol{s})}[K_h(a_t, a_t^p)]}, \tag{32}$$

and analogously,

$$\nabla_{a_t} \log q_{\theta,h}(a_t \mid \boldsymbol{s}) = \frac{\mathbb{E}_{a_t^q \sim q_\theta(\cdot \mid \boldsymbol{s})}[\nabla_{a_t} K_h(a_t, a_t^q)]}{\mathbb{E}_{a_t^q \sim q_\theta(\cdot \mid \boldsymbol{s})}[K_h(a_t, a_t^q)]}. \tag{33}$$

Applying the reverse-KL descent result in Eq. (29) to p_h and $q_{\theta,h}$ therefore gives

$$\boldsymbol{V}_{\theta,h}(a_t, \boldsymbol{s}) = \frac{\mathbb{E}_{a_t^p \sim p(\cdot \mid \boldsymbol{s})}[\nabla_{a_t} K_h(a_t, a_t^p)]}{\mathbb{E}_{a_t^p \sim p(\cdot \mid \boldsymbol{s})}[K_h(a_t, a_t^p)]} - \frac{\mathbb{E}_{a_t^q \sim q_\theta(\cdot \mid \boldsymbol{s})}[\nabla_{a_t} K_h(a_t, a_t^q)]}{\mathbb{E}_{a_t^q \sim q_\theta(\cdot \mid \boldsymbol{s})}[K_h(a_t, a_t^q)]}, \tag{34}$$

which recovers the sample-based drifting field in Eq. (6).

For example, with a Gaussian kernel $K_h(a, a') \propto \exp(-\|a - a'\|_2^2/(2h^2))$, we have

$$\nabla_a K_h(a, a') = K_h(a, a') \frac{a' - a}{h^2}. \tag{35}$$

Hence, given target samples $\{a_{t,i}^p\}_{i=1}^{N_p}$ and model samples $\{a_{t,j}^q\}_{j=1}^{N_q}$, the empirical drifting field can be written as

$$\widehat{\boldsymbol{V}}_{\theta,h}(a_t, \boldsymbol{s}) = \frac{1}{h^2} \left(\sum_{i=1}^{N_p} w_i^p a_{t,i}^p - \sum_{j=1}^{N_q} w_j^q a_{t,j}^q \right), \tag{36}$$

where

$$w_i^p = \frac{K_h(a_t, a_{t,i}^p)}{\sum_{k=1}^{N_p} K_h(a_t, a_{t,k}^p)}, \qquad w_j^q = \frac{K_h(a_t, a_{t,j}^q)}{\sum_{k=1}^{N_q} K_h(a_t, a_{t,k}^q)}. \tag{37}$$

Thus, the KDE score difference has the intuitive form of attraction toward nearby target samples and repulsion from nearby model samples.

Drifting objective as reverse-KL optimization. We finally show why moving generated samples toward the drifting field yields the gradient of the smoothed reverse KLD. Let $a_t = G_\theta(z, \boldsymbol{s})$, where z is sampled from the base distribution, and let $\tilde{a}_t = a_t + \epsilon_h$ with $\epsilon_h \sim \kappa_h$. Then $\tilde{a}_t$ follows the kernel-smoothed model distribution $q_{\theta,h}(\cdot \mid \boldsymbol{s})$. Define

$$\mathcal{L}_{\mathrm{RKL}}^h(\theta; \boldsymbol{s}) = D_{\mathrm{KL}}[q_{\theta,h}(\cdot \mid \boldsymbol{s}) \,\|\, p_h(\cdot \mid \boldsymbol{s})]. \tag{38}$$

Using the reparameterization $\tilde{a}_t = G_\theta(z, \boldsymbol{s}) + \epsilon_h$, its gradient is

$$\nabla_\theta \mathcal{L}_{\mathrm{RKL}}^h = -\mathbb{E}_{z,\epsilon_h} \left[(\partial_\theta G_\theta(z, \boldsymbol{s}))^\top \boldsymbol{V}_{\theta,h}(\tilde{a}_t, \boldsymbol{s}) \right]. \tag{39}$$

To see this explicitly, differentiating the reparameterized objective gives

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{RKL}}^h &= \mathbb{E}_{z,\epsilon_h} \left[\partial_\theta \log q_{\theta,h}(\tilde{a}_t \mid \boldsymbol{s}) + (\partial_\theta G_\theta)^\top \nabla_{\tilde{a}_t} \log \frac{q_{\theta,h}(\tilde{a}_t \mid \boldsymbol{s})}{p_h(\tilde{a}_t \mid \boldsymbol{s})} \right] \\ &= \mathbb{E}_{z,\epsilon_h} \left[(\partial_\theta G_\theta)^\top (\nabla_{\tilde{a}_t} \log q_{\theta,h} - \nabla_{\tilde{a}_t} \log p_h) \right], \end{aligned} \tag{40}$$

where the explicit density derivative vanishes because

$$\mathbb{E}_{\tilde{a}_t \sim q_{\theta,h}}[\partial_\theta \log q_{\theta,h}(\tilde{a}_t \mid \boldsymbol{s})] = \int \partial_\theta q_{\theta,h}(a \mid \boldsymbol{s})\, da = \partial_\theta 1 = 0. \tag{41}$$

Substituting the definition of $\boldsymbol{V}_{\theta,h}$ then yields Eq. (39).

Now define the drifting objective with a stop-gradient target,

$$\mathcal{L}_{\mathrm{drift}}(\theta; \boldsymbol{s}) = \mathbb{E}_{z,\epsilon_h} \left[\|G_\theta(z, \boldsymbol{s}) - \mathrm{stopgrad}(a_t + \eta \boldsymbol{V}_{\theta,h}(\tilde{a}_t, \boldsymbol{s}))\|_2^2 \right]. \tag{42}$$

Since the target inside $\mathrm{stopgrad}(\cdot)$ is treated as constant during differentiation and $a_t = G_\theta(z, \boldsymbol{s})$ numerically at the current iteration,

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{drift}} &= 2\mathbb{E}_{z,\epsilon_h} \left[(\partial_\theta G_\theta)^\top (G_\theta - \mathrm{stopgrad}(a_t + \eta \boldsymbol{V}_{\theta,h})) \right] \\ &= -2\eta\, \mathbb{E}_{z,\epsilon_h} \left[(\partial_\theta G_\theta)^\top \boldsymbol{V}_{\theta,h}(\tilde{a}_t, \boldsymbol{s}) \right] \\ &= 2\eta\, \nabla_\theta \mathcal{L}_{\mathrm{RKL}}^h(\theta; \boldsymbol{s}). \end{aligned} \tag{43}$$

Therefore, for any $\eta > 0$, minimizing the drifting objective follows the same parameter-space descent direction as minimizing the KDE-smoothed reverse KLD, up to the positive scaling factor 2η.

A.2 Sequence-Level Reverse KL Divergence in VLA

We provide the full derivation of the sequence-level reverse-KL decomposition used in Section 4.1. We assume that the model and target policies share the same initial context distribution ρ and transition dynamics P, and that $\pi_\theta(\cdot \mid \boldsymbol{s}_t)$ is absolutely continuous with respect to $\pi(\cdot \mid \boldsymbol{s}_t)$ whenever an unsmoothed density ratio is used. All required exchanges of differentiation and expectation are assumed valid.

From Eq. (8), the ratio between the model and target trajectory distributions is

$$\frac{q_\theta(\tau)}{p(\tau)} = \prod_{t=1}^{T} \frac{\pi_\theta(a_t \mid \boldsymbol{s}_t)}{\pi(a_t \mid \boldsymbol{s}_t)}, \tag{44}$$

because the common factors $\rho(\boldsymbol{s}_1)$ and $P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t)$ cancel. Taking the logarithm and expectation under q_θ gives

$$\mathcal{L}_{\mathrm{RKL}}(\theta) = \mathbb{E}_{\tau\sim q_\theta}\left[\sum_{t=1}^{T} \log \frac{\pi_\theta(a_t \mid \boldsymbol{s}_t)}{\pi(a_t \mid \boldsymbol{s}_t)}\right]. \tag{45}$$

To derive the Bellman recursion, define the reverse-KL cost-to-go as

$$J_t(\boldsymbol{s}_t) = \mathbb{E}_{\tau\sim q_\theta}\left[\sum_{k=t}^{T} \log \frac{\pi_\theta(a_k \mid \boldsymbol{s}_k)}{\pi(a_k \mid \boldsymbol{s}_k)} \,\middle|\, \boldsymbol{s}_t\right], \qquad J_{T+1} \equiv 0. \tag{46}$$

Conditioning first on a_t and then on $\boldsymbol{s}_{t+1}$ yields

$$\begin{aligned} J_t(\boldsymbol{s}_t) &= \mathbb{E}_{a_t\sim\pi_\theta(\cdot|\boldsymbol{s}_t)}\left[\log \frac{\pi_\theta(a_t \mid \boldsymbol{s}_t)}{\pi(a_t \mid \boldsymbol{s}_t)} + \mathbb{E}_{\boldsymbol{s}_{t+1}\sim P(\cdot|\boldsymbol{s}_t,a_t)}[J_{t+1}(\boldsymbol{s}_{t+1})]\right] \\ &= D_{\mathrm{KL}}\left[\pi_\theta(\cdot \mid \boldsymbol{s}_t) \,\|\, \pi(\cdot \mid \boldsymbol{s}_t)\right] + \mathbb{E}_{\substack{a_t\sim\pi_\theta(\cdot|\boldsymbol{s}_t)\\ \boldsymbol{s}_{t+1}\sim P(\cdot|\boldsymbol{s}_t,a_t)}}[J_{t+1}(\boldsymbol{s}_{t+1})], \end{aligned} \tag{47}$$

which recovers Eq. (10). In particular, $\mathcal{L}_{\mathrm{RKL}}(\theta) = \mathbb{E}_{\boldsymbol{s}_1\sim\rho}[J_1(\boldsymbol{s}_1)]$.

We next derive the corresponding gradient decomposition. Let d_t^θ denote the marginal distribution of $\boldsymbol{s}_t$ induced by q_θ. Differentiating Eq. (45) gives

$$\nabla_\theta \mathcal{L}_{\mathrm{RKL}} = \sum_{t=1}^{T} \mathbb{E}_{\tau\sim q_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid \boldsymbol{s}_t) \sum_{k=t}^{T} \log \frac{\pi_\theta(a_k \mid \boldsymbol{s}_k)}{\pi(a_k \mid \boldsymbol{s}_k)}\right]. \tag{48}$$

To obtain Eq. (48), the explicit derivative of the log-ratio sum vanishes in expectation by the score identity, $\mathbb{E}_{a_t\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid \boldsymbol{s}_t)] = 0$, while terms preceding t vanish by causality. Conditioning the remaining return on $(\boldsymbol{s}_t, a_t)$ gives the current log-ratio plus the divergence-to-go

$$Q_t^\theta(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{s}_{t+1}\sim P(\cdot|\boldsymbol{s}_t,a_t)}[J_{t+1}(\boldsymbol{s}_{t+1})]. \tag{49}$$

Therefore,

$$\nabla_\theta \mathcal{L}_{\mathrm{RKL}} = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{s}_t\sim d_t^\theta}\left[\nabla_\theta \mathcal{L}_{\mathrm{Local}}(\theta; \boldsymbol{s}_t) + \nabla_\theta \mathbb{E}_{a_t\sim\pi_\theta(\cdot|\boldsymbol{s}_t)}\left[\mathrm{stopgrad}\big(Q_t^\theta(\boldsymbol{s}_t, a_t)\big)\right]\right]. \tag{50}$$

Here, Q_t^θ is evaluated at the current policy but held fixed when differentiating the contribution of a_t. The dependence of future actions on θ is accounted for by their corresponding terms at later sequence indices. Using $a_t = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$, the future component admits the pathwise form

$$\boldsymbol{g}_{\mathrm{Future}}(\theta; \boldsymbol{s}_t) := \mathbb{E}_{\boldsymbol{z}}\left[(\partial_\theta \boldsymbol{G}_\theta)^\top \nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t)\right]. \tag{51}$$

Offline context substitution. The exact gradient in Eq. (50) is evaluated under the model-induced context distribution d_t^θ, whereas DriftOPD uses contexts from the offline demonstrations. Let d_t^π denote the target-policy context distribution and let $\boldsymbol{g}_t(\boldsymbol{s}_t; \theta)$ denote the quantity inside the brackets of Eq. (50). If $\|\boldsymbol{g}_t(\boldsymbol{s}_t; \theta)\|_2 \le M$ for all t and $\boldsymbol{s}_t$, then

$$\left\|\sum_{t=1}^{T}\left(\mathbb{E}_{d_t^\theta} - \mathbb{E}_{d_t^\pi}\right)[\boldsymbol{g}_t]\right\|_2 \le M \sum_{t=1}^{T} \sqrt{2 D_{\mathrm{KL}}\left[q_\theta^{1:t} \,\|\, p^{1:t}\right]} \le TM\sqrt{2\mathcal{L}_{\mathrm{RKL}}(\theta)}, \tag{52}$$

where $q_\theta^{1:t}$ and $p^{1:t}$ denote the corresponding trajectory-prefix distributions. The first inequality follows from Pinsker's inequality and the contraction of KLD under marginalization, and the second from $D_{\mathrm{KL}}[q_\theta^{1:t}\|p^{1:t}] \le D_{\mathrm{KL}}[q_\theta\|p]$. Thus, replacing model-generated contexts by demonstration contexts introduces an occupancy error that vanishes as the model trajectory distribution approaches the target distribution, although the bound can be loose when the two policies are far apart.

A.3 Local Term and Drifting Field

We next derive the drifting update for the local term without assuming a particular kernel family. Let $K_h(a,a') = \kappa_h(a-a')$, where κ_h is a normalized, nonnegative, differentiable density. The smoothed target and model distributions in Eq. (11) can equivalently be written as convolutions, $\pi_h = \pi * \kappa_h$ and $\pi_{\theta,h} = \pi_\theta * \kappa_h$. Accordingly, if $a_t = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$ and $\boldsymbol{\epsilon}_h \sim \kappa_h$, then $\widetilde{a}_t = a_t + \boldsymbol{\epsilon}_h$ follows $\pi_{\theta,h}(\cdot \mid \boldsymbol{s}_t)$.

KDE-based score estimation. Given samples $\{a_i^p\}_{i=1}^{N_p}$ from $\pi(\cdot \mid \boldsymbol{s}_t)$ and $\{a_j^q\}_{j=1}^{N_q}$ from $\pi_\theta(\cdot \mid \boldsymbol{s}_t)$, the empirical KDEs are

$$\widehat{\pi}_h(a \mid \boldsymbol{s}_t) = \frac{1}{N_p}\sum_{i=1}^{N_p} K_h(a, a_i^p), \qquad \widehat{\pi}_{\theta,h}(a \mid \boldsymbol{s}_t) = \frac{1}{N_q}\sum_{j=1}^{N_q} K_h(a, a_j^q). \tag{53}$$

Their score estimates follow directly by differentiating the KDE and dividing by its density,

$$\nabla_a \log \widehat{\pi}_h(a \mid \boldsymbol{s}_t) = \frac{\sum_{i=1}^{N_p} \nabla_a K_h(a, a_i^p)}{\sum_{i=1}^{N_p} K_h(a, a_i^p)}, \qquad \nabla_a \log \widehat{\pi}_{\theta,h}(a \mid \boldsymbol{s}_t) = \frac{\sum_{j=1}^{N_q} \nabla_a K_h(a, a_j^q)}{\sum_{j=1}^{N_q} K_h(a, a_j^q)}. \tag{54}$$

Thus, the population field in Eq. (13) can be estimated entirely from samples. For fixed h, the empirical density and its derivative converge to their population counterparts under standard regularity conditions, and hence the score estimate converges whenever the limiting density is positive. With a vanishing bandwidth, standard KDE conditions further recover the unsmoothed score; for a Gaussian kernel in d dimensions, a sufficient condition is $h \to 0$ and $Nh^{d+2} \to \infty$ (Härdle, 2012; Silverman, 2018).

