OUTCOME OR PROCESS? RETHINKING SUPERVISION FOR FLOW-MATCHING VLA POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching vision-language-action (VLA) policies generate actions through a solver trajectory, repeatedly evaluating a conditional velocity field at its intermediate states. This raises a fundamental question for policy distillation: should a student imitate a sampled teacher action, or the field that generates it? Distilling the teacher's final action chunk yields conditional flow-matching targets, but a single action chunk does not uniquely specify the teacher's local field or its intermediate computation. We study this choice with Path-wise Velocity Distillation (PVD), which regresses the frozen teacher's velocity at detached states along the student's own solver trajectory. Under matched teacher–student checkpoints and training exposure, PVD outperforms DAgger with action-chunk labels by 9.9 and 10.1 percentage points on ManiSkill at four- and eight-step deployment, 12.0 points on Meta-World, and 0.180 subtasks per chain on CALVIN. A one-seed ManiSkill ablation attributes more of the gain to direct teacher-velocity targets than to placing them on the student's solver trajectory. The method fails when trained and deployed at two solver steps. Under this matched setup, velocity labels on the student's own solver states yield better students than action-chunk labels.

1 INTRODUCTION

Vision-language-action (VLA) policies increasingly adopt language-model post-training: a model pretrained on broad robot data is adapted to downstream tasks (Black et al., 2025a;b), typically through supervised fine-tuning or reinforcement learning (Li et al., 2025; Chen et al., 2025; Lu et al., 2025; Tan et al., 2025; Liu & Yang, 2026). A newer paradigm uses a frozen teacher to supervise the student's own rollouts (Zhong et al., 2026). This extends DAgger's expert labeling of learner-visited states (Ross et al., 2011) and language-model on-policy distillation, where the teacher scores student-generated tokens (Agarwal et al., 2024; Yang et al., 2025; Kimi Team et al., 2026). VLA-OPD retains token-level labels for tokenized actions (Zhong et al., 2026), but many VLAs use flow-matching action heads (Black et al., 2025b; NVIDIA et al., 2025; Shukor et al., 2025), leaving what the teacher should label unresolved.

A flow-matching head generates an action chunk by integrating a velocity field through intermediate solver states (Lipman et al., 2023). At a student-visited observation, a frozen teacher can either return its final action chunk, as in DAgger (Ross et al., 2011), or provide velocities along the student's solver trajectory, as in flow distillation for image generators (Chen et al., 2026; Fang et al., 2026; Li et al., 2026).

The distinction is deeper than supervision density. A finished action chunk is the integrated outcome of the teacher's computation; a velocity query exposes the teacher's local response to an observation, an action candidate, and a solver time. Path-wise Velocity Distillation (PVD) asks the student to match the teacher's returned velocities at the states the student's own solver visits.

Robot distillation work supervises students with joint video and action targets (Yang et al., 2026), adversarial and action-target channels (Wan et al., 2026), or teacher-trajectory imitation (Jeon et al., 2026), but, to our knowledge, none holds teacher, student, and training protocol fixed while varying the label; this paper runs that comparison. We find that direct velocity supervision performs better than action-chunk labels on all three benchmarks and in every seed on the primary metric, by about ten success-rate points on ManiSkill and twelve on Meta-World; on CALVIN ABC-D, the velocity-supervised student is numerically comparable to the frozen teacher in completed subtasks.

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.

Our teachers are public RL-tuned $\pi_{0.5}$ checkpoints, paired with public SFT Bases (Black et al., 2025a; Chen et al., 2025; Yu et al., 2025). By design, teacher and student share one architecture and size, including the flow-matching action expert, and the student starts from the Base paired with each teacher. Under PVD, the student acts and runs its solver, while the teacher supplies batched velocities at every solver state (Figure 1). The DAgger baseline shares the teacher, initialization, collection schedule, and training exposure; only the target source, supervision density, and query-state location differ, as examined in Section 4.3.

We make three contributions:

1. **A matched comparison of two teacher labels.** We ask which label a frozen teacher should give at a student-visited observation: its finished action chunk, or its velocity at the states the student's own solver visits. Both labels train a velocity predictor; only the second supplies the teacher's response at the queried state (Section 3.2).
2. **Consistent gains and a one-seed decomposition.** Across three architecture-matched teacher–student pairs, direct velocity supervision improves downstream behavior over action-chunk labels. A one-seed ManiSkill ablation varies supervision density, target source, and query-state location in sequence. The switch to teacher-velocity targets accounts for more of the gain than placing the queries on the student's solver trajectory.
3. **A supervision analysis.** Action-chunk targets identify the linear-interpolant field of the teacher's action distribution over the rollout noise and are one-sample estimates of it. Velocity targets identify the teacher's own field at the queried states with no sampling noise, and a student with zero velocity loss reproduces the teacher's solver states (Section 3.3). Solver-step results locate where the measured advantage holds (Section 4.4).

2 Related Work

DAgger asks an expert to label the states the learner visits, which mitigates covariate shift (Ross et al., 2011); DART instead injects noise into the expert's demonstrations (Laskey et al., 2017). DAgger fixes where the label is taken, not what it is. AggreVaTe changes the label from an action to the expert's cost-to-go (Ross & Bagnell, 2014; Sun et al., 2017). A gate can decide when the expert is asked (Kelly et al., 2019; Hoque et al., 2022; Lee et al., 2025). A person's correction can enter a flow-matching policy as a residual velocity or an initial-noise target (Welte et al., 2026; Murray et al., 2026; Lu et al., 2026). These works motivate learner-state coverage and human intervention, but do not settle which model-level response a frozen flow teacher should provide at an internal solver query.

The expert can also be a trained policy. Policy distillation and its robot successors train the student on the teacher's outputs (Rusu et al., 2016; Chen et al., 2020; Xu et al., 2024; Shi et al., 2026; Wan et al., 2026). On-policy distillation of language models matches the teacher's token distribution on the student's own sequences (Agarwal et al., 2024). VLA-OPD keeps this match, since its VLA emits action tokens (Zhong et al., 2026). A deterministic flow-matching head, as used here, exposes no tractable token-like likelihood (Chen et al., 2025). WAM-OPD instead labels the student with the teacher's video and action targets (Yang et al., 2026).

Early few-step distillation took its targets from teacher sampling trajectories or reflow noise–sample pairs (Salimans & Ho, 2022; Liu et al., 2023). Score distillation uses the teacher's score at noised versions of the student's own samples, with different objectives (Poole et al., 2023; Luo et al., 2023; Yin et al., 2024). π-Flow regresses the student onto the teacher's velocity at the student's own sampling states with an ℓ_2 loss (Chen et al., 2026). Later work derives related objectives from KL formulations and varies the queried states and the target (Fang et al., 2026; Li et al., 2026; Zhou et al., 2026; Wei et al., 2026). FlowCTS matches student and reference trajectories started from student-visited states (Ye et al., 2026). None of these students acts in an environment. In robotics, distillation has reduced sampling cost with consistency and distribution-matching objectives (Prasad et al., 2024; Wang et al., 2025), and SnapFlow uses self-derived shortcut-velocity targets for one-step VLA generation (Luan et al., 2026). Shallow-π distills a flow-based VLA into a shallower network, and its ablations include teacher-trajectory imitation (Jeon et al., 2026). Wan et al. (2026) distill robot policies on-policy from an expressive teacher, with an action-target channel beside an adversarial one. Trajectory-consistent flow matching aligns pointwise velocity training with integrated inference in robot policies, without a teacher (Ahmed et al., 2026). In machine translation, ImitKD compares

distillation variants that differ in the source of the training states and in the token-level loss (Lin et al., 2020); Section 4.3 separates analogous factors for a flow-matching policy.

3 METHOD

3.1 SETTING: FLOW-MATCHING ACTION GENERATION AND THE TWO CHECKPOINTS

A frozen teacher supervises a flow-matching student on the rollouts the student itself produces, and the question is what the teacher should label. For a flow-matching head the label can be placed at two points, the finished action chunk or the intermediate states of the solver run that produces it. Every policy in this paper is a $\pi_{0.5}$-class vision-language-action policy (Black et al., 2025a). A vision-language backbone feeds a flow-matching action expert, which generates an action chunk. The observation o consists of camera images, the language instruction, and the robot's proprioceptive state. Let x denote the action-chunk variable, a matrix of H future actions of dimension D. Appendix B lists H and D for each benchmark. A policy with parameters θ represents a conditional velocity field $v_\theta(o, x, \tau)$, where τ is the solver time. The solver starts from a noise sample $x_0 \sim \mathcal{N}(0, I)$. The action chunk is generated by solver steps on a uniform grid $0 = \tau_0 < \cdots < \tau_K = 1$, as in π_0 (Black et al., 2025b),

$$x_{k+1} = x_k + \Delta\tau_k v_\theta(o, x_k, \tau_k), \qquad \Delta\tau_k = \tau_{k+1} - \tau_k. \tag{1}$$

The sequence $x_0, \ldots, x_K$ is the solver trajectory, and K is the number of solver steps. The executed actions are read from the final solver state x_K by undoing the action normalization and keeping the executed prefix (Appendix B).

Each benchmark has a public supervised-fine-tuned checkpoint, the Base, released with RLinf (Yu et al., 2025). The teacher is the RL-tuned checkpoint released alongside it (Chen et al., 2025). The student starts from the Base. The teacher and student share one architecture and size. This pairing is fixed by design so that the two procedures differ in the label alone and the student's solver states begin where the teacher was trained to answer. Training uses eight solver steps on ManiSkill and CALVIN and five on Meta-World.

3.2 THE SHARED LOOP AND THE TWO LABELS

Both procedures run the same loop. The student acts in the environment with its current parameters. For each action chunk it records the observation o^S, runs its own solver from the rollout noise, and executes the resulting chunk. In training, the rollout noise is one fixed vector for every chunk on ManiSkill and a Gaussian draw seeded per update on Meta-World and CALVIN; evaluation draws Gaussian noise (Table 7). The frozen teacher then labels that chunk in one of two forms, and we name the two procedures by their label. Under DAgger the label is the teacher's own finished action chunk. Under PVD it is the teacher's velocity at each of the student's intermediate solver states (Figure 1). Regressing this label is the ℓ_2 velocity imitation of π-Flow (Chen et al., 2026); here the queried states are the student's own detached solver states on a fixed uniform grid, the teacher is a same-size RL-tuned checkpoint of the same architecture, and the student's chunk is executed in the environment. Both procedures follow Ross et al. (2011) in that the expert labels the states the current student visits, with $\beta = 0$ and each chunk used once.

Under DAgger, the teacher starts from the student's rollout noise, runs its deterministic K-step solver, and returns the finished action chunk a^T; over that noise, a^T has distribution $p_T(\cdot \mid o^S)$, the teacher's action distribution on Meta-World and CALVIN and a point mass on ManiSkill, where a^T is a deterministic function of o^S. The teacher's information originates in this action chunk, but the student is trained by conditional flow matching rather than by an end-to-end loss between the student's and the teacher's finished action chunks. With one Gaussian sample x_0, drawn independently of the rollout noise, and one interpolation coordinate τ per chunk, the student evaluates its velocity predictor at $x_\tau = (1-\tau)x_0 + \tau a^T$ with target $a^T - x_0$,

$$\mathcal{L}_{\text{DAgger}} = \left\| v_\phi(o^S, x_\tau, \tau) - (a^T - x_0) \right\|_2^2, \qquad x_\tau = (1-\tau)\, x_0 + \tau\, a^T, \tag{2}$$

where τ follows the implemented schedule and a^T is in the model's normalized action space (Appendix B). The teacher solver is run once per chunk to produce this label.

