ONE STEP IS OFTEN ENOUGH: ACCELERATING VLA ACTION HEADS WITH TRAINING-FREE MARGINAL-ALIGNED WARM STARTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action models with a flow-matching action head spend several denoising steps on every policy call. We ask how few the head needs at inference, without training, and find that one step is often enough but not always. The same single step keeps or improves success on some tasks and collapses on others. The split is not visible in open loop, where one step returns an estimate of the posterior mean with a small residual of similar size on every task. It is decided in closed loop, where that residual is persistent and integrates in the state until it exceeds the loop's tolerance. A one-step head therefore needs a starting point that carries the executed motion and lies on the marginal it was trained on. Marginal-Aligned Warm Start (MAWS) provides both without training, by shifting the previously predicted action chunk, re-noising it to the flow's training marginal, and taking one step. Across models, simulated benchmarks and a real robot, MAWS lowers the residual as predicted, recovers most or all of the success that one step from noise loses, and outperforms training-free baselines that spend several evaluations per call. Because the warm start adds little to a one-step call, cutting the steps cuts the time. On π_0 the action head runs $9.1\times$ faster at an unchanged suite average.

1 INTRODUCTION

Vision-language-action (VLA) policies pair a vision-language backbone with a generative action head (Black et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025). Recently, it has become common to employ a diffusion transformer trained by flow matching as the action head (Lipman et al., 2023), which produces each action chunk by integrating a learned velocity field from noise over K denoising steps. The backbone runs once per policy call, but the action head runs K times, making the iterative denoising loop a substantial share of the overall inference latency (Oi et al., 2026). Given this structural multiplier, reducing the number of denoising steps emerges as a critical lever for accelerating the policy to meet the high-frequency demands of real-time control.

Existing answers to how few steps suffice fall into two categories. Methods that retrain the model, via distillation, consistency models, or learned priors (Prasad et al., 2024; Gao et al., 2024; Li et al., 2026; Kang et al., 2026), can successfully achieve single-step inference, but incur significant training costs. Conversely, training-free methods that reuse structures like trajectory buffers (Chen et al., 2025) or action caches (Oi et al., 2026) avoid retraining, but still require several denoising steps per call. Both lines treat the step count as a budget to be spent per call; neither asks why the same single step keeps success on one task and collapses another, which is the question that decides whether one step can be the default. Our goal is single-step execution without training. To that end we present, to our knowledge, the first systematic analysis of an unmodified policy constrained to a single step.

Empirically, a naive single-step execution yields highly inconsistent outcomes, succeeding effortlessly on some tasks while completely collapsing on others. Open-loop properties of the head, with the action head evaluated once at a fixed observation, cannot explain this split: the deterministic error of a single step is small on every task, as Fig. 1a illustrates. The determining factor lies in the closed loop, where the executed actions change the state and the next observation. A single-step policy functions as a deterministic map, causing its persistent residual to accumulate over time in the environment's state. A task begins to collapse once this integrated error exceeds the environment's

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.

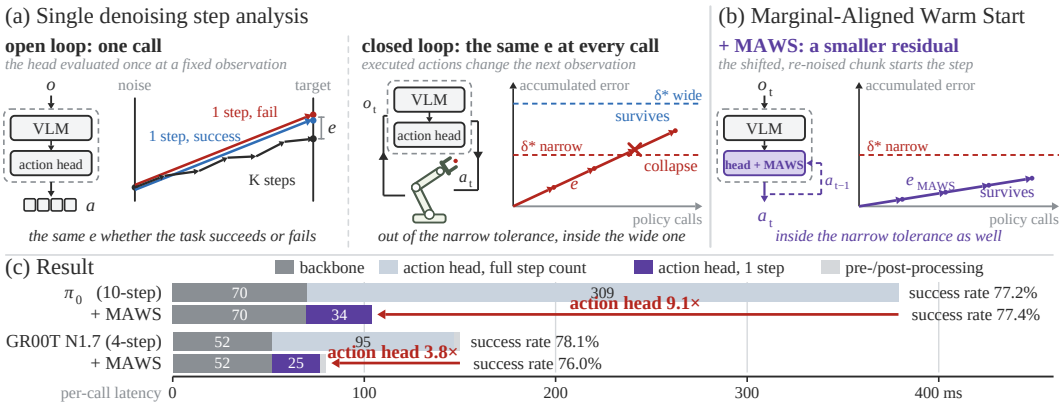


Figure 1: (a) A single denoising step in open loop and in closed loop. In open loop the one-step chunk lands the same distance e from the K-step chunk whether the task succeeds or fails; in closed loop it adds up call after call and leaves a narrow tolerance $\delta^\star$ but not a wide one. (b) MAWS starts the single step from the shifted and re-noised previous chunk, so the residual at each call is smaller and the deviation stays inside the narrow tolerance as well. (c) Per-call latency, split into backbone and action head, and success rate, measured on SimplerEnv.

tolerance threshold. We measure this tolerance directly by perturbing the executed actions, and it varies by an order of magnitude across tasks, which is why a residual of the same size is harmless on one task and fatal on another.

This diagnosis directly dictates our design. To prevent the residual from integrating, the 1-step head must be aware of its previously executed motion, while starting from a valid data distribution it was trained on. We propose Marginal-Aligned Warm Start (MAWS), which meets both requirements by shifting the previously predicted action chunk and re-noising it to the flow's training marginal before taking a single step (Fig. 1b). It keeps one chunk of state, has a single hyperparameter, the retention, and runs at close to the cost of a single step. By doing so, MAWS effectively equips a stateless VLA with a stabilizing one-step memory. The persistent error is absorbed rather than accumulated, achieving extreme acceleration without altering a single model weight or requiring any retraining.

Extensive evaluations across leading VLA models (π_0, $\pi_{0.5}$, GR00T N1.7) and diverse benchmarks (SimplerEnv, LIBERO, CALVIN, ALOHA) demonstrate the robustness of MAWS. While constraining the action head to a single denoising step, it consistently recovers performance on vulnerable tasks while maintaining baseline proficiency on others, outperforming existing step-reduction and warm-start baselines at equal computational cost. Most notably, MAWS accelerates the π_0 action head by $9.1\times$ (the action-head segment of the per-call latency in Fig. 1c) with zero degradation in average success rate (SR). Real-world deployments confirm that this extreme, training-free acceleration seamlessly transfers to physical hardware.

Our contributions are:

- **A systematic open- and closed-loop diagnosis** of single-step DiT action heads: collapse is not predictable from the flow alone and begins once the loop integrates the persistent residual beyond the environment's tolerance.
- **MAWS**, a fully training-free warm start strategy. By shifting and validly re-noising the previous chunk, it equips the sampler with a stabilizing one-step memory that prevents catastrophic error integration.
- **Extensive empirical validation** across state-of-the-art VLA models and diverse benchmarks. We demonstrate that MAWS matches or outperforms budget-matched step-reduction baselines, culminating in a $9.1\times$ faster π_0 action head deployed successfully on physical robotic hardware.

2 Related Work

2.1 Vision-Language-Action Models

VLA models attach action heads of several kinds to a vision-language backbone, among them autoregressive heads that discretize actions into tokens and decode them one at a time (Kim et al., 2024), and diffusion and flow-matching heads that generate a continuous chunk of actions by iterative denoising (Black et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025). We study the latter, since they are the choice of the strongest open checkpoints, where a chunk produced in one pass supports smooth, multimodal control at high rate. Its price is the denoising schedule, whose length multiplies the cost of every action.