Proposition 1 (Local reverse KL as drifting, restated)**.** *Let $a_t = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$ and $\widetilde{a}_t = a_t + \boldsymbol{\epsilon}_h$, where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{\epsilon}_h \sim \kappa_h$ are independent. Define*

$$\mathcal{L}_{\mathrm{drift}}(\theta; \boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{z},\boldsymbol{\epsilon}_h}\left[\left\|\boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t) - \mathrm{stopgrad}\left(a_t + \eta \boldsymbol{V}_{\theta,h}(\widetilde{a}_t, \boldsymbol{s}_t)\right)\right\|_2^2\right]. \tag{55}$$

Then, for any $\eta > 0$,

$$\nabla_\theta \mathcal{L}^h_{\mathrm{Local}}(\theta; \boldsymbol{s}_t) = \frac{1}{2\eta}\nabla_\theta \mathcal{L}_{\mathrm{drift}}(\theta; \boldsymbol{s}_t). \tag{56}$$

Proof. Since $\widetilde{a}_t = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t) + \boldsymbol{\epsilon}_h$ follows $\pi_{\theta,h}(\cdot \mid \boldsymbol{s}_t)$,

$$\mathcal{L}^h_{\mathrm{Local}} = \mathbb{E}_{\boldsymbol{z},\boldsymbol{\epsilon}_h}\left[\log \pi_{\theta,h}(\widetilde{a}_t \mid \boldsymbol{s}_t) - \log \pi_h(\widetilde{a}_t \mid \boldsymbol{s}_t)\right]. \tag{57}$$

Differentiating with respect to θ gives

$$\begin{aligned}\nabla_\theta \mathcal{L}^h_{\mathrm{Local}} &= \mathbb{E}_{\boldsymbol{z},\boldsymbol{\epsilon}_h}\left[\partial_\theta \log \pi_{\theta,h}(\widetilde{a}_t \mid \boldsymbol{s}_t)\right] + \mathbb{E}_{\boldsymbol{z},\boldsymbol{\epsilon}_h}\left[(\partial_\theta \boldsymbol{G}_\theta)^\top \nabla_{\widetilde{a}_t} \log \frac{\pi_{\theta,h}(\widetilde{a}_t \mid \boldsymbol{s}_t)}{\pi_h(\widetilde{a}_t \mid \boldsymbol{s}_t)}\right] \\ &= -\mathbb{E}_{\boldsymbol{z},\boldsymbol{\epsilon}_h}\left[(\partial_\theta \boldsymbol{G}_\theta)^\top \boldsymbol{V}_{\theta,h}(\widetilde{a}_t, \boldsymbol{s}_t)\right].\end{aligned} \tag{58}$$

The first term vanishes by the score identity,

$$\mathbb{E}_{\widetilde{a}_t \sim \pi_{\theta,h}}\left[\partial_\theta \log \pi_{\theta,h}(\widetilde{a}_t \mid \boldsymbol{s}_t)\right] = \int \partial_\theta \pi_{\theta,h}(a \mid \boldsymbol{s}_t)\, da = 0. \tag{59}$$

Now define $\widehat{a}_t = a_t + \eta \boldsymbol{V}_{\theta,h}(\widetilde{a}_t, \boldsymbol{s}_t)$. Since $\widehat{a}_t$ is treated as a stop-gradient target,

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\mathrm{drift}} &= 2\mathbb{E}_{\boldsymbol{z},\boldsymbol{\epsilon}_h}\left[(\partial_\theta \boldsymbol{G}_\theta)^\top \left(\boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t) - \mathrm{stopgrad}(\widehat{a}_t)\right)\right] \\ &= -2\eta \mathbb{E}_{\boldsymbol{z},\boldsymbol{\epsilon}_h}\left[(\partial_\theta \boldsymbol{G}_\theta)^\top \boldsymbol{V}_{\theta,h}(\widetilde{a}_t, \boldsymbol{s}_t)\right].\end{aligned} \tag{60}$$

Comparing Eq. (58) and Eq. (60) proves Eq. (56). The derivation uses only that κ_h is a differentiable probability kernel and does not require a Gaussian kernel. □

Single-positive target. For a demonstration context $\boldsymbol{s}_t$, only one target action chunk a_t^p is observed. Replacing the target conditional by $\delta_{a_t^p}$ gives $\pi_h(a \mid \boldsymbol{s}_t) = K_h(a, a_t^p)$ and

$$\boldsymbol{V}_{\theta,h}(a, \boldsymbol{s}_t) = \nabla_a \log K_h(a, a_t^p) - \frac{\mathbb{E}_{a^q \sim \pi_\theta(\cdot|\boldsymbol{s}_t)}[\nabla_a K_h(a, a^q)]}{\mathbb{E}_{a^q \sim \pi_\theta(\cdot|\boldsymbol{s}_t)}[K_h(a, a^q)]}, \tag{61}$$

which recovers Eq. (17). The first term attracts the generated action toward the demonstrated action, while the second retains the model-distribution score required by reverse-KL optimization.

The single-positive construction also has a direct relation to the full smoothed target distribution. Assume $K_h(a, a') > 0$ and define

$$\mathcal{U}_h(\theta; \boldsymbol{s}_t) = \mathbb{E}_{a_t^p \sim \pi(\cdot|\boldsymbol{s}_t)} \left[D_{\mathrm{KL}} \left[\pi_{\theta,h}(\cdot \mid \boldsymbol{s}_t) \,\|\, K_h(\cdot, a_t^p) \right] \right]. \tag{62}$$

Since $\pi_h(a \mid \boldsymbol{s}_t) = \mathbb{E}_{a_t^p}[K_h(a, a_t^p)]$,

$$\mathcal{U}_h - \mathcal{L}^h_{\mathrm{Local}} = \mathbb{E}_{a \sim \pi_{\theta,h}} \left[\log \mathbb{E}_{a_t^p} K_h(a, a_t^p) - \mathbb{E}_{a_t^p} \log K_h(a, a_t^p) \right] \geq 0, \tag{63}$$

where the inequality follows from Jensen's inequality. Thus, averaging the single-positive objective over demonstration actions upper-bounds the reverse KLD to the full kernel-smoothed target distribution. In particular, the gap vanishes when the target conditional is deterministic.

Finite-sample implementation. Proposition A.3 is an exact population identity when the field is evaluated at the kernel-perturbed sample $\widetilde{a}_t$ using the population scores. In practice, Eq. (54) is estimated from a finite set of policy samples, and Algorithm 1 evaluates the resulting field at the generated action. These introduce, respectively, finite-sample KDE error and an evaluation-point approximation. Under standard KDE regularity conditions the former vanishes as the number of policy samples increases, while the latter vanishes as the kernel bandwidth tends to zero for a continuous drifting field.

A.4 Future Term and Critic Surrogate

We now show that the future gradient contribution can be represented exactly by an ideal frozen action-value function, and then characterize the approximation error introduced by a practical learned critic. Recall

$$Q_t^\theta(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{s}_{t+1} \sim P(\cdot|\boldsymbol{s}_t, a_t)} \left[J_{t+1}(\boldsymbol{s}_{t+1}) \right]. \tag{64}$$

When computing the contribution at sequence index t, Q_t^θ is evaluated under the current policy but treated as fixed with respect to θ, as in Eq. (50).

Structure of the future action gradient. Assume that $P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t)$ admits a density differentiable with respect to a_t. Differentiating under the integral gives

$$\nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{s}_{t+1} \sim P(\cdot|\boldsymbol{s}_t, a_t)} \left[J_{t+1}(\boldsymbol{s}_{t+1}) \nabla_{a_t} \log P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t) \right]. \tag{65}$$

Define

$$\begin{aligned} I_P(\boldsymbol{s}_t, a_t) &= \mathbb{E}_{\boldsymbol{s}_{t+1} \sim P(\cdot|\boldsymbol{s}_t, a_t)} \left[\|\nabla_{a_t} \log P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t)\|_2^2 \right], \\ M_{t+1}(\boldsymbol{s}_t, a_t) &= \mathbb{E}_{\boldsymbol{s}_{t+1} \sim P(\cdot|\boldsymbol{s}_t, a_t)} \left[J_{t+1}(\boldsymbol{s}_{t+1})^2 \right]. \end{aligned} \tag{66}$$

By Cauchy–Schwarz,

$$\left\| \nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t) \right\|_2 \leq \sqrt{I_P(\boldsymbol{s}_t, a_t) M_{t+1}(\boldsymbol{s}_t, a_t)}. \tag{67}$$

If

$$\overline{D}(\theta) = \sup_{\boldsymbol{s}} \mathcal{L}_{\mathrm{Local}}(\theta; \boldsymbol{s}) < \infty, \tag{68}$$

then $0 \leq J_{t+1}(\boldsymbol{s}_{t+1}) \leq (T-t)\overline{D}(\theta)$, and therefore

$$\left\| \nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t) \right\|_2 \leq (T-t)\overline{D}(\theta) \sqrt{I_P(\boldsymbol{s}_t, a_t)}. \tag{69}$$

Hence, the future contribution can be significant only when the current action changes the successor distribution and non-negligible policy discrepancy remains in the future. In particular, the exact future contribution vanishes as the remaining local discrepancy vanishes.

Ideal frozen critic. Using $a_t = \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t)$, the exact future gradient contribution is

$$\boldsymbol{g}_{\text{Future}}(\theta; \boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{z}} \left[(\partial_\theta \boldsymbol{G}_\theta)^\top \nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t) \right]. \tag{70}$$

Consider a reference policy $\pi_{\bar{\theta}}$ and define the ideal frozen critic

$$Q_{t,\bar{\theta}}^{\text{ideal}}(\boldsymbol{s}_t, a_t) = -Q_t^{\bar{\theta}}(\boldsymbol{s}_t, a_t) + b_t(\boldsymbol{s}_t), \tag{71}$$

where $\bar{\theta}$ is fixed and $b_t(\boldsymbol{s}_t)$ is any action-independent function. Importantly, $Q_{t,\bar{\theta}}^{\text{ideal}}$ is treated as a fixed function and has no differentiable dependence on the optimized policy parameters θ.

Define

$$\mathcal{L}_{\text{Critic}}^{\text{ideal}}(\theta; \boldsymbol{s}_t) = -\mathbb{E}_{\boldsymbol{z}} \left[Q_{t,\bar{\theta}}^{\text{ideal}} \left(\boldsymbol{s}_t, \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t) \right) \right]. \tag{72}$$

Its gradient is

$$\nabla_\theta \mathcal{L}_{\text{Critic}}^{\text{ideal}} = -\mathbb{E}_{\boldsymbol{z}} \left[(\partial_\theta \boldsymbol{G}_\theta)^\top \nabla_{a_t} Q_{t,\bar{\theta}}^{\text{ideal}}(\boldsymbol{s}_t, a_t) \right]. \tag{73}$$

Since

$$\nabla_{a_t} Q_{t,\bar{\theta}}^{\text{ideal}}(\boldsymbol{s}_t, a_t) = -\nabla_{a_t} Q_t^{\bar{\theta}}(\boldsymbol{s}_t, a_t), \tag{74}$$

evaluating Eq. (73) at $\theta = \bar{\theta}$ yields

$$\nabla_\theta \mathcal{L}_{\text{Critic}}^{\text{ideal}}(\theta; \boldsymbol{s}_t)\big|_{\theta=\bar{\theta}} = \boldsymbol{g}_{\text{Future}}(\bar{\theta}; \boldsymbol{s}_t). \tag{75}$$

Thus, an ideal frozen critic exactly realizes the future gradient contribution at the reference policy. The scalar critic value itself need not match $-Q_t^{\bar{\theta}}$: any action-independent offset $b_t(\boldsymbol{s}_t)$ leaves the policy update unchanged. This establishes that the critic term is not an auxiliary regularizer in the ideal case, but an exact frozen representation of the sequence-level future contribution.

Learned critic and approximation error. In practice, the ideal critic in Eq. (71) is unavailable. DriftOPD therefore uses a frozen learned critic Q_ϕ, for which larger values indicate more desirable actions,

$$\mathcal{L}_{\text{Critic}}(\theta; \boldsymbol{s}_t) = -\mathbb{E}_{\boldsymbol{z}} \left[Q_\phi \left(\boldsymbol{s}_t, \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t) \right) \right], \tag{76}$$

with

$$\nabla_\theta \mathcal{L}_{\text{Critic}} = -\mathbb{E}_{\boldsymbol{z}} \left[(\partial_\theta \boldsymbol{G}_\theta)^\top \nabla_{a_t} Q_\phi(\boldsymbol{s}_t, a_t) \right]. \tag{77}$$

At any current policy θ, define the action-gradient mismatch

$$\Delta_t^\phi(\boldsymbol{s}_t, a_t) = \nabla_{a_t} Q_t^\theta(\boldsymbol{s}_t, a_t) + \nabla_{a_t} Q_\phi(\boldsymbol{s}_t, a_t), \tag{78}$$

and

$$\epsilon_t(\phi; \boldsymbol{s}_t) = \left(\mathbb{E}_{\boldsymbol{z}} \left[\left\| \Delta_t^\phi(\boldsymbol{s}_t, a_t) \right\|_2^2 \right] \right)^{1/2}. \tag{79}$$

If

$$\left\| \partial_\theta \boldsymbol{G}_\theta(\boldsymbol{z}, \boldsymbol{s}_t) \right\|_{\text{op}} \le L_G, \tag{80}$$

then

$$\begin{aligned} \left\| \boldsymbol{g}_{\text{Future}}(\theta; \boldsymbol{s}_t) - \nabla_\theta \mathcal{L}_{\text{Critic}}(\theta; \boldsymbol{s}_t) \right\|_2 &= \left\| \mathbb{E}_{\boldsymbol{z}} \left[(\partial_\theta \boldsymbol{G}_\theta)^\top \Delta_t^\phi(\boldsymbol{s}_t, a_t) \right] \right\|_2 \\ &\le L_G \, \epsilon_t(\phi; \boldsymbol{s}_t). \end{aligned} \tag{81}$$

Therefore, the practical critic differs from the ideal frozen surrogate only through its action-gradient mismatch on model-generated action chunks. Eq. (81) characterizes this approximation without requiring the learned critic to recover the absolute value of the future divergence, or assuming that $\epsilon_t(\phi; \boldsymbol{s}_t)$ is necessarily small. The particular demonstration-trained critic used in our experiments is one practical instantiation of this frozen surrogate.