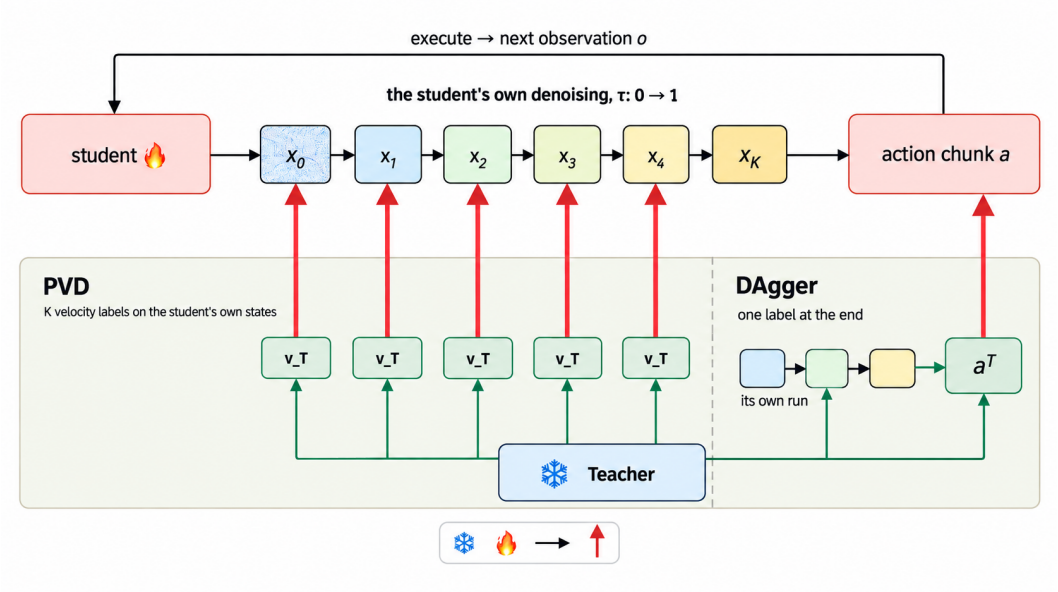


Figure 1: Same frozen teacher, same on-policy loop, two places for the label. Top: the student generates each action chunk by denoising on its own solver trajectory, $x_0 \to \cdots \to x_K$, executes the chunk, and the next observation o^S comes from its own rollout. Bottom: the teacher can label that solver trajectory in one of two ways. Under DAgger (right), the teacher runs its own solver and hands back one finished chunk a^T, a single label at the end of the solver trajectory. Under PVD (left), the teacher is queried at each of the student's own states (o^S, x_k, τ_k) and returns K velocity labels v_T along the solver trajectory; no teacher solver is run.

Under PVD, the student's current solver trajectory is sampled again as the student's parameters change. PVD queries the teacher velocity $v_T(o^S, x_k^S, \tau_k)$ at each of the K student-generated solver states and matches those values with a stop-gradient loss,

$$\mathcal{L}_{\text{path}} = \frac{1}{K} \sum_{k=0}^{K-1} \left\| v_\phi(o^S, \text{sg}[x_k^S], \tau_k) - \text{sg}\left[v_T(o^S, x_k^S, \tau_k)\right] \right\|_2^2 . \tag{3}$$

At the current update, gradients pass through neither the sampled solver trajectory nor the frozen teacher. Algorithm 1 in Appendix B lists one training step.

The two objectives differ in three ingredients: (i) supervision density, (ii) target source, and (iii) query-state location. Section 4.3 isolates them one at a time in the available seed-0 ablations. In the main comparison, both procedures make K teacher network evaluations (NFE) per chunk; PVD batches its queries, which lowers only the number of executed calls (Section 4.4).

3.3 What the Two Labels Identify

Identification. Over functions of (o, x, τ), the population minimizer of $\mathcal{L}_{\text{DAgger}}$ is the conditional mean of its target. For $\tau < 1$,

$$u^\star(o, x, \tau) = \mathbb{E}\left[a^T - x_0 \,\middle|\, x_\tau = x,\, o\right] = \frac{\mathbb{E}_{a \sim p_T(\cdot|o)}\left[w_\tau(x, a)\,(a - x)\right]}{(1-\tau)\,\mathbb{E}_{a \sim p_T(\cdot|o)}\left[w_\tau(x, a)\right]}, \tag{4}$$

with $w_\tau(x, a) = \exp\big(-\|x - \tau a\|_2^2/(2(1-\tau)^2)\big)$. This is the marginal velocity field of the linear interpolant between $\mathcal{N}(0, I)$ and $p_T(\cdot \mid o)$ under independent coupling (Lipman et al., 2023; Liu et al., 2023). Its flow carries $\mathcal{N}(0, I)$ to $p_T(\cdot \mid o)$. The field itself equals v_T only if v_T is already this field for the teacher's own samples. $\mathcal{L}_{\text{path}}$ is zero exactly when $v_\phi = v_T$ at every queried state, so its minimizer is v_T itself there. Run with a K-step solver, a student follows the field it is trained on: the DAgger student is pulled toward the flow of $u^\star$, and, by induction on k, a student with zero path loss reproduces the teacher's solver states x_k^T from the same noise.

Estimation. With one (x_0, τ) per chunk, the DAgger target $a^T - x_0$ is an unbiased one-sample estimate of $u^\star(o^S, x_\tau, \tau)$ with conditional variance $\mathrm{Var}[a^T - x_0 \mid x_\tau, o^S]$. At $\tau = 0$ we have $x_\tau = x_0$, and this variance is $\mathrm{Var}[a^T \mid o^S]$, the full spread of $p_T(\cdot \mid o^S)$; the implemented schedule has its highest density at $\tau = 0$ and draws about two thirds of its τ below $1/2$. The PVD target $v_T(o^S, x_k^S, \tau_k)$ is a deterministic function of the query and has conditional variance zero. On ManiSkill, $p_T(\cdot \mid o^S)$ is a point mass at a^T, so $u^\star(o^S, x, \tau) = (a^T - x)/(1 - \tau)$, the DAgger target also has conditional variance zero, and the target switch in Section 4.3 changes identification only. On Meta-World and CALVIN the two labels differ in identification and variance together. A control that separates them there would replace both expectations in (4) with averages over M teacher samples per query; the resulting target has limit $u^\star$, and its variance vanishes as M grows. On ManiSkill every teacher sample is a^T, so this control reduces to DAgger.

Solver grid. Write $r_k = (v_\phi - v_T)(o^S, x_k^S, \tau_k)$, so $\mathcal{L}_{\mathrm{path}} = \frac{1}{K}\sum_k \|r_k\|_2^2$. From shared noise, the student's state after the first step departs from the teacher's by $x_1^S - x_1^T = \frac{1}{K} r_0$. With a finer grid, the first query after the noise lies closer to the teacher's solver trajectory; at $K = 2$ the single interior query sits at $\tau = 1/2$, a distance $\frac{1}{2}\|r_0\|_2$ from the teacher's solver state (Section 4.4).

Remark. Proposition 1 (Appendix D) bounds the discrepancy between the coupled final solver states, $\|x_K^S - x_K^T\|_2$, by the residuals along the student's solver trajectory under an L-Lipschitz teacher field. The converse fails: agreement of the final action chunks leaves the intermediate states free. The velocity label ties the student's intermediate solver states to the teacher's, and the action-chunk label does not.

4 Experiments

4.1 Setup

Benchmarks and models. ManiSkill-25 Main is 25 object-to-plate pick-and-place variants on ManiSkill3 (Tao et al., 2024), evaluated on 320 fixed resets. CALVIN ABC-D is evaluated on 1,000 five-subtask language chains of the official D split (Mees et al., 2022). Meta-World MT50 is 50 manipulation tasks with ten variants each (Yu et al., 2020), evaluated on 500 paired episodes. Each benchmark has its own public RLinf $\pi_{0.5}$ supervised-fine-tuned Base and RL-tuned FlowSDE teacher (Black et al., 2025a; Chen et al., 2025; Yu et al., 2025), and the three pairs are independent. Table 7 in Appendix B lists all settings.

Training protocol. Both students share the frozen Base and teacher and act with their own current policy, so they visit different states once their parameters diverge. Each student trains with global batch size 8. Exposure, the matched unit, is 80,000 newly collected chunks per student, each used once, and 10,000 AdamW updates. Action format, normalization, trainable parameters (the action expert only), checkpoint cadence, and evaluation set are shared. Compute is reported as a cost in Section 4.4 and Appendix A. All runs use one eight-GPU node.

Metrics and uncertainty. ManiSkill reports `success_once` (success at any step) and, secondarily, `success_at_end` (success at the final step). CALVIN reports mean successful chain length and completion rates for one to five consecutive subtasks; Meta-World reports success rate. Students are deployed at their training step count and, on ManiSkill, also at four steps. Base and teacher are single frozen models. The main comparison reports per-seed effects, their mean, sample SD, and an exploratory 95% seed-level t interval, using three seeds on ManiSkill and CALVIN and two on Meta-World (where SD is descriptive only). Unless noted, ablations, controls, and solver-step experiments use ManiSkill seed 0, and student results are measured at 80k chunks. Differences use unrounded success counts and may differ by 0.1 from rounded table entries.

4.2 PVD versus DAgger on Three Benchmarks

On ManiSkill, PVD leads DAgger by $+9.9$ and $+10.1$ points at four- and eight-step deployment, averaged over three seeds (Table 1, Figure 5 in Appendix C). PVD leads in every seed at both deployment step counts, and the smallest single-seed difference is $+7.8$ points. Both seed-level 95% intervals exclude zero (Table 8 in Appendix C). Averaged over the five checkpoints of each run, the three-seed gap is $+10.4 \pm 0.4$ points at four-step deployment and $+12.5 \pm 0.4$ at eight. The three-seed PVD mean at four-step deployment, 85.7%, is 1.8 points below the frozen teacher's 87.5%. With

each procedure at its best training step count in seed 0 (four steps for both) and both evaluated on the same node, the gap at four-step deployment is $+2.2$ points, 86.6% against 84.4% (Section 4.4).

Table 1: Both procedures run the same on-policy loop and differ in the label. Main result at 80k on ManiSkill and Meta-World MT50. Students are trained at eight solver steps on ManiSkill and five on Meta-World. ManiSkill cells are success rate on the original 320-reset evaluation set at four- and eight-step deployment. Meta-World cells are success rate over 500 paired episodes at five steps. Student rows are mean $\pm$ SD over three training seeds on ManiSkill and over two on Meta-World, where the SD is descriptive. The Difference row is the mean $\pm$ SD of the per-seed differences. Base and teacher are single frozen models. The ManiSkill teacher's eight-step value was measured later on the same set, where a repeat of the four-step evaluation gave 279/320 (87.2%) against the original 280/320. Best student in bold. Per-seed ManiSkill differences are in Table 8. CALVIN is in Table 2.

	ManiSkill, trained at 8 steps		Meta-World, trained at 5
Variant	Deployed at 4	Deployed at 8	Deployed at 5
Base	60.9	53.8	55.0
DAgger	75.8 ± 2.1	72.7 ± 2.1	60.7 ± 6.4
PVD	$\mathbf{85.7 \pm 1.2}$	$\mathbf{82.8 \pm 0.3}$	$\mathbf{72.7 \pm 3.3}$
Difference	$+9.9 \pm 2.6$	$+10.1 \pm 1.8$	$+12.0 \pm 3.1$
teacher	87.5	84.4	76.4

On Meta-World MT50, with both students trained and deployed at five solver steps, PVD leads DAgger by $+12.0$ points over two seeds (Table 1). The per-seed gains are $+9.8$ and $+14.2$ points. PVD improves 16 of 50 tasks in both seeds and is no worse on 39. One task regresses in both seeds, and two stay at zero for both students. A paired task-cluster bootstrap within each seed excludes zero, $[+4.4, +15.6]$ points in seed 0 and $[+8.0, +21.0]$ in seed 1. Of 1,000 paired episodes over both seeds, 179 succeed under PVD alone and 59 under DAgger alone.