2.2 Efficient Inference for Diffusion-based VLAs

One line of work accelerates the backbone or the cost of a single denoising evaluation, by reusing static visual tokens across frames (Xu et al., 2025), skipping layers (Yue et al., 2024), pruning tokens and layers together with caching head features (Yang et al., 2025), or caching block features inside the head (Ji et al., 2025). Real-time chunking (Black et al., 2025) instead hides latency by solving the next chunk at full cost while the current one executes. None of these reduces the number of denoising steps, which remains the largest share of the head's cost. In generative modeling that number is reduced by better solvers (Song et al., 2021; Lu et al., 2022), by distilling few-step students (Salimans & Ho, 2022; Song et al., 2023) or by straightening the flow (Liu et al., 2023). In robotics, approaches that distil or retrain the head (Prasad et al., 2024; Gao et al., 2024; Luan et al., 2026; Hosseinkhani et al., 2026; Gao et al., 2026) reach one step, but retrain every policy and fix its operating point. Training-free alternatives keep the weights. Falcon (Chen et al., 2025) re-enters a partially denoised buffer at a depth chosen per call, ActionCache (Oi et al., 2026) reuses a cached chunk when the observation matches one it has seen, and ProbeFlow (Fang et al., 2026) and AdaVLA (Han et al., 2026) read the geometry of the flow at each call to choose the step count. All keep several steps in reserve rather than committing to one, and the last two decide from open-loop signals, which Sec. 3 shows do not determine whether a single step succeeds. We compare at a matched number of function evaluations (NFE) throughout.

2.3 Temporal Priors for Action Sampling

Initializing a solver from a temporally shifted previous solution is standard practice in model predictive control (Rawlings et al., 2017; Diehl et al., 2005; Williams et al., 2017), and the idea has been carried to learned action samplers. RTI-DP (Duan et al., 2025) starts a truncated diffusion schedule from the shifted previous chunk, Falcon keeps a buffer of recent trajectories, ActionCache retrieves cached chunks, and temporal ensembling (Zhao et al., 2023) averages overlapping predictions. Two recent works add the prior through training, as spatiotemporal (Li et al., 2026) or temporal (Kang et al., 2026) conditioning. For a learned denoiser, a prior is admissible only on the marginal it was trained on, a constraint that classical solvers do not have. RTI-DP injects the clean chunk directly and the others do not address the constraint, whereas our warm start re-noises the prior onto the training marginal at inference and follows it with a single step, on unmodified policies.

3 Diagnosing Single-Step Execution

To reduce denoising steps without severely harming task performance, we first look at what happens when each model is constrained to a single step instead of its full step count. The outcome is far from uniform. Across the sixteen cells of Fig. 2, two models on eight SimplerEnv tasks under the protocol of Sec. 5.1, the change in SR ranges from a 26-point drop to a 23-point gain. The same task can collapse under one architecture and improve under another, and a single model can yield opposite outcomes on similar tasks. Single-step viability is therefore neither a property of the task alone nor of the model alone. To isolate the cause, we evaluate the action head in open loop at fixed observations (Sec. 3.1) and in closed loop acting in the environment (Sec. 3.2).

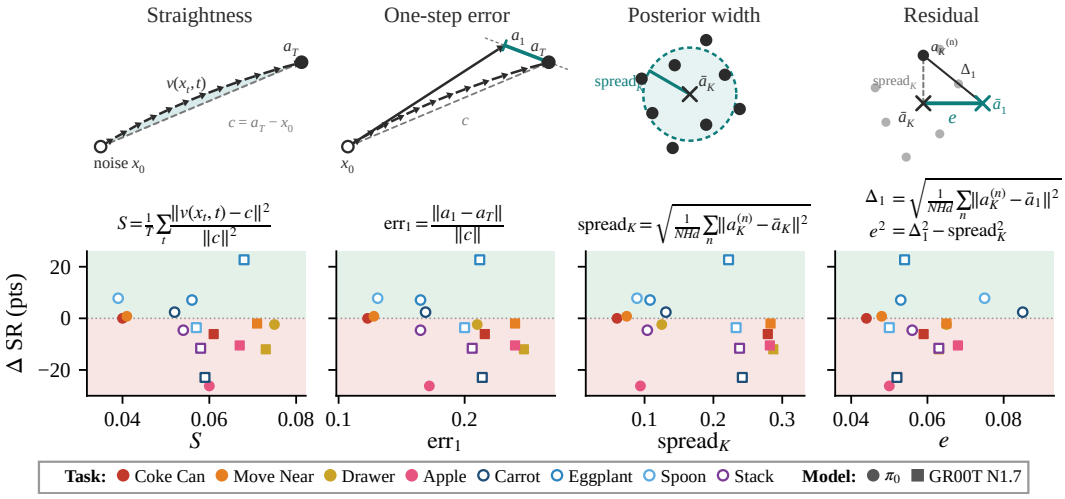


Figure 2: Top: the four open-loop probes of the flow, each measured at real observations of every task. Bottom: one-step outcome of each SimplerEnv task, plotted against each probe, under two VLA models. ΔSR is SR at a single step minus SR at the full step count, in points, so positive values (green band) mean that one step does better and negative values (red band) that it does worse.

3.1 Open-Loop Characterization: One Step is a Conditional Mean

The single step as an estimator. Let o denote the observation, a the action chunk, $\varepsilon \sim \mathcal{N}(0, I)$ the noise, and $x_s = (1-s)\,\varepsilon + s\,a$ the point on the flow path at time $s \in [0,1]$. The action head is trained to predict the velocity $a - \varepsilon$ from x_s, s and o. Because the interpolant is linear in a and ε, the ideal field $v^\star(x,s) = \mathbb{E}[a - \varepsilon \mid x_s{=}x, o]$ makes a single Euler step land on the posterior mean, $x + (1-s)\,v^\star(x,s) = \mathbb{E}[a \mid x_s{=}x, o]$. From pure noise ($s{=}0$), this resolves to $\mathbb{E}[a \mid o]$, regardless of the specific noise drawn. We call this conditional mean the *center*, and describe a learned head's departure from it by the *spread* (the standard deviation of the K-step chunk across noise seeds) and the *residual* e (the distance from the 1-step output to the center). A learned head only approximates $v^\star$, so its one-step output depends weakly, but not negligibly, on the noise drawn. Thus, what one step returns is not a cruder draw from the posterior but an estimate of its mean. It is nearly deterministic, with a little spread, and sits a residual e from the center, the largest of any step count. As K grows the spread returns, since an exact flow carries the noise into the posterior's width, while the residual shrinks, since later steps execute where the velocity field is more accurate (Chen et al., 2023). K effectively acts as a **mean dial** between the center and a full posterior sample.

Open-loop probes. The top row of Fig. 2 draws the four probes we read the flow with, the straightness S, the one-step error err_1, the posterior width spread_K and the residual e, and the bottom row plots the one-step outcome of every task against each. If the cause of collapse were visible in open loop, the failing tasks should score worse on one of them. The center itself is not observed, so we estimate it by the mean $\bar{a}_K$ of $N{=}8$ full-step chunks drawn with different noise seeds, with norms over the $H \times d$ entries of a chunk. The distance Δ_1 from the mean one-step chunk to these samples then decomposes as $\Delta_1^2 = e^2 + \mathrm{spread}_K^2$, because the chunks scatter around their mean, so the residual e is the part of Δ_1 that is a fixed offset from the center rather than the sampler's own scatter. Notably, none of these static probes successfully order the tasks by outcome. Each probe varies by at most a factor of two, and the one π_0 collapse (Apple) sits inside the range of the surviving tasks on every axis. The two models do separate on err_1 and spread_K, implying that GR00T N1.7's step retains more of the noise, but it still orders no task. The residual does not track the outcome either, as under GR00T N1.7 Carrot and Eggplant yield nearly identical residuals (0.052 and 0.054) while diverging by 23 points in opposite directions. One step therefore returns an estimate of the center off by a small residual of the same order on every task, and an error of that size everywhere cannot explain outcomes from collapse to gain. The difference must lie in how the loop treats it.

