COCOA: ACTION-CONDITIONED CORRECTION FOR VISION-LANGUAGE-ACTION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models have shown strong performance on complex robot manipulation tasks by generating low-level action chunks from multimodal observations and language instructions. However, an erroneous action chunk can derail long-horizon execution and lead to task failure. To address this challenge, we propose *COCOA*, an action-conditioned correction training framework that learns a separate corrective policy to revise action chunks proposed by a base VLA before execution. Given the current visual observations, language instruction, robot state, and a proposed action chunk, the corrective policy generates a corrected action chunk for execution. To train the corrective policy, *COCOA* introduces a two-stage training procedure that combines supervised flow-matching training with correction preference optimization for flow-based VLAs. We apply *COCOA* to three flow-based VLA backbones, including $\pi_{0.5}$, FLOWER VLA, and Xiaomi-Robotics-0. Our experiments show that *COCOA* achieves consistent improvements on two simulation benchmarks, SimplerEnv and CALVIN, as well as on real-robot tasks.

1 INTRODUCTION

Generalist robot manipulation requires translating visual observations and natural-language instructions into precise low-level control across diverse tasks, scenes, and embodiments. VLA models address this goal by adapting pretrained vision-language representations to robot action prediction (Driess et al., 2023; Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024). Recent flow-based VLAs model continuous action distributions through flow matching and predict action chunks: sequences of future robot commands conditioned on the current observation and task (Black et al., 2024; Intelligence et al., 2025; Reuss et al., 2025; Cai et al., 2026b).

Despite this progress, long-horizon manipulation remains brittle. A VLA may generate a suboptimal action chunk due to distribution shift, compounding errors, or stochastic action generation (Sendai et al., 2025; Zhang et al., 2026; Pan et al., 2026). Once executed in the physical environment, even a single erroneous action chunk can have lasting consequences: the robot may push an object out of reach, collide with the workspace, or close the gripper prematurely. Such errors can induce irreversible physical changes or drive the robot into states from which the VLA policy is difficult to recover (Ma et al., 2026; Pan et al., 2026). Fig. 1 shows an example where the VLA model is instructed to open the top drawer and place the apple inside. At t_2, the base VLA (top) generates a misaligned action chunk that causes the robot to inadvertently push the drawer back instead of opening it further, leaving the VLA policy unable to recover later in the trajectory.

Ensuring that VLA policies generate correct and safe action chunks is therefore critical, yet remains challenging. Scaling the base VLA, collecting more demonstrations, or fine-tuning the full policy can improve overall task performance (Intelligence et al., 2025; Kim et al., 2025), but these approaches primarily improve the action generator itself. Standard VLA systems still directly execute the generated action chunk without an explicit mechanism to inspect and revise the proposed actions before execution (Intelligence et al., 2025; Kim et al., 2025; Bjorck et al., 2025; Reuss et al., 2025; Cai et al., 2026b). This motivates a complementary question: *can a VLA policy benefit from a separate model that revises its proposed action chunks before execution?* The proposed action chunk provides a concrete representation of the base policy's intended behavior. Jointly conditioning on this proposal together with the current visual observation, robot state, and language instruction pro-

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.

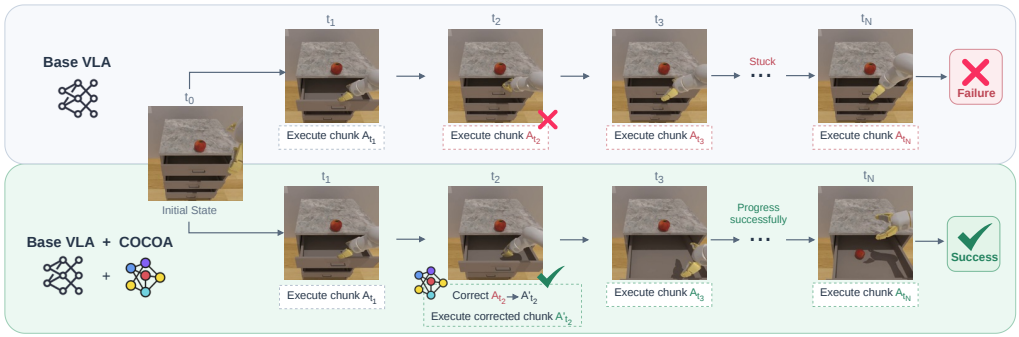


Figure 1: **Motivation example.** The task is "Open the top drawer and place the apple into the top drawer" from the SimplerEnv benchmark. **Top: Base VLA only.** At t_1, the robot partially opens the drawer. At t_2, the robot is expected to grasp the upper edge of the drawer and pull it further open. However, due to spatial misalignment in the proposed action chunk A_{t_2}, the gripper instead contacts the upper edge of the drawer and pushes the drawer back rather than pulling it further open. The robot subsequently becomes stuck against the tabletop and fails to complete the task. **Bottom: Base VLA + COCOA.** The trajectory is identical through t_1. At t_2, COCOA revises the proposed action chunk A_{t_2} into A'_{t_2} before execution. The corrected chunk successfully opens the drawer further, allowing the robot to continue the task and ultimately place the apple into the drawer.

vides a richer signal for determining whether and how the proposed action should be revised. We therefore learn a separate corrective policy that takes both the VLA input context and its proposed action chunk as input and generates an improved action chunk before execution. This yields a modular formulation in which the base VLA remains responsible for task-conditioned action generation, while the corrective policy specializes in proposal-conditioned action correction.

We propose COCOA, an *action-conditioned correction training framework* for VLA policies. Given the current visual observations, language instruction, robot state, and an action chunk proposed by a base VLA as input, COCOA learns a separate *corrective policy* that generates an improved action chunk for execution. We instantiate the corrective policy from a pretrained flow-based VLA and augment it with an action encoder that incorporates the proposed action chunk as an additional conditioning signal. COCOA proposes a two-stage training for developing the corrective policy. First, supervised flow-matching training provides a cold start for action correction. Second, we introduce *correction preference optimization*, adapting Self-Improving Robust Preference Optimization (SRPO) (Choi et al., 2025) to flow-based action prediction, to further improve action correction while discouraging unnecessary modification of already desirable actions. We also build an automated pipeline to construct action pairs as correction training data.

Our contributions are:

- We introduce COCOA, an action-conditioned training framework for VLA correction that learns a separate corrective policy to revise full action chunks proposed by a frozen base VLA before execution.
- We develop a two-stage training procedure that combines supervised flow matching with correction preference optimization for flow-based VLAs, using automatically constructed proposal–expert pairs to learn action correction without manual correction annotations.
- We demonstrate the effectiveness and generality of COCOA across three VLA backbones, $\pi_{0.5}$, FLOWER VLA, and Xiaomi-Robotics-0, on CALVIN and SimplerEnv benchmarks. COCOA consistently improves base VLA policies, increasing FLOWER VLA's five-instruction success rate on CALVIN from 77.8% to 81.9% and Xiaomi-Robotics-0's overall success rate on SimplerEnv Visual Matching from 85.5% to 90.4%.

2 Related Work

Vision-Language-Action Models and Action Chunking Vision-language-action (VLA) models adapt pretrained vision-language representations to language-conditioned robot control (Driess

et al., 2023; Zitkovich et al., 2023; Kim et al., 2024). Action-sequence policies such as ACT (Zhao et al., 2023) and Diffusion Policy (Chi et al., 2025) demonstrated the benefits of predicting temporally coherent action chunks, while recent flow-based VLAs combine multimodal backbones with continuous action decoders, including the π_0 family (Black et al., 2024; Driess et al., 2025; Intelligence et al., 2026), FLOWER (Reuss et al., 2025), Evo-1 (Lin et al., 2026), and Xiaomi-Robotics-0 (Cai et al., 2026b). Other work augments VLAs with reasoning, future prediction, world models, or visual subgoals (LeCun, 2022; Bi et al., 2025; Wang et al., 2025; Cen et al., 2025; Lv et al., 2025; Zhu et al., 2025; 2026; Zhang et al., 2025; Zhao et al., 2025; Cai et al., 2026a; Gu et al., 2025). While these approaches improve action generation, COCOA instead learns to revise an action chunk proposed by a base VLA before execution.

Action Refinement and Self-Correction Residual policies learn additive corrections to fixed controllers (Silver et al., 2018; Davchev et al., 2022), while Policy Decorator extends this idea to frozen imitation policies through online residual RL (Yuan et al., 2025). Recent VLA methods perform correction in different ways: SC-VLA generates corrective actions after failures (Li et al., 2024a), A2C2 applies per-step residual corrections during action-chunk execution (Sendai et al., 2025), and CycleVLA, VLA-SCT, and VLA-Corrector detect failures or deviations and trigger backtracking, termination, or replanning (Ma et al., 2026; Zhang et al., 2026; Pan et al., 2026). Other approaches use world imagination, iterative action refinement, or value-guided sampling (Liu et al., 2026; Chen et al., 2026; Song et al., 2026). In contrast, COCOA performs pre-execution correction with an offline-trained, proposal-conditioned corrective VLA that replaces a full continuous action chunk proposed by a frozen base policy, without online interaction or post-failure recovery.