On CALVIN ABC-D, a chain of five instructions ends at its first failure (Mees et al., 2022). PVD leads DAgger in every seed, by $+0.180$ subtasks per chain on average (Table 2). The seed-level interval is $[+0.14, +0.22]$ (per-seed values in Table 11 in Appendix C). A paired sequence bootstrap within each seed excludes zero (seed-0 pairs in Table 12). The mean per-position gain goes from $+1.6$ points for one subtask to $+5.4$ for all five (Figure 10 in Appendix C). In every seed, the five-subtask gain is at least three times the one-subtask gain. The three-seed PVD mean of 4.479 is numerically comparable to the frozen teacher's 4.474, with every seed within 0.03. Appendix E has fourteen paired rollouts on the three benchmarks, including a seed-0 chain that DAgger fails at the first subtask and PVD completes (Figure 11).

Table 2: CALVIN ABC-D at 80k, official-D evaluation set of 1,000 five-subtask chains. Average number of subtasks completed in a row, and the rate of completing the first n subtasks in a row for $n = 1$ to 5. Student rows are mean $\pm$ SD over three training seeds. The Difference row is the mean $\pm$ SD of the three per-seed differences. Base and teacher are single frozen models. Best student in bold. Per-seed values are in Table 11 in Appendix C.

		Subtasks completed in a row, %				
Variant	Avg. length	1	2	3	4	5
Base	3.746	92.6	82.7	73.9	67.0	58.4
DAgger	4.299 ± 0.010	97.2 ± 0.3	92.4 ± 0.3	86.2 ± 0.2	80.7 ± 0.4	73.4 ± 0.5
PVD	$\mathbf{4.479 \pm 0.025}$	$\mathbf{98.8 \pm 0.2}$	$\mathbf{95.0 \pm 0.5}$	$\mathbf{90.1 \pm 0.7}$	$\mathbf{85.2 \pm 0.9}$	$\mathbf{78.8 \pm 0.3}$
Difference	$+0.180 \pm 0.016$	$+1.6 \pm 0.2$	$+2.6 \pm 0.4$	$+3.9 \pm 0.5$	$+4.5 \pm 1.0$	$+5.4 \pm 0.4$
teacher	4.474	98.7	95.3	90.0	84.7	78.7

On ManiSkill, the `success_at_end` difference follows the `success_once` difference at eight-step deployment, $+10.0$ points, but is -1.5 at four, with an interval that crosses zero (Table 8). In the later teacher evaluations on one node, the teacher falls from 87.2% `success_once` to 62.8% `success_at_end` at four steps, a loss of 24.4 points against 9.7 at eight, and both students exceed

that `success_at_end` (69.7% for DAgger, 68.2% for PVD). On 320 held-out resets disjoint from the original set, the three-seed mean gain is +9.9 points at four steps and +9.1 at eight. On a stratified set of 1,024 fixed object, receptacle, and scene combinations, the three-seed mean gains are +15.5 and +14.0 (Figure 9 in Appendix C). An out-of-distribution evaluation follows the π_{RL} ManiSkill protocol (Chen et al., 2025), with seed-0 students, eight-step deployment, and one evaluation seed. Over its six test conditions (300 episodes), PVD reaches 51.7%, DAgger 39.7%, the Base 29.3%, and the teacher 51.7% (Table 13 in Appendix C).

4.3 One-Factor Ablations and Offline Controls

The main comparison changes supervision density, teacher signal, and query states (Section 3.2). ManiSkill seed-0 controls vary these factors in sequence (Table 3); all train from the Base with an eight-step solver and current-student rollouts. At 80k chunks, PVD leads DAgger by +12.8 points at four-step deployment and +12.2 at eight. A 320-episode evaluation near 80% success has a binomial standard error of about 2.2 points.

Table 3: ManiSkill seed-0 ablations at 80k. Each of rows two to four changes one factor from the row above; the second row also moves the interpolation times from the DAgger schedule to the solver grid $\tau_k = k/8$. Executed calls are per 80k chunks; PVD batches its queries. GPU-h is the full 80k run on one node (Table 5). The last row uses teacher-rollout observations. Best per deployment step count is bold.

Variant	Points per chunk	Teacher signal	States	Executed calls	GPU-h	Deploy 4	Deploy 8
DAgger	1	action chunk	interpolant	640k	64.50	73.4	70.3
Eight-point control	8	action chunk	interpolant	640k	81.91	74.1	76.3
Interpolant-velocity control	8	velocity	interpolant	720k	82.07	81.9	**82.5**
PVD	8	velocity	solver trajectory	80k, batched	75.50	**86.3**	**82.5**
teacher-rollout BC	1	action chunk	interpolant	640k	65.60	77.5	72.8

Supervision density. The eight-point control retains the teacher's action-chunk label but evaluates its conditional flow-matching target at eight interpolant times per chunk, the solver-grid times $\tau_k = k/8$ of PVD, rather than at one time drawn from the DAgger schedule. At 80k chunks, it exceeds DAgger by +0.6 points at four-step deployment and +5.9 at eight-step deployment. Across the five checkpoints, PVD leads the eight-point control by an average of +9.5 points at four steps and +7.6 at eight (Figure 2 and Table 10 in Appendix C). Additional queries of the action-chunk target can help, particularly at eight-step deployment, but do not close the gap to direct velocity supervision in this control.

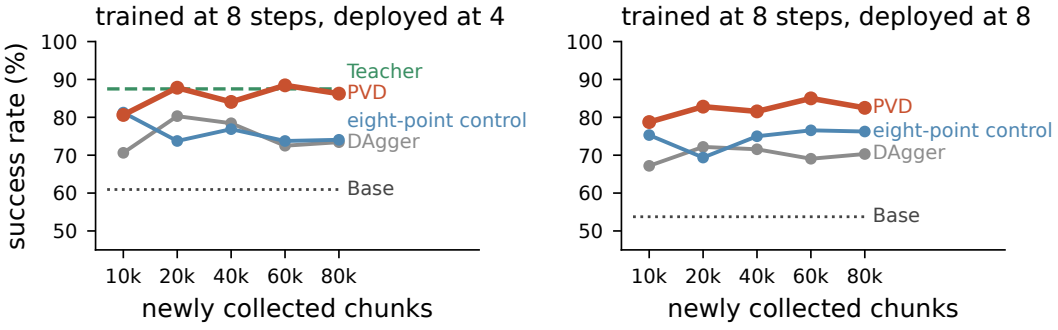


Figure 2: Success rate over training for ManiSkill seed 0. All students train with eight solver steps and deploy with four (left) or eight (right). PVD exceeds DAgger at every checkpoint. The eight-point control exceeds PVD only at the first four-step-deployment checkpoint. The dashed line is the frozen teacher at four steps (its eight-step value is 84.4%, Table 1), and the dotted line is the untrained Base. The interpolant-velocity control was evaluated at 80k only (Table 3).

Teacher signal and query states. The interpolant-velocity control retains the eight interpolant queries of the eight-point control but replaces their action-chunk targets with direct evaluations of the teacher's velocity field. This change adds $+7.8$ points at four-step deployment and $+6.3$ at eight, accounting for more than half of the respective DAgger-to-PVD gaps at 80k. The next change, from interpolant queries to the student's own solver states, adds $+4.4$ points at four-step deployment and none at eight; these two placement increments differ by about 1.5 standard errors. The target switch accounts for more of the gain than placement on the student's solver trajectory at both deployment step counts. At eight-step deployment, the $+6.3$-point target increment is numerically close to the $+5.9$-point density increment. The action-chunk target at the student's solver states was not evaluated.

Moving the velocity queries to the student's solver trajectory also removes the teacher's sequential solver run. Relative to the interpolant-velocity control, executed teacher calls fall from 720,000 to 80,000 batched calls, and full-node training cost falls by 8%, from 82.07 to 75.50 GPU-hours (Table 3).

Online collection. The frozen-rollout control trains both students on the same buffer of 20,000 observations collected by the frozen Base, so neither training procedure collects new observations with its updated student. At 20k, PVD leads DAgger by $+7.8$ points at four-step deployment and $+15.6$ at eight (Table 10 in Appendix C). The corresponding leads under online collection at that checkpoint are $+7.5$ and $+10.6$ points (Table 6 in Appendix A). Here the advantage does not require online collection.

The teacher's own rollout states. The teacher-rollout BC trains a student from the Base offline on observations generated by the teacher, labeled with the teacher's final action chunk. It reaches 77.5% at four-step deployment and 72.8% at eight, compared with 73.4% and 70.3% for DAgger (Table 3). Under this seed-0 action-chunk comparison, collecting observations with the current student provides no measured advantage over using teacher-rollout observations. PVD still exceeds the teacher-rollout BC by $+8.8$ and $+9.7$ points at four and eight steps. Together with the frozen-rollout control, this argues against attributing the measured PVD advantage to online environment collection alone.

4.4 Solver Steps and Cost

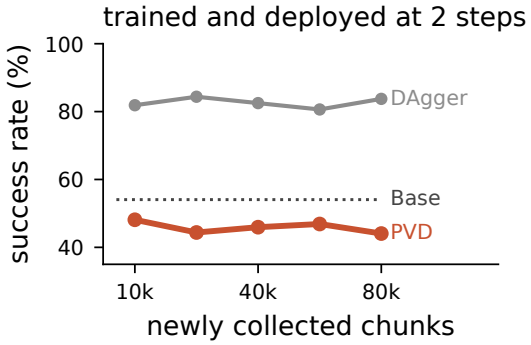


Figure 3: Both students trained and deployed at two solver steps, ManiSkill seed 0. PVD falls below the untrained Base at every checkpoint, while DAgger stays above 80%.

The gain depends on the training step count, with both students trained and deployed at the same count. At eight steps PVD leads by $+12.2$ points in seed 0 (82.5% against 70.3%) and by $+10.1$ over three seeds. At four steps it leads by $+2.2$ points at 80k (86.6% against 84.4%) and by $+2.2$ to $+5.9$ over five checkpoints. This pair was evaluated on a second node, where repeat evaluations of one checkpoint differ by 0.6 to 2.5 points. The DAgger teacher solves at eight steps in the main comparison and at four in the four- and two-step runs. Evaluated from the fixed training noise, the teacher reaches 86.6% at eight steps and 85.6% at four, so the DAgger labels come from a teacher at its reported level. Compared across the two nodes, DAgger trained at four steps is 10.9 points above DAgger trained at eight and deployed at four (73.4%). The four-step-trained and eight-step-trained PVD students differ by 1.6 points at four-step deployment on the same node (84.7% against 86.3%, Figure 7 in Appendix C); the 84.7% comes from the same four-step-trained 80k checkpoint that scored 86.6% on the second node. At two steps the teacher itself reaches 34.1%. PVD, whose

zero-loss solution reproduces the teacher's solver states, stays at 44–48% at every checkpoint, below the untrained Base's 54.1% at two steps. DAgger, labeled by the teacher's four-step solver, stays above 80% (Figure 3); the 80k gap is -39.7 points. Two-step deployment alone does not account for the failure, since DAgger deploys well there.

Across deployment step counts, the eight-step-trained PVD student on ManiSkill reaches 86.3%, 85.6%, 82.5%, and 81.6% at four, six, eight, and sixteen steps (Table 9 in Appendix C). Moving deployment from four to eight steps lowers success rate by 3.8 points for both training step counts (Figure 7 in Appendix C). On Meta-World, both seed-0 students train at five steps. PVD reaches 66.4%, 75.0%, 49.6%, and 55.8% at four, five, eight, and ten steps. DAgger reaches 65.6% at four and 65.2% at five and eight. The Base's 55.0% is measured at five steps only.