3.2 Closed-Loop Diagnosis: The Integration of Errors

To determine how the loop treats an error, we add a random perturbation to every chunk the full-step policy executed, of root mean square (RMS) size δ in the probes' normalized action units, either drawn independently at every call (iid) or fixed for the episode (bias). Fig. 3 shows the resulting tolerance curves, and both kinds degrade success at large enough δ but not at the same size. A fresh perturbation changes direction at every call and largely cancels, whereas a persistent one adds up in the state, because actions are displacements, so the loop tolerates fresh perturbations several times larger than persistent ones. We define the *tolerance* $\delta^\star$ as the persistent perturbation that halves the SR. It varies by an order of magnitude across tasks and tracks the slack the goal leaves, narrowest where a part has to be placed inside another and widest where an object only has to reach a region.

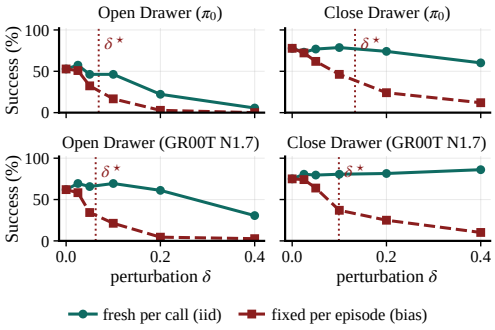


Figure 3: Success rate of the full-step policy when a perturbation of RMS δ is added to every executed chunk, fresh at every call (iid) or fixed for the episode (bias), on Open and Close Drawer under π_0 and GR00T N1.7. The dotted line marks the tolerance $\delta^\star$.

The two departures of Sec. 3.1 fall on the two sides of this divide. The spread is fresh by construction, since the noise is drawn anew at every call. Conversely, the residual is persistent. It is a deterministic function of the observation, consecutive observations are highly correlated, and its size changes slowly along an episode, as Table S.2 records, so it enters the loop as a bias. This suggests a simple rule: **a single-step policy begins to collapse once its residual exceeds the loop's tolerance, $e > \delta^\star$, and survives while it does not.** On all eight π_0 tasks the rule matches the outcome, with e and with no other probe, against the tolerances Fig. 3 measures on the drawers and Sec. 5.4.1 on the rest. On Open Drawer of tolerance 0.07, Δ_1=0.13 would predict a collapse that does not happen, while e≈0.06 predicts the small loss we observe, recorded before the perturbation was run. Under GR00T N1.7 the residual varies along the episode and, read where the loop acts, separates the two drawer tasks. It reaches 0.08 to 0.10 during handling on Open Drawer, above a tolerance of about 0.06, and stays near 0.07 on Close Drawer, below 0.10. This is the compounding of smooth deterministic policies (Simchowitz et al., 2025), which the K-step spread, a state-dependent stochasticity, escapes.

The same divide between fresh and persistent error explains the inversions. A K-step sampler redraws from the posterior at every call, a fresh perturbation the size of its spread, whereas the center never redraws and pays nothing. Three of four π_0 Bridge tasks therefore gain 2 to 8 points at one step, and the fourth is Stack Cube, the one whose spread exceeds its tolerance.

4 Marginal-Aligned Warm Start (MAWS)

4.1 Starting the Step on the Training Marginal

A policy call at control step t receives o_t and returns a chunk $a_t \in \mathbb{R}^{H\times d}$, integrated with the learned velocity $v_\theta(x_s, s, o_t)$, and the loop executes its first $r \le H$ actions. Sec. 3 leaves the single step with two requirements. Its persistent residual must be brought below the tolerance $\delta^\star$, and it must remain an estimate of the center rather than a sample. Both point at the starting point x_s, the only input through which the head can be told what motion is underway. From s=0 the step must estimate the whole motion from o_t alone, whereas from a later flow time it lands on $\mathbb{E}[a \mid x_s, o_t]$, conditioned on what x_s carries and taken where the field is more accurate (Sec. 3.1). The previously predicted chunk is the natural carrier.

The action head, however, is defined only on its training marginal, where at time s the input is the answer plus noise of width exactly $1 - s$. A clean chunk placed at $s < 1$ breaks this contract, and in practice the head returns almost no correction and replays the stale chunk (the raw-initialization rows of Table 2). The prior must therefore enter the flow as a draw of the marginal, *re-noised* with fresh ε. The noise supplies the width the head expects to remove, so the step remains a center estimate, and the flow time s at which it enters fixes how much of the prior it keeps. Unlike SDEdit (Meng et al., 2022), which noises an external guide once and runs the rest of the schedule to trade

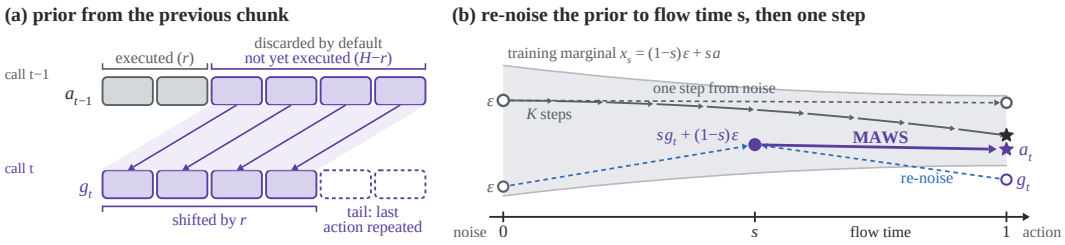


Figure 4: One policy call with MAWS. (a) The prior g_t: the r executed actions of the previous chunk are dropped, the $H-r$ remaining ones slide forward, and the vacated tail repeats the last action. (b) Flow time from noise to action, with the training marginal drawn as a funnel: the full-step sampler takes K steps from noise to its output, and one step from the same noise lands a residual away from it, while MAWS re-noises g_t to time s inside the marginal and takes one step to a_t, landing closer.

fidelity against realism, MAWS re-noises the policy's own shifted chunk at every control step and takes a single step, since the aim is a smaller persistent residual, not a new sample.

4.2 Constructing the Prior and Taking a Step

At control step t we shift the previous chunk a_{t-1} by the replan interval r, so that its remaining $H - r$ actions come first, and fill the r vacated tail positions with the last action (the tail fill). The result is the prior g_t (Fig. 4a). If $r{=}H$, g_t is the last action repeated. The prior enters the flow at retention s by explicit re-noising,

$$x_s = (1 - s)\varepsilon + s \cdot g_t, \quad \varepsilon \sim \mathcal{N}(0, I), \tag{1}$$

which is a sample of the training marginal at time s with g_t in place of the data. The head then takes a single step to the end of the flow,

$$a_t = x_s + (1 - s)v_\theta(x_s, s, o_t), \tag{2}$$

as in Fig. 4b and Algorithm 1 in Appendix A.2. Each call costs one function evaluation plus a shift and a noise mix below the timer's resolution. At $s{=}0$ the start is pure noise and the call is one step from noise, at $s{=}1$ the head is not consulted and the previous chunk is replayed, and in between the step corrects the prior only in part, so s sets how much of it survives. The loop is then an *exponential moving average* over chunks, longer in memory as s grows, the temporal ensemble of ACT (Zhao et al., 2023) inside the sampler. The retention is the sole hyperparameter (Sec. 5.1). On the first call of an episode, and whenever the instruction changes, $g_t{=}0$ and $s{=}0$ reduce the same equations to one step from noise, so every call costs one evaluation. MAWS stores one chunk per environment and changes neither the weights, the backbone pass and its cache, nor the replan interval.

5 Experiments

5.1 Setup

Models and protocol. We evaluate π_0 and GR00T N1.7 on SimplerEnv (Li et al., 2024) Visual Matching, the Fractal suite (Google robot) and the Bridge suite (WidowX), under each harness's official protocol (Appendix A.3). π_0 runs at its full step count of ten and predicts four actions per call, of which the harness executes two on Fractal and all four on Bridge; GR00T N1.7 runs at four steps, predicts forty and executes one on Fractal and four on Bridge. Every number is the mean over three seeds, and per-call latency is measured on one NVIDIA A40. The retention is $s{=}0.25$ where the whole chunk is executed ($r{=}H$) and $s{=}0.5$ otherwise (Sec. 5.4.2 examines this choice).

