RoboMonkey: Scaling Test-Time Sampling and Verification for Vision-Language-Action Models

Jacky Kwok[1] Christopher Agia[1,†] Rohan Sinha[1,†] Matt Foutter[1,†]
Shulu Li[2] Ion Stoica[2] Azalia Mirhoseini[1] Marco Pavone[1,3]

[1]Stanford University [2]UC Berkeley [3]NVIDIA Research

***Abstract*—Vision-Language-Action (VLA) models have demonstrated remarkable capabilities in visuomotor control, yet ensuring their robustness in unstructured real-world environments remains a persistent challenge. In this paper, we investigate test-time scaling through the lens of sampling and verification as means to enhance the robustness and generalization of VLAs. We first demonstrate that the relationship between action error and the number of generated samples follows an exponentiated power law across a range of VLAs, indicating the existence of inference-time scaling laws. Building on these insights, we introduce RoboMonkey, a test-time scaling framework for VLAs. At deployment, RoboMonkey samples a small set of actions from a VLA, applies Gaussian perturbation and majority voting to construct an action proposal distribution, and then uses a Vision Language Model (VLM)-based verifier to select the optimal action. We propose a synthetic data generation pipeline for training such VLM-based action verifiers, and demonstrate that scaling the synthetic dataset consistently improves verification and downstream accuracy. Through extensive simulated and hardware experiments, we show that pairing existing VLAs with RoboMonkey yields significant performance gains, achieving a 25% absolute improvement on out-of-distribution tasks and 8% on in-distribution tasks. Additionally, when adapting to new robot setups, we show that fine-tuning both VLAs and action verifiers yields a 7% performance increase compared to fine-tuning VLAs alone. Project website: https://robomonkey-vla.github.io.**

I. Introduction

Foundation models, pre-trained on extensive internet-scale data, have demonstrated significant potential in robotics domains. Recent advancements in Vision-Language-Action (VLA) models [2, 18, 12, 3, 4] have shown that scaling up training compute on large-scale robotics datasets [17, 30] can improve their capabilities and generalization. Despite these advancements, VLAs exhibit diverse failure modes during deployment [1, 39], such as imprecise grasping, task progression failure, and collision with surrounding objects. Addressing these limitations could accelerate the deployment of robots in unstructured real-world environments.

Efforts to improve the robustness and generalization of VLAs have gradually shifted from the pre-training to the post-training phase. In the pre-training stage, previous work emphasizes scaling up data collection [55, 30, 46], optimizing training data mixtures [15, 18], and developing model architectures [9, 2, 6, 53] that can be effectively adapted for robot control. More recently, we have observed a paradigm shift toward developments in the post-training phase, e.g., fine-tuning VLAs for multi-step reasoning with chain-of-thought [49, 10, 52] and aligning VLAs with preferences [51, 50, 20]. However, beyond pre-training and post-training, less attention has been paid to scaling the amount of compute used during deployment, as VLA models are typically designed to generate a single action chunk per observation.

Humans naturally allocate more time when encountering challenging problems. For Large Language Models (LLMs), this principle has been validated by applying additional compute at test time [5, 40, 34, 8, 41, 19]. Specifically, repeatedly sampling candidate solutions from a model has been shown to enhance the capabilities of LLMs across multiple domains, including mathematics, coding, chat, and summarization [5, 7, 11]. This raises the question of whether test-time scaling with repeated sampling may also benefit robotics. More precisely, we ask in this work: given an observation and task instruction, can we improve the precision and robustness of VLAs by repeatedly sampling and verifying actions at deployment?

We answer this question in two parts. First, we systematically investigate the benefits of scaling test-time compute in the domain of static manipulation tasks, using off-the-shelf generalist VLA models as base policies. Through our experiments, we find that the relationship between action error and the number of generated samples follows an exponentiated power law across a range of VLAs, demonstrating the existence of inference-time scaling laws. This finding aligns with the power-law scaling [36, 5] observed in LLMs and suggests that, when paired with a robust verifier, repeated sampling can significantly boost the performance of any off-the-shelf VLA model. Interestingly, different sampling techniques—repeatedly sampling actions from VLAs, Gaussian perturbation applied to a few actions, and random sampling—exhibit a similar scaling pattern. Among these, we find Gaussian perturbation to be the most cost-effective approach and it is therefore adopted in deployment. To our knowledge, our work is the first to characterize inference-time scaling laws for VLAs.