Preference-Based Policy Improvement Preference learning improves policies from comparisons between desirable and undesirable outputs (Christiano et al., 2017; Ouyang et al., 2022; Rafailov et al., 2023). Diffusion-DPO and Flow-DPO extend preference optimization to continuous generative models (Wallace et al., 2024; Liu et al., 2025), while Action Preference Optimization applies preference learning to VLA refinement using human interventions (Xia et al., 2025). COCOA instead automatically constructs preferred/dispreferred action pairs from demonstrations and model-generated actions and applies the correction component of SRPO (Choi et al., 2025) to a separate corrective policy while keeping the base VLA frozen.

3 PRELIMINARIES

We consider language-conditioned robot manipulation with VLA policies. At control step t, given an observation o_t and a natural-language instruction ℓ, a VLA model is typically trained to maximize the log-likelihood of an action chunk A_t: $\max_\theta \mathbb{E}_{(A_t,o_t,\ell)\sim\mathcal{D}} \log\big(\pi_\theta(A_t|o_t,\ell)\big)$. The observation o_t contains one or more images $I_t^1,\dots,I_t^M$ and a proprioceptive state q_t, such that $o_t = \{I_t^1,\dots,I_t^M,q_t\}$. The action chunk A_t consists of future actions: $A_t = [a_t, a_{t+1},\dots,a_{t+H-1}] \in \mathbb{R}^{H\times d_a}$ where H is the action horizon and d_a is the action dimension.

Recent flow-based VLAs (Black et al., 2024; Driess et al., 2025; Reuss et al., 2025; Cai et al., 2026b; Lin et al., 2026) typically combine a pretrained vision-language model (VLM) with a transformer-based flow-matching action module. The VLM encodes the visual observations and language instruction into multimodal representations, while the action module generates continuous action chunks conditioned on these representations and the robot proprioceptive state. Specifically, a VLA is trained to minimize the conditional flow matching loss:

$$\mathcal{L}_{\mathrm{F}}(\theta) = \mathbb{E}_{(o_t,\ell,A_t),\epsilon,\tau}\left[\left\|\epsilon - A_t - v_\theta((A_t)_\tau,\tau,o_t,\ell)\right\|^2\right]. \tag{1}$$

Under a rectified-flow formulation, we sample Gaussian noise $\epsilon \sim \mathcal{N}(0,I)$ and a flow time $\tau \in [0,1]$, then constructs the interpolated action

$$(A_t)_\tau = (1-\tau)A_t + \tau\epsilon. \tag{2}$$

The model predicts a velocity field $v_\theta((A_t)_\tau,\tau,o_t,\ell)$ that is trained to match the target velocity $u_t = \epsilon - A_t$. At inference time, the VLA starts from Gaussian noise and integrates the learned velocity field to produce a continuous action chunk.

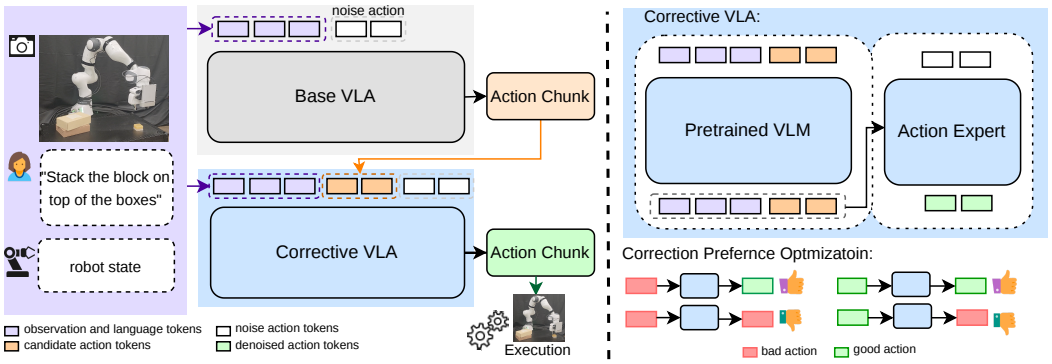


Figure 2: Overview of COCOA. A base VLA proposes a candidate action chunk from the observation and language instruction, and a corrective VLA conditions on that proposal to predict a repaired action chunk before execution.

4 METHOD

4.1 OVERVIEW

COCOA improves a pretrained VLA by learning a separate corrective policy π_c. Given the current observation o_t, language instruction ℓ, and an initial action chunk $\widetilde{A}_t \sim \pi_b(\cdot \mid o_t, \ell)$ proposed by the base VLA, the corrective policy π_c generates a corrected action chunk

$$A'_t \sim \pi_c(\cdot \mid o_t, \ell, \widetilde{A}_t). \tag{3}$$

The robot executes A'_t instead of $\widetilde{A}_t$. This design decouples action correction from action generation: the base VLA remains frozen, while the corrective policy specializes in mapping locally suboptimal action chunks to higher-quality chunks in the same action space.

We initialize the corrective policy from a pretrained VLA model. Following the VLA architecture described in the preliminaries, which consists of a VLM backbone and a flow-based action head, COCOA augments the VLM backbone with an action encoder that incorporates the proposed action chunk as an additional conditioning signal. Specifically, the proposed action chunk $\widetilde{A}_t \in \mathbb{R}^{H \times d_a}$ is flattened and passed through the action encoder, which produces K continuous action tokens. We additionally prepend a learnable special token <action> to the action-token sequence. These candidate-action tokens are then concatenated with the observation and language tokens and jointly processed by the VLM backbone, allowing the resulting hidden states to encode the visual context, language instruction, and proposed action chunk.

The flow-based action head remains unchanged. It denoises a noisy target action chunk while conditioning on the hidden states produced by the VLM backbone, which now contain information from the observation, instruction, and candidate action.

4.2 STAGE I: SFT COLD-START FOR ACTION CORRECTION

The first stage initializes the correction behavior with supervised fine-tuning. We construct a dataset $\mathcal{D}_{\text{correct}} = \{(o_i, \ell_i, A_i^l, A_i^w)\}$, where A_i^l is a lower-quality action chunk sampled from a weak VLA (Sec.4.4) and A_i^w is the corresponding expert action chunk from the demonstration dataset. The correction model is trained to predict A_i^w while conditioning on A_i^l:

$$\mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{(o,\ell,A^l,A^w),\epsilon,\tau}\left[\left\|\epsilon - A^w - v_\theta((A^w)_\tau, \tau, o, \ell, A^l)\right\|^2\right], \tag{4}$$

where $(A^w)_\tau = (1-\tau)A^w + \tau\epsilon$. This is the standard rectified-flow objective in Eq. 1, with the only change being that the bad action chunk A^l is provided as additional context. We initialize the corrective VLA policy from a weak VLA checkpoint (Sec. 4.4) and randomly initialize the new action encoder and the special action token.

4.3 Stage II: Action Correction via Preference Optimization

Cold-start training teaches the correction model to reproduce expert chunks, but never which of two candidates is better, and nothing in Eq. 4 discourages over-correction: editing a chunk that was already good. Stage II therefore replaces this absolute target with a *relative* one that rewards moving toward the preferred chunk and penalizes moving away from it.

We build on Self-Improving Robust Preference Optimization (SRPO) (Choi et al., 2025): given an input chunk A, the correction model should return a chunk A' preferred to A while staying close to the cold-start model π_{ref}. Written as an objective to be minimized,

$$\min_{\pi_{\mathrm{c}}} \mathbb{E}_{\substack{A\sim\pi_{\mathrm{b}}(\cdot|o,\ell) \\ A'\sim\pi_{\mathrm{c}}(\cdot|o,\ell,A)}} \left[-\,p(A' \succ A \mid o,\ell) + \beta D_{\mathrm{KL}}(\pi_{\mathrm{c}}(A' \mid o,\ell,A) \,\|\, \pi_{\mathrm{ref}}(A' \mid o,\ell,A))\right], \quad (5)$$

where $p(A' \succ A \mid o,\ell)$ is the probability that A' is preferred to A, and $\beta > 0$ controls deviation from π_{ref}. Only π_{c} is trained, initialized from Stage I; the base VLA stays frozen.

A key challenge in adapting SRPO to a flow-based VLA is that the chunk-level likelihood $\pi(A' \mid o,\ell,A)$ is intractable for both π_{c} and π_{ref}, as its evaluation requires marginalizing over all flow trajectory paths $A'_{\tau_{0:T}}$. Following previous works (Liu et al., 2025; Wallace et al., 2024), we instead upper-bound the KL term with the joint trajectory divergence $D_{\mathrm{KL}}\big(\pi_{\mathrm{c}}(A'_{\tau_{0:T}} \mid o,\ell,A) \,\|\, \pi_{\mathrm{ref}}(A'_{\tau_{0:T}} \mid o,\ell,A)\big)$ over the sampling trajectory $A'_{\tau_{0:T}}$ (see details in Appendix A.1), giving the objective

$$\min_{\pi_{\mathrm{c}}} \mathbb{E}_{\substack{A\sim\pi_{\mathrm{b}}(\cdot|o,\ell) \\ A'_{\tau_{0:T}}\sim\pi_{\mathrm{c}}(\cdot|o,\ell,A)}} \big[-\,p(A' \succ A \mid o,\ell) + \beta D_{\mathrm{KL}}\big(\pi_{\mathrm{c}}(A'_{\tau_{0:T}} \mid o,\ell,A) \,\|\, \pi_{\mathrm{ref}}(A'_{\tau_{0:T}} \mid o,\ell,A)\big)\big]. \quad (6)$$

Still, Eq. 6 cannot be minimized directly: it requires p, which is unknown since only binary preference labels $A^w \succ A^l$ are observed. Its minimizer is available in closed form, $\pi^\dagger \propto \pi_{\mathrm{ref}}(A'_{\tau_{0:T}} \mid o,\ell,A)\exp\big(p(A' \succ A \mid o,\ell)/\beta\big)$. Using the convention $p(A \succ A) = \frac{1}{2}$ to eliminate the normalizer, this is equivalent to the identity

$$p(A' \succ A \mid o,\ell) = \frac{1}{2} + \beta\left[\log\frac{\pi^\dagger(A'_{\tau_{0:T}} \mid o,\ell,A)}{\pi_{\mathrm{ref}}(A'_{\tau_{0:T}} \mid o,\ell,A)} - \log\frac{\pi^\dagger(A_{\tau_{0:T}} \mid o,\ell,A)}{\pi_{\mathrm{ref}}(A_{\tau_{0:T}} \mid o,\ell,A)}\right]. \quad (7)$$

SRPO therefore fits π_{c} to $\pi^\dagger$ by minimizing the ℓ_2 residual of this identity, evaluated on labelled preference pairs (see details in Appendix A.2).