Baselines. We compare with three training-free alternatives on the π_0 Fractal suite, at the number of function evaluations each actually spends. Falcon (Chen et al., 2025) runs at its official setting. ActionCache (Oi et al., 2026) is our port of the published method ($N_{\text{hit}}{=}1$), with a cache of 10k chunks filled by a full-step pass over the same scene enumeration before evaluation, a convention that favours it, run at the published hit threshold $T_{\text{hit}}{=}0.85$ and at a lower threshold of 0.75 that accepts more cached chunks. The RTI-DP-style rows (Duan et al., 2025) feed the shifted previous chunk without re-noising and enter the schedule K' steps from its clean end.

Table 1: Success rate (%) on SimplerEnv benchmark, mean±sd over seeds. For each model, the first row samples from noise at the number of denoising steps of the checkpoint's official configuration, which we call the full step count, and the second at a single step; +MAWS is our training-free method (Sec. 4). The NFE column gives function evaluations per policy call. Bold marks the best entry of a block, red the lowest average, and parentheses the change of the average against the full step count.

Fractal (Google robot)	NFE	Coke Can	Move Near	Drawer	Apple	Avg
π_0	10	97.3±0.3	77.4±1.0	66.4±2.7	**42.6±1.9**	77.2
π_0	1	97.3±0.3	78.2±1.2	64.0±2.1	16.4±4.4	73.6 (-3.6)
+MAWS (Ours)	1	**99.0±0.3**	**82.4±3.7**	**67.7±2.3**	25.6±3.9	**77.4** (+0.2)
GR00T N1.7	4	97.2±1.2	**96.9±1.3**	**62.5±5.3**	**14.5±7.0**	**78.1**
GR00T N1.7	1	91.1±0.8	94.9±0.6	50.5±5.8	4.0±0.5	71.1 (-7.0)
+MAWS (Ours)	1	**97.3±1.0**	94.6±1.5	57.6±7.6	12.2±4.7	76.0 (-2.1)
Bridge (WidowX)	NFE	Carrot	Eggplant	Spoon	Stack	Avg
π_0	10	55.0±1.9	84.4±3.0	83.3±2.5	49.9±3.5	68.2
π_0	1	57.4±4.2	91.5±0.5	**91.1±1.3**	45.3±0.2	71.3 (+3.1)
+MAWS (Ours)	1	**62.4±0.9**	**92.1±2.6**	90.1±0.6	**54.2±1.9**	**74.7** (+6.5)
GR00T N1.7	4	**65.4±2.7**	49.7±3.6	**84.2±0.8**	**37.4±0.6**	**59.2**
GR00T N1.7	1	42.5±4.7	**72.4±4.0**	80.6±2.0	25.8±1.5	55.3 (-3.9)
+MAWS (Ours)	1	46.7±5.2	62.1±3.6	82.1±5.4	33.8±4.5	56.2 (-3.0)

5.2 Main results

Table 1 compares sampling from noise at the full step count, sampling from noise in a single step, and MAWS. At one evaluation per call MAWS improves on the one-step baseline in all four model–benchmark pairs. On π_0, it matches or passes the full step count (77.4 against 77.2 on Fractal, 74.7 against 68.2 on Bridge) with a tenth of the evaluations. On GR00T N1.7, it recovers most of the drop (76.0 against 78.1, 56.2 against 59.2) at a quarter. The two shortfalls are the ones Sec. 3 predicts. On π_0 Apple, the collapse regime where the residual exceeds the tolerance, no starting point rescues a single step, but a second step closes the gap, as the step sweep in Appendix A.6 shows. On GR00T N1.7 the single step itself removes less of the noise (Table S.1), so the warm start recovers most of the drop but not all of it. Results on further benchmarks and models (LIBERO, CALVIN and ALOHA under π_0 and $\pi_{0.5}$) are in Appendix A.4, where Tables S.3 and S.4 show the same pattern. The procedure also carries over to CogACT, a VLA model with a diffusion action head sampled with DDIM rather than a flow, where it recovers most of the drop of one step from noise (Table S.5). Since MAWS leaves a call at nearly the price of one step, the action head runs **9.1**× faster on π_0 and **3.8**× on GR00T N1.7, and the whole call 3.6× and 1.9× faster, the rest being the backbone's share, as Fig. 1c breaks down.

5.3 Comparison with other methods

Table 2 places MAWS beside the baselines. Falcon, which re-enters a partially denoised buffer at a depth chosen per call, and ActionCache at its published threshold, which reuses a cached chunk when the observation matches one it has seen, average within two points of MAWS at 3 to 8 evaluations per call, and neither reaches one. Falcon re-enters partway down the schedule and a cache miss runs the full count, so their cost grows on the harder tasks, where scenes repeat less (Falcon spends nine evaluations per call on Apple). ActionCache also needs its cache filled before evaluation, whereas MAWS keeps one chunk of state and needs no cache. Lowering the ActionCache threshold brings its cost to 1.5 NFEs, but the reused chunks then fit the scene less often, and its average falls by four points, with the largest losses on Drawer and Apple. On Apple, where one step from noise collapses, Falcon and ActionCache at the published threshold score above MAWS at five to nine evaluations per call, and a second warm step closes most of that gap (Appendix A.6). The RTI-DP-style rows collapse to 9.9 and 18.5, the failure Sec. 4.1 attributes to leaving the training marginal. In addition, methods that retrain the action head for one step report on LIBERO under $\pi_{0.5}$, and Appendix A.5 compares MAWS with them there.

Table 2: Success rate (%) against training-free baselines on the π_0 Fractal suite, mean±sd over seeds. NFE is the measured, episode-weighted number of function evaluations per policy call; MAWS runs on the same checkpoint at one step, and the settings of each baseline are given in Sec. 5.1. Parentheses give the change of the average against the full step count.

Fractal (Google robot)	NFE	Coke Can	Move Near	Drawer	Apple	Avg
π_0	10	97.3±0.3	77.4±1.0	66.4±2.7	**42.6±1.9**	77.2
π_0	1	97.3±0.3	78.2±1.2	64.0±2.1	16.4±4.4	73.6 (-3.6)
Falcon (Chen et al., 2025)	7.6	96.8±0.8	78.1±2.8	64.5±2.2	33.6±3.9	75.6 (-1.6)
ActionCache (Oi et al., 2026)	3.2	95.7±0.9	**84.4±1.5**	66.2±0.9	33.6±2.1	**77.4** (+0.2)
ActionCache (Oi et al., 2026)	1.5	94.3±0.3	83.2±1.5	56.0±1.7	25.3±1.9	73.0 (-4.2)
RTI-DP-style (Duan et al., 2025)	3	0.0±0.0	25.6±0.9	45.7±1.5	0.0±0.0	18.5 (-58.7)
RTI-DP-style (Duan et al., 2025)	1	0.0±0.0	10.6±1.0	27.9±0.7	0.0±0.0	9.9 (-67.3)
MAWS (Ours)	1	**99.0±0.3**	82.4±3.7	**67.7±2.3**	25.6±3.9	**77.4** (+0.2)

5.4 Analysis

5.4.1 Residual under MAWS

To test the claim of Sec. 3 that the warm start lowers the persistent residual, we measure the residual of the MAWS step, e_{MAWS}, at real observations of a MAWS rollout, next to the residual e of the single step from noise at the same frames, with the probe of Sec. 3.1. Table 3 reports both on every π_0 task at the operating points of Table 1. The MAWS step lowers the residual on every task, by 17 to 35 percent. The second requirement of Sec. 4.1, that the step remain an estimate of the center, is checked by $\text{spread}_{\text{MAWS}}$ (the spread of the MAWS output over noise draws). At 0.02 to 0.05 against 0.06 to 0.12 for the full-step posterior, re-noising does not turn the step back into a sample, so what MAWS changes is the persistent error, not the fresh one. The prior itself sits 0.08 to 0.25 from the posterior mean, farther than the center of the step from noise, yet the step taken from it lands closer, so the gain comes from starting at a flow time where the head is more accurate, the mean dial of Sec. 3.1, not from copying the previous chunk. Read against the tolerances, the residuals match Table 1. On Stack Cube the residual falls from 0.056 to 0.047 under a tolerance of 0.09 and the one-step loss of five points turns into a gain of four, while on Apple it falls to 0.034, still above a tolerance of under 0.05, and the recovery is partial. With more steps the residual heals on its own and the spread returns, so the benefit of the prior fades while the cost of a stale extrapolation remains (Appendix A.6).