Second, we investigate whether capitalizing on these scaling laws with a learned action verifier can improve policy robustness, guided by the intuition from classic complexity theory that verifying proposals is often easier than generating a solution to a task. To do so, we present a preference-based learning recipe to automatically curate synthetic action

[†]Equal contribution. Correspondence to: jackykwok@stanford.edu

comparisons for large-scale imitation learning datasets and use it to train a 7B VLM-based action verifier. Our results show that increasing the synthetic preference dataset size leads to consistent performance improvements. We then introduce our test-time scaling framework, RoboMonkey. During deployment, RoboMonkey samples a small batch of actions from a VLA, applies Gaussian perturbation and majority voting to construct an action proposal distribution, and then uses the fine-tuned VLM-based verifier to select the optimal action. Through extensive evaluations, we demonstrate that pairing existing VLAs with RoboMonkey substantially enhances their precision and robustness.

The contributions of this paper are summarized as follows:

1) We propose efficient methods for action sampling, and demonstrate that the relationship between action error and the number of samples follows an approximate power law across a range of VLAs.
2) We present a scalable pipeline for automatically generating synthetic action preferences along with a method for training a VLM-based action verifier.
3) We show that our test-time scaling framework significantly enhances VLA performance, achieving a 25% absolute improvement in real-world out-of-distribution tasks and 8% on in-distribution SIMPLER environments.
4) We demonstrate that fine-tuning both VLAs and action verifiers yields a 7% performance increase compared to fine-tuning VLAs alone on the LIBERO-Long benchmark.

II. Related Work

Vision Language Action Models: Recent advancements in robotics have seen a shift toward training multi-task generalist robot policies [16] on large robotics datasets [46, 14] collected on diverse scenes and robot embodiments. In this landscape, several robotics foundation models have emerged. π_0, OpenVLA, PaLM-E, and RT-X [2, 18, 12, 3, 4] have demonstrated strong generalization capabilities by combining Transformer architectures [45] or diffusion policies [9] with imitation learning. While these generalist policies demonstrate out-of-the-box capabilities for controlling robots, they may still fail due to distribution shift and compounding prediction errors. Our evaluation demonstrates that RoboMonkey significantly improves the robustness and generalization of these generalist policies at deployment.

Out-of-Distribution Robustness: The challenge of learning-based systems performing unreliably on data that differs from their training distribution is documented across robotics literature [38, 1, 39]. Researchers have approached this challenge through various methodologies, including robust training and adapting models to varying environmental distribution shifts [38, 56, 27]. A significant breakthrough came with the emergence of Foundation Models (FMs). Recently, FM is widely adopted in robotics. For instance, several prior works [23, 28, 37] explore employing VLMs to generate sequences of high-level action plans, which are then executed through low-level policy. In contrast, rather than using them primarily for hierarchical planning, RoboMonkey and several concurrent works [48, 47] employ FMs as action verifiers that evaluate the low-level actions generated by robot policies.

Repeated Sampling: The methodology of applying additional computation at test time has demonstrated remarkable success across various domains. For LLMs, repeated sampling has proven effective in enhancing performance across diverse tasks, including mathematical problem-solving, coding, and text summarization [7, 5, 13]. In robotics, V-GPS [29] adopts a related strategy by training a value function with offline RL to re-rank candidate actions, selecting those that lead to better outcomes. RoboMonkey introduces a more scalable data curation pipeline and model architecture for training the action verifier. Our experimental results show that pairing existing VLAs with our verifier substantially improves both task performance and generalization compared to prior verifier-based approaches. Instead of relying on naive action sampling from robot policies, RoboMonkey uses Gaussian perturbation to efficiently generate diverse candidate actions and integrates inference-time techniques such as majority voting to guide the verification process.