Both students use 640k teacher NFE per 80k chunks on ManiSkill and 400k on Meta-World. DAgger makes one sequential call per teacher solver step. PVD batches a chunk's queries into one call and so makes K times fewer teacher calls. PVD still spends more full-node GPU-hours on all three benchmarks, $+17\%$ on ManiSkill, $+41\%$ on CALVIN, and $+25\%$ on Meta-World (Table 5 in Appendix A). At matched GPU-hours on ManiSkill in seed 0, every PVD checkpoint from 20k on exceeds every DAgger checkpoint up to 80k. The comparison holds at both deployment step counts (Table 6 in Appendix A). The eight-point control spends 81.91 GPU-hours and trails PVD at every checkpoint except 10k at four-step deployment.

5 Discussion

Both procedures train a velocity predictor, but from different teacher targets: the DAgger baseline uses a sampled action chunk to construct a conditional flow-matching target, whereas PVD uses the teacher's velocity at states visited by the student's solver. Under the matched protocol, PVD wins in every seed on the primary metric, by about ten percentage points on ManiSkill, twelve on Meta-World MT50, and 0.18 subtasks per CALVIN ABC-D chain on average. In the ManiSkill seed-0 ablation, the switch to teacher-velocity targets accounts for the larger share of this gap, and placement on the student's solver trajectory for a smaller one.

The advantage has empirical and explanatory limits. On ManiSkill seed 0, it falls from $+12.2$ points with eight-step training and deployment to $+2.2$ at four steps and -39.7 at two (Section 4.4). At eight and four steps PVD comes within about two points of the teacher, 82.8% (three-seed mean) against 84.4% and 86.6% (seed 0) against 87.2%, with the teacher values from the later evaluations on one node. At two steps, where the teacher's own solver reaches 34.1%, PVD also fails and falls below the Base, consistent with the Meta-World drop at eight-step deployment (Table 9). It needs a teacher whose solver works at that step count. Why the velocity label wins is open. Target variance does not explain the ManiSkill gain, since both targets have zero conditional variance there (Section 3.3); PVD also leads on Meta-World and CALVIN, where the action-chunk target has variance. Teacher and student share an architecture and size, so this study isolates the choice of label. It is limited to simulation, and PVD uses fewer teacher calls but more total GPU-hours. Privileged teachers and real-robot transfer remain unevaluated.

PVD supervises the student with the teacher's velocity responses at the student's own solver states during training. DAgger labels the states the student visits in the environment; PVD does the same one level down, at solver states. At a different deployment step count, the student visits solver states that were never labeled; this is a distribution shift in solver states. Our hypothesis is that the labels constrain the student only where queries are dense and inside the teacher's reliable region. Both failures fit this hypothesis: the two-step grid labels only two solver states per chunk and lies outside the teacher's reliable region, where its own solver reaches 34.1%, and the Meta-World student trained at five steps meets unlabeled states at eight and falls below the Base. The same hypothesis predicts a weaker velocity label whenever the student's solver states fall outside the teacher's reliable region, as with a teacher of different lineage, a smaller student, or a privileged-input teacher. Jittered query times in training, off-grid velocity errors measured offline, and a student not initialized from the teacher's paired Base would test it.

REPRODUCIBILITY STATEMENT

Both procedures share one fixed training protocol (Section 4.1) and one evaluation protocol per benchmark (Section 4). ManiSkill, CALVIN, and Meta-World numbers are taken from archived run reports with SHA256 hashes recorded in the artifact registry. The registry, the exact ManiSkill and CALVIN evaluation sets, the per-seed result tables, and the resource records will be released with the code. Public Base and teacher checkpoints are identified by repository and revision in the released configuration files. The figure-generation script reproduces every plot from the tabulated values.

AI USE STATEMENT

In this work, we used generative AI tools (large language model agents) for the following tasks with required disclosure: implement methods (the training and evaluation pipeline was written by agents from the authors' written experiment specifications and executed on the compute nodes by agents), design or provide feedback on research methodology or experiments (agent-written review passes proposed the eight-point and interpolant-velocity controls that the authors then authorized), propose or refine hypotheses (the mechanism decomposition into target type and query-state location was refined in agent-written analyses), and interpret results (agents computed the reported statistics and drafted the run summaries from which the interpretation in this paper was developed). Generative AI tools drafted the analysis in Section 3.3, including its statements and derivations, and the proof of Proposition 1 in Appendix D, from definitions the authors supplied; the authors checked every statement, derivation, and proof step. We have not used generative AI tools to generate synthetic data sets or to assist with translation. Cleaning and reformatting data sets and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for the following tasks with recommended disclosure: draft parts of the paper under a written set of claim-evidence rules that fixed the permitted claims, their evidence, and the statistical language; edit the paper to improve readability; create the data plots from tabulated values and the schematic Figures 1, 6, and 8 with an image-generation model; create or edit software code; identify relevant literature; and format references. We have reviewed all AI-assisted work. The authors designed the research question, the method, and the matched comparison; set every training authorization and stopping rule; and reviewed every section, table, number, and reference. Every number in the paper is taken from archived run reports produced during the experiments, with SHA256 hashes recorded in the artifact registry. No reference was generated from model memory; each BibTeX entry was checked against its arXiv record or its published record by title and authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Riad Ahmed, Sujosh Nag, Moniruzzaman Akash, Mostafa Hussein, and Momotaz Begum. Trajectory-consistent flow matching for robust visuomotor policy learning. *arXiv preprint arXiv:2605.08511*, 2026. URL https://arxiv.org/abs/2605.08511.

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 *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

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*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 66–75. PMLR, 2020. URL https://proceedings.mlr.press/v100/chen20a.html. Presented at CoRL 2019.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. π-Flow: Policy-based few-step generation via imitation distillation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1vAte8dsap.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, et al. π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025. URL https://arxiv.org/abs/2510.25889.

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. URL https://arxiv.org/abs/2605.08063.

Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, Berlin, Heidelberg, 2 edition, 1993. ISBN 978-3-540-56670-0. doi: 10.1007/978-3-540-78862-1.

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S. Brown, and Ken Goldberg. ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 598–608. PMLR, 2022. URL https://proceedings.mlr.press/v164/hoque22a.html. Presented at CoRL 2021.

Boseong Jeon, Yunho Choi, and Taehan Kim. Shallow-π: Knowledge distillation for flow-based VLAs. *arXiv preprint arXiv:2601.20262*, 2026. URL https://arxiv.org/abs/2601.20262.

Michael Kelly, Chelsea Sidrane, Katherine Rose Driggs-Campbell, and Mykel J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pp. 8077–8083. IEEE, 2019. doi: 10.1109/ICRA.2019.8793698. URL https://doi.org/10.1109/ICRA.2019.8793698.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

Michael Laskey, Jonathan Lee, Roy Fox, Anca D. Dragan, and Ken Goldberg. DART: noise injection for robust imitation learning. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, volume 78 of *Proceedings of Machine Learning Research*, pp. 143–156. PMLR, 2017. URL https://proceedings.mlr.press/v78/laskey17a.html.

Sung-Wook Lee, Xuhui Kang, and Yen-Ling Kuo. Diff-DAgger: Uncertainty estimation with diffusion policy for robotic manipulation. In *IEEE International Conference on Robotics and Automation, ICRA 2025, Atlanta, GA, USA, May 19-23, 2025*, pp. 4845–4852. IEEE, 2025. doi: 10.1109/ICRA55743.2025.11127730. URL https://doi.org/10.1109/ICRA55743.2025.11127730.

Haoran Li, Yuhui Chen, Wenbo Cui, Weiheng Liu, et al. Survey of Vision-Language-Action Models for Embodied Manipulation. *arXiv preprint arXiv:2508.15201*, 2025. URL https://arxiv.org/abs/2508.15201.

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*, 2026. URL https://arxiv.org/abs/2605.15055.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6121–6133. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.494. URL https://doi.org/10.18653/v1/2020.emnlp-main.494.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Ziheng Liu and Quantao Yang. TEMPO: Semantic-Action Decoupled RL Post-Training for Vision-Language-Action Models. *arXiv preprint arXiv:2608.07314*, 2026. URL https://arxiv.org/abs/2608.07314.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, et al. VLA-RL: Towards Masterful and General Robotic Manipulation with Scalable Reinforcement Learning. *arXiv preprint arXiv:2505.18719*, 2025. URL https://arxiv.org/abs/2505.18719.

Junjie Lu, Xinyao Qin, Yuhua Jiang, Kaixin Wang, Chuheng Zhang, Bin Liang, Jun Yang, Min Xu, and Li Zhao. UniSteer: Unified noise steering for efficient human-guided VLA adaptation, 2026. URL https://arxiv.org/abs/2605.10821.

Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. SnapFlow: One-step action generation for flow-matching VLAs via progressive self-distillation. *arXiv preprint arXiv:2604.05656*, 2026. URL https://arxiv.org/abs/2604.05656.

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. doi: 10.52202/075280-3344. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f115f619b62833aadc5acb058975b0e6-Abstract-Conference.html.

Oier Mees, Lukás Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics Autom. Lett.*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108. URL https://doi.org/10.1109/LRA.2022.3180108.

Michael Murray, Daphne Chen, Simran Bagaria, Dean Fortier, Tess Hellebrekers, Galen Mullins, Harshavardhan Gajarla, Oier Mees, Maya Cakmak, and Andrey Kolobov. FlowDAgger: Human-in-the-loop adaptation of generative robot policies in latent space, 2026. URL https://arxiv.org/abs/2607.08877.

NVIDIA, Johan Bjorck, Fernando Castañeda, et al. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=FjNys5c7VyY.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency Policy: Accelerated visuomotor policies via consistency distillation. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado (eds.), *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.071. URL https://doi.org/10.15607/RSS.2024.XX.071.

Stephane Ross and J. Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning, 2014. URL https://arxiv.org/abs/1406.5979.

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*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL https://arxiv.org/abs/1511.06295.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Jin Shi, Brady Zhang, and Yishun Lu. Offline semantic guidance for efficient vision-language-action policy distillation, 2026. URL https://arxiv.org/abs/2605.16241.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, et al. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Wen Sun, Arun Venkatraman, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3309–3318. PMLR, 2017. URL https://proceedings.mlr.press/v70/sun17d.html.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive Post-Training for Vision-Language-Action Models. *arXiv preprint arXiv:2505.17016*, 2025. URL https://arxiv.org/abs/2505.17016.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, et al. ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI. *arXiv preprint arXiv:2410.00425*, 2024. URL https://arxiv.org/abs/2410.00425.

Zhenglin Wan, Jingxuan Wu, Xingrui Yu, Chubin Zhang, Mingcong Lei, Bo An, Ivor W. Tsang, and Yang You. Adversarial dual on-policy distillation from expressive teacher. In *International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/61158.

Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.),

Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025, volume 267 of *Proceedings of Machine Learning Research*, pp. 63399–63416. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25ba.html.

Qingyan Wei, Guangzhao Li, Xiaobing Tu, Yinggui Wang, Xiantao Zhang, Jinkui Ren, Xiaohong Liu, and Linfeng Zhang. STEP-OPD: Rethinking output targets and internal dynamics in on-policy distillation for diffusion models. *arXiv preprint arXiv:2608.04887*, 2026. URL https://arxiv.org/abs/2608.04887.