Table 3: Residual of one step from noise (e) and of the MAWS step (e_{MAWS}) at the same observations of a MAWS rollout, the spread of the MAWS step, and the tolerance $\delta^\star$. A bound marks a tolerance outside the injected range.

Task	$\delta^\star$	e	e_{MAWS}	$\text{spread}_{\text{MAWS}}$
Coke Can	–	.034	.022	.020
Move Near	.11	.044	.029	.022
Drawer	.10	.059	.046	.050
Apple	$<$.05	.047	.034	.029
Carrot	.17	.070	.056	.027
Eggplant	.17	.061	.047	.022
Spoon	.15	.061	.047	.019
Stack	.09	.056	.047	.020

5.4.2 Retention and the replan interval

Fig. 5 sweeps the retention s on the two long-horizon benchmarks, which differ in how much of each chunk the loop executes. On ALOHA ($r{<}H$) success rises to a broad optimum at $s{=}0.5$, close to the full step count, and falls only as s approaches one, where the head is barely consulted and the stale prior is replayed. On CALVIN ($r{=}H$) the prior is pure extrapolation, since the tail is the last action repeated, and the optimum moves down to $s{=}0.25$, with $s{=}0.5$ already collapsing. Keeping less of the prior where the whole chunk is executed is the rule of Sec. 5.1, $s{=}0.25$ for $r{=}H$ and 0.5 otherwise, and we apply it throughout rather than tuning s per task.

5.4.3 Sensitivity to the tail fill

The shifted chunk leaves r vacated tail positions, filled by repeating the last action. Fig. 6 compares this with a linear continuation of the chunk's trend, the same trend damped, the trend scaled by its lag-one autocorrelation, and zeros. Every fill that continues the motion improves on one step from noise by five to nine points, which is the gain of starting from the prior at all. The four differ by a few

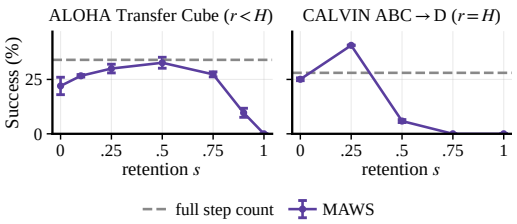


Figure 5: Success rate (%) of MAWS against the retention s for π_0 on two benchmarks that execute different fractions of the chunk. Dashed lines mark the full step count.

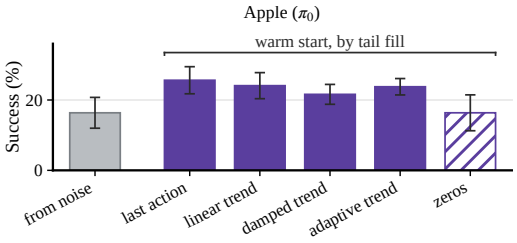


Figure 6: Success rate (%) of MAWS by the fill of the vacated tail positions, against one-step sampling from noise.

Table 4: Success rate (%) on the real robot: six tasks under a $\pi_{0.5}$ policy fine-tuned on our teleoperation data. Head and E2E are the median latency of the denoising segment and of the whole client round trip (Appendix A.7), and the shift and re-noising of MAWS fall outside the head. Late calls exceed the 66.7 ms period of 15 Hz control.

	NFE	T1	T2	T3	T4	T5	T6	Avg	Head (ms)	E2E (ms)	Late calls (%)
$\pi_{0.5}$	10	75	85	35	35	50	45	54.2	17.5	67.9	89
$\pi_{0.5}$	1	70	95	**75**	15	**75**	**75**	67.5 (+13.3)	1.7	52.1	0
+MAWS (Ours)	1	**80**	**100**	**75**	**75**	55	70	**75.8** (+21.6)	1.8	54.7	0

points, within the spread over seeds, whereas zeros, which breaks the motion instead of continuing it, returns to the level of sampling from noise. What the step needs is a tail that carries the chunk forward, and which continuation supplies it matters little. Where $r<H$ the vacated positions lie past the r actions the loop executes, so the tail is replaced before it is ever acted on and only has to place x_s on the marginal, and where $r=H$ the whole prior is that extrapolation and the retention drops accordingly.

5.5 Real-world experiments

We deploy MAWS on a 7-DoF Franka Research 3 arm driven at 15 Hz by a $\pi_{0.5}$ policy fine-tuned on our own teleoperation data (action horizon 16, replan interval 8), on six tabletop tasks, four pick-and-place tasks into a basket or onto a plate (T1 to T4) and two stacking tasks (T5 and T6); the tasks, the setup, the training and the latency measurement are in Appendix A.7. The policy runs at its full step count of ten, at one step from noise, and with MAWS at the setting of Sec. 5.1, so the variants differ only in the sampler. In Table 4, MAWS averages 75.8%, one step from noise 67.5% and the full step count 54.2%. The largest change is on T4, where one step from noise falls to 15% and MAWS returns to 75%. On the harder T5 and T6 one step from noise already exceeds the full step count by 25 to 30 points and MAWS gives part of that back on T5, the regime where the center is already enough. On the RTX 5090 that serves the robot, a full-step call takes 67.9 ms from request to returned chunk and exceeds the 15 Hz period on 89% of calls, whereas a MAWS call takes 54.7 ms and never exceeds it.

6 Conclusion

We asked how few denoising steps a flow-matching action head needs at inference and found that the answer is not a property of the head. One step returns an estimate of the posterior mean with a small residual on every task, and a task collapses once the loop, which integrates that persistent residual, exceeds its tolerance. The single step therefore needs a starting point that carries the executed motion and lies on the training marginal, which MAWS builds from the previous chunk by shifting and re-noising. Without changing a weight, it recovers single-step success across models, benchmarks and a real robot. Its limits follow from the same diagnosis. One step is not enough where the residual stays above the tolerance even from the warm start, as on π_0 Apple, where MAWS reaches 25.6 against 42.6 and needs a second step to come back. Choosing the retention per call from the measured residual is a natural next step, although estimating that residual online would spend what the method saves.