III. Preliminaries

We consider a Markov Decision Process $M = (\mathcal{S},\ \mathcal{A},\ P,\ R)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ denote the robot's state and action spaces, respectively. In this work, both the state and action spaces are 7-dimensional vector spaces corresponding to the robot's end effector pose and characterized by three translational states $(x,\ y,\ z) \in \mathbb{R}^3$ and three rotational states $(u,\ v,\ w) \in \mathbb{R}^3$, while the last dimension corresponds to a binary state $g \in \{0,\ 1\}$ indicating whether the end effector gripper is open. $a_t = [\Delta x_t,\ \Delta y_t,\ \Delta z_t,\ \Delta u_t,\ \Delta v_t,\ \Delta w_t,\ g_t]'$ indicates the desired magnitude and direction to augment each state variable at time step t. Further, $P(s' \mid s,\ a) \in [0,\ 1]$ represents the robot's non-deterministic transition dynamics from the current state $s \in \mathcal{S}$ with action $a \in \mathcal{A}$ to the candidate state $s' \in \mathcal{S}$, and $R:\ \mathcal{S} \times \mathcal{A} \times \mathcal{I} \to \mathbb{R}$ provides the reward for choosing action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$ under the language instruction $I \in \mathcal{I}$, where $\mathcal{I}$ is the set of possible instructions. Our framework assumes access to a language-conditioned robot policy $\pi_\theta : \mathcal{S} \times \mathcal{I} \to \mathcal{A}$, parameterized by $\theta \in \mathbb{R}^{|\theta|}$, from which we can sample multiple actions given the state at timestep t and the language instruction $I \in \mathcal{I}$. Additionally, we assume access to a dataset of N_D expert demonstrations performing a suite of manipulation tasks: $\mathcal{D} = \{(\tau^i,\ I^i)\}_{i=1}^{N_D}$, where each demonstration constitutes a valid trajectory $\tau^i = (s_0^i, a_0^i, \ldots, s_T^i)$ under the transition dynamics to time horizon $T \in \mathbb{N}_+$. We further curate an auxiliary dataset $\mathcal{D}_{\text{buf}} \subset \mathcal{D}$ comprising of tuples $(s_t,\ a_t^*,\ I)$, where $a_t^* \in \mathcal{A}$ is the ground-truth action taken by the expert at state s_t under instruction I. We assume that the generalist policy π_θ is fine-tuned to imitate expert demonstrations on this dataset by minimizing the standard imitation learning objective as $\mathcal{L}(\theta;\ \mathcal{D}) = -\mathbb{E}_{(s_t^j,\ a_t^j,\ I^j)\sim\mathcal{D}}\left[\log\ \pi_\theta(a_t^j \mid s_t^j,\ I^j)\right]$.