We instantiate this loss on the pairs (A^w, A^l) of Sec. 4.4, conditioning on each member in turn and averaging the trajectory log-ratios over the forward process q defined in Appendix A:

$$\begin{aligned}\mathcal{L}_{\mathrm{SI}}(\pi_{\mathrm{c}}) = \mathbb{E}_{(A^w,A^l,o,\ell)\sim\mathcal{D}}\Bigg[&\left(\frac{1}{2} - \beta\,\mathbb{E}_q\left[\log\frac{\pi_{\mathrm{c}}(A^w_{\tau_{0:T}} \mid o,\ell,A^l)}{\pi_{\mathrm{ref}}(A^w_{\tau_{0:T}} \mid o,\ell,A^l)} - \log\frac{\pi_{\mathrm{c}}(A^l_{\tau_{0:T}} \mid o,\ell,A^l)}{\pi_{\mathrm{ref}}(A^l_{\tau_{0:T}} \mid o,\ell,A^l)}\right]\right)^2 \\ &+ \left(\frac{1}{2} - \beta\,\mathbb{E}_q\left[\log\frac{\pi_{\mathrm{c}}(A^w_{\tau_{0:T}} \mid o,\ell,A^w)}{\pi_{\mathrm{ref}}(A^w_{\tau_{0:T}} \mid o,\ell,A^w)} - \log\frac{\pi_{\mathrm{c}}(A^l_{\tau_{0:T}} \mid o,\ell,A^w)}{\pi_{\mathrm{ref}}(A^l_{\tau_{0:T}} \mid o,\ell,A^w)}\right]\right)^2\Bigg],\end{aligned} \quad (8)$$

where $A_{\tau_{0:T}}$ denotes the sampling trajectory ending at $A_{\tau_0} = A$ and $\mathbb{E}_q$ averages over trajectories drawn from the forward process. Conditioned on A^l the model must prefer the expert chunk, which is the repair behavior; conditioned on A^w it must still prefer A^w, a fixed-point constraint against over-correction.

Following prior work that adapts DPO to flow matching and diffusion (Liu et al., 2025; Wallace et al., 2024), Eq. 8 can be simplified via Jensen's inequality into a tractable form in velocity space, giving the upper bound we minimize, with $\tau \sim \mathcal{U}[0,1]$, $\epsilon^w, \epsilon^l \sim \mathcal{N}(0, I)$, interpolants A^w_τ, A^l_τ as in

Eq. 2, and v_{ref} the velocity field of π_{ref}:

$$\widehat{\mathcal{L}}_{\text{SI}}(\theta) = \mathbb{E}_{\substack{(A^w, A^l, o, \ell) \sim \mathcal{D} \\ \tau, \epsilon^w, \epsilon^l}} \left[\left(\frac{1}{2} + \frac{\beta}{2}(1-\tau)^2 \begin{bmatrix} \|\epsilon^w - A^w - v_\theta(A^w_\tau, \tau, o, \ell, A^l)\|^2 \\ - \|\epsilon^w - A^w - v_{\text{ref}}(A^w_\tau, \tau, o, \ell, A^l)\|^2 \\ - \|\epsilon^l - A^l - v_\theta(A^l_\tau, \tau, o, \ell, A^l)\|^2 \\ + \|\epsilon^l - A^l - v_{\text{ref}}(A^l_\tau, \tau, o, \ell, A^l)\|^2 \end{bmatrix} \right)^2 + \left(\frac{1}{2} + \frac{\beta}{2}(1-\tau)^2 \begin{bmatrix} \|\epsilon^w - A^w - v_\theta(A^w_\tau, \tau, o, \ell, A^w)\|^2 \\ - \|\epsilon^w - A^w - v_{\text{ref}}(A^w_\tau, \tau, o, \ell, A^w)\|^2 \\ - \|\epsilon^l - A^l - v_\theta(A^l_\tau, \tau, o, \ell, A^w)\|^2 \\ + \|\epsilon^l - A^l - v_{\text{ref}}(A^l_\tau, \tau, o, \ell, A^w)\|^2 \end{bmatrix} \right)^2 \right], \tag{9}$$

where a single flow time τ and a single pair of noise draws ϵ^w, ϵ^l are shared across all eight squared errors, correlating the terms and lowering gradient variance (see details in Appendix A.3).[1]

4.4 Data Construction

Given an expert demonstration dataset $\mathcal{D}_{\text{expert}}$ (usually collected through human teleoperation) for a downstream task domain, we split it evenly into two subsets, $\mathcal{D}_1$ and $\mathcal{D}_2$. To construct the training dataset $\mathcal{D}_{\text{correct}} = \{(o_i, \ell_i, A^l_i, A^w_i)\}$, we need a pair of a bad action and a good action for each observation and language instruction input. We use the expert actions in the demonstration dataset as the good actions A^w. To obtain the bad actions A^l, we first build a weak VLA model by fine-tuning the pretrained VLA model on $\mathcal{D}_1$. We then use the weak VLA model to sample action chunks on $\mathcal{D}_2$. Since the weak VLA model has not been trained on $\mathcal{D}_2$, we assume that the sampled actions are worse than the corresponding expert actions A^w in $\mathcal{D}_2$ and therefore use them as the bad actions A^l. This yields the action-pair dataset $\mathcal{D}'_2$. We use 75% of $\mathcal{D}'_2$ for Stage 1 cold-start training and the remaining 25% for Stage 2 correction preference training.

5 Experiments And Results

5.1 Experimental Setup

We evaluate COCOA with three flow-based VLA backbones: OpenPi $\pi_{0.5}$ (Driess et al., 2025), FLOWER VLA (Reuss et al., 2025), and Xiaomi-Robotics-0 (Cai et al., 2026b). For each backbone, we use the VLA model fine-tuned on the corresponding benchmark as the base policy, which generates candidate action chunks during evaluation. The corrective policy is initialized from the original pretrained VLA checkpoint (before fine-tuning on the downstream benchmark) and is trained using the COCOA framework. At inference time, the base policy first proposes an action chunk. At each *correction step*, the corrective policy conditions on the proposed action chunk and generates a corrected action chunk for execution. To balance correction frequency and computational overhead, we apply correction to every Nth action chunk proposed by the base policy. We set $N = 5$ in all experiments (that is, correction is applied to the 5th, 10th, ... action chunks).

We train and evaluate COCOA with all three VLA backbones on two simulation benchmarks, CALVIN (Mees et al., 2022) and SimplerEnv. We additionally evaluate $\pi_{0.5}$ with COCOA on three real-world manipulation tasks using a Franka Research 3 robot. We use four 80GB H100 GPUs for training and four 24GB NVIDIA RTX A5000 for evaluation.

[1] The full SRPO objective mixes this loss with a joint generative-policy term as $(1-\alpha)\mathcal{L}_{\text{joint}} + \alpha\widehat{\mathcal{L}}_{\text{SI}}$; since only the correction model is trained, we use $\alpha = 1$.

Method	No. Instructions Completed in a Row (%) ↑					Avg. Len. ↑
	1	2	3	4	5	
Diff-P-CNN	63.5%	35.3%	19.4%	10.7%	6.4%	1.35±0.05
MDT	63.1%	42.9%	24.7%	15.1%	9.1%	1.55
RoboFlamingo	82.4%	61.9%	46.6%	33.1%	23.5%	2.47
SuSIE	87.0%	69.0%	49.0%	38.0%	26.0%	2.69
DeerVLA	84.8%	72.3%	54.9%	44.6%	33.5%	2.90
GR-1	85.4%	71.2%	59.6%	49.7%	40.1%	3.06
OpenVLA	91.3%	77.8%	62.0%	52.1%	43.5%	3.27
3DDA	93.8%	80.3%	66.2%	53.3%	41.2%	3.35
RoboDual	94.4%	82.7%	72.1%	62.4%	54.4%	3.66
OpenPi $\pi_{0.5}$	91.4%	82.7%	77.4%	72.6%	67.7%	3.92
OpenPi $\pi_{0.5}$ + COCOA	92.5% ↑ 1.1 pp	85.2% ↑ 2.5 pp	79.9% ↑ 2.5 pp	75.6% ↑ 3.0 pp	69.5% ↑ 1.8 pp	4.03 ↑ 2.8%
MoDE	96.2%	88.9%	81.1%	71.8%	63.5%	4.01
Seer	96.3%	91.6%	86.1%	80.3%	74.0%	4.28
VPP	95.7%	91.2%	86.3%	81.0%	75.0%	4.29
FLOWER VLA	99.4%	95.8%	90.7%	84.9%	77.8%	4.53
FLOWER VLA + COCOA	99.6% ↑ 0.2 pp	97.2% ↑ 1.4 pp	93.9% ↑ 3.2 pp	88.6% ↑ 3.7 pp	81.9% ↑ 4.1 pp	4.61 ↑ 1.8%
Xiaomi-Robotics-0	**100.0%**	**98.7%**	96.1%	92.6%	88.1%	4.75
Xiaomi-Robotics-0 + COCOA	**100.0%** 0.0 pp	98.4% ↓ 0.3 pp	**96.3%** ↑ 0.2 pp	**92.9%** ↑ 0.3 pp	**88.5%** ↑ 0.4 pp	**4.76** ↑ 0.2%

Table 1: CALVIN ABC→D long-horizon evaluation. Bold and underlined values denote the best and second-best results in each column. Annotations in the COCOA rows report changes from the corresponding baselines: percentage points (pp) for success rates and relative percentages for average sequence length ↑/↓ indicate gains/drops.