B EXPERIMENT PROTOCOL

B.1 MODEL CONFIGURATIONS

Table 4: Model configurations used in our experiments

Model	Benchmark	Action objective	Distillation trainable parameters	Action dim.	Chunk length	Teacher steps	One-step parameterization
$\pi_{0.5}$	RoboCasa365	Flow matching	Action expert 693.4M	32	50	10	$z - v(z, t{=}1)$
$\pi_{0.5}$-LoRA	RoboCasa365	Flow matching	LoRA + input/output proj. + timestep MLP, 9.1M	32	50	10	$z - v(z, t{=}1)$
	LIBERO				10		
	RoboTwin 2.0				50		
GR00T N1.5	RoboCasa365	Flow matching	Action-generation head 1,068.8M	32	16	4	$z + v(z, t{=}0)$
GR00T N1.6	RoboCasa365	Flow matching	Action-generation head 1,418.6M	128	50	4	$z + v(z, t{=}0)$
ABot-M0	RoboTwin 2.0	Direct clean action	Action-generation head 164.2M	14	50	10	Direct x_0
	LIBERO				10		
SmolVLA	LIBERO	Flow matching	Action-generation head 97.5M	32	50	10	$z - v(z, t{=}1)$

Table 5: Starting checkpoints used for each model and benchmark

Model	Benchmark / setting	Starting checkpoint
$\pi_{0.5}$	RoboCasa365	`robocasa/robocasa365_checkpoints`
	LIBERO	`gs://openpi-assets/checkpoints/pi05_libero/`
	RoboTwin 2.0	`gs://openpi-assets/checkpoints/pi05_base/params` (fine-tuned by us for 75k steps with batch size 64 on `TianxingChen/RoboTwin2.0`, 50 tasks)
GR00T N1.5	RoboCasa365	`robocasa/robocasa365_checkpoints`
GR00T N1.6	RoboCasa365	`Abhi03/grootn16_robocasa365_multitask_learning`
	Real world	`nvidia/GR00T-N1.6-3B`
ABot-M0	LIBERO	`acvlab/ABot-M0-LIBERO`
	RoboTwin 2.0	`acvlab/ABot-M0-RoboTwin2`
SmolVLA	LIBERO	`HuggingFaceVLA/smolvla_libero`

Model configurations. Tables 4 and 5 summarize the policy configurations and starting checkpoints used throughout our experiments. Depending on the benchmark, we either use a released task-specific checkpoint directly or fine-tune a base checkpoint before distillation, as detailed in Table 5. Each one-step student is initialized from its corresponding teacher checkpoint, and during distillation only the parameters listed in Table 4 are updated while the remaining policy parameters are kept frozen.

LoRA fine-tuning. For $\pi_{0.5}$-LoRA, we insert LoRA (Hu et al., 2022) modules with $r = 16$ and $\alpha = 16$ into the query, key, value, output, gate, up, and down projections of the action-expert transformer. This replaces 126 linear layers with LoRA-augmented layers and contributes 6.93M trainable LoRA parameters. The action input/output projections and timestep MLP remain fully trainable, resulting in 9.1M trainable parameters in total. We use no LoRA dropout or bias term, and the LoRA scaling is $\alpha/r = 1$.

B.2 DRIFTOPD TRAINING DETAILS

Table 6: Common student optimization settings

Hyperparameter	Setting
Training steps	10,000
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$
Learning rate	$3 \times 10^{-5} \rightarrow 3 \times 10^{-7}$
LR schedule	Linear decay; no learning-rate warmup
Weight decay	0
Gradient clipping	1.0
Precision	bfloat16 autocast with FP32 master weights
EMA	Maximum decay 0.9995 with 1,000-step EMA warmup
Initialization	Full copy of the corresponding starting policy checkpoint
Default training seed	42
Noise	$z \sim \mathcal{N}(0, I)$, resampled at every microstep
Antithetic sampling	Disabled
Noise scale	1.0

One-step distillation settings. The common optimization settings used for DriftOPD and the one-step baselines are summarized in Table 6. Unless otherwise specified, all students are trained for 10k optimization steps and evaluated using their EMA parameters.

Table 7: DriftOPD-specific hyperparameters

Hyperparameter	Setting
Kernel	Laplacian, $K_R(a, a') \propto \exp(-\|a - a'\|/R)$
Bandwidths	$R \in \{0.02, 0.05, 0.2\}$
Current-policy samples M	64 by default (8 gradient-carrying + 56 detached) 32 (8 gradient-carrying + 24 detached) on RoboCasa365 for $\pi_{0.5}$
Positive sample	One aligned demonstration action, separate from the M policy samples
Default KDE pool	64 policy samples + 1 demonstration sample
Input perturbation	Laplace noise ϵ_R with scale R on generated policy samples
Drifting step η	1.0 for all bandwidths
Force normalization	Current-step global RMS over batch, generated-sample, and flattened action dimensions
RMS floor	10^{-8} before the square root
Multi-band aggregation	Compute one normalized loss per bandwidth and average the losses
λ	0.01, fixed
Critic warmup	linear warmup over the first 500 steps
Critic gradient normalization	Drift-to-critic gradient-norm ratio with EMA 0.99
Gradient-ratio range	$[0.1, 10]$
Simulation Q clipping	Disabled
Real-world Q clipping	Hard clip when generated-action Q exceeds demonstration Q

DriftOPD-specific settings. Table 7 summarizes the implementation settings specific to the drifting and critic objectives. Following prior work (Deng et al., 2026; Zhang et al., 2026), we use a Laplacian affinity kernel and the three-bandwidth set $R \in \{0.02, 0.05, 0.2\}$. As in Zhang et al. (2026), the policy-sample set is built from both gradient-carrying and detached generations. M counts current-policy samples only, of which eight retain gradients and the remainder are detached, and the aligned demonstration action enters the kernel pool as the single positive. We use $M = 64$ (8 gradient-carrying + 56 detached) by default, giving a kernel pool of 65 samples. On RoboCasa365, $\pi_{0.5}$ students instead use $M = 32$ (8 + 24), i.e. a pool of 33.

Details for the drifting field. Before the kernel is evaluated, Laplace noise ϵ_R with scale R is added to the generated and detached policy samples (Zhang et al., 2026), so the drifting force is evaluated at perturbed actions $\tilde{a} = a + \epsilon_R$. The demonstration positive is left unperturbed, and the same noise realization enters the regression target of the one-step student. For each bandwidth, the drifting force is normalized using statistics from the current optimization step rather than a running RMS:

$$\widehat{\boldsymbol{V}}_R = \frac{\boldsymbol{V}_R}{\sqrt{\max(\mathbb{E}_{b,c,s}[\boldsymbol{V}_R^2], 10^{-8})}}. \tag{82}$$

Here, b, c, and s index the conditioning batch, generated policy samples, and flattened action coordinates, respectively. The drifting loss is computed independently for each bandwidth and then averaged across the three bandwidths. Normalizing each bandwidth before averaging keeps the target displacement independent of the number of bandwidths, whereas summing the normalized fields would scale it with $|R|$ and destabilize early training.

Implementation details for the critic loss. The critic term uses a base weight $\lambda = 0.01$ with no decay schedule, ramped linearly from zero over the first 500 optimization steps so that the critic does not steer the student while its one-step outputs are still far from the data manifold. To keep the two terms commensurable, the critic weight is rescaled by the ratio of gradient norms measured at the generated actions, $r = \|\nabla_a \mathcal{L}_{\text{Drift}}\|_2 / \|\nabla_a Q_\phi\|_2$, clipped to $[0.1, 10]$ and smoothed by an exponential moving average with coefficient 0.99. The effective weight is $\lambda_{\text{eff}}(t) = \lambda\, w(t)\, \bar{r}(t)$, where $w(t)$ denotes the warmup factor. The critic-weight analysis in Figure 3 evaluates $\lambda \in \{0, 0.001, 0.01, 0.1, 0.3, 1.0\}$ while holding all other optimization and stabilization settings fixed, and the final simulation experiments use $\lambda = 0.01$. No trust region or clipping of the critic score is applied in simulation. The real-world runs additionally hard-clip the critic term when the generated-action Q_ϕ exceeds the demonstration-action Q_ϕ.

Table 8: Model-specific batch configurations. N_c denotes the number of task contexts per optimization step, and batch size denotes the number of gradient-carrying policy samples per context. Students evaluated on LIBERO-Plus are the LIBERO students, so they share the LIBERO configuration.

Benchmark	Model	N_c (# tasks)	Batch size	Gradient-carrying samples / step
RoboCasa365	$\pi_{0.5}$ (full fine-tuning)	4	8	32
	$\pi_{0.5}$ (LoRA r=16)	4	8	32
	GR00T N1.5	16	8	128
	GR00T N1.6	16	8	128
RoboTwin 2.0	$\pi_{0.5}$ (LoRA r=16)	4	8	32
	ABot-M0	64	8	512
LIBERO / LIBERO-Plus	$\pi_{0.5}$ (LoRA r=16)	4	8	32
	ABot-M0	64	8	512
	SmolVLA	16	8	128

Mini-batch configurations. Table 8 summarizes the batch configuration for each model and benchmark. Every optimization step draws N_c task contexts and generates eight gradient-carrying one-step chunks per context, so a step is backpropagated through $8N_c$ policy samples. We keep the per-context count at eight everywhere and vary only N_c, which ranges from 4 for $\pi_{0.5}$ to 64 for ABot-M0, because the drifting loss is a per-task quantity and widening N_c marginalizes over more of the task distribution per step. Within a given model and benchmark all four students share the same N_c and batch size, so sCD, MFD, Drift, and DriftOPD see an identical sample budget and any gap between them reflects the objective alone. Each context also supplies one cached ground-truth chunk and 56 or 24 detached negatives, which shape the loss but carry no gradient and are excluded from the count above. We realize N_c on a single device as the per-microstep task count times the number of gradient accumulation steps, holding the product fixed across students, so no multi-GPU training is needed to reproduce these settings. Students evaluated on LIBERO-Plus are the LIBERO students and inherit the LIBERO configuration.

B.3 CRITIC TRAINING

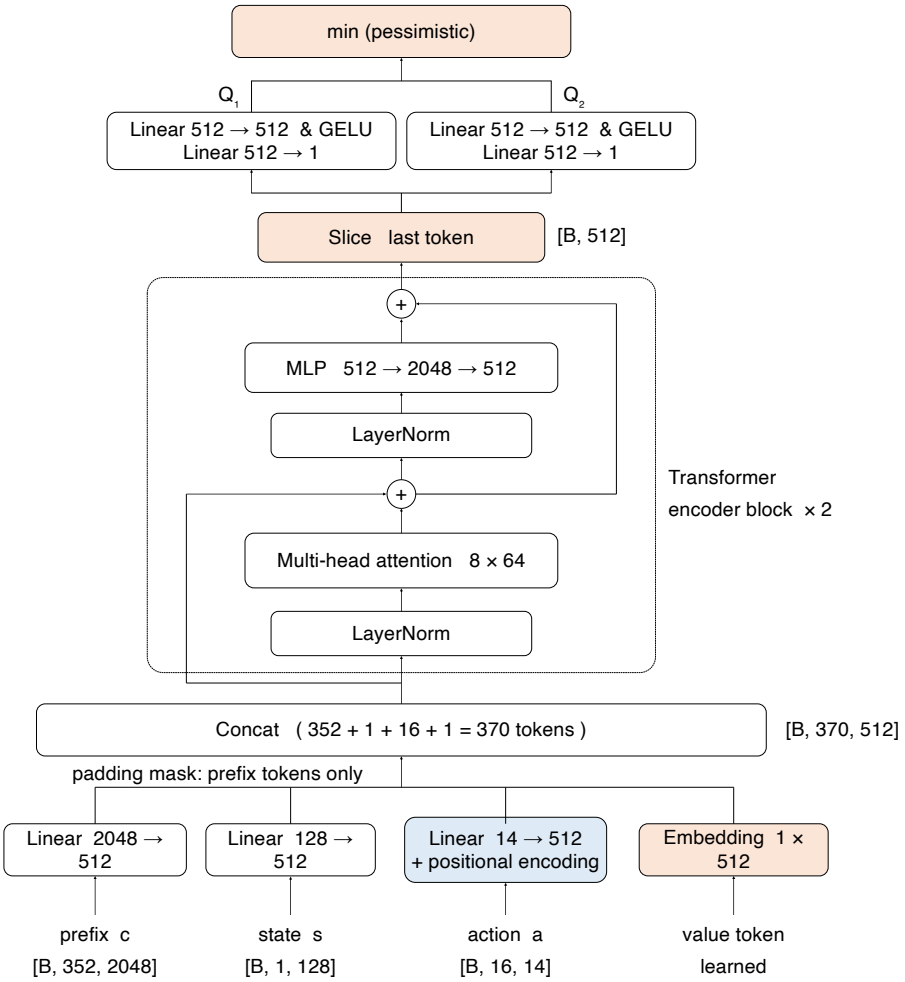


Figure 5: Architecture of the demonstration-trained critic used in DriftOPD. The frozen VLA prefix, the robot state, and the candidate action chunk are encoded jointly, and a learned value token is read out by twin Q-heads.

Overview. DriftOPD queries the critic for an action gradient rather than an absolute value, so the critic must vary informatively with the action chunk in a neighborhood of demonstrated behavior. We therefore build on the Q-chunk-Former formulation of VGAS (Xu et al., 2026) but adapt both the architecture and the training objective to that requirement. We do not reuse a released VGAS critic. Instead we train one critic per policy backbone and embodiment from offline demonstrations alone, so that the critic reads the same frozen prefix the student is distilled against. As shown in Figure 5, the critic consumes the frozen VLA prefix together with the robot state and an action chunk, encodes them with a lightweight bidirectional Transformer, and reads an action-conditioned value out of a learned value token through twin Q-heads. The critic is trained once and then frozen for the whole of policy distillation. Table 9 lists the architecture and optimization settings in full.

Training objective. Each training sample is a triple $(s_t, a_t, R_t) \sim \mathcal{D}$, where s_t collects the frozen VLA prefix tokens, the robot state, and the language instruction, a_t is the demonstrated action chunk executed at s_t, and R_t is its Monte-Carlo return. The two Q-heads are combined pessimistically,

$$Q_\phi(s_t, a_t) = \min\big(Q_1(s_t, a_t), Q_2(s_t, a_t)\big), \tag{83}$$

and the objective pairs a value-regression term with two terms that induce action dependence,

$$\mathcal{L}_{\text{Critic}}(\phi) = \mathcal{L}_{\text{MC}}(\phi) + \alpha\, \mathcal{L}_{\text{Cal-QL}}(\phi) + \mathcal{L}_{\text{EGR}}(\phi), \qquad \alpha = 1.0. \tag{84}$$

Table 9: Architecture and optimization settings of the demonstration-trained critic.

Component	Setting
Input	Frozen VLA prefix tokens + state token + action-chunk tokens + learned value token
Encoder	2-layer pre-norm bidirectional Transformer encoder
Hidden dimension	512
Attention heads	8
FFN dimension	2048
Dropout	0
Q-heads	Two independent MLP heads; $\min(Q_1, Q_2)$ used for policy optimization
Q-head MLP	$512 \to 512 \to 1$ with GELU and LayerNorm
Training data	Offline demonstrations only
Sampling	Task-balanced uniform sampling over tasks, then a uniform row within the sampled task
Held-out split	Simulation: tasks satisfying `task_index % 10 == 0` excluded from critic training Real world: episodes satisfying `episode_index % 10 == 0` excluded from critic training
Target	Monte-Carlo return
Discount factor	$\gamma = 0.998$
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$
Learning rate	10^{-4}
Weight decay	10^{-6}
Batch size	64
Training steps	20,000
LR schedule	1,000-step warmup followed by cosine decay
Gradient clipping	10
Critic seed	0
Q-output normalization	None
Objective	MC regression + Cal-QL + EGR
Cal-QL	Coefficient 1.0, eight OOD action candidates
OOD candidates	Four local perturbations with $\sigma = 0.3$ + four marginal samples
EGR	Anchor weight 1.0, ranking weight 5.0
EGR scales	β and $d_{\max}$ automatically calibrated from each cache

Monte-Carlo regression. Our demonstration caches retain only a few frames per episode, so consecutive chunk pairs (s_t, s_{t+H}) are rarely both present and temporal-difference bootstrapping is impractical. All demonstrations are successful and the reward is defined by frame position, so the return is computed directly from the episode and used as a regression target,

$$\mathcal{L}_{\mathrm{MC}}(\phi) = \mathbb{E}_{(s_t,a_t,R_t)\sim\mathcal{D}}\left[\left(Q_\phi(s_t,a_t) - R_t\right)^2\right], \qquad R_t = \sum_{k\geq 0}\gamma^k r_{t+k}, \quad \gamma = 0.998. \tag{85}$$

No target network or bootstrapping is used, so the critic is trained by pure regression onto returns available from the recorded episodes.