AI USE STATEMENT

We used a generative AI coding assistant (Claude, Anthropic) as a tool throughout this work, under the direction of the authors. Of the tasks for which the ICLR policy requires disclosure, AI assistance was used to implement methods and experiment harnesses, including our ports of the baselines and the open-loop and injection probes; to provide feedback on the design of experiments proposed by the authors; and to assist in interpreting results by tabulating and cross-checking measurements. Generative AI was not used to generate data sets, to formulate or prove mathematical claims, or to propose the research question or the central hypotheses, which were set by the authors; translation, dataset cleaning, qualitative analysis and survey design do not apply to this work. Of the tasks with recommended disclosure, AI assistance was used to draft and edit parts of the text for readability, to create the figures and tables from the measured numbers, to search for and summarize related work, and to format references. All AI-assisted code was reviewed by the authors and validated against published baseline numbers before use, every reported number was checked against the raw experiment logs, and all AI-assisted text was revised and approved by the authors. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies inference-time acceleration of robot manipulation policies in simulation and on a single robot arm in a laboratory. No human-subject data or personal data were collected; the real-robot demonstrations were recorded by the authors teleoperating the arm. All simulated benchmarks and pretrained checkpoints are publicly released by their authors and were used under their licenses. Faster policies lower the latency of robot control but do not change what a policy is trained to do, and we see no application of this work that raises concerns beyond those of the underlying policies. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

The method is fully specified by Eqs. 1 and 2 and Algorithm 1 (Appendix A.2), with its single hyperparameter and every exception stated in Sec. 5.1. The evaluation protocols of all benchmarks, including episode counts, seeds and replan intervals, are given in Appendix A.3, the baseline settings in Sec. 5.3, and the real-robot hardware, training and latency-measurement details in Appendix A.7. The open-loop probes and the injection experiment are defined in Sec. 3 with their raw values in Appendix A.1. Code for MAWS, the probes and the baseline ports, together with the evaluation scripts used for every table, will be released with the camera-ready version.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Haojun Chen, Minghao Liu, Chengdong Ma, Xiaojian Ma, Zailin Ma, Huimin Wu, Yuanpei Chen, Yifan Zhong, Mingzhi Wang, Qing Li, and Yaodong Yang. Falcon: Fast visuomotor policies via partial denoising. In *International Conference on Machine Learning (ICML)*, 2025.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R. Zhang. Sampling is as easy as learning the score: Theory for diffusion models with minimal data assumptions. In *International Conference on Learning Representations (ICLR)*, 2023.

Moritz Diehl, Hans Georg Bock, and Johannes P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5): 1714–1736, 2005.

Yufei Duan, Hang Yin, and Danica Kragic. Real-time iteration scheme for diffusion policy. *arXiv preprint arXiv:2508.05396*, 2025.

Zhou Fang, Jiaqi Wang, Yi Zhou, and Qiongfeng Shi. Probeflow: Training-free adaptive flow matching for vision-language-action models. *arXiv preprint arXiv:2603.17850*, 2026.

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.

Zifeng Gao, Guanxing Lu, Tianxing Chen, Wenxun Dai, Ziwei Wang, Chao Shang, Wenbo Ding, and Yansong Tang. ManiCM: Real-time 3D diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

Sunghwan Han, Youngtae Han, and Youngmin Yi. Adavla: Adaptive step flow matching for training-free acceleration of vision-language-action models. *arXiv preprint arXiv:2608.29208*, 2026.

Kian Hosseinkhani, Qinhe Peng, George Shramko, Mehran Aghabozorgi, Jianing Qian, Tristan Engst, Alireza Moazeni, Dinesh Jayaraman, and Ke Li. Imle-vla: Fast single-step action generation for vision-language-action policies. *arXiv preprint arXiv:2609.10915*, 2026.

Kangye Ji, Yuan Meng, Hanyun Cui, Ye Li, Jianbo Zhou, Shengjia Hua, Lei Chen, and Zhi Wang. Block-wise adaptive caching for accelerating diffusion policy. *arXiv preprint arXiv:2506.13456*, 2025.

Sinjae Kang, Chanyoung Kim, Kaixin Wang, Li Zhao, and Kimin Lee. WarmPrior: Straightening flow-matching policies with temporal priors. *arXiv preprint arXiv:2605.13959*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

Jinhao Li, Yuxuan Cong, Yingqiao Wang, Hao Xia, Shan Huang, Yijia Zhang, Ningyi Xu, and Guohao Dai. STEP: Warm-started visuomotor policies with spatiotemporal consistency prediction. *arXiv preprint arXiv:2602.08245*, 2026.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Conference on Robot Learning (CoRL)*, 2024.

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

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

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022.

NVIDIA, Johan Bjorck, Fernando Castaneda, Nikita Cherniadev, Xingye Da, Runyu Ding, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Ryuji Oi, Hikari Otsuka, Kosuke Matsushima, Yuki Ichikawa, Masato Motomura, Tatsuya Kaneko, and Daichi Fujiki. Actioncache: Training-free acceleration for vision-language-action models with action caching and refinement. *arXiv preprint arXiv:2607.06370*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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 Stirling, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky.
$pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems (RSS)*, 2024.

James B. Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2nd edition, 2017.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. *arXiv preprint arXiv:2503.09722*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *arXiv preprint arXiv:2506.10100*, 2025.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

A Appendix

A.1 Open-loop probes

Two tables support the analysis of Sec. 3. Table S.1 holds the open-loop probes behind Fig. 2, measured once at fixed observations, and Table S.2 follows the residual along the episode, which is what the closed-loop reading of Sec. 3.2 needs.

Table S.1: Fixed-observation properties of the flow for every task. Fig. 2 defines S, err_1, spread_K and Δ_1, e follows from the decomposition above, and spread_1 is what spread_K is for the one-step chunk. Every entry is measured at real observations, three progress points per episode, over three episodes with two noise seeds for S and err_1 and two episodes with eight noise seeds for the rest. K is the full step count, ten for π_0 and four for GR00T N1.7. S and err_1 are dimensionless and the other four are in each model's own normalized action units, so they compare across models only loosely.

π_0, *SimplerEnv*	S	err_1	spread_1	spread_K	Δ_1	e
Coke Can	.040	.123	.012	.060	.074	.044
Move Near	.041	.128	.014	.074	.088	.048
Open Drawer	.063	.189	.014	.118	.132	.058
Close Drawer	.087	.231	.016	.131	.149	.071
Apple	.060	.172	.013	.094	.107	.050
Carrot	.052	.169	.020	.131	.157	.085
Spoon	.039	.131	.018	.089	.117	.075
Stack	.054	.165	.015	.104	.119	.056
Eggplant	.056	.165	.017	.108	.120	.053
GR00T N1.7, SimplerEnv	S	err_1	spread_1	spread_K	Δ_1	e
Coke Can	.061	.216	.222	.279	.285	.059
Move Near	.071	.240	.222	.283	.290	.065
Open Drawer	.066	.226	.227	.282	.288	.054
Close Drawer	.079	.267	.228	.291	.300	.072
Apple	.067	.240	.227	.282	.290	.068
Carrot	.059	.214	.205	.242	.248	.052
Spoon	.057	.200	.201	.233	.238	.050
Stack	.058	.206	.206	.238	.248	.063
Eggplant	.068	.212	.191	.222	.229	.054

Table S.2: Residual e along the episode, at policy calls 0 to 60 of the full-step policy (π_0: two episodes and eight noise seeds per call; GR00T N1.7: three environments and eight seeds; dashes where every episode had already ended). Tolerances: π_0 Open Drawer 0.07 and Apple below 0.05; GR00T N1.7 Open Drawer about 0.06 and Close Drawer 0.10.

	call	0	5	10	15	20	25	30	35	40	45	50	55	60
π_0	Open Drawer	.077	.049	.038	.047	.040	.046	.043	.072	.057	.049	.063	.060	–
π_0	Apple	.072	.036	.042	.073	.071	.037	.046	.062	.072	.068	.059	.035	.030
GR00T N1.7	Open Drawer	.044	.051	.069	.062	.078	.087	.075	.085	.096	.081	.067	.086	.088
GR00T N1.7	Close Drawer	.073	.069	.076	.083	.080	.062	.070	–	–	–	–	–	–

The residual is not measured directly but read off a decomposition of Δ_1. Write the N full-step chunks at one observation as $a_K^{(n)} = \bar{a}_K + \delta_n$, so that the deviations δ_n from their mean sum to zero. Expanding their squared distance to the mean one-step chunk $\bar{a}_1$ then leaves no cross term,

$$\frac{1}{N}\sum_n \left\| a_K^{(n)} - \bar{a}_1 \right\|^2 = \left\| \bar{a}_K - \bar{a}_1 \right\|^2 + \frac{1}{N}\sum_n \|\delta_n\|^2,$$

which is $\Delta_1^2 = e^2 + \mathrm{spread}_K^2$, with e the distance between the one-step mean and the full-step mean. We list all three because Δ_1 and spread_K are probes in their own right and the decomposition says what Δ_1 contains. The scatter is the fresh part, which the full-step sampler also pays at every call. The offset is the persistent part, which only the single step carries.