Edgar Welte, Yitian Shi, Rosa Wolf, Maximillian Gilles, and Rania Rayyes. FlowCorrect: Efficient interactive correction of generative flow policies for robotic manipulation, 2026. URL https://arxiv.org/abs/2602.22056.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic generalist policy distillation via reinforcement learning, 2024. URL https://arxiv.org/abs/2412.09858.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Liuhaichen Yang, Zhuang Jiang, Chenchao Sheng, and Zezhi Tang. WAM-OPD: On-policy distillation for world action models, 2026. URL https://arxiv.org/abs/2608.22364.

Kaiyang Ye, Yuan Ge, Junxiang Zhang, Bei Li, Ziming Zhu, Haishu Zhao, Xiaoqian Liu, Chenglong Wang, Jingbo Zhu, Zhengtao Yu, and Tong Xiao. FlowCTS: On-policy continuous trajectory supervision of flow models. *arXiv preprint arXiv:2607.24522*, 2026. URL https://arxiv.org/abs/2607.24522.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 6613–6623. IEEE, 2024. doi: 10.1109/CVPR52733.2024.00632. URL https://doi.org/10.1109/CVPR52733.2024.00632.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, et al. RLinf: Flexible and Efficient Large-scale Reinforcement Learning via Macro-to-Micro Flow Transformation. *arXiv preprint arXiv:2509.15965*, 2025. URL https://arxiv.org/abs/2509.15965.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020. URL https://proceedings.mlr.press/v100/yu20a.html. Presented at CoRL 2019.

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. URL https://arxiv.org/abs/2603.26666.

Wei Zhou, Xiongwei Zhu, Zelin Xu, Bo Dong, Lixue Gong, Yongyuan Liang, Meng Chu, Leigang Qu, Lingdong Kong, Wei Liu, and Tat-Seng Chua. DanceOPD: On-policy generative field distillation. *arXiv preprint arXiv:2606.27377*, 2026. URL https://arxiv.org/abs/2606.27377.

A Resource Accounting

All training and evaluation runs use one node of the configuration in Table 4. GPU-hours below are full-node GPU-hours, summed over the eight GPUs, as recorded in the run records.

Table 4: Compute environment shared by all runs.

GPUs	8× NVIDIA RTX PRO 6000D, one node
CPU	Intel Xeon Platinum 8473C
Software	PyTorch 2.11.0 built against CUDA 13.0; NVIDIA driver 595.84, CUDA 13.2

Table 5: Resource accounting for the 80k protocol, seed 0; ManiSkill, CALVIN, and Meta-World blocks from top to bottom. NFE counts teacher network evaluations; executed calls reflect batching and sequential scheduling. The interpolant-velocity control queries the teacher for the action chunk and for eight velocities per chunk, hence its larger NFE. All runs use one eight-GPU node, so wall-clock time is one eighth of the GPU-hours. The teacher-rollout BC's teacher calls are its eight-step rollouts, made before the student is trained. Peak memory is the largest per-GPU allocation during training, in GiB; the Meta-World runs and the teacher-rollout BC did not record it.

Variant	Chunks	Updates	NFE	Executed calls	GPU-h	Mem. (GiB)
DAgger	80k	10k	640k	640k	64.50	18.1
teacher-rollout BC	80k	10k	640k	640k	65.60	–
Eight-point control	80k	10k	640k	640k	81.91	19.1
Interpolant-velocity control	80k	10k	1,280k	720k	82.07	18.1
PVD	80k	10k	640k	80k (batched)	75.50	19.1
DAgger (CALVIN, K=8)	80k	10k	640k	640k	45.46	18.1
PVD (CALVIN, K=8)	80k	10k	640k	80k (batched)	63.99	18.1
DAgger (Meta-World, K=5)	80k	10k	400k	400k	26.74	–
PVD (Meta-World, K=5)	80k	10k	400k	80k (batched)	33.31	–

The other seeds cost the following, DAgger and PVD respectively: ManiSkill seed 1, 64.72 and 76.84 GPU-hours; ManiSkill seed 2, 65.85 for DAgger, with the PVD figure missing from the exported cost records; CALVIN seed 1, 43.18 and 61.04; CALVIN seed 2, 43.22 and 61.19; Meta-World seed 1, 26.47 and 33.72. The frozen-rollout control cost 11.23 GPU-hours to generate the shared buffer and 5.42 and 10.92 GPU-hours for the two 20k training runs. One Meta-World evaluation of one student takes under half a GPU-hour.

Table 6: Cumulative training GPU-hours at each checkpoint for the seed-0 ManiSkill students, with the success rate of that checkpoint at four-step and eight-step deployment, in percent. GPU-hours are measured to the checkpoint's write time and therefore fall slightly short of the full-run totals in Table 5.

	DAgger			PVD		
Checkpoint	GPU-hours	K=4	K=8	GPU-hours	K=4	K=8
10k	8.5	70.6	67.2	10.0	80.6	78.8
20k	16.6	80.3	72.2	19.3	87.8	82.8
40k	32.3	78.4	71.6	37.8	84.1	81.6
60k	48.2	72.5	69.1	56.4	88.4	85.0
80k	64.3	73.4	70.3	75.3	86.3	82.5

Table 7: Shared training protocol and per-benchmark settings. The solver is forward Euler on a uniform grid in every run. Both students on a benchmark use the same values; only the supervision target differs. The learning rate, AdamW ϵ, and clipping norm are those of RLinf's released $\pi_{0.5}$ ManiSkill PPO configuration, fixed before any evaluation and not tuned for either method.

Setting	ManiSkill-25 Main	CALVIN ABC-D	Meta-World MT50
Tasks	25 object-to-plate pick-and-place variants on ManiSkill3 (Tao et al., 2024)	five-subtask language sequences, official D split (Mees et al., 2022)	50 manipulation tasks, ten variants each (Yu et al., 2020)
Base / teacher	RLinf $\pi_{0.5}$ SFT / RL-FlowSDE	RLinf $\pi_{0.5}$ CALVIN SFT / RL-FlowSDE	RLinf $\pi_{0.5}$ Meta-World SFT / RL-FlowSDE
Student initialization	public Base	public Base	public Base
Acting policy	current student only ($\beta = 0$)	current student only ($\beta = 0$)	current student only ($\beta = 0$)
Student rollout solver	eight steps (K=8)	eight steps (K=8)	five steps (K=5)
Teacher query grid	K=8 (8 NFE per chunk)	K=8 (8 NFE per chunk)	K=5 (5 NFE per chunk)
Training rollout noise	one fixed vector for every chunk and seed: $H \times 32$ values evenly spaced over $[-1, 1]$ (`linspace`)	Gaussian, derived from the seed and the update	Gaussian, from a per-update seed
Evaluation rollout noise	Gaussian	Gaussian	Gaussian
Action horizon / executed prefix	8 / 5 actions (5×7)	5 / 5 actions (5×7)	5 / 5 actions (5×4)
Newly collected chunks / optimizer updates	80,000 / 10,000	80,000 / 10,000	80,000 / 10,000
Global batch size	8	8	8
Chunk reuse	none, each chunk used once	none, each chunk used once	none, each chunk used once
Optimizer	AdamW, lr 7.91×10^{-6}, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-5}$, weight decay 0.01, gradient clip 1.0	same as ManiSkill	same as ManiSkill
Trainable parameters	action expert only (208 tensors)	action expert only (208 tensors)	action expert only
Checkpoints	10k, 20k, 40k, 60k, 80k chunks	10k, 20k, 40k, 60k, 80k chunks; reported point 80k	10k, 20k, 40k, 60k, 80k chunks; reported point 80k
Evaluation set	320 fixed resets per checkpoint, K=4 and K=8; held-out reset set and stratified population set (three-seed means)	1,000 official-D sequences, K=8	500 paired episodes, K=5; 160-action limit, stops on first success
Primary metric	`success_once`	average successful sequence length	`success_once`
Training seeds	0, 1, 2	0, 1, 2	0, 1
Controls	eight-point, interpolant-velocity, frozen-rollout, two-step solver, teacher-rollout BC	main comparison only	deployment step count

B Training and Evaluation Settings

Flow time for the DAgger loss is drawn once per chunk as $s \sim \mathrm{Beta}(1.5, 1)$ and $\tau = 0.999\,(1 - s)$, so most of the mass of τ lies near the noise end; one Gaussian x_0 is also drawn per chunk. The eight-point control and the interpolant-velocity control evaluate their targets at the eight solver-grid times $\tau_k = k/8$, $k = 0, \ldots, 7$, which are the query times of PVD. All eight points of a chunk share one x_0 and one a^T, so they lie on a single interpolant. Actions are normalized per dimension by

the checkpoint's quantile statistics, $\tilde{a} = 2\,(a - q_{01})/(q_{99} - q_{01} + 10^{-6}) - 1$, and zero-padded to 32 dimensions. The DAgger loss is averaged over the valid entries of the chunk, with the padded dimensions excluded.

Algorithm 1 PVD training step (one new chunk; batched in practice)

Require: frozen teacher v_T, student v_ϕ initialized from Base, solver grid $\tau_0 < \cdots < \tau_K$
1: observe o^S from the current student's rollout; take the rollout noise x_0^S (Table 7)
2: **for** $k = 0, \ldots, K-1$ **do**
3: $\quad x_{k+1}^S \leftarrow x_k^S + \Delta\tau_k\, v_\phi(o^S, x_k^S, \tau_k)$ ▷ student's own solver, no gradient through the chain
4: **end for**
5: $\tilde{v}_k \leftarrow \mathrm{sg}[v_T(o^S, x_k^S, \tau_k)]$ for all k ▷ one batched teacher query
6: execute the chunk decoded from x_K^S in the environment
7: $\mathcal{L}_{\text{path}} \leftarrow \frac{1}{K}\sum_k \|v_\phi(o^S, \mathrm{sg}[x_k^S], \tau_k) - \tilde{v}_k\|_2^2$; update ϕ by one AdamW step

Table 7 lists the shared training protocol and per-benchmark settings. The Meta-World runs use RLinf commit `bde6c918`. The Meta-World Base and teacher are the public Hugging Face releases `RLinf/RLinf-Pi05-MetaWorld-SFT` and `RLinf/RLinf-Pi05-MetaWorld-RL-FlowSDE`. The run records identify their weight files by SHA256 prefixes `6e877ae8e2c9c33a` and `e14684cd8e6938ec`, which match the released `model.safetensors` files. The seed-0 80k student checkpoints have prefixes `2af9fbf4052d70a0` for DAgger and `9625f3ee138c56dc` for PVD. Each training rank sampled its own environment stream over all 50 tasks. Episodes ran until success, environment termination, or 160 actions, with no segment reset. Over 80,000 chunks, the DAgger and PVD students completed 3,941 and 4,530 episodes in seed 0 and 3,956 and 4,572 in seed 1, respectively. Every task was visited at least 60 times per student and seed.

C Additional Tables and Figures

Figures 5, 6, and 8 show the main result, the control design, and both labels in solver time. For ManiSkill, this section gives per-seed differences (Table 8), control contrasts (Table 10), the solver-step matrix (Figure 7), evaluation sets with cost (Figure 9), and the cost and success of the ablation sequence (Figure 4). Table 9 sweeps the deployment step count on ManiSkill and Meta-World. Tables 13 and 14 give the ManiSkill out-of-distribution set and the Meta-World task-shift set. For CALVIN, this section gives per-seed values (Table 11), seed-0 paired chains (Table 12), and success by chain position (Figure 10).