Negative candidates. Monte-Carlo regression alone constrains the critic only at demonstrated state–action pairs, which leaves its local variation in the action argument unsupervised. For each sample we therefore draw eight out-of-distribution (OOD) action chunks from an equal mixture of local and marginal candidates,

$$a' \sim \nu(\cdot \mid s_t, a_t) = \frac{1}{2}\underbrace{\mathcal{N}\big(a_t, \sigma^2 I\big)}_{\text{local, } \sigma=0.3} + \frac{1}{2}\underbrace{\mathcal{D}_a(\cdot)}_{\text{marginal}}, \tag{86}$$

where $\mathcal{D}_a$ is the empirical distribution of demonstrated action chunks recorded at other observations, and four candidates are drawn from each component. The local component populates a neighbor-

hood of the demonstrated chunk, while the marginal component supplies chunks that are plausible in the dataset but mismatched to the current context.

Action dependence. The first of the two candidate-based terms is a conservative objective in the style of Cal-QL (Nakamoto et al., 2023), which suppresses OOD action values while calibrating them against the Monte-Carlo return,

$$\mathcal{L}_{\text{Cal-QL}}(\phi) = \mathbb{E}_{s_t}\left[\log \sum_{a' \sim \nu} \exp\left(\max\left(Q_\phi(s_t, a'), R_t\right)\right) - Q_\phi(s_t, a_t)\right]. \tag{87}$$

The second, EGR, adds distance-aware supervision so that the critic varies smoothly with the distance from the demonstrated chunk,

$$\mathcal{L}_{\text{EGR}}(\phi) = w_{\text{anchor}}\, \ell_{\text{anchor}}(\phi) + w_{\text{rank}}\, \ell_{\text{rank}}(\phi), \qquad w_{\text{anchor}} = 1.0, \quad w_{\text{rank}} = 5.0. \tag{88}$$

Here ℓ_{anchor} ties the value of a candidate at distance $d(a', a_t)$ to the margin-shifted anchor $Q_\phi(s_t, a_t) - \beta\, d(a', a_t)/d_{\max}$, while ℓ_{rank} penalizes candidate pairs whose predicted ordering disagrees with their distance ordering, encouraging $Q_\phi(s_t, a'_i) > Q_\phi(s_t, a'_j)$ whenever $d(a'_i, a_t) < d(a'_j, a_t)$. The scale β and the distance normalization $d_{\max}$ are calibrated from each critic cache rather than fixed across embodiments, since the action spaces differ in magnitude.

Together, Eq. (87) and Eq. (88) make Q_ϕ action-dependent in a neighborhood of demonstrated behavior. This is the property required by the critic surrogate in Section 4.3, which consumes the action gradient $\nabla_{a_t} Q_\phi(s_t, a_t)$ rather than the absolute critic value.

B.4 Baseline Implementations

All one-step baselines start from the same teacher checkpoint, read the same cached demonstrations and conditioning, and use the optimizer and effective context batch reported in Table 6 and 8. We follow the hyperparameters of the original methods where they are specified and run no baseline-specific sweep. The implemented settings are listed in Table 10.

Table 10: Implementation details of the one-step distillation baselines.

Method	Implementation
One-step Teacher	Direct one-step generation from the corresponding teacher checkpoint without additional distillation. For ABot-M0, the baseline uses its dedicated one-step generator.
sCD	$P_{\text{mean}} = -1.0$, $P_{\text{std}} = 1.4$, $\sigma_d = 1$, tangent normalization $c = 0.1$, JVP tangent estimation with central finite-difference fallback $\epsilon = 10^{-3}$, and adaptive-weighting learning rate 10^{-4}.
MFD	Maximum step size 0.02, solver-step cap 32, minimum interval 0.02. TTUR ratio $K = 10$ with logit-normal time sampling on LIBERO and RoboTwin 2.0; $K = 50$ with uniform time sampling on RoboCasa365.
Drift	Identical kernel, bandwidth, perturbation, normalization, sampling and student-optimization settings as DriftOPD, with the critic term disabled.
DriftOPD	Drift objective combined with the frozen demonstration-trained critic using the settings in Tables 7 and 9.

One-step Teacher. The teacher is evaluated with a single generation step and no additional distillation, which isolates how much of the one-step gap is already closed by the pretrained flow head. For ABot-M0 this baseline uses the dedicated one-step generator shipped with the backbone rather than a truncated sampler.

sCD. We use the distillation form of the continuous-time consistency model (Lu & Song, 2025), in which the probability-flow ODE is supplied by the frozen pretrained teacher F^{Q} rather than by an unbiased noise estimator. With $\sigma_d = 1$, fixed by the teachers having been trained with $\epsilon \sim \mathcal{N}(0, I)$, each step draws $x_0 \sim \mathcal{D}$, $z \sim \mathcal{N}(0, \sigma_d^2 I)$ and a time from the log-normal proposal

$$\tau \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2), \qquad t = \arctan\left(e^{\tau}/\sigma_d\right), \qquad x_t = \cos(t)\, x_0 + \sin(t)\, z, \tag{89}$$

with $P_{\text{mean}} = -1.0$ and $P_{\text{std}} = 1.4$. The tangent that the consistency condition requires is

$$g = -\cos^2(t)\Big(\sigma_d F_{\theta^-} - \tfrac{\mathrm{d}x_t}{\mathrm{d}t}\Big) - r\,\cos(t)\sin(t)\Big(x_t + \sigma_d\,\tfrac{\mathrm{d}F_{\theta^-}}{\mathrm{d}t}\Big), \qquad \tfrac{\mathrm{d}x_t}{\mathrm{d}t} = \sigma_d F^{\mathrm{Q}}\big(x_t/\sigma_d, t\big), \quad (90)$$

where $r \in [0,1]$ is a linear warmup factor. The term $\mathrm{d}F_{\theta^-}/\mathrm{d}t$ is a directional derivative in time and is obtained with a Jacobian-vector product; since sCM removes the EMA target and sets $\theta^- = \mathrm{sg}[\theta]$, the primal output of that JVP is exactly F_{θ^-}, so no separate forward pass is needed for it. If forward-mode automatic differentiation is unavailable for a backbone, the implementation falls back permanently to central finite differences with $\epsilon = 10^{-3}$. The tangent is normalized as $g \leftarrow g/(\|g\| + c)$ with $c = 0.1$, and the loss uses the adaptive weighting w_φ of the original method,

$$\mathcal{L}_{\text{sCD}}(\theta, \varphi) = \frac{e^{w_\varphi(t)}}{D}\big\|F_\theta - F_{\theta^-} - g\big\|^2 - w_\varphi(t), \quad (91)$$

with D the action-chunk dimension and w_φ trained at learning rate 10^{-4}.

MFD. Mean Flow Distillation (Zhao et al., 2026a) is data-free, so the demonstration bank that Drift attracts toward is unused and only the teacher, the conditioning and the step budget are shared. Three heads sit on the frozen backbone: the frozen teacher u^{Q}, the student generator G_θ, and an auxiliary flow v_ϕ that tracks the student's own output distribution. Each outer iteration performs K auxiliary updates followed by one student update. The auxiliary head is fit by flow matching on student samples,

$$\mathcal{L}_{\text{aux}}(\phi) = \mathbb{E}\Big[\big\|v_\phi(X_\tau, \tau) - (X_1 - X_0')\big\|^2\Big], \quad X_1 = \mathrm{sg}[G_\theta(X_0)], \quad X_\tau = (1-\tau)X_0' + \tau X_1, \quad (92)$$

with $\tau \sim \mathcal{U}[0,1]$ and X_0, X_0' independent noise draws. The two-timescale ratio K keeps v_ϕ an accurate model of the student distribution before each student step. For the student update an interval $0 \le s \le t \le 1$ is drawn, the state $X_s = s\,G_\theta(X_0) + (1-s)X_0'$ is formed, and both flows are integrated from X_s over $[s,t]$ to give mean velocities

$$\hat{U}^{\mathrm{P}} = \frac{\Phi(v_\phi, X_s, s, t) - X_s}{t - s}, \qquad \hat{U}^{\mathrm{Q}} = \frac{\Phi(u^{\mathrm{Q}}, X_s, s, t) - X_s}{t - s}, \qquad \Delta = \hat{U}^{\mathrm{P}} - \hat{U}^{\mathrm{Q}}, \quad (93)$$

where Φ denotes the ODE solver. Following the original derivation we drop the Jacobian of the solver with respect to its input, which gives the student gradient

$$\nabla_\theta \mathcal{L}_{\text{MFD}} = \mathbb{E}\Big[s\,\big(\nabla_\theta G_\theta(X_0)\big)^\top \Delta\Big]. \quad (94)$$

We implement Eq. (94) as the surrogate $\mathcal{L} = \langle \mathrm{sg}[\Delta], X_s \rangle$, whose autograd gradient is exactly the expression above. A mean-squared error on Δ would instead backpropagate through the solver and optimize a different objective.

Drift. Drift is the drifting distillation objective of Zhang et al. (2026), applied to action chunks. Generated samples are pushed by a drift field built from an affinity kernel over a single positive sample drawn from the demonstration cache and detached negatives drawn from the student's own output, so the student is attracted toward the demonstration distribution without a per-sample regression target. We use the Laplace kernel $k(d) = \exp(-d/R)$ on scale-normalized distances, compute the loss at each temperature and average across temperatures, and keep the three modifications of the reference implementation: a single softmax, Laplace perturbation of the inputs, and partially detached negatives. The feature over which the kernel acts is the flattened action chunk, and one drift problem is instantiated per task in the micro-batch. Kernel, bandwidth, perturbation, normalization, time sampling and student-optimization settings are listed in Table 7.

DriftOPD. DriftOPD augments the Drift objective with the off-policy distillation term introduced in Section 4.3, which supplies the action gradient $\nabla_a Q_\phi$ of the frozen demonstration-trained critic. Every other setting is shared with Drift, so the two differ by exactly this one term and their gap is attributable to the critic. The critic is trained as described in Section B.3 and the coupling weight and normalization are given in Table 7.

B.5 Simulation Evaluation Protocol

We follow the official benchmark protocols for replanning intervals and episode termination, while environment initialization follows the evaluation code released with the corresponding teacher models. Task sets, episode counts and the resulting settings are summarized in Table 11.

Table 11: Simulation benchmark configurations.

Benchmark	Task set	# Tasks	Episodes per task	Initialization	Actions before replanning	Termination
RoboCasa365	`pretrain` split (leaderboard)	50	30	$\pi_{0.5}$: seed 7 GR00T: no explicit seed	$\pi_{0.5}$: 5 GR00T N1.5: 16 GR00T N1.6: 8	Success or task horizon
RoboTwin 2.0	50 tasks `demo_clean`	50	30	Accepted-seed manifest	25	Success or task-specific limit
LIBERO	4 suites $\times$ 10 tasks	40	50	Seed 7 50 official initial states	$\pi_{0.5}$: 5 ABot / SmolVLA: 10	Success or suite horizon
LIBERO-Plus	Perturbed LIBERO tasks	10,030	1	Seed 7 first official initial state	$\pi_{0.5}$: 5 ABot / SmolVLA: 10	Success or suite horizon

RoboCasa365. We use the `pretrain` evaluation split with 18 `atomic_seen`, 16 `composite_seen` and 16 `composite_unseen` tasks, each evaluated over 30 episodes. For $\pi_{0.5}$ the environment is initialized with seed 7, so the compared methods share the same initial conditions and differ only in the stochasticity of policy generation. The official GR00T evaluators do not fix an environment seed, so those evaluations are not treated as paired. Thus, for Table 1, we follow the official GR00T evaluation protocol, while paired episode comparisons are provided as demo videos on the project page. An episode ends when the environment reports success or when the task horizon from the benchmark registry is exhausted.

RoboTwin 2.0. We evaluate 50 tasks with the ALOHA-AgileX embodiment under the `demo_clean` configuration, which disables background, lighting, camera, table-height and clutter randomization. The simulator runs at a physics timestep of $1/250\,\mathrm{s}$. Each predicted action chunk is executed for 25 environment actions before replanning, and an episode ends on success or when the task-specific action limit is reached.

To make initial conditions reproducible we build an accepted-seed manifest per task. Candidate seeds are drawn in increasing order from 100000 and run with the benchmark expert, and a seed is accepted only when both its motion plan and its task-success check pass. Seeds the expert cannot solve are discarded, because such episodes fail for every policy and separate no methods while still consuming the budget. Filling 30 accepted episodes takes about 36 candidates per task at the median, so the seeds actually executed span 100000 to 100190 and form a non-contiguous subset of that interval rather than a fixed block. The manifest is shared across the compared methods, so a given episode index refers to the same initial condition for each of them, and the executed seed and outcome of every episode are recorded so that two runs can be aligned episode by episode. The short, medium and long groups reported in Table 2 correspond to average demonstration trajectory lengths of $<$150, 150–279 and $\geq$280 steps.

LIBERO and LIBERO-Plus. For LIBERO we evaluate the four standard suites with 10 tasks per suite and 50 episodes per task. Each task uses seed 7 and the 50 benchmark-provided initial states exactly once, so the compared methods again share initial conditions. The policy consumes agent-view and wrist RGB observations resized with padding to 224×224. Every episode begins with 10 no-op steps for physical stabilization, and the maximum horizons are 220, 280, 300 and 520 steps for LIBERO-Spatial, LIBERO-Object, LIBERO-Goal and LIBERO-10. An episode ends immediately on success and is otherwise counted as a failure once its horizon is reached.

LIBERO-Plus reuses the LIBERO students without further training and follows the same execution protocol and suite horizons. Its 10,030 task instances are perturbed variants of the standard LIBERO tasks, so each instance is already a distinct condition and is evaluated for a single episode from the first benchmark-provided initial state. All LIBERO-Plus evaluations are run in one compute environment so that rendering does not differ across nodes.

B.6 Real-World Data and Evaluation



(a) MolmoAct 2 Research Kit (b) Objects used in real-world experiments

Figure 6: Real-world experimental setup. (a) MolmoAct 2 Research Kit on the YAM leader–follower bimanual platform, with top and wrist RealSense cameras providing RGB observations. The right arm is used for unimanual tasks and both arms for bimanual tasks. (b) Objects and containers used for the unimanual pick-and-place and bimanual handover tasks.

Platform and demonstrations. All real-world experiments are conducted on the YAM bimanual platform shown in Figure 6a, using the right arm for unimanual tasks and both arms for bimanual tasks. Two operators collected 638 demonstration episodes spanning twelve tasks, teleoperating the follower arms at roughly 200 Hz while the three RealSense cameras recorded at approximately 30 Hz. Robot state and action streams are linearly interpolated onto camera timestamps so that every frame carries a matched observation-action pair, yielding 366,373 aligned frames. The policy observes RGB only, letterboxed from 640×480 to 256×256, and predicts a seven-dimensional action per arm, consisting of six arm joint values and one continuous gripper value. Actions are normalized to $[-1, 1]$ using $1^{\text{st}}/99^{\text{th}}$ percentile min-max scaling, preventing rare teleoperation outliers from compressing the effective action range. The full collection and GR00T N1.6 teacher fine-tuning recipe are listed in Table 12.