The chunk-level multimodality score, the distance from the sample mean to its nearest sample over the median inter-sample distance, lies in $[0.41, 0.50]$ on π_0 and $[0.57, 0.62]$ on GR00T N1.7 and is omitted from the table. The GR00T N1.7 rows use the head's inputs recorded during short rollouts, three environments and calls 0, 5 and 10 of the first episode, fewer where it ended earlier, and the seven real dimensions of its padded action output. All of their columns come from the same frames.

Algorithm 1 One policy call with MAWS

Require: observation o_t; previous chunk a_{t-1} (none on the first call); retention s; replan interval r

if a_{t-1} is none **then**
  $g_t \leftarrow 0$, $s \leftarrow 0$ ▷ first call: one step from noise
else
  $g_t \leftarrow a_{t-1}$ shifted by r, tail filled with its last action
end if
$\varepsilon \sim \mathcal{N}(0, I)$; $x_s \leftarrow (1-s)\,\varepsilon + s\,g_t$
$a_t \leftarrow x_s + (1-s)\,v_\theta(x_s, s, o_t)$
execute the first r actions of a_t; keep a_t for the next call

Table S.3: Success rate (%) on LIBERO ($\pi_{0.5}$, four suites, 500 episodes each at the default replan r=5, mean±sd over three seeds). Bold marks the best entry of a column, red the lowest average, and parentheses the change of the average against the full step count.

LIBERO	NFE	Spatial	Object	Goal	Long	Avg
$\pi_{0.5}$	10	98.5±0.2	98.1±0.2	97.8±0.5	92.7±0.3	96.8
$\pi_{0.5}$	1	98.5±0.3	98.5±0.5	97.7±1.0	91.7±0.5	96.6 (-0.2)
+MAWS (Ours)	1	**98.6**±0.2	**99.3**±0.2	**98.7**±0.2	**94.4**±0.4	**97.7** (+0.9)

A.2 One policy call with MAWS

Algorithm 1 lists one policy call, including the first call of an episode, which samples from noise.

A.3 Evaluation protocol

SimplerEnv. Visual Matching with the official task sets and episode counts of each harness, using the open-pi-zero PyTorch checkpoints of π_0 (Fractal and Bridge) and NVIDIA's SimplerEnv checkpoints of GR00T N1.7. Fractal (Google robot): Pick Coke Can in three orientations (3×100 episodes), Move Near (240), Open and Close Drawer (2×108) and Place Apple in Closed Top Drawer (108), 864 episodes per seed. Bridge (WidowX): Carrot on Plate, Eggplant in Basket, Spoon on Towel and Stack Cube, 240 enumerated episodes each. Every episode is capped at 300 environment steps. The tail-fill ablation (Fig. 6) runs on Place Apple at these counts, 108 episodes per point and three seeds. π_0 uses seeds 42/43/44 and GR00T N1.7 seeds 42/52/62. The injection experiment (Fig. 3) is a single seed, and its GR00T N1.7 curves were collected with a longer episode cap and rescored to the 300-step one. The one-step and MAWS rows run on the same server with the same client, reported NFE is counted on the server, and latency is measured with the policy server alone on the GPU.

Long-horizon benchmarks. ALOHA transfer-cube (π_0, JAX openpi): 100 episodes per point, three seeds. CALVIN ABC→D (π_0 and $\pi_{0.5}$, RLinf checkpoints): 1000 chains per point, three seeds, success measured at chain end under the harness protocol of 480 steps per chain, a within-paper comparison rather than the official CALVIN score. LIBERO ($\pi_{0.5}$): 500 episodes per suite, three seeds, at the default replan interval r=5.

A.4 Additional benchmarks

Tables S.3 and S.4 report the one-step and warm-start rows on LIBERO, CALVIN and ALOHA under the protocol above, and Table S.5 the same step reduction on a diffusion action head (CogACT sampled with DDIM).

A.5 Comparison with one-step methods that train

Methods that retrain the action head for one step report on LIBERO under $\pi_{0.5}$ and not on SimplerEnv, so the comparison is made there (Table S.6), against the numbers each paper reports for its own ten-step policy and its one-step result. SnapFlow (Luan et al., 2026) distils the flow into

Table S.4: Success rate (%) on the two long-horizon benchmarks: CALVIN ABC→D (success at chain end, 1000 chains per seed) and ALOHA TransferCube (100 episodes), three seeds each.

CALVIN	NFE	Chain
$\pi_{0.5}$	5	46.5±0.6
$\pi_{0.5}$	1	45.9±0.6 (-0.6)
+MAWS	1	**49.8**±1.0 (+3.3)
π_0	5	28.0±0.5
π_0	1	25.1±0.6 (-2.9)
+MAWS	1	**40.7**±0.2 (+12.7)
ALOHA	NFE	Cube
π_0	10	**34.0**±3.0
π_0	1	22.0±4.0 (-12.0)
+MAWS	1	32.7±2.5 (-1.3)

Table S.5: Success rate (%) of CogACT, whose action head is a *diffusion* model sampled with DDIM, on SimplerEnv Pick Coke Can, pooled over the three orientations (25 episodes each, three seeds).

CogACT	NFE	Coke Can
CogACT (DDIM)	10	**95.1**±1.5
CogACT (DDIM)	1	72.9±3.4 (-22.2)
+MAWS	1	90.7±3.5 (-4.4)

Table S.6: LIBERO under $\pi_{0.5}$: one-step methods that retrain the head, each against the ten-step policy of its own paper, next to MAWS on the openpi checkpoint. Protocols differ (SnapFlow 10 episodes per task, IMLE-VLA 50, DriftingVLA 100 over three seeds, ours 50 per task over three seeds), so absolute numbers are not comparable across blocks; Δ is the change from each block's own ten-step policy.

	NFE	Spatial	Object	Goal	Long	Avg	Δ
SnapFlow (Luan et al., 2026), progressive self-distillation							
$\pi_{0.5}$	10	98.0	100.0	96.0	97.0	97.8	
$\pi_{0.5}$	1	96.0	99.0	98.0	94.0	96.8	−1.0
+SnapFlow	1	99.0	100.0	99.0	97.0	98.8	+1.0
IMLE-VLA (Hosseinkhani et al., 2026), head retrained with conditional IMLE							
$\pi_{0.5}$	10	97.2	99.0	97.8	96.0	97.5	
+IMLE-VLA	1	98.0	99.8	98.2	96.0	98.0	+0.5
DriftingVLA (Gao et al., 2026), new one-step head							
$\pi_{0.5}$	10	98.4	99.5	98.1	92.4	97.1	
$\pi_{0.5}$	1	95.3	96.0	94.3	90.5	94.0	−3.1
+DriftingVLA	1	99.7	99.9	99.3	94.3	98.3	+1.2
MAWS *(Ours), training-free*							
$\pi_{0.5}$	10	98.5	98.1	97.8	92.7	96.8	
$\pi_{0.5}$	1	98.5	98.5	97.7	91.7	96.6	−0.2
+MAWS	1	98.6	99.3	98.7	94.4	97.7	+0.9

a one-step student by progressive self-distillation. IMLE-VLA (Hosseinkhani et al., 2026) retrains the action head under a conditional IMLE objective. DriftingVLA (Gao et al., 2026) trains a new one-step head. Evaluation protocols differ across papers, so we compare each method's change over its own ten-step policy: retraining gains 0.5 to 1.2 points at one step, and MAWS gains 0.9 on the released checkpoint without touching the weights, from a one-step-from-noise drop that is already small in our runs (0.2 points, against the 1.0 and 3.1 that two of the papers report for their own one-step baselines).