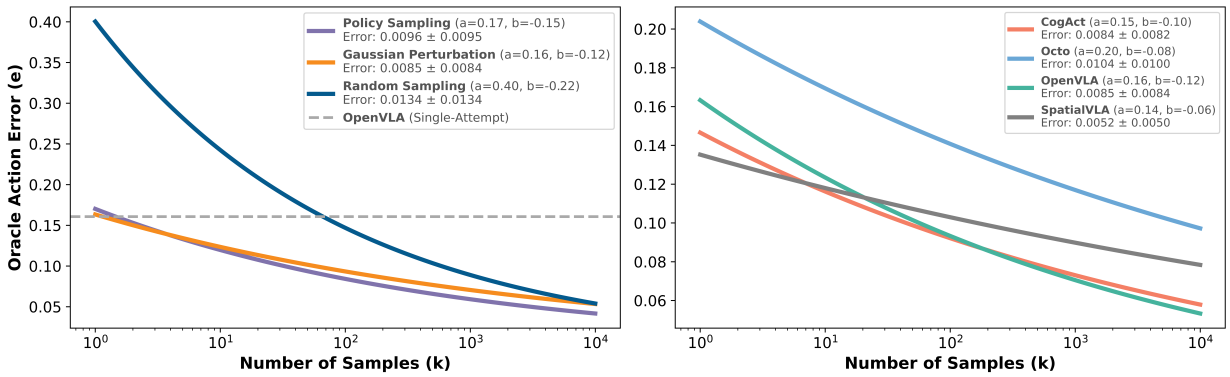

Fig. 1. **Left:** We observe that action error consistently decreases as we scale the number of generated actions across multiple sampling approaches, assuming the presence of an oracle verifier. Repeatedly sampling actions from robot policies, applying Gaussian perturbation to a few sampled actions, and even random sampling of action tokens all outperform single-attempt OpenVLA. **Right:** We find that the relationship between action error and the number of samples generated through Gaussian perturbation follows an approximate power law across a range of VLA models, including CogACT, Octo, OpenVLA, and SpatialVLA. For power law fitting, we model the logarithm of action error e as a function of the number of samples k: $\log(e) \approx \log(a) + b \cdot \log(k)$.

IV. Inference-Time Scaling Law

The relationship between a VLA's action error and its training compute [35, 24] has been well-documented. However, the potential benefits of scaling test-time compute for VLAs remain largely underexplored. To bridge this gap, we conduct a detailed analysis on the Bridge V2 Dataset [46], examining the relationship between the number of generated samples and action error.

Concretely, we uniformly sample 1,000 $(s,\ a^*,\ I)$ tuples from our auxiliary dataset $\mathcal{D}_{\text{buf}}$. For each tuple, we generate 10,000 actions using various sampling approaches and compute the Normalized Root Mean Squared Error (RMSE) between the ground-truth action a^* and each sampled actions $\{a_1,\ a_2,\ \ldots,\ a_{10,000}\}$.

We evaluate three sampling approaches: **Random sampling**: candidate actions are generated by uniformly sampling discrete action tokens for each dimension based on the scheme introduced by Brohan et al. [4] **Policy sampling**: actions are repeatedly sampled from a robot policy $\pi_\theta(a \mid s,\ I)$ with a positive temperature. **Gaussian perturbation**: sampling only 4 actions from a robot policy $\pi_\theta(a \mid s,\ I)$, then fitting a Gaussian distribution from which all candidate actions are drawn (see Section V-C for details).

The result is shown in the left plot of Figure 1. Assuming the presence of an oracle verifier that always selects the action with the lowest RMSE, we observed that as we scale the number of generated samples, the action error consistently decreases across all sampling methods. Our key findings are: (1) sampling more than 100 actions uniformly at random outperforms greedy decoding using OpenVLA; (2) using policy sampling to repeatedly generate actions from a VLA consistently yields the lowest action error; and (3) Gaussian perturbation achieves nearly identical performance compared to policy sampling while being computationally more efficient. A comprehensive latency analysis is provided in Section VI-E.

The right plot of Figure 1 demonstrates that scaling the number of generated samples with Gaussian perturbation is effective across various generalist robot policies, including CogACT, Octo, OpenVLA, and SpatialVLA [21, 43, 18, 32]. We find that the relationship between action error and the number of samples often follows an exponentiated power law. Specifically, for OpenVLA, the RMSE decreases by 59.3% when sampling 10,000 actions. Overall, we offer a new perspective on how we might approach general robot foundation models. Rather than framing robot control purely as a generation problem, our results suggest that viewing it through the lens of verification—generating diverse candidates and verifying them—can substantially improve performance. We hope our findings will motivate and guide the development of scalable action verifiers for robot policies.

V. Proposed Approach: RoboMonkey

After establishing the potential for scaling test-time compute for robotics in Section IV, we now present RoboMonkey, a framework that leverages a learned action verifier to scale test-time compute. We first describe our method for curating a synthetic action preference dataset, followed by reward modeling and inference-time techniques used within RoboMonkey's generate-then-verify pipeline.