Table 12: Real-world demonstration data and GR00T N1.6 teacher fine-tuning

Item	Setting
Robot	I2RT YAM leader–follower system
Demonstrations	Unimanual (right arm): 424 episodes, 269,089 frames, 8 tasks Bimanual: 214 episodes, 97,284 frames, 4 tasks
Teleoperation rate	≈ 200 Hz
Camera rate	≈ 30 Hz
Policy cameras	Top, right, and left Intel RealSense, RGB only
Raw resolution	640×480
Model image input	256×256 with letterbox preprocessing
Robot state	Follower-arm state
Action target	Leader-arm action
Action dimension	Unimanual: 7 (six arm joints and one gripper value) Bimanual: 14 (twelve arm joints and two gripper values)
Gripper	Continuous value in $[0, 1]$
Temporal alignment	Linear interpolation of state and action to camera timestamps
Normalization	$1^{\text{st}}/99^{\text{th}}$ percentile min-max scaling to $[-1, 1]$
Data usage	All demonstrations used for teacher fine-tuning; no held-out split
Teacher model	GR00T N1.6
Teacher fine-tuning	20,000 steps, batch size 64, AdamW, learning rate 10^{-4}, weight decay 10^{-5}, warmup ratio 0.05, cosine decay
Hardware	$1\times$ NVIDIA B200

Table 13: Objects used in the real-world experiments. Dimensions are height $\times$ width $\times$ depth in centimeters; for spherical and circular items the diameter is reported instead.

Object	Size	Role
Small basket (white, shallow)	$20.2 \times 14.5 \times 6$	Target container
Plate (white, round)	diameter 22.5	Target container
Pot (stainless steel)	diameter 16, height 9.2	Target container
Plush tennis ball	diameter ≈ 12	Grasped object
Plush soccer ball	diameter ≈ 12	Grasped object
Plush basketball	diameter ≈ 5	Grasped object
Plush baseball	diameter ≈ 5	Grasped object
Plush black ball	diameter ≈ 5	Grasped object

Task design. We construct four unimanual pick-and-place tasks with different arm motions and end-effector orientations, as well as two bimanual pick-and-handover-and-place tasks using the objects and containers shown in Figure 6b and Table 13. The four unimanual tasks are qualitatively grouped into Easy, Moderate, Hard, and Challenging based on their task characteristics and the multi-step teacher success rates. For the two bimanual tasks, we swap the object and target locations to create left-to-right and right-to-left handover settings.

Table 14: Real-world evaluation configuration

Item	Setting
Tasks	Unimanual: four pick-and-place tasks Bimanual: two pick-and-handover-and-place tasks
Evaluation settings	15 fixed scenes per task
Pairing	Teacher session captures the scenes; all methods replay the same scenes in the same order
Control frequency	30 Hz
Inference interval	0.3 s, i.e. about 9 commanded control steps between replans
Inference mode	Asynchronous rate-limited inference
Inference hardware	Local NVIDIA GeForce RTX 4090 (24 GB); no external inference server
Success annotation	Manual binary success/failure by the operator
Scene verification	RGB and depth similarity logged against the teacher-session reference
Reset	Return to a predefined basic pose, then settle
Episode timeout	40 s
Maximum commanded joint speed	$0.75\,\mathrm{rad/s}$
Command timeout	0.5 s

Paired scene replay. Each task is evaluated on 15 fixed scenes. The multi-step teacher session runs first and records those scenes; the one-step teacher, sCD, MFD, Drift, and DriftOPD then replay the same 15 scenes in the same order. To ensure paired initial conditions, we require both the top and wrist views to achieve PSNR $\geq$ 28 dB and SSIM $\geq$ 0.95 relative to the corresponding reference scene. Thus, every method is evaluated under closely matched visual environments. Success is annotated manually as a binary outcome by the operator running the session. The remaining execution settings are given in Table 14.

Safety and exclusions. Joint-speed and command-timeout limits are enforced by the robot controller independently of the evaluated policy, so they apply identically to every method. Joint-limit recovery occasionally requires an extra reset attempt between episodes; because this happens before policy execution begins, no accepted evaluation episode is discarded by it. Episodes affected by operator error are identified and repeated before evaluation rather than being dropped afterwards. Any episode in which the two robot arms collide with each other, or in which a robot arm collides with the table or the top-camera support structure, is counted as a task failure.

B.7 Hardware and Software

Table 15: Hardware used for training and evaluation

Workload	Hardware	Usage
Student distillation	NVIDIA B200, 180 GB	1 GPU per run
Critic training	NVIDIA B200, 180 GB	1 GPU per run
Real-world teacher fine-tuning	NVIDIA B200, 180 GB	1 GPU per run
Simulation evaluation	RoboTwin 2.0 & LIBERO-Plus: NVIDIA A100-SXM4-40GB RoboCasa365 & LIBERO: NVIDIA B200	1 evaluation lane per GPU
Real-world inference	NVIDIA GeForce RTX 4090, 24 GB	Local robot workstation

The hardware used for training and evaluation is summarized in Table 15. All student-distillation and critic-training runs use one GPU per process.

Table 16: Benchmark-specific software environments. Each benchmark uses its own Python environment because the simulator dependencies conflict.

Component	RoboTwin 2.0	RoboCasa365	LIBERO / LIBERO-Plus
Benchmark code	RoboTwin, upstream commit `c3ddfa8b` (23 May 2026); camera images rendered with the rasterization shader instead of the upstream 32-spp ray-tracing path	RoboCasa 1.0.1 (vendored)	LIBERO (vendored)
Simulator	SAPIEN 3.0.0b1	MuJoCo 3.3.1	MuJoCo 2.3.7 ($\pi_{0.5}$, ABot-M0); 3.8.1 (SmolVLA)
Physics timestep	$1/250\,\mathrm{s}$	Benchmark default	Benchmark default
robosuite	—	1.5.2	1.4.0
Motion planning	mplib 0.2.1, nvidia-curobo 0.7.8, warp-lang 1.15.0	—	—
Python	3.10.20	3.11.15	3.11.15
PyTorch	2.7.1+cu128	2.7.1+cu128	2.11.0+cu128 ($\pi_{0.5}$); 2.7.1+cu128 (ABot-M0, SmolVLA)
NumPy	1.26.4	2.2.5	1.26.4
Gymnasium	0.29.1	0.29.1	0.29.1

Each benchmark keeps a separate Python environment because their simulator dependencies conflict: RoboCasa365 requires robosuite 1.5.2 with MuJoCo 3.3.1, whereas LIBERO is pinned to robosuite 1.4.0 with MuJoCo 2.3.7. Within LIBERO-Plus the SmolVLA evaluator additionally requires a newer MuJoCo than the other two policies. The RoboCasa365 and LIBERO copies are vendored into our codebase without upstream revision metadata, so we report the versions recoverable from the environments rather than a commit hash.

B.8 Oscillation Metric

Task success rate is the primary metric for both simulation and real-world experiments. On the real robot we additionally report a command oscillation metric in Table 25, because success rate alone does not say how a policy reaches the goal. A one-step student can complete a task while emitting commands that reverse direction from step to step, which stresses the hardware and is not visible in a binary outcome. We therefore measure oscillation as a stability metric for real-robot deployment.

The commanded joint trajectory carries both the intended motion and any high-frequency jitter on top of it. We separate the two by smoothing the trajectory and taking the residual as the jitter. For each accepted episode,

$$\hat{x}_j = \mathrm{SG}_{w=15,p=2}(x_j),$$
$$\mathrm{Osc} = \frac{180}{\pi} \cdot \frac{1}{J} \sum_{j=1}^{J} \sqrt{\frac{1}{N} \sum_{t=1}^{N} \left(x_j(t) - \hat{x}_j(t)\right)^2}, \tag{95}$$

where $x_j(t)$ is the commanded position of arm joint j at timestep t and $\mathrm{SG}_{w=15,p=2}$ is a second-order Savitzky-Golay filter with a 15-sample window. At the 30 Hz command rate that window spans about 0.5 s, so the filter passes deliberate reaching motion and leaves only faster reversals in the residual. The residual RMS is averaged over the J arm joints, excluding the grippers ($J = 6$ for unimanual and $J = 12$ for bimanual), and converted from radians to degrees, so Osc is read as a typical command deviation in degrees.

We compute Osc per episode and report the unweighted mean over the 15 accepted episodes of each task–method pair. Failed episodes are included: a policy that oscillates its way into failure should not be rewarded by having those episodes dropped.

C Full Results

This section expands the aggregate tables of the main paper into per-task results and adds the two LIBERO benchmarks, which the main paper reports only in summary. All success rates are per-task rates over the episode counts given in Table 11.

RoboCasa365 (Tables 17–20). These four tables are the per-task version of Table 1, for $\pi_{0.5}$ with full fine-tuning, $\pi_{0.5}$ with LoRA (r=16), GR00T N1.5 and GR00T N1.6. Each keeps the three `pretrain` suites separate, and the *Avg.* rows reproduce the corresponding entries of the main table. The per-task view shows that the suite averages rest on a few tasks: most `composite_unseen` tasks are at 0% for every method.

RoboTwin 2.0 (Tables 21 and 22). These two tables are the per-task version of Table 2, for $\pi_{0.5}$-LoRA and ABot-M0. Tasks are ordered by average demonstration trajectory length and grouped into the same `short`, `medium` and `long` horizons as the main table, so the horizon trend reads directly down each column. The backbones differ at long horizons: $\pi_{0.5}$ loses 14.5 points from the multi-step to the one-step teacher, which the students partly recover, whereas for ABot-M0 both teachers sit at 36.97% and no such gap exists.

LIBERO (Table 23). This table is not summarized in the main paper. It reports success rate for each of the four standard suites and the 40-task average. Excluding SmolVLA, both backbones are near saturation on `libero_spatial`, `libero_object` and `libero_goal`, so methods separate mainly through the long-horizon `libero_10`. The one exception is MFD on ABot-M0, which collapses to 0% everywhere.

LIBERO-Plus (Table 24). This table is likewise new. It breaks the 10,030 perturbed task instances down by perturbation factor, so each row isolates one axis of robustness. Success rates track the perturbation rather than the distillation method: for $\pi_{0.5}$-LoRA every method stays within about one point of the teacher on each factor, and the spread across factors (71% on camera viewpoints versus 98% on lighting) far exceeds the spread across methods.

Training curves (Figure 7). The figure shows one-step distillation losses on RoboCasa365. One caveat is needed to read it: the drifting objective $\mathcal{L}_{\mathrm{Drift}}$ is RMS-normalized, so its value stays near 1 throughout training by construction and carries no information about optimization progress. A flat drift curve is the expected behavior, not a stalled run. For DriftOPD the total loss is plotted with its drift term (dashed) and critic term $-Q$ (red); only the critic term moves on a meaningful scale, and it is what separates DriftOPD from Drift.

Pre-normalization drift magnitude (Figure 8). Because that normalization removes the scale of $\mathcal{L}_{\mathrm{Drift}}$, we additionally plot the bandwidth-wise RMS of the drifting term before normalization. This is the quantity that actually changes over training, and shows how the drift field's magnitude is distributed across kernel bandwidths.

Table 17: Per-task success rates of $\pi_{0.5}$ (full fine-tuning) and its one-step students on RoboCasa365. Each task is evaluated over 30 episodes. The multi-step Teacher column is taken from the RoboCasa365 leaderboard (https://robocasa.ai/leaderboard.html).

Model	Teacher steps	Task	Teacher	1-step				
				Teacher	sCD	MFD	Drift	DriftOPD
atomic_seen (18 tasks)								
$\pi_{0.5}$	10	CloseBlenderLid	–	6.67	0.00	3.33	3.33	3.33
		CloseFridge	–	23.33	46.67	46.67	30.00	56.67
		CloseToasterOvenDoor	–	0.00	0.00	6.67	3.33	0.00
		CoffeeSetupMug	–	13.33	3.33	10.00	13.33	6.67
		NavigateKitchen	–	0.00	0.00	3.33	0.00	0.00
		OpenCabinet	–	30.00	30.00	36.67	43.33	43.33
		OpenDrawer	–	53.33	63.33	53.33	66.67	83.33
		OpenStandMixerHead	–	63.33	30.00	26.67	30.00	53.33
		PickPlaceCounterToCabinet	–	43.33	46.67	66.67	70.00	76.67
		PickPlaceCounterToStove	–	76.67	73.33	70.00	70.00	66.67
		PickPlaceDrawerToCounter	–	36.67	33.33	46.67	36.67	53.33
		PickPlaceSinkToCounter	–	86.67	86.67	76.67	93.33	90.00
		PickPlaceToasterToCounter	–	0.00	0.00	6.67	3.33	3.33
		SlideDishwasherRack	–	73.33	66.67	80.00	60.00	76.67
		TurnOffStove	–	6.67	3.33	6.67	0.00	6.67
		TurnOnElectricKettle	–	0.00	0.00	3.33	0.00	6.67
		TurnOnMicrowave	–	0.00	0.00	16.67	3.33	13.33
		TurnOnSinkFaucet	–	50.00	26.67	60.00	53.33	50.00
		Avg. (atomic_seen)	39.6	31.30	28.33	34.44	32.22	**38.33**
composite_seen (16 tasks)								
$\pi_{0.5}$	10	DeliverStraw	–	0.00	0.00	0.00	0.00	0.00
		GetToastedBread	–	0.00	0.00	0.00	0.00	0.00
		KettleBoiling	–	0.00	3.33	0.00	0.00	3.33
		LoadDishwasher	–	6.67	0.00	10.00	20.00	16.67
		PackIdenticalLunches	–	0.00	0.00	3.33	3.33	0.00
		PreSoakPan	–	3.33	10.00	3.33	23.33	40.00
		PrepareCoffee	–	0.00	0.00	3.33	0.00	0.00
		RinseSinkBasin	–	0.00	0.00	3.33	0.00	3.33
		ScrubCuttingBoard	–	0.00	0.00	0.00	0.00	0.00
		SearingMeat	–	0.00	0.00	0.00	0.00	0.00
		SetUpCuttingStation	–	0.00	0.00	6.67	0.00	0.00
		StackBowlsCabinet	–	3.33	3.33	10.00	16.67	16.67
		SteamInMicrowave	–	0.00	0.00	0.00	3.33	0.00
		StirVegetables	–	0.00	0.00	0.00	0.00	3.33
		StoreLeftoversInBowl	–	0.00	0.00	0.00	0.00	0.00
		WashLettuce	–	0.00	0.00	0.00	13.33	3.33
		Avg. (composite_seen)	7.1	0.83	1.04	2.50	5.00	**5.42**
composite_unseen (16 tasks)								
$\pi_{0.5}$	10	ArrangeBreadBasket	–	0.00	0.00	0.00	0.00	0.00
		ArrangeTea	–	0.00	0.00	0.00	0.00	0.00
		BreadSelection	–	0.00	0.00	0.00	0.00	0.00
		CategorizeCondiments	–	0.00	0.00	0.00	0.00	0.00
		CuttingToolSelection	–	0.00	0.00	0.00	0.00	0.00
		GarnishPancake	–	0.00	0.00	0.00	0.00	0.00
		GatherTableware	–	0.00	0.00	0.00	0.00	0.00
		HeatKebabSandwich	–	0.00	0.00	0.00	0.00	0.00
		MakeIceLemonade	–	0.00	0.00	0.00	0.00	0.00
		PanTransfer	–	0.00	0.00	0.00	0.00	0.00
		PortionHotDogs	–	0.00	0.00	0.00	0.00	0.00
		RecycleBottlesByType	–	13.33	13.33	13.33	13.33	13.33
		SeparateFreezerRack	–	0.00	0.00	0.00	0.00	0.00
		WaffleReheat	–	0.00	0.00	0.00	0.00	0.00
		WashFruitColander	–	0.00	0.00	0.00	3.33	13.33
		WeighIngredients	–	0.00	0.00	0.00	0.00	0.00
		Avg. (composite_unseen)	1.2	0.83	0.83	0.83	1.04	**1.67**

Table 18: Per-task success rates of $\pi_{0.5}$ (LoRA r=16) and its one-step students on RoboCasa365. Each task is evaluated over 30 episodes. The multi-step Teacher column is taken from the RoboCasa365 leaderboard (https://robocasa.ai/leaderboard.html).