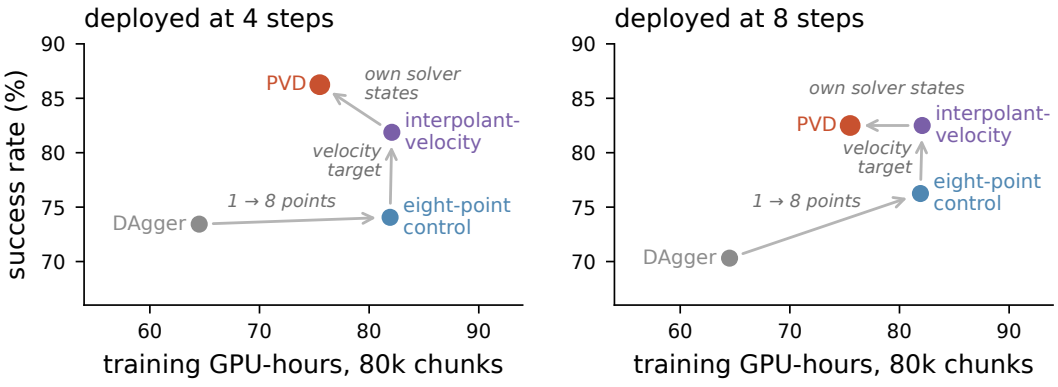


Figure 4: Full-node training GPU-hours against success rate at 80k for the ManiSkill seed-0 sequence in Table 3, with four-step (left) and eight-step (right) deployment. Each arrow names the factor changed from the preceding variant. Moving the velocity queries from interpolant states to the student's solver states lowers cost relative to the interpolant-velocity control because PVD does not run the teacher's solver. PVD nevertheless costs more than DAgger over the complete 80k run (Table 5 in Appendix A).

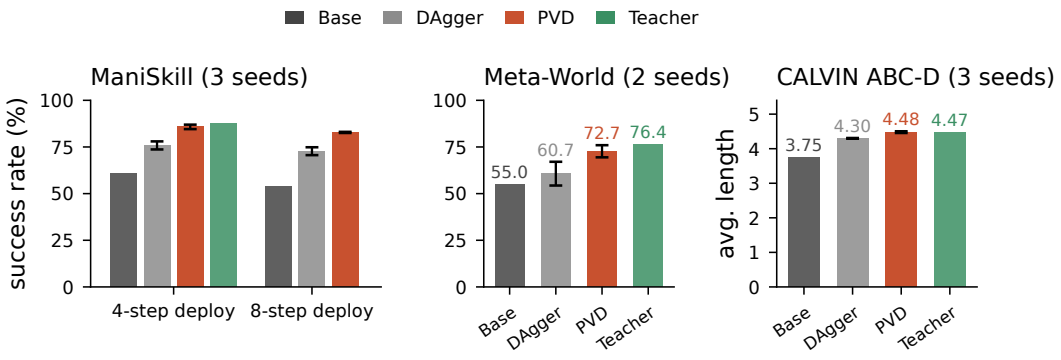


Figure 5: Main result. Left: ManiSkill success rate at 80k on the original 320-reset evaluation set at four-step and eight-step deployment; the teacher bar is at four steps, and the teacher's eight-step value is 84.4% (Table 1). Center: Meta-World MT50 success rate at 80k, five-step training and deployment. Right: CALVIN ABC-D average successful sequence length on the official-D evaluation set. Student bars are means over three seeds (two on Meta-World) with one standard deviation; Base and teacher are single frozen models.

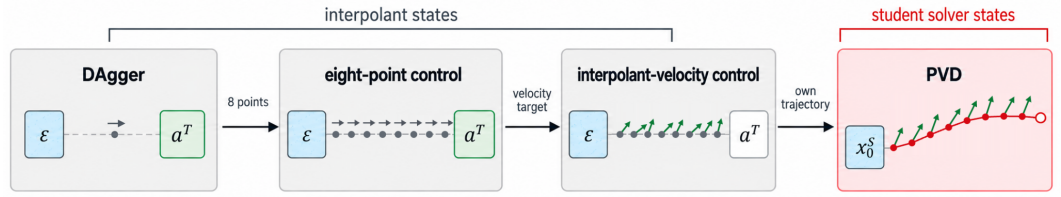


Figure 6: Design of the one-factor ablations. Each arrow names the difference between neighbors, so the series DAgger → eight-point control → interpolant-velocity control → PVD attributes the gain to point count, target type, and point location in turn. The first arrow also moves the interpolation times from the DAgger schedule to the solver grid $\tau_k = k/8$, which the eight-point control and the interpolant-velocity control share with PVD. The order of the series is the order of analysis, not of importance. All four variants are on-policy in the environment; the first three evaluate the loss on the interpolant between a Gaussian sample ε (x_0 in Section 3.2) and the teacher's chunk a^T, and only PVD evaluates it on the student's own solver trajectory. Dark arrows: targets built from a^T; green arrows: teacher-velocity labels. A green a^T box is the action-chunk label; a white one is only the end of the interpolant.

The per-seed Meta-World success rates averaged in Table 1, at five steps, are 65.2 and 56.2 for DAgger and 75.0 and 70.4 for PVD in seeds 0 and 1, differences of +9.8 and +14.2 points. The Meta-World run reports also carry a difficulty-macro average: success rate is computed within four task groups of 28, 11, 6, and 5 tasks and the four group rates are averaged with equal weight, so the two small hard groups count more than in the episode-level success rate. The macro average uses the same per-episode success event and moves with success rate here: for the seed-0 PVD student it is

Table 8: Per-seed PVD minus DAgger differences at 80k on the original ManiSkill evaluation set, in percentage points, with an exploratory seed-level 95% t interval.

		Per-seed difference					
Metric	Deploy	Seed 0	Seed 1	Seed 2	Mean	SD	95% t interval
`success_once`	K=4	+12.81	+7.81	+9.06	+9.90	2.60	[+3.43, +16.36]
`success_once`	K=8	+12.19	+9.06	+9.06	+10.10	1.80	[+5.62, +14.59]
`success_at_end`	K=4	+3.44	−0.94	−6.88	−1.46	5.18	[−14.32, +11.40]
`success_at_end`	K=8	+13.13	+6.88	+10.00	+10.00	3.13	[+2.24, +17.76]

Table 9: Deployment step-count sweep, seed 0, 80k students, in percent. Top: Meta-World MT50, both students trained at five steps and deployed at four, five, eight, and ten steps on the same 500-episode evaluation set; Base 55.0 and teacher 76.4 at five steps; the ten-step point was run for PVD only. Bottom: ManiSkill, the K=8-trained PVD student deployed at four, six, eight, and sixteen steps on the original 320-reset evaluation set.

Meta-World, trained at K=5	Deploy K=4	K=5	K=8	K=10
DAgger	65.6	65.2	65.2	–
PVD	66.4	75.0	49.6	55.8
ManiSkill, trained at K=8	**Deploy K=4**	**K=6**	**K=8**	**K=16**
PVD, `success_once`	86.3	85.6	82.5	81.6
PVD, `success_at_end`	69.4	75.9	75.6	72.5

57.8, 67.1, 37.1, and 41.4 at four, five, eight, and ten steps, against 44.6 for Base and 70.2 for the teacher at five steps.

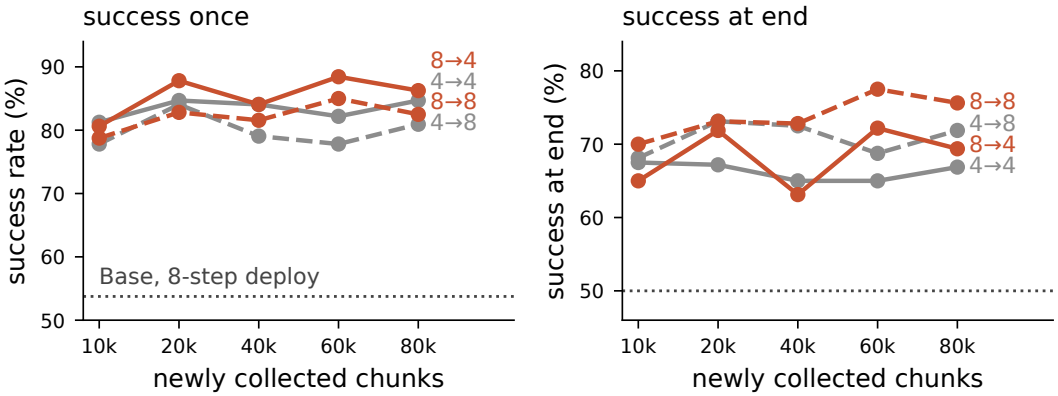


Figure 7: Training and deployment step counts for PVD, seed 0. Students trained at K=4 (gray) and K=8 (orange) are deployed at K=4 (solid) and K=8 (dashed). The left plot shows `success_once` and the right plot `success_at_end`. Deploying at K=8 lowers `success_once` and raises `success_at_end` for both students. At 80k, four- and eight-step training differ by 1.6 points in `success_once` at either deployment step count.

Details of the teacher-rollout BC. The teacher runs its own eight-step solver in the environment for 80,000 chunks and the environment executes the teacher's chunk. A student initialized from Base is then trained offline on those observations with the teacher's final action chunk as the label, for 10,000 updates under the same optimizer and trainable parameters as the two on-policy students. The offline student learns from a teacher that succeeds on 87.5% of the evaluation set at four steps; a weaker teacher would supply a worse training set. Table 3 in the main text reports the teacher-rollout BC's success rate; on the seed-0 evaluation set at 80k it scores 77.50 / 72.81 `success_once` and 71.88 / 66.88 `success_at_end` at K=4 / K=8.

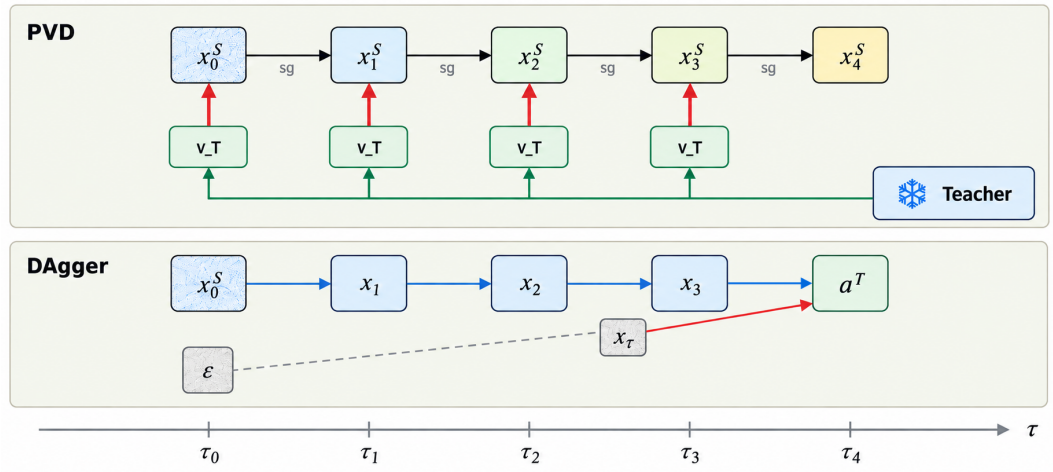


Figure 8: Where the labels sit in solver time for one action chunk ($K = 4$ shown; red arrows are labels). DAgger (bottom) runs the teacher's solver (blue arrows) from the student's initial noise x_0^S and takes one label, the teacher's final chunk a^T. It uses a^T as a conditional flow-matching target at one interpolant state x_τ between an independent Gaussian sample ε and a^T, not at a student solver state. PVD (top) takes K labels, the teacher's velocity queried at each of the student's own states x_k^S, and applies each at that step; the states are detached (sg), so the loss is a sum of K local terms with no gradient through the chain.