5.2 Main Results

5.2.1 CALVIN Results

CALVIN is a language-conditioned benchmark for long-horizon robot manipulation. Following the standard ABC→D evaluation setting, policies are trained on environments A, B, and C and evaluated on 1,000 sequences of five consecutive language instructions in the unseen environment D. We report the percentage of evaluation sequences that successfully complete at least k instructions, for $k \in \{1, \ldots, 5\}$, as well as the average number of completed instructions per sequence.

As Table 1 shows, COCOA consistently improves the aggregate long-horizon performance of all three evaluated VLA backbones. For $\pi_{0.5}$, COCOA increases the average completed sequence length from 3.92 to 4.03 and the five-instruction success rate from 67.7% to 69.5% (+1.8 pp). The gains are larger for FLOWER VLA, where COCOA improves the average sequence length from 4.53 to 4.61 and the five-instruction success rate from 77.8% to 81.9% (+4.1 pp). Xiaomi-Robotics-0 already achieves strong performance with an average sequence length of $4.75/5$, leaving limited headroom for further improvement; nevertheless, COCOA slightly increases the average sequence length to 4.76 and the five-instruction success rate from 88.1% to 88.5%. Overall, these results show that COCOA improves long horizon manipulation performance across different VLA base models.

5.2.2 SimplerEnv Results

SimplerEnv is a real-to-sim benchmark in which robot policies trained on real-world trajectories are evaluated in simulation (Li et al., 2024b). It supports two robot platforms: Google Robot and WidowX. The Google Robot benchmark provides two evaluation settings: *Visual Matching*, which reduces the visual gap between simulation and real-world observations, and *Variant Aggregation*, which evaluates policies under variations in background, lighting, distractors, and table texture. Following the standard protocol, we train each VLA policy with COCOA on the RT-1 Fractal dataset and report task success rates and their average across four task categories.

As shown in Table 2, COCOA improves the overall success rate of all three VLA backbones under both evaluation settings. COCOA yields clear gains for both FLOWER VLA and Xiaomi-Robotics-0: Under Visual Matching, COCOA improves FLOWER VLA from 61.3% to 64.0% (↑ 2.7 pp) and Xiaomi-Robotics-0 from 85.5% to 90.4% (↑ 4.9 pp). Under Variant Aggregation, COCOA improves FLOWER VLA from 52.9% to 54.2% (↑ 1.3 pp) and Xiaomi-Robotics-0 from 74.7% to 77.5% (↑ 2.8 pp). The task-level results reveal that the gains are concentrated on the more challenging manipulation tasks. The largest gains occur on the challenging Drawer Apple task, where Xiaomi-Robotics-0 improves by 14.8 pp under Visual Matching and 11.6 pp under Variant Aggregation, and a similar pattern appears for the other backbones FLOWER VLA and $\pi_{0.5}$. The Drawer Apple task (as shown in Fig. 1) has a substantially longer execution horizon: Xiaomi-Robotics-0's successful episodes average 148.1–150.3 action steps, compared with 16.9–47.1 steps for the other task categories (Appendix Table 4). These results indicate that action-conditioned correction is especially

Visual Matching	Pick Coke Can	Move Near	Open/Close Drawer	Drawer Apple	Overall
Octo-Base	17.0%	4.2%	22.7%	0%	11.0%
OpenVLA	16.3%	46.2%	35.6%	0%	24.5%
RT-1	85.7%	44.2%	73.0%	6.5%	52.4%
RT-1-X	56.7%	31.7%	59.7%	40.7%	47.2%
RT-2-X	78.7%	77.9%	25.0%	7.4%	47.3%
Magma	75.0%	53.0%	58.9%	8.3%	48.8%
RoboVLMs	77.3%	61.7%	43.5%	24.1%	51.7%
SpatialVLA	86.0%	77.9%	57.4%	0%	55.3%
FlowerVLA	84.3%	82.9%	66.7%	11.1%	61.3%
FlowerVLA + COCOA	84.3% 0.0 pp	83.1% ↑ 0.2 pp	69.2% ↑ 2.5 pp	19.4% ↑ 8.3 pp	64.0% ↑ 2.7 pp
$\pi_0.5$	88.3%	77.9%	56.9%	47.2%	67.6%
$\pi_0.5$ + COCOA	86.7% ↓ 1.6 pp	76.7% ↓ 1.2 pp	61.6% ↑ 4.7 pp	52.8% ↑ 5.6 pp	69.5% ↑ 1.9 pp
ThinkAct	92.0%	72.4%	50.0%	-	-
MolmoAct	77.7%	77.1%	60.0%	-	-
EO-1	98.0%	83.8%	71.3%	52.8%	76.5%
Xiaomi-Robotics-0	98.7%	88.8%	79.6%	75.0%	85.5%
Xiaomi-Robotics-0 + COCOA	**99.0%** ↑ 0.3 pp	**91.7%** ↑ 2.9 pp	**81.0%** ↑ 1.4 pp	**89.8%** ↑ 14.8 pp	**90.4%** ↑ 4.9 pp
Variant Aggregation	**Pick Coke Can**	**Move Near**	**Open/Close Drawer**	**Drawer Apple**	**Overall**
Octo-Base	0.6%	3.1%	1.1%	0%	1.2%
OpenVLA	54.5%	47.7%	17.7%	0.0%	30.0%
RT-1	89.8%	50.0%	32.3%	2.6%	43.7%
RT-1-X	49.0%	32.3%	29.4%	10.1%	30.2%
RT-2-X	82.3%	79.2%	35.3%	20.6%	54.4%
Magma	68.6%	78.5%	59.0%	24.0%	57.5%
RoboVLMs	75.6%	60.0%	10.6%	0%	36.6%
$\pi_0.5$	69.0%	62.7%	42.3%	34.4%	52.1%
$\pi_0.5$ + COCOA	69.9% ↑ 0.9 pp	61.1% ↓ 1.6 pp	45.8% ↑ 3.5 pp	33.3% ↓ 1.1 pp	52.5% ↑ 0.4 pp
SpatialVLA	88.0%	72.7%	41.8%	6.3%	52.2%
FlowerVLA	70.4%	68.0%	56.6%	16.4%	52.9%
FlowerVLA + COCOA	70.2% ↓ 0.2 pp	68.7% ↑ 0.7 pp	57.2% ↑ 0.6 pp	20.5% ↑ 4.1 pp	54.2% ↑ 1.3 pp
ThinkAct	84.0%	63.8%	47.6%	-	-
MolmoAct	76.1%	61.3%	**78.8%**	-	-
EO-1	**91.6%**	**81.7%**	55.0%	23.8%	63.0%
Xiaomi-Robotics-0	88.2%	76.8%	67.2%	66.7%	74.7%
Xiaomi-Robotics-0 + COCOA	89.1% ↑ 0.9 pp	77.2% ↑ 0.4 pp	65.3% ↓ 1.9 pp	**78.3%** ↑ 11.6 pp	**77.5%** ↑ 2.8 pp

Table 2: Evaluations results of SimplerEnv Google Robot. Annotations in the COCOA rows report changes in success rate in percentage points (pp) compared with the corresponding baselines; ↑/↓ indicate gains/drops

beneficial on tasks requiring longer or more precise manipulation sequences, where an intermediate action error can more easily derail subsequent execution.

5.3 Cost and Latency Analysis

COCOA introduces additional inference overhead because each correction step requires a forward pass through a separate corrective VLA policy. Since correction is applied only once every five base-policy action chunks, this additional inference is invoked sparsely, and the corrected chunk replaces rather than adds to the corresponding base-policy chunk. We report the number of corrected chunks, inference time, and estimated completion time on successful episodes in Appendix Tables 4–7. On SimplerEnv, for Xiaomi-Robotics-0, successful episodes contain on average 0.55–7.12 corrected chunks across the four tasks under Visual Matching and 0.76–7.16 under Variant Aggregation. Across these tasks, corrective inference increases inference time by 0.18–1.51 s under Visual Matching and 0.19–1.50 s under Variant Aggregation, resulting in an increase of 0.43–1.73 s (1.3–12.6%) and 0.29–1.29 s (2.2–6.8%) in estimated completion time, respectively.