Model	Teacher steps	Task	Teacher	1-step				
				Teacher	sCD	MFD	Drift	DriftOPD
atomic_seen (18 tasks)								
$\pi_{0.5}$ (LoRA r=16)	10	CloseBlenderLid	–	6.67	0.00	0.00	3.33	3.33
		CloseFridge	–	23.33	10.00	43.33	20.00	43.33
		CloseToasterOvenDoor	–	0.00	0.00	3.33	0.00	6.67
		CoffeeSetupMug	–	13.33	10.00	10.00	16.67	10.00
		NavigateKitchen	–	0.00	0.00	3.33	0.00	0.00
		OpenCabinet	–	30.00	23.33	16.67	36.67	33.33
		OpenDrawer	–	53.33	56.67	40.00	56.67	70.00
		OpenStandMixerHead	–	63.33	60.00	6.67	53.33	53.33
		PickPlaceCounterToCabinet	–	43.33	60.00	53.33	46.67	66.67
		PickPlaceCounterToStove	–	76.67	83.33	66.67	70.00	70.00
		PickPlaceDrawerToCounter	–	36.67	36.67	30.00	36.67	46.67
		PickPlaceSinkToCounter	–	86.67	70.00	76.67	93.33	96.67
		PickPlaceToasterToCounter	–	0.00	3.33	10.00	16.67	30.00
		SlideDishwasherRack	–	73.33	70.00	60.00	80.00	73.33
		TurnOffStove	–	6.67	0.00	10.00	6.67	10.00
		TurnOnElectricKettle	–	0.00	0.00	13.33	3.33	10.00
		TurnOnMicrowave	–	0.00	0.00	10.00	13.33	23.33
		TurnOnSinkFaucet	–	50.00	43.33	56.67	60.00	60.00
		Avg. (atomic_seen)	39.6	31.30	29.26	28.33	34.07	**39.26**
composite_seen (16 tasks)								
$\pi_{0.5}$ (LoRA r=16)	10	DeliverStraw	–	0.00	0.00	0.00	0.00	0.00
		GetToastedBread	–	0.00	0.00	0.00	0.00	0.00
		KettleBoiling	–	0.00	0.00	3.33	0.00	10.00
		LoadDishwasher	–	6.67	6.67	10.00	20.00	23.33
		PackIdenticalLunches	–	0.00	0.00	0.00	0.00	0.00
		PreSoakPan	–	3.33	16.67	20.00	10.00	13.33
		PrepareCoffee	–	0.00	0.00	0.00	0.00	0.00
		RinseSinkBasin	–	0.00	3.33	0.00	3.33	3.33
		ScrubCuttingBoard	–	0.00	0.00	0.00	0.00	0.00
		SearingMeat	–	0.00	0.00	0.00	0.00	0.00
		SetUpCuttingStation	–	0.00	0.00	0.00	0.00	3.33
		StackBowlsCabinet	–	3.33	0.00	6.67	6.67	20.00
		SteamInMicrowave	–	0.00	0.00	0.00	0.00	0.00
		StirVegetables	–	0.00	0.00	0.00	6.67	3.33
		StoreLeftoversInBowl	–	0.00	0.00	0.00	0.00	0.00
		WashLettuce	–	0.00	0.00	3.33	3.33	3.33
		Avg. (composite_seen)	7.1	0.83	1.67	2.71	3.12	**5.00**
composite_unseen (16 tasks)								
$\pi_{0.5}$ (LoRA r=16)	10	ArrangeBreadBasket	–	0.00	0.00	0.00	0.00	0.00
		ArrangeTea	–	0.00	0.00	0.00	0.00	0.00
		BreadSelection	–	0.00	0.00	0.00	0.00	0.00
		CategorizeCondiments	–	0.00	0.00	0.00	0.00	0.00
		CuttingToolSelection	–	0.00	0.00	3.33	6.67	6.67
		GarnishPancake	–	0.00	0.00	0.00	0.00	0.00
		GatherTableware	–	0.00	0.00	0.00	0.00	0.00
		HeatKebabSandwich	–	0.00	0.00	0.00	0.00	0.00
		MakeIceLemonade	–	0.00	0.00	0.00	0.00	0.00
		PanTransfer	–	0.00	0.00	0.00	0.00	0.00
		PortionHotDogs	–	0.00	0.00	0.00	0.00	3.33
		RecycleBottlesByType	–	13.33	13.33	13.33	13.33	13.33
		SeparateFreezerRack	–	0.00	0.00	0.00	0.00	0.00
		WaffleReheat	–	0.00	0.00	0.00	0.00	0.00
		WashFruitColander	–	0.00	0.00	3.33	3.33	0.00
		WeighIngredients	–	0.00	0.00	0.00	0.00	0.00
		Avg. (composite_unseen)	1.2	0.83	0.83	1.25	**1.46**	**1.46**

Table 19: Per-task success rates of GR00T N1.5 and its one-step students on RoboCasa365. Each task is evaluated over 30 episodes. The multi-step Teacher column is taken from the RoboCasa365 leaderboard (https://robocasa.ai/leaderboard.html).

Model	Teacher steps	Task	Teacher	1-step				
				Teacher	sCD	MFD	Drift	DriftOPD
atomic_seen (18 tasks)								
GR00T N1.5	4	CloseBlenderLid	–	6.67	3.33	10.00	16.67	26.67
		CloseFridge	–	73.33	60.00	90.00	56.67	70.00
		CloseToasterOvenDoor	–	43.33	33.33	53.33	23.33	26.67
		CoffeeSetupMug	–	16.67	16.67	3.33	33.33	16.67
		NavigateKitchen	–	6.67	0.00	3.33	3.33	6.67
		OpenCabinet	–	56.67	36.67	63.33	73.33	53.33
		OpenDrawer	–	53.33	43.33	56.67	46.67	63.33
		OpenStandMixerHead	–	80.00	73.33	70.00	86.67	80.00
		PickPlaceCounterToCabinet	–	63.33	53.33	73.33	70.00	86.67
		PickPlaceCounterToStove	–	80.00	86.67	56.67	70.00	73.33
		PickPlaceDrawerToCounter	–	46.67	53.33	63.33	33.33	36.67
		PickPlaceSinkToCounter	–	80.00	73.33	83.33	90.00	96.67
		PickPlaceToasterToCounter	–	63.33	33.33	43.33	50.00	43.33
		SlideDishwasherRack	–	30.00	53.33	43.33	50.00	63.33
		TurnOffStove	–	6.67	10.00	3.33	3.33	13.33
		TurnOnElectricKettle	–	40.00	53.33	43.33	30.00	23.33
		TurnOnMicrowave	–	0.00	16.67	13.33	20.00	6.67
		TurnOnSinkFaucet	–	26.67	13.33	10.00	33.33	30.00
		Avg. (atomic_seen)	50.7	42.96	39.63	43.52	43.89	**45.37**
composite_seen (16 tasks)								
GR00T N1.5	4	DeliverStraw	–	0.00	0.00	0.00	0.00	0.00
		GetToastedBread	–	0.00	0.00	0.00	0.00	0.00
		KettleBoiling	–	6.67	3.33	3.33	6.67	6.67
		LoadDishwasher	–	33.33	30.00	23.33	10.00	20.00
		PackIdenticalLunches	–	10.00	0.00	3.33	3.33	0.00
		PreSoakPan	–	10.00	10.00	33.33	13.33	13.33
		PrepareCoffee	–	0.00	0.00	0.00	0.00	0.00
		RinseSinkBasin	–	6.67	6.67	26.67	13.33	10.00
		ScrubCuttingBoard	–	36.67	6.67	13.33	13.33	20.00
		SearingMeat	–	0.00	0.00	0.00	0.00	0.00
		SetUpCuttingStation	–	0.00	6.67	13.33	10.00	6.67
		StackBowlsCabinet	–	6.67	0.00	6.67	20.00	26.67
		SteamInMicrowave	–	0.00	0.00	3.33	0.00	3.33
		StirVegetables	–	0.00	0.00	0.00	6.67	6.67
		StoreLeftoversInBowl	–	3.33	0.00	0.00	0.00	0.00
		WashLettuce	–	10.00	3.33	10.00	33.33	23.33
		Avg. (composite_seen)	14.8	7.71	4.17	**8.54**	8.12	**8.54**
composite_unseen (16 tasks)								
GR00T N1.5	4	ArrangeBreadBasket	–	0.00	0.00	0.00	0.00	0.00
		ArrangeTea	–	0.00	0.00	0.00	0.00	0.00
		BreadSelection	–	16.67	0.00	0.00	3.33	13.33
		CategorizeCondiments	–	0.00	0.00	0.00	3.33	0.00
		CuttingToolSelection	–	0.00	0.00	6.67	0.00	0.00
		GarnishPancake	–	0.00	0.00	3.33	0.00	3.33
		GatherTableware	–	0.00	0.00	0.00	0.00	0.00
		HeatKebabSandwich	–	0.00	0.00	0.00	0.00	0.00
		MakeIceLemonade	–	3.33	0.00	3.33	0.00	0.00
		PanTransfer	–	0.00	0.00	0.00	0.00	0.00
		PortionHotDogs	–	0.00	0.00	6.67	3.33	6.67
		RecycleBottlesByType	–	6.67	16.67	3.33	16.67	16.67
		SeparateFreezerRack	–	0.00	0.00	0.00	0.00	0.00
		WaffleReheat	–	6.67	3.33	6.67	6.67	3.33
		WashFruitColander	–	3.33	0.00	10.00	0.00	3.33
		WeighIngredients	–	3.33	6.67	3.33	3.33	0.00
		Avg. (composite_unseen)	2.7	2.50	1.67	2.71	2.29	**2.92**

Table 20: Per-task success rates of GR00T N1.6 and its one-step students on RoboCasa365. Each task is evaluated over 30 episodes. The multi-step Teacher column is taken from the RoboCasa365 leaderboard (https://robocasa.ai/leaderboard.html).

Model	Teacher steps	Task	Teacher	1-step				
				Teacher	sCD	MFD	Drift	DriftOPD
atomic_seen (18 tasks)								
GR00T N1.6	4	CloseBlenderLid	–	3.33	6.67	0.00	10.00	13.33
		CloseFridge	–	66.67	80.00	63.33	90.00	76.67
		CloseToasterOvenDoor	–	56.67	60.00	30.00	66.67	50.00
		CoffeeSetupMug	–	36.67	36.67	3.33	20.00	26.67
		NavigateKitchen	–	6.67	0.00	0.00	16.67	16.67
		OpenCabinet	–	30.00	33.33	33.33	66.67	63.33
		OpenDrawer	–	53.33	43.33	6.67	23.33	26.67
		OpenStandMixerHead	–	83.33	86.67	80.00	86.67	93.33
		PickPlaceCounterToCabinet	–	56.67	63.33	53.33	60.00	53.33
		PickPlaceCounterToStove	–	60.00	63.33	43.33	56.67	60.00
		PickPlaceDrawerToCounter	–	23.33	36.67	10.00	30.00	30.00
		PickPlaceSinkToCounter	–	80.00	83.33	80.00	76.67	73.33
		PickPlaceToasterToCounter	–	50.00	56.67	43.33	40.00	60.00
		SlideDishwasherRack	–	46.67	80.00	43.33	70.00	66.67
		TurnOffStove	–	3.33	6.67	0.00	3.33	13.33
		TurnOnElectricKettle	–	70.00	70.00	73.33	56.67	90.00
		TurnOnMicrowave	–	43.33	63.33	23.33	46.67	33.33
		TurnOnSinkFaucet	–	36.67	60.00	6.67	30.00	43.33
		Avg. (atomic_seen)	51.1	44.81	**51.67**	32.96	47.22	49.44
composite_seen (16 tasks)								
GR00T N1.6	4	DeliverStraw	–	0.00	0.00	0.00	0.00	0.00
		GetToastedBread	–	0.00	0.00	0.00	0.00	0.00
		KettleBoiling	–	0.00	6.67	0.00	0.00	6.67
		LoadDishwasher	–	10.00	3.33	0.00	6.67	10.00
		PackIdenticalLunches	–	0.00	0.00	0.00	10.00	6.67
		PreSoakPan	–	3.33	6.67	3.33	3.33	3.33
		PrepareCoffee	–	0.00	0.00	0.00	0.00	0.00
		RinseSinkBasin	–	13.33	13.33	3.33	10.00	16.67
		ScrubCuttingBoard	–	6.67	10.00	6.67	20.00	20.00
		SearingMeat	–	0.00	0.00	0.00	0.00	0.00
		SetUpCuttingStation	–	0.00	0.00	0.00	3.33	0.00
		StackBowlsCabinet	–	10.00	6.67	0.00	0.00	6.67
		SteamInMicrowave	–	0.00	0.00	0.00	3.33	3.33
		StirVegetables	–	0.00	3.33	0.00	0.00	3.33
		StoreLeftoversInBowl	–	0.00	0.00	0.00	0.00	0.00
		WashLettuce	–	0.00	6.67	0.00	6.67	3.33
		Avg. (composite_seen)	9.4	2.71	3.54	0.83	3.96	**5.00**
composite_unseen (16 tasks)								
GR00T N1.6	4	ArrangeBreadBasket	–	0.00	0.00	0.00	0.00	0.00
		ArrangeTea	–	0.00	0.00	0.00	0.00	0.00
		BreadSelection	–	3.33	3.33	0.00	0.00	3.33
		CategorizeCondiments	–	0.00	0.00	0.00	0.00	0.00
		CuttingToolSelection	–	0.00	0.00	0.00	0.00	0.00
		GarnishPancake	–	0.00	0.00	0.00	0.00	0.00
		GatherTableware	–	0.00	0.00	0.00	0.00	0.00
		HeatKebabSandwich	–	0.00	0.00	0.00	0.00	0.00
		MakeIceLemonade	–	0.00	0.00	0.00	3.33	0.00
		PanTransfer	–	0.00	0.00	0.00	0.00	0.00
		PortionHotDogs	–	0.00	0.00	0.00	0.00	0.00
		RecycleBottlesByType	–	6.67	6.67	16.67	6.67	13.33
		SeparateFreezerRack	–	0.00	0.00	0.00	0.00	0.00
		WaffleReheat	–	0.00	0.00	0.00	0.00	0.00
		WashFruitColander	–	3.33	0.00	0.00	0.00	0.00
		WeighIngredients	–	0.00	0.00	0.00	0.00	0.00
		Avg. (composite_unseen)	1.7	0.83	0.62	**1.04**	0.62	**1.04**

Table 21: Per-task success rates of $\pi_{0.5}$ and its LoRA-based ($r = 16$) student policies on RoboTwin 2.0 with the ALOHA 14-DoF embodiment. Tasks are ordered by average demonstration trajectory length and grouped into `short`, `medium` and `long` horizons. Each task is evaluated over 30 episodes from a shared accepted-seed manifest.