A. Synthetic Data Generation Pipeline

In this section, we outline our approach for generating synthetic action comparisons, which leverages an existing demonstration dataset $\mathcal{D}$ to produce action pairs with high-quality preference labels without the need for human annotation. Specifically, for each tuple $(s_t,\ a_t^*,\ I)$ from our auxiliary dataset $\mathcal{D}_{\text{buf}}$, we use a reference robot policy to generate N candidate actions. To ensure diversity among the samples, we apply clustering algorithms, reducing these candidates to K representative actions. Subsequently, we construct $\binom{K}{2}$ pairwise comparisons and compute the RMSE between each sampled action, $\{a_t^1,\ a_t^2,\ \ldots,\ a_t^K\}$, and the ground-truth action, a_t^*. Then, the "winning" action, a_t^W, and the "losing" action, a_t^L, between any two actions a_t^i and a_t^j are determined

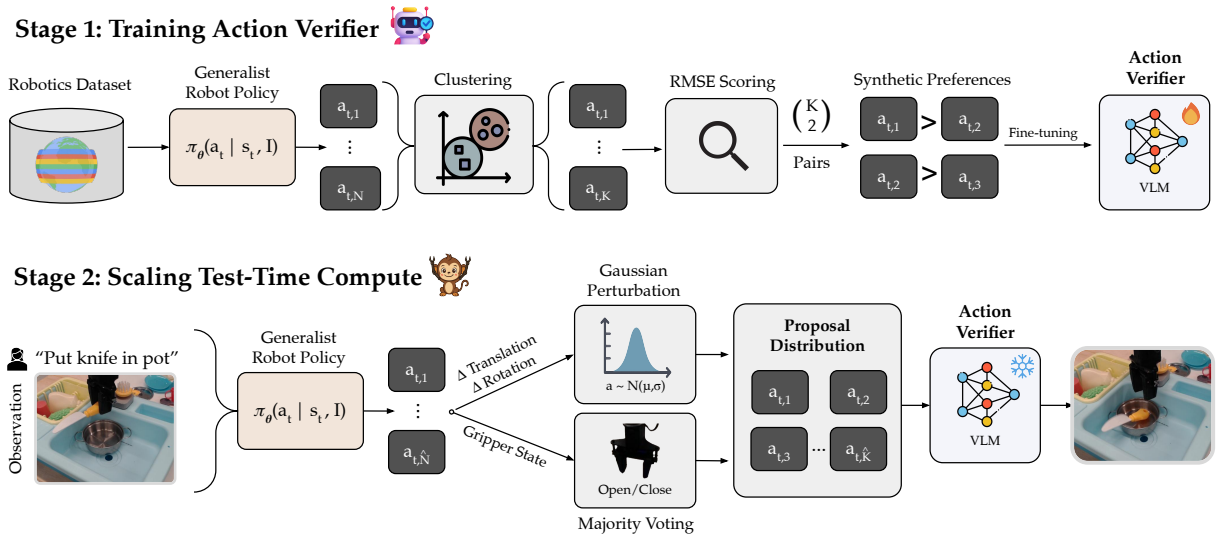

Fig. 2. **Stage 1: Training the Action Verifier.** Given an imitation learning dataset, we sample N candidate actions per state from a generalist robot policy, and apply clustering to reduce them to K representative actions. We construct $\binom{K}{2}$ synthetic action comparisons and assign preferences based on the RMSE between each sampled action and the ground-truth action. This synthetic preference dataset is then used to fine-tune a VLM-based action verifier. **Stage 2: Scaling Test-Time Compute.** At deployment, we sample $\hat{N}$ initial actions from the generalist robot policy based on the given task instruction and observation. We fit a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ to the translation and rotation components $(\Delta x,\ \Delta y,\ \Delta z,\ \Delta u,\ \Delta v,\ \Delta w)$ of these actions, and use majority voting to determine the gripper state. This creates an action proposal distribution from which we can efficiently sample candidate actions with negligible overhead. Finally, we use the fine-tuned VLM-based verifier to evaluate these $\hat{K}$ candidate actions and select the optimal action.

as follows:

$$(a_t^W, a_t^L) = \begin{cases} (a_t^i, a_t^j) & \text{if RMSE}(a_t^i, a_t^*) < \text{RMSE}(a_t^j, a_t^*) \\ (a_t^j, a_t^i) & \text{otherwise} \end{cases}$$

We use this procedure to instantiate our action preference dataset $\mathcal{D}_{\text{comp}}$ consisting of tuples $(a_t^W,\ a_t^L,\ a_t^*,\ s_t,\ I)$. Following Ouyang et.al [31], we take all $\binom{K}{2}$ pairwise comparisons from identical initial conditions $(s_t,\ I)$ and group these together as a single batch for training.