On CALVIN, successful five-instruction sequences contain on average 13.75, 5.36, and 5.28 corrected chunks for OpenPi $\pi_{0.5}$, FLOWER VLA, and Xiaomi-Robotics-0, respectively. The corresponding inference time increases from 8.833 s to 10.320 s for OpenPi $\pi_{0.5}$, from 6.315 s to 7.391 s for FLOWER VLA, and from 7.470 s to 8.492 s for Xiaomi-Robotics-0. Because corrected chunks replace the original base-policy chunks, the robot motion duration remains nearly unchanged, and the resulting increase in estimated completion time is smaller: 21.970 s to 23.317 s (+6.1%), 17.898 s to 18.941 s (+5.8%), and 19.010 s to 20.008 s (+5.2%), respectively. Overall, COCOA improves policy performance with a moderate latency overhead primarily due to the additional corrective-model inference.

5.4 Real Robot Experiment

Our real-robot setup consists of a Franka Research 3 manipulator equipped with two calibrated Femto Mega cameras: a wrist-mounted camera and a fixed third-person camera overlooking the workspace. The policy receives synchronized observations from both cameras together and predicts



Figure 3: COCOA deployment on 3 tasks in the real world using an FR3 manipulator robot.

Table 3: **Real-robot Evaluation.** Success rate (%, ↑) on three tasks. Avg. is the unweighted mean across tasks

Method	Pick-place↑	Clear table↑	Stack box↑	Avg.↑
$\pi_{0.5}$	90.0%	55.0%	70.0%	71.7%
$\pi_{0.5}$ + COCOA	95.0%	65.0%	85.0%	81.7%

joint-position actions. We evaluate on three manipulation tasks, as shown in Fig. 3. For each task, we collect 50 teleoperated demonstrations using a Meta Quest 2, with variations in object positions and camera viewpoints. We fine-tune $\pi_{0.5}$ on these demonstrations to obtain the base VLA policy and train a separate corrective policy using COCOA. At test time, both policies are executed closed-loop under the same evaluation protocol. We perform 20 trials per task with varied initial object configurations and report task success rate. Videos of both successful and failed trials are provided in the supplementary material. Table 3 shows that COCOA improves real-robot performance across all three tasks.

6 Limitation and Future Work

Our current implementation has several limitations. First, COCOA applies correction at a fixed frequency, although the action chunk generated by the base VLA policy may not require correction; future work could learn an adaptive monitor that estimates action quality or uncertainty and invokes correction only when necessary. Second, each correction requires an additional inference, introducing computational overhead that can be important for latency-sensitive applications; lightweight corrective policies or asynchronous correction could further improve efficiency. Finally, our training data assumes that actions sampled from a weaker VLA are lower quality than the corresponding demonstration actions. While this enables automatic construction of correction pairs without manual annotations, these preference labels are approximate. Future work could incorporate execution outcomes from simulated rollouts, task rewards, or human feedback to construct more reliable correction preferences.

7 Conclusion

We introduced COCOA, a training framework that learns a separate corrective VLA policy to revise action chunks proposed by a base VLA before execution. COCOA includes a two-stage training that combines supervised flow-matching training with correction preference optimization to learn effective action correction. We apply and evaluate COCOA on three flow-based VLA backbones spanning different model scales: $\pi_{0.5}$, FLOWER VLA, and Xiaomi-Robotics-0. Experimental results show that COCOA consistently improves the performance of different base VLA policies, including strong policies that are already highly competitive, and achieves state-of-the-art performance on both CALVIN and SimplerEnv. These results demonstrate the effectiveness and generality of COCOA and establish action-conditioned correction as an effective approach for improving VLA policies.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks:

- Polishing author-written text for clarity, readability, and grammar.
- Searching for and identifying relevant literature.
- Assisting with the implementation and debugging of code for our training and evaluation methods.
- Providing feedback on research experiments. Specifically, we used generative AI tools to help analyze training loss and training behavior for debugging and hyperparameter tuning.

We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, assist with translation, clean or reformat datasets, or support qualitative or thematic data analysis.

We reviewed all AI-assisted work. LLM-generated or LLM-assisted code was reviewed, tested, and verified for correctness by the authors. All AI-assisted edits to the paper were reviewed by the authors, and related literature identified with the assistance of generative AI was verified by the authors.

We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the source code in the anonymous GitHub repository: `https://anonymous.4open.science/r/cocoa_iclr_2027-3214`

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

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

Kevin Black, Noah Brown, 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.

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026a.

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, Feng Qiu, Heng Qu, Yifei Su, Qiao Sun, Dong Wang, Donghao Wang, Yunhong Wang, Rujie Wu, Diyun Xiang, Yu Yang, Hangjun Ye, Yuan Zhang, and Quanyun Zhou. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026b.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Jiayi Chen, Wenxuan Song, Shuai Chen, Jingbo Wang, Zhijun Li, and Haoang Li. DFM-VLA: Iterative action refinement for robot manipulation via discrete flow matching. *arXiv preprint arXiv:2603.26320*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Eugene Choi, Arash Ahmadian, Matthieu Geist, Olivier Pietquin, and Mohammad Gheshlaghi Azar. Self-improving robust preference optimization. *arXiv preprint*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration: Adapting dmps for contact-rich manipulation. *IEEE Robotics and Automation Letters*, 7(2):4488–4495, 2022.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

Chenyang Gu, Jiaming Liu, Hao Chen, Runzhong Huang, Qingpo Wuwu, Zhuoyang Liu, Xiaoqi Li, Ying Li, Renrui Zhang, Peng Jia, et al. Manualvla: A unified vla model for chain-of-thought manual generation and robotic manipulation. *arXiv preprint arXiv:2512.02013*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi_0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

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

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. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf.

Chenxuan Li, Jiaming Liu, Guanqun Wang, Xiaoqi Li, Sixiang Chen, Liang Heng, Chuyan Xiong, Jiaxin Ge, Renrui Zhang, Kaichen Zhou, and Shanghang Zhang. A self-correcting vision-language-action model for fast and slow system manipulation. *arXiv preprint arXiv:2405.17418*, 2024a.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024b.

Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Chenyv Liu, Wentao Tan, Lei Zhu, Fengling Li, Jingjing Li, Guoli Yang, and Heng Tao Shen. Self-correcting VLA: Online action refinement via sparse world imagination. *arXiv preprint arXiv:2602.21633*, 2026.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, and Mingwu Zheng. Improving video generation with human feedback. *arXiv preprint*, 2025.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025.

Chenyang Ma, Kai Lu, Guangyu Yang, Jiuming Liu, Shitong Xu, Bill Byrne, Ioannis Havoutis, Niki Trigoni, and Andrew Markham. CycleVLA: Proactive self-correcting vision-language-action models via subtask backtracking and minimum bayes risk decoding. *arXiv preprint arXiv:2601.02295*, 2026.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Yi Pan, Miao Pan, Qi Lu, Jiaming Huang, Man Zhang, Siteng Huang, Xin Li, Jie Zhang, Yongliang Shen, Xuhong Zhang, and Wenqi Zhang. VLA-corrector: Lightweight detect-and-correct inference for adaptive action horizon. *arXiv preprint arXiv:2607.01804*, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies. *Conference on Robot Learning*, 2025.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for VLA action chunks. *arXiv preprint arXiv:2509.23224*, 2025.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Jiarui Li, Qi Lv, Yiwen Tang, Li Kang, Heng Zhou, Xianqiang Gao, Yuhang Tang, Xiaofan Li, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, Dong Wang, and Xuelong Li. FM-Steer: Enhance generalist policies with value-guided cascaded denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13407–13418, 2026.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. *arXiv preprint arXiv:2506.19850*, 2025.

Wenke Xia, Yichu Yang, Hongtao Wu, Xiao Ma, Tao Kong, and Di Hu. Human-assisted robotic policy refinement via action preference optimization. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=e5jGTEiJMT.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*, 2025.

Wentao Zhang, Aolan Sun, Wentao Mo, Xiaoyang Qu, Yuxin Zheng, and Jianzong Wang. From knowing to doing precisely: A general self-correction and termination framework for VLA models. *arXiv preprint arXiv:2602.01811*, 2026.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1702–1713, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: A fine-grained world model for robot manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9834–9844, October 2025.

Fangqi Zhu, YAN Zhengyang, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. WMPO: World model-based policy optimization for vision-language-action models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qE2FyvRvuF.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023.

A Derivation of the Self-Improvement Loss for Flow Models

We omit (o, ℓ) for simplicity. We denote the input and output chunks by C and A, respectively, corresponding to A and A' in Section 4.3. Let $A_{\tau_{0:T}}$ be a denoising trajectory with $0 = \tau_0 < \cdots < \tau_T = 1$ and $A_{\tau_0} = A$. We use the Gaussian forward process q and reverse-process parameterization of Wallace et al. (2024), with $q(A_\tau \mid A) = \mathcal{N}(a_\tau A, \sigma_\tau^2 I)$ and a shared noise prior and transition covariances for π_{c} and π_{ref}.

A.1 Preference Optimization over Denoising Trajectories

By the chain rule for KL divergence,

$$\begin{aligned} &D_{\mathrm{KL}}\left(\pi_{\mathrm{c}}(A_{\tau_{0:T}} \mid C) \,\|\, \pi_{\mathrm{ref}}(A_{\tau_{0:T}} \mid C)\right) \\ &\quad = D_{\mathrm{KL}}\left(\pi_{\mathrm{c}}(A \mid C) \,\|\, \pi_{\mathrm{ref}}(A \mid C)\right) \\ &\qquad + \mathbb{E}_{A \sim \pi_{\mathrm{c}}(\cdot \mid C)} D_{\mathrm{KL}}\left(\pi_{\mathrm{c}}(A_{\tau_{1:T}} \mid A, C) \,\|\, \pi_{\mathrm{ref}}(A_{\tau_{1:T}} \mid A, C)\right). \end{aligned} \tag{10}$$

The last term is nonnegative. Substituting this bound into Eq. 5 gives Eq. 6.