Model	Teacher steps	Task	Teacher	1-step				
				Teacher	sCD	MFD	Drift	DriftOPD
short (18 tasks, 77–147 demo steps)								
$\pi_{0.5}$-LoRA	10	click_bell	30.00	80.00	90.00	100.00	86.67	86.67
		click_alarmclock	50.00	80.00	73.33	100.00	83.33	86.67
		grab_roller	100.00	100.00	100.00	100.00	100.00	100.00
		turn_switch	40.00	50.00	56.67	66.67	43.33	50.00
		lift_pot	33.33	33.33	50.00	43.33	40.00	60.00
		beat_block_hammer	53.33	36.67	16.67	26.67	63.33	40.00
		move_playingcard_away	90.00	90.00	93.33	70.00	83.33	93.33
		press_stapler	90.00	86.67	96.67	100.00	93.33	100.00
		pick_diverse_bottles	46.67	36.67	53.33	30.00	53.33	70.00
		pick_dual_bottles	56.67	60.00	50.00	50.00	66.67	80.00
		place_phone_stand	33.33	33.33	33.33	10.00	30.00	36.67
		place_object_stand	90.00	83.33	90.00	66.67	96.67	96.67
		adjust_bottle	93.33	96.67	93.33	63.33	90.00	86.67
		place_object_scale	56.67	60.00	60.00	33.33	70.00	56.67
		move_pillbottle_pad	70.00	60.00	70.00	23.33	80.00	53.33
		place_a2b_right	63.33	66.67	56.67	66.67	70.00	60.00
		place_fan	66.67	33.33	40.00	10.00	30.00	40.00
		place_mouse_pad	30.00	16.67	43.33	20.00	13.33	23.33
		Avg. (short)	60.74	61.30	64.81	54.44	66.30	67.78
medium (21 tasks, 149–278 demo steps)								
$\pi_{0.5}$-LoRA	10	stamp_seal	26.67	23.33	23.33	23.33	20.00	30.00
		place_a2b_left	66.67	50.00	63.33	60.00	66.67	63.33
		move_can_pot	93.33	63.33	70.00	13.33	70.00	70.00
		rotate_qrcode	76.67	63.33	66.67	43.33	73.33	70.00
		move_stapler_pad	20.00	30.00	23.33	23.33	40.00	33.33
		place_container_plate	96.67	86.67	93.33	80.00	96.67	90.00
		place_bread_skillet	76.67	70.00	63.33	70.00	66.67	73.33
		scan_object	23.33	13.33	20.00	10.00	33.33	43.33
		place_empty_cup	96.67	93.33	90.00	83.33	100.00	96.67
		place_shoe	43.33	23.33	36.67	33.33	33.33	40.00
		open_laptop	93.33	90.00	90.00	86.67	93.33	83.33
		handover_mic	46.67	30.00	36.67	16.67	46.67	53.33
		place_dual_shoes	36.67	26.67	13.33	10.00	3.33	13.33
		place_bread_basket	63.33	56.67	80.00	56.67	66.67	70.00
		place_burger_fries	90.00	90.00	93.33	90.00	93.33	93.33
		dump_bin_bigbin	86.67	86.67	70.00	60.00	70.00	83.33
		place_object_basket	83.33	50.00	73.33	40.00	63.33	66.67
		shake_bottle	100.00	100.00	100.00	86.67	100.00	96.67
		place_can_basket	66.67	26.67	43.33	26.67	50.00	60.00
		put_object_cabinet	26.67	10.00	36.67	6.67	13.33	16.67
		shake_bottle_horizontally	100.00	100.00	96.67	93.33	100.00	100.00
		Avg. (medium)	67.30	56.35	61.11	48.25	61.90	64.13
long (11 tasks, 282–625 demo steps)								
$\pi_{0.5}$-LoRA	10	handover_block	30.00	0.00	6.67	0.00	3.33	6.67
		place_cans_plasticbox	83.33	83.33	53.33	13.33	86.67	90.00
		stack_bowls_two	86.67	76.67	73.33	40.00	73.33	86.67
		stack_blocks_two	93.33	53.33	76.67	10.00	76.67	63.33
		hanging_mug	10.00	3.33	3.33	0.00	6.67	6.67
		blocks_ranking_rgb	90.00	70.00	66.67	30.00	90.00	73.33
		blocks_ranking_size	43.33	33.33	40.00	23.33	43.33	43.33
		stack_bowls_three	56.67	33.33	26.67	3.33	53.33	46.67
		stack_blocks_three	20.00	10.00	3.33	0.00	6.67	16.67
		open_microwave	83.33	76.67	56.67	56.67	76.67	73.33
		put_bottles_dustbin	10.00	6.67	0.00	3.33	16.67	6.67
		Avg. (long)	55.15	40.61	36.97	16.36	48.48	46.67
Avg. (50 tasks)			**62.27**	**54.67**	**57.13**	**43.47**	**60.53**	**61.60**

Table 22: Per-task success rates of ABot-M0 and its student policies on RoboTwin 2.0 with the ALOHA 14-DoF embodiment. Tasks are ordered by average demonstration trajectory length and grouped into `short`, `medium` and `long` horizons. Each task is evaluated over 30 episodes from a shared accepted-seed manifest.

Model	Teacher steps	Task	Teacher	1-step			
				Teacher	sCD	MFD	Drift	DriftOPD
short (18 tasks, 77–147 demo steps)								
ABot-M0	10	click_bell	96.67	100.00	90.00	0.00	80.00	93.33
		click_alarmclock	100.00	100.00	96.67	3.33	93.33	90.00
		grab_roller	100.00	93.33	83.33	0.00	93.33	90.00
		turn_switch	80.00	63.33	70.00	23.33	73.33	53.33
		lift_pot	100.00	93.33	93.33	0.00	90.00	6.67
		beat_block_hammer	86.67	83.33	80.00	0.00	83.33	83.33
		move_playingcard_away	40.00	53.33	50.00	0.00	46.67	50.00
		press_stapler	10.00	16.67	13.33	16.67	36.67	30.00
		pick_diverse_bottles	33.33	53.33	66.67	0.00	43.33	53.33
		pick_dual_bottles	46.67	63.33	73.33	0.00	56.67	56.67
		place_phone_stand	43.33	40.00	56.67	0.00	56.67	36.67
		place_object_stand	76.67	76.67	60.00	0.00	63.33	73.33
		adjust_bottle	83.33	76.67	73.33	0.00	76.67	80.00
		place_object_scale	86.67	86.67	86.67	0.00	86.67	86.67
		move_pillbottle_pad	90.00	70.00	70.00	0.00	86.67	90.00
		place_a2b_right	76.67	83.33	80.00	0.00	73.33	80.00
		place_fan	90.00	86.67	60.00	0.00	70.00	70.00
		place_mouse_pad	23.33	26.67	10.00	0.00	33.33	40.00
		Avg. (short)	70.19	70.37	67.41	2.41	69.07	64.63
medium (21 tasks, 149–278 demo steps)								
ABot-M0	10	stamp_seal	50.00	50.00	40.00	0.00	50.00	60.00
		place_a2b_left	90.00	90.00	83.33	0.00	83.33	90.00
		move_can_pot	73.33	60.00	50.00	0.00	73.33	76.67
		rotate_qrcode	90.00	73.33	70.00	0.00	80.00	70.00
		move_stapler_pad	26.67	26.67	20.00	0.00	23.33	33.33
		place_container_plate	93.33	96.67	90.00	0.00	80.00	80.00
		place_bread_skillet	80.00	60.00	63.33	0.00	53.33	63.33
		scan_object	50.00	36.67	50.00	0.00	46.67	53.33
		place_empty_cup	96.67	96.67	96.67	0.00	80.00	90.00
		place_shoe	46.67	36.67	50.00	0.00	40.00	43.33
		open_laptop	93.33	83.33	90.00	6.67	80.00	90.00
		handover_mic	66.67	53.33	40.00	0.00	60.00	60.00
		place_dual_shoes	33.33	30.00	26.67	0.00	23.33	23.33
		place_bread_basket	90.00	76.67	86.67	0.00	76.67	76.67
		place_burger_fries	90.00	86.67	76.67	0.00	76.67	76.67
		dump_bin_bigbin	93.33	80.00	90.00	0.00	83.33	93.33
		place_object_basket	56.67	66.67	83.33	0.00	76.67	70.00
		shake_bottle	100.00	100.00	100.00	46.67	100.00	100.00
		place_can_basket	73.33	70.00	66.67	0.00	63.33	53.33
		put_object_cabinet	63.33	50.00	30.00	0.00	30.00	26.67
		shake_bottle_horizontally	96.67	100.00	100.00	33.33	100.00	100.00
		Avg. (medium)	73.97	67.78	66.83	4.13	65.71	68.10
long (11 tasks, 282–625 demo steps)								
ABot-M0	10	handover_block	0.00	3.33	0.00	0.00	6.67	3.33
		place_cans_plasticbox	96.67	70.00	66.67	0.00	93.33	83.33
		stack_bowls_two	100.00	96.67	86.67	0.00	83.33	86.67
		stack_blocks_two	56.67	66.67	56.67	0.00	56.67	66.67
		hanging_mug	10.00	0.00	0.00	0.00	20.00	16.67
		blocks_ranking_rgb	26.67	46.67	33.33	0.00	30.00	33.33
		blocks_ranking_size	10.00	6.67	6.67	0.00	0.00	3.33
		stack_bowls_three	83.33	76.67	66.67	0.00	60.00	66.67
		stack_blocks_three	6.67	0.00	3.33	0.00	3.33	16.67
		open_microwave	10.00	33.33	16.67	10.00	36.67	46.67
		put_bottles_dustbin	6.67	6.67	0.00	0.00	3.33	3.33
		Avg. (long)	36.97	36.97	30.61	0.91	35.76	38.79
Avg. (50 tasks)			**64.47**	**61.93**	**59.07**	**2.80**	**60.33**	**60.40**

Table 23: Average task success rates on LIBERO, reported separately for each evaluation suite. Each task is evaluated over 50 episodes under paired environment settings.

Model	Teacher steps	Suite	Teacher	1-step				
				Teacher	sCD	MFD	Drift	DriftOPD
$\pi_{0.5}$-LoRA	10	libero_spatial (10)	98.8%	99.2%	98.6%	99.2%	98.2%	98.4%
		libero_object (10)	98.2%	98.4%	97.2%	97.2%	98.4%	97.4%
		libero_goal (10)	98.0%	98.2%	97.4%	98.2%	98.0%	97.0%
		libero_10 (10)	92.4%	93.0%	89.2%	87.8%	91.8%	91.8%
		Average (40)	**96.85%**	**97.20%**	**95.60%**	**95.60%**	**96.60%**	**96.15%**
SmolVLA	10	libero_spatial (10)	66.6%	76.6%	69.6%	36.8%	64.0%	68.0%
		libero_object (10)	92.0%	95.4%	89.4%	78.8%	90.2%	89.6%
		libero_goal (10)	78.0%	78.8%	73.6%	76.6%	76.6%	76.6%
		libero_10 (10)	43.0%	58.0%	43.4%	22.0%	44.6%	41.8%
		Average (40)	**69.90%**	**77.20%**	**69.00%**	**53.55%**	**68.85%**	**69.00%**
ABot-M0	10	libero_spatial (10)	98.8%	98.8%	99.2%	0.0%	99.2%	99.6%
		libero_object (10)	99.8%	99.6%	99.8%	0.0%	99.4%	99.8%
		libero_goal (10)	99.0%	98.4%	98.6%	0.0%	98.4%	98.4%
		libero_10 (10)	96.6%	95.2%	94.6%	0.0%	94.0%	94.6%
		Average (40)	**98.55%**	**98.00%**	**98.05%**	**0.00%**	**97.75%**	**98.10%**

Table 24: Average task success rates on LIBERO-Plus, reported separately for each evaluation suite. Each of the 10,030 perturbed task instances is evaluated over a single episode under paired environment settings.

Model	Teacher steps	Suite	Teacher	1-step				
				Teacher	sCD	MFD	Drift	DriftOPD
$\pi_{0.5}$-LoRA	10	Sensor Noise (1,601)	86.70%	84.51%	85.38%	86.20%	85.51%	86.38%
		Camera Viewpoints (1,599)	71.36%	71.79%	70.61%	72.61%	70.86%	71.23%
		Robot Initial States (1,550)	75.16%	71.74%	73.16%	72.97%	74.39%	74.58%
		Language Instructions (1,537)	84.65%	86.21%	84.65%	85.75%	85.88%	85.49%
		Objects Layout (1,525)	85.90%	85.77%	86.69%	86.23%	86.75%	86.56%
		Light Conditions (1,142)	97.81%	97.11%	96.50%	96.58%	97.81%	96.76%
		Background Textures (1,076)	95.45%	96.56%	95.26%	94.89%	96.47%	96.93%
		Average (10,030)	**84.24%**	**83.69%**	**83.55%**	**84.04%**	**84.28%**	**84.35%**
SmolVLA	10	Sensor Noise (1,601)	0.00%	0.06%	0.06%	0.12%	0.06%	0.00%
		Camera Viewpoints (1,599)	0.00%	0.00%	0.06%	0.00%	0.00%	0.13%
		Robot Initial States (1,550)	0.32%	0.26%	0.52%	0.19%	0.52%	0.52%
		Language Instructions (1,537)	5.01%	6.18%	5.99%	3.84%	4.88%	6.57%
		Objects Layout (1,525)	10.62%	10.69%	9.05%	13.51%	11.67%	11.28%
		Light Conditions (1,142)	14.62%	21.28%	17.86%	12.00%	17.51%	19.18%
		Background Textures (1,076)	9.29%	9.20%	9.01%	8.27%	10.13%	10.41%
		Average (10,030)	**5.09%**	**6.03%**	**5.39%**	**4.95%**	**5.69%**	**6.12%**
ABot-M0	10	Sensor Noise (1,601)	86.51%	87.26%	87.26%	0.00%	87.07%	87.57%
		Camera Viewpoints (1,599)	59.35%	58.47%	57.41%	0.00%	57.54%	58.29%
		Robot Initial States (1,550)	68.65%	67.48%	67.29%	0.00%	67.29%	67.74%
		Language Instructions (1,537)	87.18%	86.73%	86.21%	0.00%	88.03%	87.77%
		Objects Layout (1,525)	83.02%	83.21%	82.95%	0.00%	82.95%	82.69%
		Light Conditions (1,142)	96.85%	95.53%	96.32%	0.00%	95.36%	95.18%
		Background Textures (1,076)	93.22%	92.01%	93.31%	0.00%	92.75%	92.10%
		Average (10,030)	**80.89%**	**80.37%**	**80.28%**	**0.00%**	**80.38%**	**80.48%**

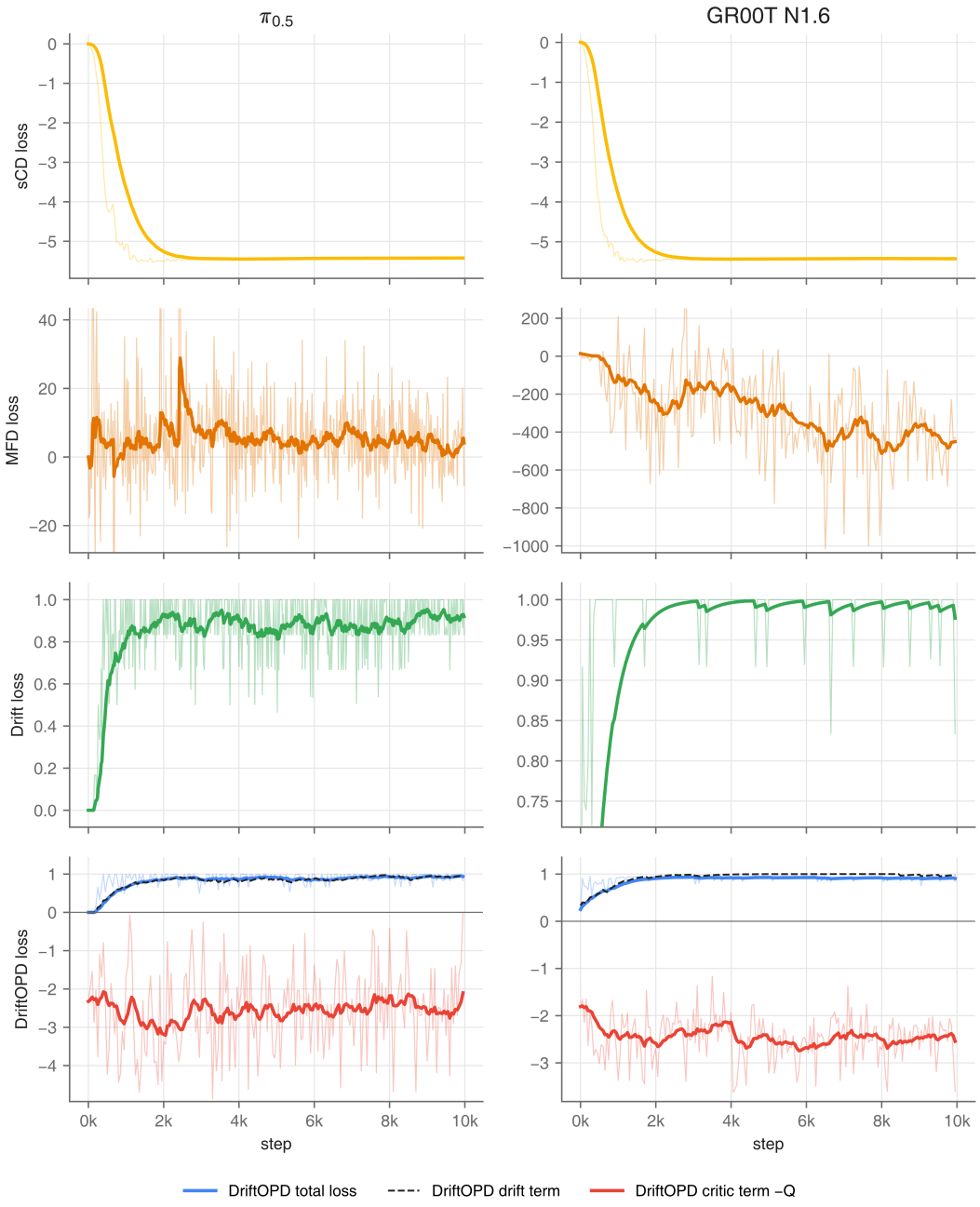


Figure 7: Training curves of one-step distillation on RoboCasa365. Faint lines are logged values and bold lines are EMAs. For DriftOPD, the total loss is shown with its drift term (dashed) and critic term $-Q$ (red).