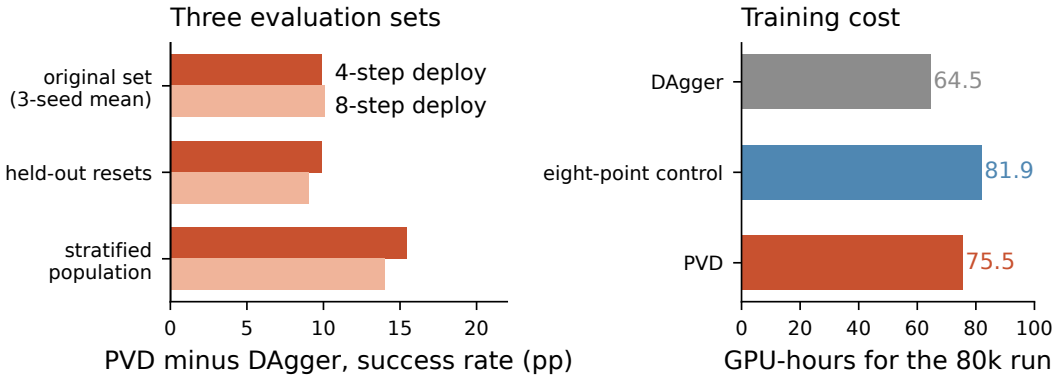


Figure 9: The left plot shows PVD minus DAgger `success_once` at 80k on the three ManiSkill evaluation sets, each as a three-seed mean. The right plot shows full-node GPU-hours of the seed-0 80k ManiSkill run for each variant. PVD costs more than DAgger and less than the eight-point control.

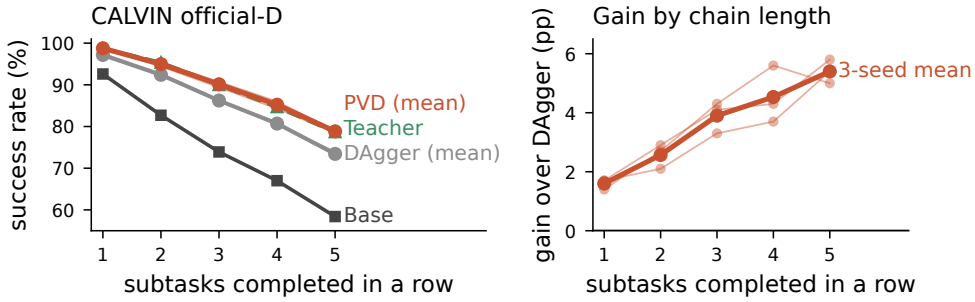


Figure 10: CALVIN ABC-D official-D. The left plot shows the rate of completing the first n subtasks in a row, for $n = 1$ to 5. Curves are Base, teacher, and the three-seed means of DAgger and PVD, with per-seed curves in light color. The right plot shows the PVD minus DAgger gain in that rate, per seed in light color and as the three-seed mean.

Table 10: Ablations and controls: what each constrains and what remains open. Values are `success_once` contrasts in percentage points on the seed-0 evaluation set; the basis (five-checkpoint mean or single checkpoint) is stated per row.

Control	Main contrast	What it constrains	What remains open
Eight-point control	PVD − eight-point: $+9.50$ / $+7.63$ ($K{=}4$/$K{=}8$, five-ckpt mean); $+12.19$ / $+6.25$ (80k)	More supervised states alone	State location, target type, online collection
Interpolant-velocity control	interpolant-velocity − eight-point: $+7.81$ / $+6.25$ (80k); PVD − interpolant-velocity: $+4.38$ / 0.00 (80k)	At the same states, with zero target variance in both, the label switch carries part, not all, of the contrast	Cost not matched, state location
Frozen-rollout control	PVD − DAgger: $+7.81$ / $+15.63$ (20k)	Online collection is not necessary for the gain	Whether online collection still contributes in the main run
teacher-rollout BC	teacher-rollout BC − DAgger: $+4.06$ / $+2.50$ (80k); PVD − teacher-rollout BC: $+8.75$ / $+9.69$ (80k)	Student rollouts with the action-chunk label	Teacher quality on this evaluation set
Two-step-solver pair	PVD − DAgger: -33.8 to -40.0 (five checkpoints)	A gain at every solver step count	Internal cause, traces not retained

Table 11: CALVIN ABC-D official-D results, 1,000 chains per model. Base and teacher are frozen and evaluated once. PVD and DAgger are shown per training seed and as the three-seed mean.

			Subtasks completed in a row, %				
Model	Seed	Avg. length	1	2	3	4	5
Base	–	3.746	92.6	82.7	73.9	67.0	58.4
DAgger	0	4.310	97.5	92.7	86.4	80.4	74.0
	1	4.290	96.9	92.3	86.1	80.6	73.1
	2	4.298	97.1	92.2	86.2	81.1	73.2
	mean	4.299	97.2	92.4	86.2	80.7	73.4
PVD	0	4.500	98.9	95.4	90.7	86.0	79.0
	1	4.452	98.6	94.4	89.4	84.3	78.5
	2	4.486	98.8	95.1	90.3	85.4	79.0
	mean	4.479	98.8	95.0	90.1	85.2	78.8
teacher	–	4.474	98.7	95.3	90.0	84.7	78.7

Table 12: Paired outcomes of the 1,000 official-D chains for CALVIN seed 0 at 80k. A chain counts as a success when all five subtasks are completed in a row. Rows are DAgger and columns PVD; each pair shares its subtask sequence and initial state.

	PVD fails	PVD succeeds	Total
DAgger fails	144	116	260
DAgger succeeds	66	674	740
Total	210	790	1,000

Table 13: Out-of-distribution evaluation on ManiSkill, seed-0 students at 80k deployed at $K=8$, one evaluation seed. The conditions follow the π_{RL} ManiSkill protocol (Chen et al., 2025). They cover an unseen instruction, an image perturbation, held-out object sets with their training-set controls, and two position shifts. Cells are `success_once` counts out of 50 episodes. The test-only row covers six conditions and 300 episodes and excludes the two training-set controls. The last row adds the two training-set controls, for a total of eight conditions and 400 episodes.

Condition	Type	Base	teacher	DAgger	PVD
instruct_test	instruction	25	44	34	46
vision_image_test	image	30	34	43	35
multi_carrot_test	held-out carrot set	4	10	5	16
multi_carrot_train_control	training carrot set	4	7	4	8
multi_plate_test	held-out plate set	8	26	19	19
multi_plate_train_control	training plate set	9	20	7	18
position_change_to_test	position shift	12	20	14	19
position_test	position, test set	9	21	4	20
Test-only, 6 conditions, % of 300		29.3	51.7	39.7	51.7
All 8 conditions, % of 400		25.3	45.5	32.5	45.3

Table 14: Task-shift evaluation, Meta-World, seed-0 students at 80k deployed at $K=5$: five ML45 held-out task names, 20 paired episodes each, `success_once` counts. These tasks were in the MT50 training pool of every model, so the evaluation measures robustness on a task subset, not generalization to unseen tasks.

Task	Base	teacher	DAgger	PVD
bin-picking-v3	4	7	3	8
box-close-v3	7	15	8	13
door-lock-v3	7	17	15	15
door-unlock-v3	7	16	11	15
hand-insert-v3	4	12	3	13
All five, % of 100	29.0	67.0	40.0	64.0

D Proof of Proposition 1

Proposition 1. *Fix o^S, let x_k^T be the teacher's own solver states from the student's noise on the same uniform grid, and let $r_k = v_\phi(o^S, x_k^S, \tau_k) - v_T(o^S, x_k^S, \tau_k)$, so that $\mathcal{L}_{\text{path}} = \frac{1}{K}\sum_k \|r_k\|^2$ for one chunk. If $v_T(o^S, \cdot, \tau_k)$ is L-Lipschitz for every k, then $\|x_K^S - x_K^T\|^2 \le \frac{e^{2L}-1}{2L}\,\mathcal{L}_{\text{path}}$ for any K, where the prefactor extends continuously to 1 at $L = 0$; an L_D-Lipschitz action decoder scales the bound by L_D^2. If every $r_k = 0$, the two solver trajectories coincide.*

Write $h = \Delta\tau_k = 1/K$ and $C_K(L) = \frac{1}{K}\sum_{j=0}^{K-1}(1 + L/K)^{2j}$. Let $E_k^S(x) = x + h\, v_\phi(o^S, x, \tau_k)$ and $E_k^T(x) = x + h\, v_T(o^S, x, \tau_k)$ be one student and one teacher solver step, so that $x_{k+1}^S = E_k^S(x_k^S)$ and $x_{k+1}^T = E_k^T(x_k^T)$ with $x_0^S = x_0^T = x_0$. Let $F_k = E_{K-1}^T \circ \cdots \circ E_k^T$ be the teacher's solver from step k to the end, with F_K the identity, so that $F_k = F_{k+1} \circ E_k^T$. The hybrid final state $y_k = F_k(x_k^S)$ lets the student take the first k steps and the teacher the rest, so $y_0 = x_K^T$ and $y_K = x_K^S$. Consecutive hybrids differ in one step:

$$y_{k+1} - y_k = F_{k+1}\big(E_k^S(x_k^S)\big) - F_{k+1}\big(E_k^T(x_k^S)\big), \qquad E_k^S(x_k^S) - E_k^T(x_k^S) = h\, r_k.$$

Each teacher step is $(1 + hL)$-Lipschitz, so F_{k+1}, a composition of $K - 1 - k$ of them, is $(1 + hL)^{K-1-k}$-Lipschitz and $\|y_{k+1} - y_k\| \le h\,(1 + hL)^{K-1-k}\,\|r_k\|$. The hypothesis enters only through $E_1^T, \ldots, E_{K-1}^T$. It is applied along teacher solves that start from student states and leave both solver trajectories, so it is stated for all x. Nothing is assumed about v_ϕ. Telescoping from y_0 to y_K,

$$\|x_K^S - x_K^T\| \le \sum_{k=0}^{K-1} h\,(1 + hL)^{K-1-k}\,\|r_k\|,$$

and the Cauchy–Schwarz inequality with weights h gives, with $j = K - 1 - k$ in the first factor,

$$\|x_K^S - x_K^T\|^2 \le \Big(\sum_{k=0}^{K-1} h\,(1 + hL)^{2(K-1-k)}\Big)\Big(\sum_{k=0}^{K-1} h\,\|r_k\|^2\Big) = C_K(L)\,\mathcal{L}_{\text{path}}.$$

The uniform grid makes the weights h equal the loss weights $1/K$. The stop-gradients leave the loss value unchanged, so the right side is the training loss as computed. If D maps x_K to the executed chunk and $\|D(x) - D(x')\| \le L_D\|x - x'\|$, the executed chunks obey the same bound times L_D^2. The argument is the standard error propagation for one-step solvers (Hairer et al., 1993), with the teacher's solver in place of the exact flow.

If every $r_k = 0$, each student step is a teacher step, $x_{k+1}^S = E_k^T(x_k^S)$, and induction from $x_0^S = x_0^T$ gives $x_k^S = x_k^T$ for every k; this part needs no Lipschitz hypothesis. For $K \ge 2$, equal final states do not imply zero loss. The query times τ_k are distinct, so a smooth student field can take any value at each (x_k^S, τ_k). Choose the first $K - 1$ values freely and set $v_\phi(o^S, x_{K-1}^S, \tau_{K-1}) = (x_K^T - x_{K-1}^S)/h$: the student lands on x_K^T from any intermediate states, with a positive loss whenever they leave the teacher's solver trajectory.

Average the bound over the noise x_0 at fixed o^S. The shared noise is a coupling of the two chunk distributions at o^S, so their squared 2-Wasserstein distance is at most $L_D^2\,C_K(L)$ times the expected loss. Averaging over o^S as well needs one L for all observations.