A.6 Denoising step sweep

Fig. S.1 varies the step count with and without the warm start at matched budget. Taking k steps of length $(1 - s)/k$ from x_s runs MAWS at k steps, where the first call of an episode also uses

Table S.7: Success rate (%) at two denoising steps, sampling from noise and with MAWS, against the checkpoint at its full step count and against MAWS at one step, mean over seeds. SimplerEnv rows are episode-weighted suite averages.

	full steps	full	MAWS 1 step	from noise 2 steps	MAWS 2 steps
π_0, Fractal	10	77.2	77.4	75.0	**79.1**
π_0, Bridge	10	68.2	74.7	**76.2**	73.4
GR00T N1.7, Fractal	4	78.1	76.0	75.5	**78.9**
GR00T N1.7, Bridge	4	59.2	56.2	56.7	**60.8**

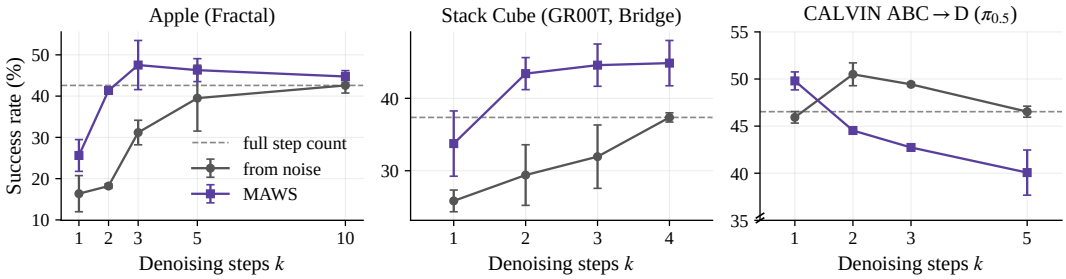


Figure S.1: Success rate as a function of the number of denoising steps k, sampling from noise and with MAWS, mean±sd over seeds; the dashed line is the checkpoint at its full step count (10 for π_0, 4 for GR00T N1.7, 5 for $\pi_{0.5}$). Panels: Apple (π_0, Fractal), Stack Cube (GR00T N1.7, Bridge) and CALVIN ABC→D ($\pi_{0.5}$, $r{=}H$).

k steps so that every call costs k evaluations. Adding steps after MAWS moves the output along the mean dial of Sec. 3.1. The residual shrinks and the spread returns, so the MAWS curve peaks where the residual crosses the loop's tolerance and then approaches the from-noise curve. In the collapse regime the peak is at two or three steps (Apple peaks at $k{=}3$, above the ten-step score), and on GR00T, whose single step retains more of the noise, two MAWS steps recover the full-step average on both suites (Sec. 5). On GR00T the prior keeps paying through the full count. On Stack Cube the MAWS curve sits above the from-noise curve at every k and above the four-step score from $k{=}2$ on, because a less complete single step leaves a larger residual for the prior to remove. CALVIN under $\pi_{0.5}$, whose full step count is five and where the whole chunk is executed ($r{=}H$), shows the other end. Sampling from noise peaks at two steps, four points above the full count, and the warm start helps only at one step, where it matches that peak at half the cost; from two steps on the MAWS curve falls below the from-noise curve and keeps falling, ending well under the full count. The prior's benefit is tied to the single step, where it offsets the persistent residual. Its cost, an extrapolated chunk the loop has already moved past, is paid at every step count. With more steps the residual heals on its own, the benefit is gone and the cost remains.

Table S.7 isolates the two-step point at matched budget, next to MAWS at one step. A second MAWS step recovers the full-step score in every collapse case we have (π_0 Apple, GR00T on both suites), adding 1.7 to 4.6 points over one step, and it is unnecessary where one step already sufficed: on π_0 Bridge two from-noise steps exceed the full count on their own, and there they also edge MAWS; on CALVIN they beat it by six points. The second step is therefore a remedy for the cases the analysis flags, not a general improvement.

A.7 Real-world experiment

Hardware. A 7-DoF Franka Research 3 arm, with state and action both given by the joint positions and a binary gripper state, controlled at 15 Hz. Two Zed 2i cameras give a primary and a secondary view and a Zed mini on the wrist a third, and following the openpi convention the policy receives the primary view. Fig. S.2 shows the setup.

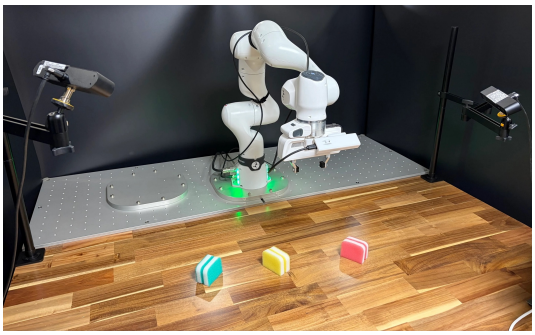


Figure S.2: Real-robot evaluation setup. The Franka Research 3 arm faces a tabletop holding the objects of the six tasks, with the two Zed 2i cameras in front and the Zed mini on the wrist.

Tasks. Six tabletop tasks, given to the policy as the following instructions. T1: pick the sponge and place it into the basket. T2: pick the white plush and place it into the basket. T3: pick the sponge and place it into the plate. T4: pick the white plush and place it into the plate. T5: stack the yellow cup into the blue cup. T6: place the red sponge onto the yellow sponge. T1 to T4 are pick-and-place tasks with 25 demonstrations each, and T5 and T6 require aligning one object on another and have 50 each. Fig. S.3 shows one rollout per task.

Policy. $\pi_{0.5}$ fine-tuned for 20k steps on four A40 GPUs on our own dataset, collected by an expert teleoperating the arm with a Quest 3 controller. AdamW (β_1=0.9, β_2=0.95, weight decay 10^{-10}), peak learning rate 2.5×10^{-5} with 1000 warmup steps and cosine decay to 2.5×10^{-6}, gradient clipping at 1.0, global batch size 32. Actions are delta positions, the action horizon is H=16 and the executed prefix is r=8. Each task is evaluated over 20 trials per variant. The full step count of the checkpoint is ten, and MAWS uses s=0.5 with one step from noise on the first call, as in the simulated benchmarks.

Latency. Measured on the RTX 5090 that serves the robot. Over 200 calls, the prefix pass (image and language encoding) takes 39.7 ms and each denoising step 1.76 ms, so the denoising segment takes 17.5 ms at the full step count, 1.7 ms at one step and 1.8 ms with MAWS. The end-to-end figures of Table 4 replay 254 requests from six dataset episodes through the server on the same path as the robot client. On top of the sampler they carry the server time (input resizing, normalization and tokenization and output de-normalization, about 8 ms in every variant) and the client round trip (msgpack serialization and websocket transport, about 2 ms more), which is what the control loop waits for and is compared with the 66.7 ms period of 15 Hz control. Its p99 is 70.7 ms at the full step count, 54.6 ms at one step and 58.1 ms with MAWS.



T1: ***Pick sponge and place it into the basket***



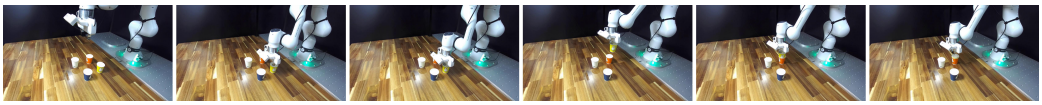
T2: ***Pick white plush and place it into the basket***



T3: ***Pick sponge and place it into the plate***



T4: ***Pick white plush and place it into the plate***



T5: ***Stack yellow cup into the blue cup***



T6: ***Place the red sponge onto the yellow sponge***

Figure S.3: Qualitative robot manipulation rollouts across six language-conditioned tasks. Each row shows six temporally ordered frames.