A.2 Self-Improvement Loss from Pairwise Preferences

For a fixed input C and trajectory distribution π, the objective in Eq. 6 can be written as

$$\begin{aligned} &-\mathbb{E}_\pi[p(A \succ C)] + \beta D_{\mathrm{KL}}\big(\pi(\cdot \mid C) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid C)\big) \\ &\quad = \beta D_{\mathrm{KL}}\big(\pi(\cdot \mid C) \,\|\, \pi^\dagger(\cdot \mid C)\big) - \beta \log Z(C), \end{aligned}$$

where

$$\pi^\dagger(A_{\tau_{0:T}} \mid C) = \pi_{\mathrm{ref}}(A_{\tau_{0:T}} \mid C) \exp(p(A \succ C)/\beta)\, / Z(C), \tag{11}$$

and $Z(C)$ is the normalization factor. Since KL divergence is nonnegative, $\pi^\dagger$ minimizes this objective. Taking logarithms of Eq. 11 gives

$$\begin{aligned} p(A \succ C) &= \beta \log \frac{\pi^\dagger(A_{\tau_{0:T}} \mid C)}{\pi_{\mathrm{ref}}(A_{\tau_{0:T}} \mid C)} + \beta \log Z(C), \\ \frac{1}{2} &= \beta \log \frac{\pi^\dagger(C_{\tau_{0:T}} \mid C)}{\pi_{\mathrm{ref}}(C_{\tau_{0:T}} \mid C)} + \beta \log Z(C), \end{aligned}$$

where the second equality uses $A = C$ and $p(C \succ C) = \frac{1}{2}$. Subtracting the second equality from the first eliminates $Z(C)$ and yields

$$p(A \succ C) = \frac{1}{2} + \beta \left[\log \frac{\pi^\dagger(A_{\tau_{0:T}} \mid C)}{\pi_{\mathrm{ref}}(A_{\tau_{0:T}} \mid C)} - \log \frac{\pi^\dagger(C_{\tau_{0:T}} \mid C)}{\pi_{\mathrm{ref}}(C_{\tau_{0:T}} \mid C)}\right]. \tag{12}$$

We replace $\pi^\dagger$ by π_{c} and fit Eq. 12 by least squares using binary preference labels (Choi et al., 2025). Since sampling conditional denoising trajectories is intractable, we approximate them with $q(A_{\tau_{1:T}} \mid A)$.

For a labelled pair $A^w \succ A^l$, we evaluate the residual for each input chunk. Substituting $(C, A) = (A^l, A^w)$ with target label 1 gives

$$\delta(A^l, A^w) = \frac{1}{2} - \beta\, \mathbb{E}_q \left[\log \frac{\pi_{\mathrm{c}}(A^w_{\tau_{0:T}} \mid A^l)}{\pi_{\mathrm{ref}}(A^w_{\tau_{0:T}} \mid A^l)} - \log \frac{\pi_{\mathrm{c}}(A^l_{\tau_{0:T}} \mid A^l)}{\pi_{\mathrm{ref}}(A^l_{\tau_{0:T}} \mid A^l)}\right], \tag{13}$$

Substituting $(C, A) = (A^w, A^l)$ with target label 0 gives

$$\delta(A^w, A^l) = -\frac{1}{2} - \beta\, \mathbb{E}_q \left[\log \frac{\pi_{\mathrm{c}}(A^l_{\tau_{0:T}} \mid A^w)}{\pi_{\mathrm{ref}}(A^l_{\tau_{0:T}} \mid A^w)} - \log \frac{\pi_{\mathrm{c}}(A^w_{\tau_{0:T}} \mid A^w)}{\pi_{\mathrm{ref}}(A^w_{\tau_{0:T}} \mid A^w)}\right]. \tag{14}$$

Squaring and summing gives the self-improvement loss in Eq. 8:

$$\mathcal{L}_{\mathrm{SI}}(\pi_{\mathrm{c}}) = \mathbb{E}_{\mathcal{D}}\left[\delta(A^l, A^w)^2 + \delta(A^w, A^l)^2\right].$$

A.3 From Noise Prediction to Velocity Prediction

Fix a preference pair (A^w, A^l). Let $\mathcal{L}_C(\theta)$ denote the squared loss for input C in Eq. 8, where θ parameterizes π_c:

$$\mathcal{L}_C(\theta) = \begin{cases} \delta(A^l, A^w)^2, & C = A^l, \\ \delta(A^w, A^l)^2, & C = A^w. \end{cases}$$

For $b \in \{w, l\}$, define the log-ratio at step i:

$$r_i^b = \log \frac{\pi_c(A^b_{\tau_{i-1}} \mid A^b_{\tau_i}, C)}{\pi_{\text{ref}}(A^b_{\tau_{i-1}} \mid A^b_{\tau_i}, C)}. \tag{15}$$

The shared prior cancels, so

$$\log \frac{\pi_c(A^b_{\tau_{0:T}} \mid C)}{\pi_{\text{ref}}(A^b_{\tau_{0:T}} \mid C)} = \sum_{i=1}^{T} r_i^b = T\mathbb{E}_i[r_i^b], \qquad i \sim \mathcal{U}\{1, \dots, T\}.$$

Set $\tau = \tau_i$ and draw $A^w_\tau \sim q(\cdot \mid A^w)$ and $A^l_\tau \sim q(\cdot \mid A^l)$. The inner expectation $\mathbb{E}_q$ below is over the forward-process posteriors

$$A^w_{\tau_{i-1}} \sim q(\cdot \mid A^w_{\tau_i}, A^w), \qquad A^l_{\tau_{i-1}} \sim q(\cdot \mid A^l_{\tau_i}, A^l).$$

By Jensen's inequality,

$$\begin{aligned} \mathcal{L}_C(\theta) &= \left(\tfrac{1}{2} - \beta T\, \mathbb{E}_{\tau, A^w_\tau, A^l_\tau} \mathbb{E}_q \left[r_i^w - r_i^l\right]\right)^2 \\ &\le \mathbb{E}_{\tau, A^w_\tau, A^l_\tau} \left(\tfrac{1}{2} - \beta T\, \mathbb{E}_q \left[r_i^w - r_i^l\right]\right)^2. \end{aligned} \tag{16}$$

Write $q^b = q(A^b_{\tau_{i-1}} \mid A^b_{\tau_i}, A^b)$ and $\pi^b = \pi(A^b_{\tau_{i-1}} \mid A^b_{\tau_i}, C)$ for $\pi \in \{\pi_c, \pi_{\text{ref}}\}$. Using $\mathbb{E}_q[r_i^b] = D_{\text{KL}}(q^b \| \pi^b_{\text{ref}}) - D_{\text{KL}}(q^b \| \pi^b_c)$ gives

$$\mathcal{L}_C(\theta) \;\le\; \mathbb{E}_{\tau, A^w_\tau, A^l_\tau} \left(\tfrac{1}{2} + \beta T \left[D_{\text{KL}}(q^w \| \pi^w_c) - D_{\text{KL}}(q^w \| \pi^w_{\text{ref}}) - D_{\text{KL}}(q^l \| \pi^l_c) + D_{\text{KL}}(q^l \| \pi^l_{\text{ref}})\right]\right)^2, \tag{17}$$

At $i = 1$, the corresponding term is the reconstruction log-likelihood ratio. For either chunk $A = A^b$, let $\mu_q, \mu_\theta, \mu_{\text{ref}}$ denote the posterior and reverse-process means. The two reverse processes share covariance $s_\tau^2 I$. Set $A_\tau = a_\tau A + \sigma_\tau \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, and write $\widehat{\epsilon}_m$ for $\widehat{\epsilon}_m(A_\tau, \tau, C)$. The noise-prediction parameterization gives

$$\mu_q - \mu_m = k_\tau(\widehat{\epsilon}_m - \epsilon), \qquad m \in \{\theta, \text{ref}\},$$

where k_τ depends only on the noise schedule. The covariance and entropy terms cancel in the KL difference, giving

$$\begin{aligned} &D_{\text{KL}}(q^b \| \pi^b_c) - D_{\text{KL}}(q^b \| \pi^b_{\text{ref}}) \\ &= \frac{1}{2s_\tau^2} \left(\|\mu_q - \mu_\theta\|^2 - \|\mu_q - \mu_{\text{ref}}\|^2\right) \\ &= \frac{\omega_\tau}{2} \left(\|\epsilon - \widehat{\epsilon}_\theta\|^2 - \|\epsilon - \widehat{\epsilon}_{\text{ref}}\|^2\right), \qquad \omega_\tau = \frac{k_\tau^2}{s_\tau^2}. \end{aligned}$$

Let $\widehat{\epsilon}^b_m = \widehat{\epsilon}_m(A^b_\tau, \tau, C)$, with $A^b_\tau = a_\tau A^b + \sigma_\tau \epsilon^b$ and $\epsilon^b \sim \mathcal{N}(0, I)$. Substituting into Eq. 17 gives

$$\mathcal{L}_C(\theta) \;\le\; \mathbb{E}_{\tau, \epsilon^w, \epsilon^l} \left(\tfrac{1}{2} + \tfrac{\beta T \omega_\tau}{2} \left[\|\epsilon^w - \widehat{\epsilon}^w_\theta\|^2 - \|\epsilon^w - \widehat{\epsilon}^w_{\text{ref}}\|^2 - \|\epsilon^l - \widehat{\epsilon}^l_\theta\|^2 + \|\epsilon^l - \widehat{\epsilon}^l_{\text{ref}}\|^2\right]\right)^2, \tag{18}$$

For training, we use uniform timestep weights, setting $\omega_\tau = 1$ and absorbing T into β.