For $L > 0$ the geometric sum gives $C_K(L) = \big((1 + L/K)^{2K} - 1\big)/(2L + L^2/K)$. The numerator increases with K and the denominator decreases, so $C_K(L)$ increases with K: $C_1 = 1$, $C_2 = 1 + L/2 + L^2/8$, and $C_K(L) \to (e^{2L} - 1)/(2L)$, the constant the same argument gives for the exact flow. This limit is the constant stated in Proposition 1, which therefore holds for every K. Of all $K \ge 2$, the two-step solver has the smallest constant per unit of loss. The bound does not predict the two-step failure of Section 4.4.

E Paired Rollouts

Each strip pairs the seed-0 DAgger and PVD students at 80k on one evaluation episode. The two rollouts start from the same reset and, on CALVIN, follow the same five instructions. Frames are

taken from the recorded rollouts through the evaluation camera. On ManiSkill and Meta-World they are the start, quarter, half, three-quarter, and end of each rollout, and the ManiSkill frames are cropped to the tabletop. On CALVIN each frame is the last of its subtask, and subtasks after a failure are not run. Students are deployed at eight steps on ManiSkill, five on Meta-World, and eight on CALVIN. The episodes shown are ones on which PVD succeeds and DAgger fails. The counts in both directions are: on CALVIN seed 0, 116 chains succeed under PVD alone and 66 under DAgger alone (Table 12); on Meta-World over both seeds, 179 and 59 of the 1,000 paired episodes (Section 4.2). Figures 12 to 14 hold thirteen strips; the fourteenth is Figure 11.

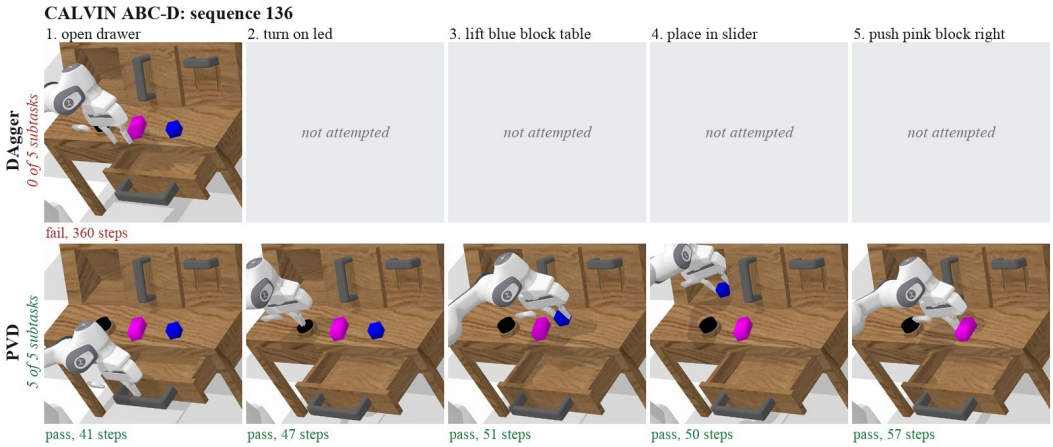


Figure 11: One CALVIN ABC-D chain, official-D sequence 136, seed-0 students at 80k. Both start from the same initial state with the same five instructions. Each frame is the last of its subtask. DAgger fails the first subtask at the 360-action limit and the chain stops. PVD completes all five.

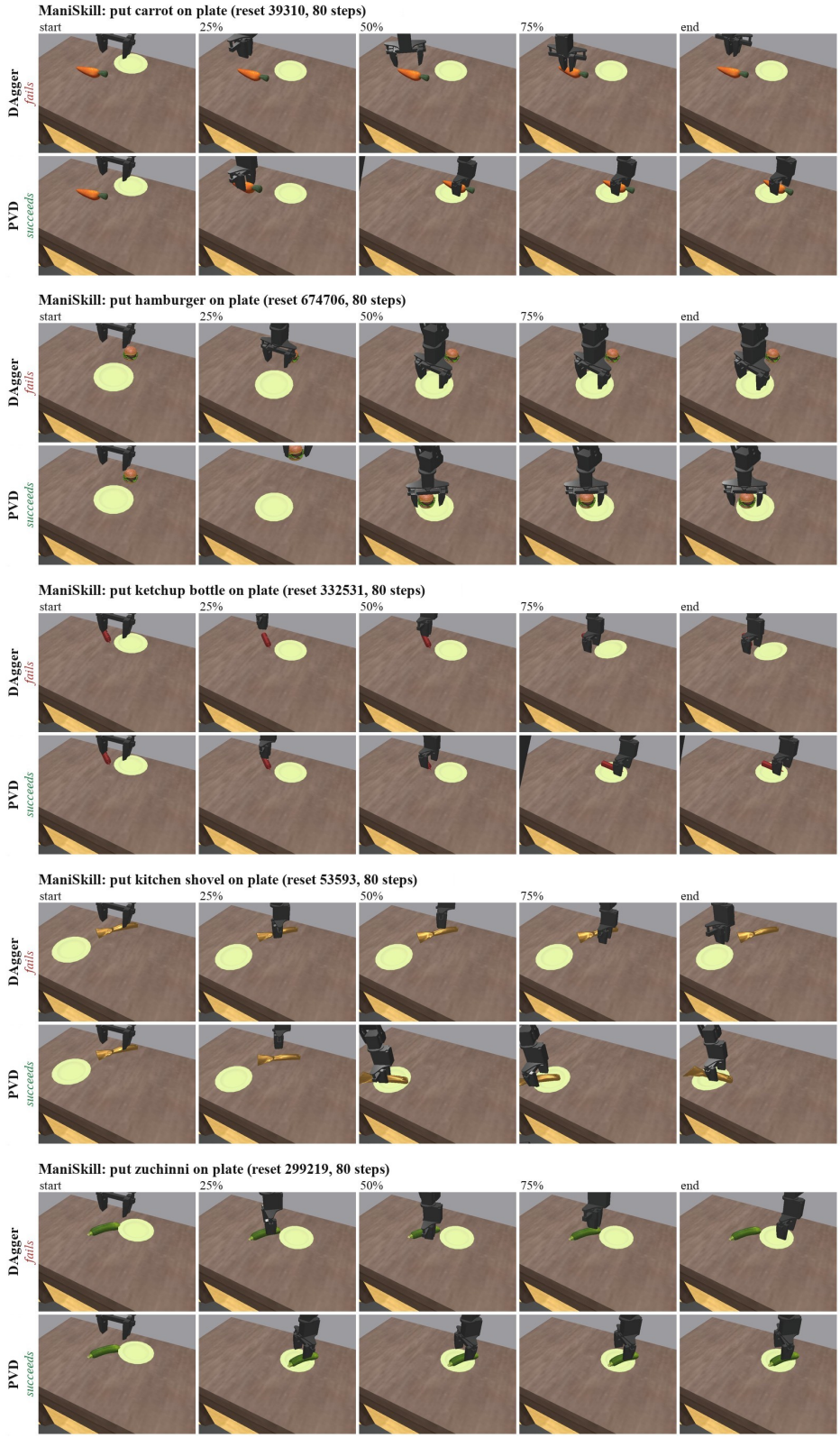


Figure 12: ManiSkill-25 Main, five paired rollouts at eight-step deployment: carrot, hamburger, ketchup bottle, kitchen shovel, and zucchini onto the plate. Each pair shares its reset id.

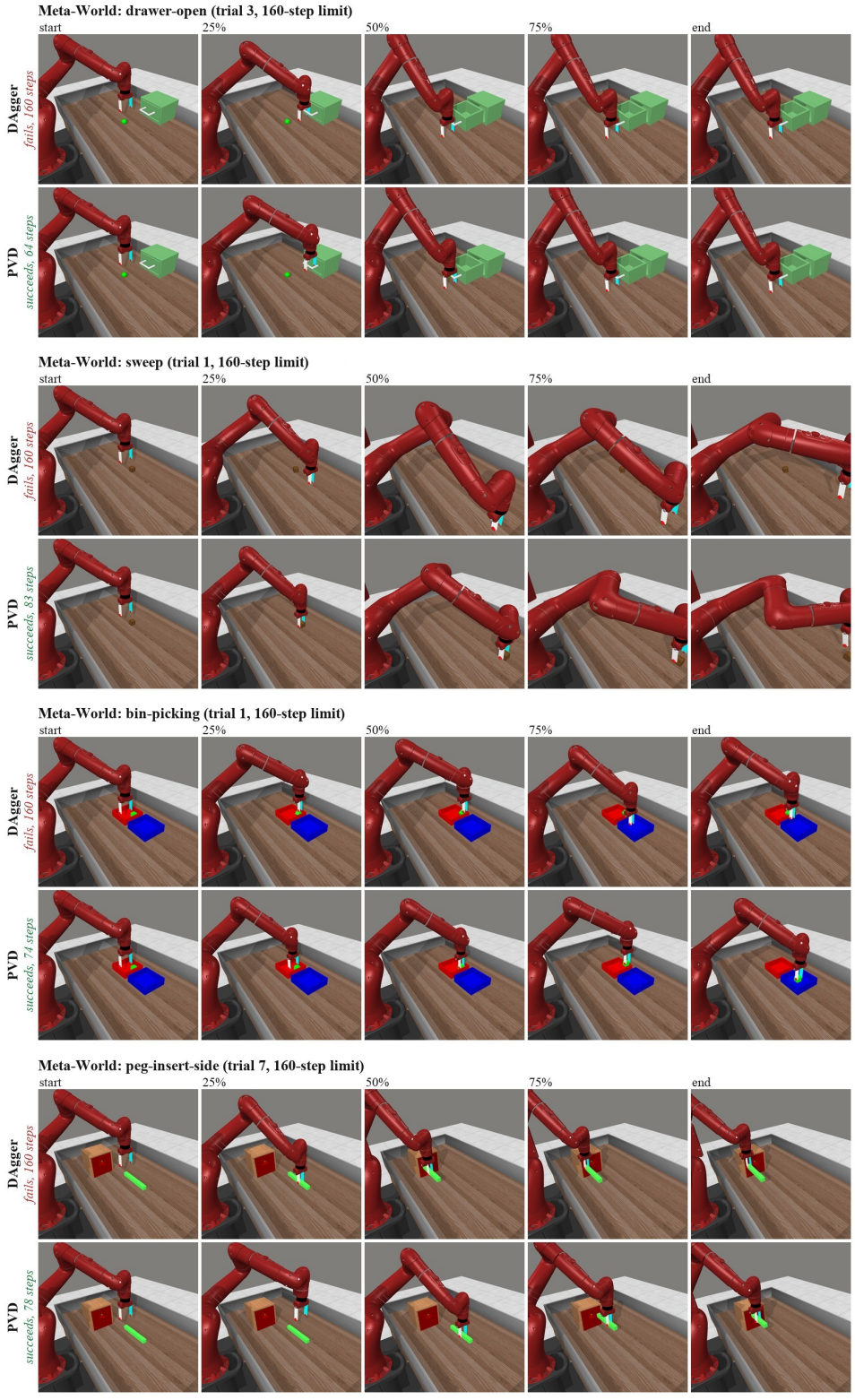


Figure 13: Meta-World MT50, four paired rollouts at five-step deployment: drawer-open, sweep, bin-picking, and peg-insert-side. An episode ends at the first success or at 160 actions, so the PVD rows span fewer actions.



Figure 14: CALVIN ABC-D, four paired five-subtask chains at eight-step deployment, official-D sequences 48, 793, 760, and 584. Gray cells are subtasks not run because the chain had already failed.

F ARTIFACT NOTES

The exact ManiSkill and CALVIN run-record and report hashes, code and checkpoint availability, and author and affiliation information are artifact-stage additions.