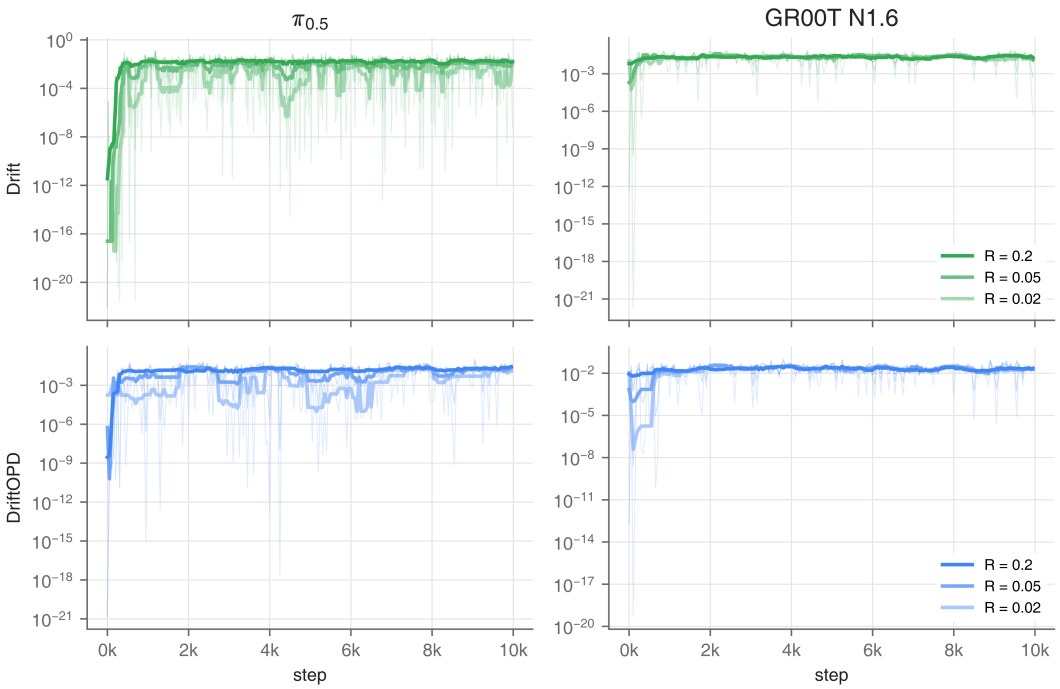


Figure 8: Bandwidth-wise RMS of the drifting loss term before normalization, for the drifting distillation (Drift) and DriftOPD on RoboCasa365.

D Additional Experiments

D.1 Oscillation Analysis for Real-World Robot

Table 25: Real-world comparison of task success rate and action oscillation across the multi-step teacher and one-step policies. Values in parentheses indicate partial success rates (e.g., picking the object but failing to place or handover it), and oscillation is computed according to Eq. (95).

Task	Success Rate (%)						Oscillation (°)					
	4-step	1-step					4-step	1-step				
	Teacher	Teacher	sCD	MFD	Drift	DriftOPD	Teacher	Teacher	sCD	MFD	Drift	DriftOPD
unimanual (right arm)												
Easy	93.3	40.0	0.0 (60.0)	0.0 (0.0)	86.7	**93.3**	1.70	1.77	4.64	7.55	**1.97**	2.01
Moderate	73.3	46.7	6.7 (20.0)	0.0 (0.0)	**60.0**	**60.0**	1.61	1.90	4.81	7.77	**1.83**	1.89
Hard	66.7	13.3	0.0 (0.0)	0.0 (0.0)	60.0	**66.7**	1.65	1.79	4.87	7.69	**1.70**	1.89
Challenging	46.7	0.0	0.0 (0.0)	0.0 (0.0)	**13.3**	**13.3**	1.64	2.03	5.01	7.66	1.71	**1.68**
Avg	70.0	25.0	1.7	0.0	55.0	**58.3**	1.65	1.87	4.83	7.67	**1.80**	1.87
bimanual												
Handover (LR)	73.3	60.0	0.0 (0.0)	0.0 (0.0)	53.3	**80.0**	1.69	2.08	3.89	5.44	**2.05**	2.19
Handover (RL)	86.7	53.3	0.0 (26.7)	0.0 (20.0)	40.0	**73.3**	1.69	1.93	3.75	5.16	2.11	**2.02**
Avg	80.0	56.7	0.0	0.0	46.7	**76.7**	1.69	2.00	3.82	5.30	**2.08**	2.10

Oscillations for real world experiments. Table 25 is the full version of Table 3, covering every task and every method we ran on the real robot with GR00T N1.6 as the teacher, in a right-arm unimanual variant (7-D actions) and a bimanual variant (14-D actions). Oscillation is reported in degrees, measured on the joint-position commands the policy streams to the follower arms at 30 Hz and averaged over the arm joints of each embodiment. sCD and MFD oscillate severely, by a wide margin over the 4-step teacher, and the same pattern holds on every task and on both embodiments. Drift and DriftOPD instead stay at the level of the teacher, so collapsing the sampler to a single step costs them no smoothness while sCD and MFD pay for it with visible jitter.

D.2 Stability Across LoRA Ranks

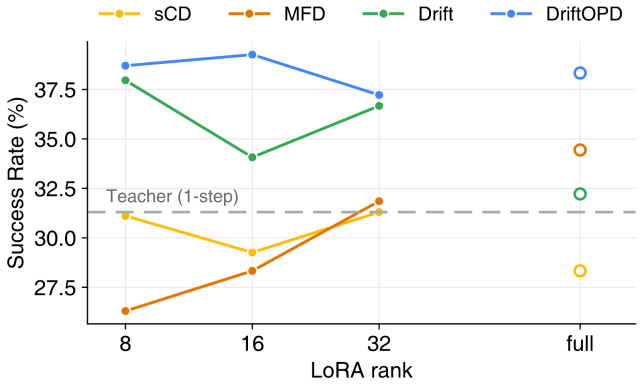


Figure 9: LoRA rank sweep for $\pi_{0.5}$ on RoboCasa365 (`atomic_seen`). Hollow markers: full fine-tuning; dashed line: 1-step teacher (31.30%).

LoRA ranks. The LoRA rank sets how much capacity the student has to move away from the teacher it is initialized from, and one-step distillation is exactly the regime where that budget matters: too little capacity and no objective can close the multi-step gap, too much and the student is free to drift away from the teacher in ways the objective must control. An objective that only wins at one rank would therefore say little. We sweep the rank for $\pi_{0.5}$ on `atomic_seen` and compare all objectives at each setting (Figure 9). DriftOPD outperforms the other objectives at every rank and at full fine-tuning.

D.3 Analysis for Single-Positive Drifting Field

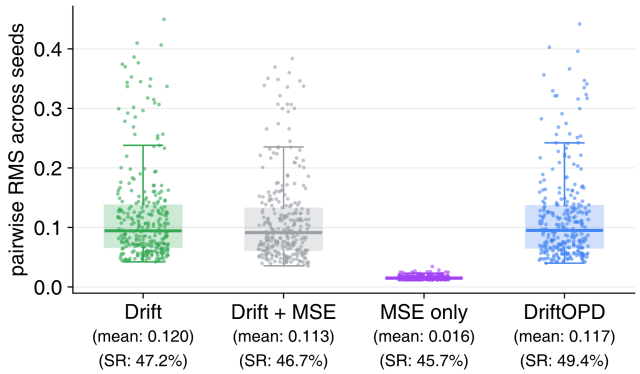


Figure 10: Seed-to-seed action diversity at a fixed observation (GR00T N1.6, RoboCasa365, 10k-step students). For 288 training observations (18 tasks $\times$ 16), each student samples $K = 32$ one-step action chunks that differ only in input noise. The y-axis is the pairwise RMS among the 32 chunks (12 action dims $\times$ 16 steps, normalized space); each dot is one observation, boxes show the median and IQR, and the mean is given under each label. MSE only collapses to a near-deterministic output (0.016), while Drift (0.120) and DriftOPD (0.117) keep about 7$\times$ more spread.

Drifting loss vs. MSE loss. Yang et al. (2026b) argue that a drifting field built from a single positive degenerates into a mean squared error (MSE) objective unless additional practical strategies are used. Since our formulation uses one positive per context, we test directly whether it suffers from that degeneracy.

Figure 10 measures seed-to-seed diversity rather than success, because the failure mode in question is a policy that ignores its input noise. We draw 18 tasks at random and 16 observations from each, and at every one of those 288 observations each student generates 32 action chunks that differ only

in the sampled noise $z \sim \mathcal{N}(0, I)$. *Drift + MSE* and *MSE only* denote a plain sum of the drifting objective and an MSE term, and distillation with the MSE term alone; both follow the training protocol of Table 6. The MSE-only student collapses to an almost deterministic map: its spread is 0.016, an order of magnitude below Drift (0.120) and DriftOPD (0.117), so it discards the initial noise entirely. Success rate alone would not have exposed this, as it degrades far less than the diversity does.

Drawing multiple independent noise samples per observation during training did not change the measured diversity, so the collapse is not simply a matter of how many samples each context sees. We instead attribute the gap to three ingredients that our field inherits from Deng et al. (2026) and from the distillation setting itself: the self-masking heuristic, which excludes each generated sample from its own kernel sum; initialization from pretrained weights, which places the student in a region where the teacher's noise dependence already exists; and the multi-bandwidth kernel, which keeps a repulsive component alive at scales where a single bandwidth would have saturated. Drifting distillation applied to VLAs is therefore free of the collapse that the single-positive analysis predicts.

D.4 Success Field Drifting

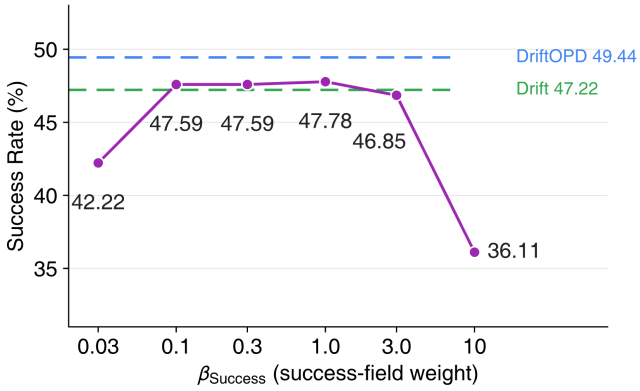


Figure 11: Effect of the success-field weight β_{Success} on GR00T N1.6 (RoboCasa365 `atomic_seen`, 18 tasks $\times$ 30 episodes). Dashed lines indicate Drift (47.22) and DriftOPD (49.44). Success rate remains flat for $\beta_{\text{Success}} \in [0.1, 3.0]$, drops at $\beta_{\text{Success}} = 0.03$, and collapses at $\beta_{\text{Success}} = 10$.

Chunk-level success signal: additional success drifting field. DriftOPD supplies its success signal per context, through a critic conditioned on the observation the student is distilling against. We ask whether that conditioning is necessary, or whether one-step drifting distillation can also be aided by a success signal defined at the level of the action chunk alone. Suppose successful actions form a manifold in chunk space, so that whether a chunk succeeds is a property of the chunk itself rather than of the observation it was produced for. Success could then be supervised globally, leaving the ground-truth target untouched and adding a separate matching term against the distribution of successful chunks. Writing the marginal distribution of the student over observations as

$$\bar{q}_\theta(a) = \mathbb{E}_o[q_\theta(a \mid o)], \tag{96}$$

the local field is the usual per-context drifting field and the success field is its global counterpart,

$$\begin{aligned} V_{\text{GT}}(a, o) &= \nabla_a \log p_{\text{GT}}(a \mid o) - \nabla_a \log q_\theta(a \mid o), \\ V_{\text{succ}}(a) &= \nabla_a \log p_{\text{succ}}(a) - \nabla_a \log \bar{q}_\theta(a). \end{aligned} \tag{97}$$

The two are combined with a single coefficient,

$$\mathcal{L} = \mathcal{L}_{\text{GT}} + \beta_{\text{success}} \mathcal{L}_{\text{succ}}, \qquad \mathcal{L}_{\text{succ}} = \mathbb{E}_{a \sim \bar{q}_\theta}\left[\left\| a - \text{sg}(a + \eta V_{\text{succ}}) \right\|^2\right]. \tag{98}$$

The sweep in Figure 11 suggests the chunk-level signal is a weak substitute for the conditioned one. Over a broad range of β_{success} the success rate stays close to the Drift baseline. A global success field can therefore be added without breaking training, but it brings no substantial gain to one-step

drifting distillation. Whether a chunk succeeds appears to remain a function of the observation it was generated for. Marginalizing the success signal over observations discards much of what makes it useful, and the per-context (i.e., episode) form DriftOPD uses is what carries it.

E Limitations and Future Work

Our current instantiation uses a critic trained only from demonstrations, which allows DriftOPD to remain fully rollout-free. While an ideal critic can exactly provide the future-gradient contribution of the sequence-level reverse KLD objective, the practical critic provides an offline approximation whose action-gradient quality is limited by demonstration coverage. This limitation is particularly evident when the one-step teacher already matches its multi-step counterpart, as in several LIBERO and LIBERO-Plus settings (Tables 23 and 24), where the current critic provides little additional improvement beyond the teacher/student baseline. Importantly, this behavior reflects the limitation of the learned critic rather than an intrinsic ceiling of the sequence-level OPD formulation. A stronger critic with broader coverage could provide more informative guidance toward action sequences that improve beyond the multi-step policy, motivating future work on Q-functions that more accurately capture how the current action chunk influences subsequent task progress.

Finally, although we study continuous action-chunk VLAs, the proposed sequence-level OPD formulation is not restricted to this setting. For autoregressive VLAs that predict discrete actions or action tokens, step-wise conditional probabilities are directly available, making the local reverse-KL term and its future contribution potentially more direct to instantiate than for continuous flow-based action experts. Extending sequence-level OPD to such next-action prediction models is therefore a natural direction for future work.