For rectified flow, sample $\tau \sim \mathcal{U}[0, 1]$ and use the interpolation $A_\tau = (1 - \tau)A + \tau\epsilon$ from Eq. 2. The noise and velocity errors satisfy the following identity (Liu et al., 2025).

Lemma 1. *Let $A_\tau = (1 - \tau)A + \tau\epsilon$ and $u_A = \epsilon - A$. For a velocity prediction v, define the noise prediction $\widehat{\epsilon}[v] = A_\tau + (1 - \tau)v$. Then*

$$\left\|\epsilon - \widehat{\epsilon}[v]\right\|_2^2 = (1 - \tau)^2 \left\|u_A - v\right\|_2^2. \tag{19}$$

Proof. By substitution,

$$\epsilon - \widehat{\epsilon}[v] = \epsilon - A_\tau - (1-\tau)v = (1-\tau)(\epsilon - A - v) = (1-\tau)(u_A - v).$$

Taking squared norms gives Eq. 19. □

Applying Lemma 1 to the reweighted loss gives the velocity loss for input C:

$$\widehat{\mathcal{L}}_C(\theta) = \mathbb{E}_{\tau,\,\epsilon^w,\,\epsilon^l}\left(\frac{1}{2} + \frac{\beta}{2}(1-\tau)^2 \begin{bmatrix} \|\epsilon^w - A^w - v_\theta(A^w_\tau, \tau, C)\|^2 \\ - \|\epsilon^w - A^w - v_{\mathrm{ref}}(A^w_\tau, \tau, C)\|^2 \\ - \|\epsilon^l - A^l - v_\theta(A^l_\tau, \tau, C)\|^2 \\ + \|\epsilon^l - A^l - v_{\mathrm{ref}}(A^l_\tau, \tau, C)\|^2 \end{bmatrix}\right)^2. \tag{20}$$

Summing over the two input chunks gives Eq. 9:

$$\widehat{\mathcal{L}}_{\mathrm{SI}}(\theta) = \mathbb{E}_{\mathcal{D}}\left[\widehat{\mathcal{L}}_{A^l}(\theta) + \widehat{\mathcal{L}}_{A^w}(\theta)\right].$$

The same $\tau, \epsilon^w, \epsilon^l$ are used for both conditioning chunks and both models.

B SimplerEnv Action Execution and Timing

Tables 4 and 5 report action execution counts and estimated durations for successful Google Robot episodes in SimplerEnv, selected independently for each approach. Open/Close Drawer pools the two tasks, and Overall is weighted by the number of successful episodes within each setting. Means include episodes with zero corrections.

Estimated duration sums inference, nominal motion duration, and other controller processing. Motion duration is executed steps divided by the control rate; setup, warm-up, teardown, and artifact writing are excluded. These completion costs should be read alongside the success rates in Table 2, since the successful episodes and task mixture can differ between methods.

Visual Matching Task	Model	Avg. chunks	Avg. steps	Avg. correction chunks	Avg. correction steps
Pick Coke Can	$\pi_{0.5}$	6.42	24.08	—	—
	$\pi_{0.5}$ + COCOA	6.35	23.71	0.91	3.13
	FLOWER VLA	4.05	18.63	—	—
	FLOWER VLA + COCOA	4.13	18.90	0.52	1.39
	Xiaomi	4.56	16.88	—	—
	Xiaomi + COCOA	4.75	17.61	0.55	1.37
Move Near	$\pi_{0.5}$	6.73	25.49	—	—
	$\pi_{0.5}$ + COCOA	6.62	25.08	1.01	3.61
	FLOWER VLA	5.51	25.63	—	—
	FLOWER VLA + COCOA	5.63	26.52	0.85	2.89
	Xiaomi	5.31	19.83	—	—
	Xiaomi + COCOA	5.53	20.73	0.83	2.67
Open/Close Drawer	$\pi_{0.5}$	12.02	46.73	—	—
	$\pi_{0.5}$ + COCOA	12.29	47.69	2.07	8.04
	FLOWER VLA	8.38	40.06	—	—
	FLOWER VLA + COCOA	8.97	43.05	1.51	5.79
	Xiaomi	9.17	35.27	—	—
	Xiaomi + COCOA	10.13	38.98	1.57	5.95
Drawer Apple	$\pi_{0.5}$	38.73	149.76	—	—
	$\pi_{0.5}$ + COCOA	37.32	144.47	7.09	27.94
	FLOWER VLA	34.08	169.17	—	—
	FLOWER VLA + COCOA	33.70	165.46	6.35	25.81
	Xiaomi	38.05	150.29	—	—
	Xiaomi + COCOA	37.52	148.11	7.12	27.97
Visual Aggregation Task	**Model**	**Avg. chunks**	**Avg. steps**	**Avg. correction chunks**	**Avg. correction steps**
Pick Coke Can	$\pi_{0.5}$	8.21	31.21	—	—
	$\pi_{0.5}$ + COCOA	8.09	30.65	1.28	4.72
	FLOWER VLA	4.73	21.96	—	—
	FLOWER VLA + COCOA	4.81	22.31	0.68	2.11
	Xiaomi	5.25	19.53	—	—
	Xiaomi + COCOA	5.38	20.00	0.76	2.31
Move Near	$\pi_{0.5}$	8.00	30.55	—	—
	$\pi_{0.5}$ + COCOA	8.04	30.74	1.24	4.63
	FLOWER VLA	5.75	26.95	—	—
	FLOWER VLA + COCOA	5.79	27.17	0.89	3.15
	Xiaomi	6.30	23.88	—	—
	Xiaomi + COCOA	6.39	24.12	0.96	3.42
Open/Close Drawer	$\pi_{0.5}$	15.43	60.29	—	—
	$\pi_{0.5}$ + COCOA	14.59	56.95	2.49	9.72
	FLOWER VLA	8.78	41.90	—	—
	FLOWER VLA + COCOA	8.80	42.57	1.49	5.71
	Xiaomi	11.62	45.05	—	—
	Xiaomi + COCOA	12.15	47.06	2.04	7.75
Drawer Apple	$\pi_{0.5}$	39.03	151.40	—	—
	$\pi_{0.5}$ + COCOA	41.93	162.78	8.00	31.56
	FLOWER VLA	32.61	161.23	—	—
	FLOWER VLA + COCOA	33.02	163.71	6.48	25.60
	Xiaomi	37.82	149.32	—	—
	Xiaomi + COCOA	37.64	148.70	7.16	28.31

Table 4: Average action execution counts on **successful episodes only** of the Google Robot tasks in SimplerEnv. Each approach uses its own successful episodes. Xiaomi denotes Xiaomi-Robotics-0. Correction chunks and steps are included in the corresponding total counts; means include successful episodes with zero corrections. Open/Close Drawer pools the two tasks. For models without COCOA, — marks correction metrics as not applicable.

Visual Matching Task	Model	Avg. estimated total time (s)	Avg. inference time (s)	Avg. motion duration (s)
Pick Coke Can	$\pi_{0.5}$	10.384	2.330	8.026
	$\pi_{0.5}$ + COCOA	10.544	2.615	7.902
	FLOWER VLA	6.844	0.629	6.209
	FLOWER VLA + COCOA	7.315	0.797	6.484
	Xiaomi	6.589	0.929	5.628
	Xiaomi + COCOA	7.021	1.111	5.870
Move Near	$\pi_{0.5}$	10.973	2.448	8.496
	$\pi_{0.5}$ + COCOA	11.112	2.726	8.359
	FLOWER VLA	9.378	0.826	8.544
	FLOWER VLA + COCOA	9.947	0.998	8.891
	Xiaomi	7.757	1.110	6.611
	Xiaomi + COCOA	8.242	1.292	6.911
Open/Close Drawer	$\pi_{0.5}$	19.792	4.165	15.577
	$\pi_{0.5}$ + COCOA	21.054	5.106	15.896
	FLOWER VLA	14.607	1.244	13.352
	FLOWER VLA + COCOA	16.329	1.912	14.347
	Xiaomi	13.688	1.868	11.756
	Xiaomi + COCOA	15.414	2.352	12.992
Drawer Apple	$\pi_{0.5}$	63.421	13.342	49.922
	$\pi_{0.5}$ + COCOA	64.002	15.691	48.156
	FLOWER VLA	61.335	4.903	56.389
	FLOWER VLA + COCOA	62.147	5.129	56.91
	Xiaomi	57.823	7.462	50.098
	Xiaomi + COCOA	58.597	8.967	49.371
Visual Aggregation Task	**Model**	**Avg. estimated total time (s)**	**Avg. inference time (s)**	**Avg. motion duration (s)**
Pick Coke Can	$\pi_{0.5}$	13.868	3.416	10.404
	$\pi_{0.5}$ + COCOA	13.541	3.293	10.216
	FLOWER VLA	8.029	0.704	7.320
	FLOWER VLA + COCOA	8.317	0.729	7.417
	Xiaomi	7.584	1.038	6.510
	Xiaomi + COCOA	7.929	1.225	6.667
Move Near	$\pi_{0.5}$	13.528	3.300	10.182
	$\pi_{0.5}$ + COCOA	13.558	3.278	10.247
	FLOWER VLA	9.832	0.843	8.982
	FLOWER VLA + COCOA	9.891	0.839	9.030
	Xiaomi	9.265	1.262	7.959
	Xiaomi + COCOA	9.559	1.475	8.040
Open/Close Drawer	$\pi_{0.5}$	26.502	6.324	20.096
	$\pi_{0.5}$ + COCOA	25.224	6.178	18.982
	FLOWER VLA	15.263	1.287	13.966
	FLOWER VLA + COCOA	16.142	1.593	14.341
	Xiaomi	17.405	2.309	15.017
	Xiaomi + COCOA	18.590	2.824	15.686
Drawer Apple	$\pi_{0.5}$	66.266	15.600	50.467
	$\pi_{0.5}$ + COCOA	72.318	17.878	54.260
	FLOWER VLA	58.669	4.884	53.742
	FLOWER VLA + COCOA	60.813	5.961	54.812
	Xiaomi	57.471	7.441	49.772
	Xiaomi + COCOA	58.757	8.937	49.568

Table 5: Estimated duration on **successful episodes only** of the Google Robot tasks in SimplerEnv. Each approach uses its own successful episodes. Xiaomi denotes Xiaomi-Robotics-0. Values are mean seconds per successful episode. Estimated total equals inference plus motion plus other processing, before rounding. Inference includes base-model and correction-model requests, with preprocessing and communication; motion duration is executed steps divided by 3 Hz. Other processing excludes inference and simulator computation. Open/Close Drawer pools both tasks. Estimates assume sequential inference and motion with overhead measured in simulation. The $\pi_{0.5}$ baseline timing run encountered CPU contention on the shared host during part of the evaluation; its measured latencies may be elevated.

C CALVIN ACTION EXECUTION AND TIMING

Tables 6 and 7 report action execution counts and estimated durations on CALVIN ABC→D for each approach's own successful sequences, defined as completing all five instructions. Positions 1–5 use the same successful-sequence cohort and report the indicated instruction alone. Overall first sums execution across all five instructions within each successful sequence, then averages those sequence totals. Means include sequences with zero corrections.

Restricting this analysis to complete sequences holds the number of completed instructions fixed. Successful cohorts can still differ between approaches, so these completion costs should be interpreted alongside the success rates in Table 1. Estimated duration uses the same inference-plus-motion-plus-processing definition as for SimplerEnv, with motion duration computed at the CALVIN control rate.

Subtask position	Model	Avg. chunks	Avg. steps	Avg. correction chunks	Avg. correction steps
1	OpenPi $\pi_{0.5}$	15.30	74.54	—	—
	OpenPi $\pi_{0.5}$ + COCOA	15.49	75.37	2.69	12.99
	FLOWER VLA	6.86	64.10	—	—
	FLOWER VLA + COCOA	6.95	65.24	1.04	9.63
	Xiaomi-Robotics-0	7.10	66.37	—	—
	Xiaomi-Robotics-0 + COCOA	7.18	67.02	1.08	10.16
2	OpenPi $\pi_{0.5}$	17.43	85.20	—	—
	OpenPi $\pi_{0.5}$ + COCOA	17.22	84.12	3.05	14.88
	FLOWER VLA	7.74	72.93	—	—
	FLOWER VLA + COCOA	7.75	73.13	1.14	10.70
	Xiaomi-Robotics-0	7.81	73.63	—	—
	Xiaomi-Robotics-0 + COCOA	7.81	73.77	1.15	10.72
3	OpenPi $\pi_{0.5}$	15.99	77.98	—	—
	OpenPi $\pi_{0.5}$ + COCOA	15.80	77.01	2.74	13.38
	FLOWER VLA	7.43	69.81	—	—
	FLOWER VLA + COCOA	7.49	70.20	1.07	9.98
	Xiaomi-Robotics-0	7.23	67.88	—	—
	Xiaomi-Robotics-0 + COCOA	7.16	67.07	1.01	9.37
4	OpenPi $\pi_{0.5}$	15.35	74.74	—	—
	OpenPi $\pi_{0.5}$ + COCOA	14.96	72.76	2.59	12.59
	FLOWER VLA	7.39	69.22	—	—
	FLOWER VLA + COCOA	7.23	67.85	1.04	9.62
	Xiaomi-Robotics-0	7.12	66.96	—	—
	Xiaomi-Robotics-0 + COCOA	7.14	67.01	1.01	9.38
5	OpenPi $\pi_{0.5}$	15.63	76.16	—	—
	OpenPi $\pi_{0.5}$ + COCOA	15.43	75.10	2.68	13.01
	FLOWER VLA	7.55	70.81	—	—
	FLOWER VLA + COCOA	7.40	69.50	1.06	9.86
	Xiaomi-Robotics-0	7.32	68.83	—	—
	Xiaomi-Robotics-0 + COCOA	7.25	67.92	1.04	9.66
Overall	OpenPi $\pi_{0.5}$	79.69	388.62	—	—
	OpenPi $\pi_{0.5}$ + COCOA	78.91	384.37	13.75	66.85
	FLOWER VLA	36.96	346.87	—	—
	FLOWER VLA + COCOA	36.82	345.93	5.36	49.79
	Xiaomi-Robotics-0	36.58	343.68	—	—
	Xiaomi-Robotics-0 + COCOA	36.54	342.79	5.28	49.29

Table 6: Average action execution counts on **successful episodes only** on CALVIN ABC→D. Each approach uses its own sequences that complete all five instructions. Positions 1–5 report counts for the indicated instruction alone, averaged over these successful sequences. Overall sums counts across all five instructions within each successful sequence, then averages over the same sequences. A chunk is counted once at least one of its actions is executed. Correction chunks and steps are included in the corresponding total counts; means include successful sequences with zero corrections. For models without COCOA, — marks correction metrics as not applicable.

Subtask position	Model	Avg. estimated total time (s)	Avg. inference time (s)	Avg. motion duration (s)
1	OpenPi $\pi_{0.5}$	4.216	1.695	2.485
	OpenPi $\pi_{0.5}$ + COCOA	4.574	2.025	2.512
	FLOWER VLA	3.313	1.172	2.137
	FLOWER VLA + COCOA	3.580	1.401	2.175
	Xiaomi-Robotics-0	3.720	1.489	2.212
	Xiaomi-Robotics-0 + COCOA	3.969	1.713	2.234
2	OpenPi $\pi_{0.5}$	4.811	1.931	2.840
	OpenPi $\pi_{0.5}$ + COCOA	5.104	2.259	2.804
	FLOWER VLA	3.758	1.322	2.431
	FLOWER VLA + COCOA	4.002	1.560	2.438
	Xiaomi-Robotics-0	4.065	1.593	2.454
	Xiaomi-Robotics-0 + COCOA	4.295	1.817	2.459
3	OpenPi $\pi_{0.5}$	4.408	1.772	2.599
	OpenPi $\pi_{0.5}$ + COCOA	4.669	2.065	2.567
	FLOWER VLA	3.600	1.269	2.327
	FLOWER VLA + COCOA	3.844	1.500	2.340
	Xiaomi-Robotics-0	3.748	1.469	2.263
	Xiaomi-Robotics-0 + COCOA	3.902	1.649	2.236
4	OpenPi $\pi_{0.5}$	4.228	1.702	2.491
	OpenPi $\pi_{0.5}$ + COCOA	4.415	1.955	2.425
	FLOWER VLA	3.573	1.261	2.307
	FLOWER VLA + COCOA	3.715	1.450	2.262
	Xiaomi-Robotics-0	3.688	1.440	2.232
	Xiaomi-Robotics-0 + COCOA	3.891	1.641	2.234
5	OpenPi $\pi_{0.5}$	4.307	1.733	2.539
	OpenPi $\pi_{0.5}$ + COCOA	4.555	2.016	2.503
	FLOWER VLA	3.655	1.290	2.360
	FLOWER VLA + COCOA	3.800	1.480	2.317
	Xiaomi-Robotics-0	3.790	1.480	2.294
	Xiaomi-Robotics-0 + COCOA	3.952	1.671	2.264
Overall	OpenPi $\pi_{0.5}$	21.970	8.833	12.954
	OpenPi $\pi_{0.5}$ + COCOA	23.317	10.320	12.812
	FLOWER VLA	17.898	6.315	11.562
	FLOWER VLA + COCOA	18.941	7.391	11.531
	Xiaomi-Robotics-0	19.010	7.470	11.456
	Xiaomi-Robotics-0 + COCOA	20.008	8.492	11.426

Table 7: Estimated duration on **successful episodes only** on CALVIN ABC→D. Each approach uses its own sequences that complete all five instructions. Positions 1–5 report mean seconds for the indicated instruction alone among these successful sequences. Overall reports mean total seconds across all five instructions per successful sequence. Estimated total equals inference plus motion plus other processing. Inference includes base-model and correction-model requests, with preprocessing and communication; motion is executed steps divided by the configured control rate. Other processing excludes inference and simulator computation. Estimates assume sequential inference and motion, excluding setup, warm-up, teardown, and artifact writing.