B. Reward Modeling

The loss function for training the reward model follows the Bradley-Terry model [44] with additional modifications to account for preference levels. Formally, we define the ground truth preference level as $\Delta_t^* = \left|\text{RMSE}(a_t^W,\ a_t^*) - \text{RMSE}(a_t^L,\ a_t^*)\right|$ and the predicted preference level from our action verifier $R_\phi : A \times S \times \mathcal{I} \to \mathbb{R}$, parameterized by $\phi \in \mathbb{R}^{|\phi|}$, as $\hat{\Delta}_t = \left|R_\phi(a_t^W,\ s_t,\ I) - R_\phi(a_t^L,\ s_t,\ I)\right|$. These components are then integrated into our loss function for training the reward model:

$$\mathcal{L}(\phi) = -\mathbb{E}_{(a_t^W,\ a_t^L,\ a_t^*,\ s_t,\ I)\sim\mathcal{D}_{\text{comp}}}\Big[\log\ \sigma\left(R_\phi(a_t^W,\ s_t,\ I)\right. \\ \left. - R_\phi(a_t^L,\ s_t,\ I) - \alpha\left\|\Delta_t^* - \hat{\Delta}_t\right\|_2^2\right)\Big] \quad (1)$$

where $\sigma : \mathbb{R} \to [0,\ 1]$ is the sigmoid function and $\alpha \in \mathbb{R}$ is a hyperparameter to control the magnitude of the preference level. We find that including the margin component $\left\|\Delta_t^* - \hat{\Delta}_t\right\|_2^2$ improves the accuracy, particularly when distinguishing between clearly different actions. For more detailed analysis and ablation studies, please refer to Appendices F. The action verifier uses LLaVA-7B [26, 42] as the backbone and replaces its final unembedding layer with a reward head. The architecture integrates ViT-Large [33] as the vision encoder and uses a MLP to map the visual features into the same dimensionality as the word embedding space of the language model.

C. Action Sampling and Verification

A visualization of the pipeline is shown in Figure 2. Formally, at each timestep t during deployment under instruction I, RoboMonkey first samples $\hat{N}$ candidate actions from a VLA model $\pi_\theta(a \mid s_t,\ I;\ \mathcal{T})$ with a positive temperature $\mathcal{T}$, yielding a set of candidate actions $\hat{A} = \{\hat{a}_t^1,\ \ldots,\ \hat{a}_t^{\hat{N}}\} \in \mathbb{R}^{m\times\hat{N}}$. Given these samples, we determine the gripper action g_t via majority voting over the discrete gripper component: $g_t = \text{mode}(\{g_t^i\}_{i=1}^{\hat{N}})$. We then fit a Gaussian distribution $\mathcal{N}(\mu_t, \Sigma_t)$ to both the translational components $\{[\Delta\hat{x}_t^i,\ \Delta\hat{y}_t^i,\ \Delta\hat{z}_t^i]'\}_{i=1}^{\hat{N}}$ and rotational components $\{[\Delta\hat{u}_t^i,\ \Delta\hat{v}_t^i,\ \Delta\hat{w}_t^i]'\}_{i=1}^{\hat{N}}$. RoboMonkey then samples $\hat{K}$ new actions from this proposal distribution and appends the fixed gripper state g_t to each, forming a refined action set $\tilde{A} = \{\tilde{a}_t^1,\ \ldots,\ \tilde{a}_t^{\hat{K}}\} \in \mathbb{R}^{m\times\hat{K}}$. Finally, each action $\tilde{a}_t^i$ is scored using our reward model $R_\phi(\tilde{a}_t^i,\ s_t,\ I)$ from which we select the action with the highest reward for execution $a_t = \arg\max_{\tilde{a}_t^i \in \{\tilde{a}_t^1,\ \ldots,\ \tilde{a}_t^{\hat{K}}\}} R_\phi(\tilde{a}_t^i,\ s_t,\ I)$. Algorithm 1 presents our detailed test-time scaling pipeline.

VI. Experiments

The goal of our experiments is to evaluate the effectiveness of scaling test-time compute for generalist robot policies. We evaluate RoboMonkey across both simulated and real-world

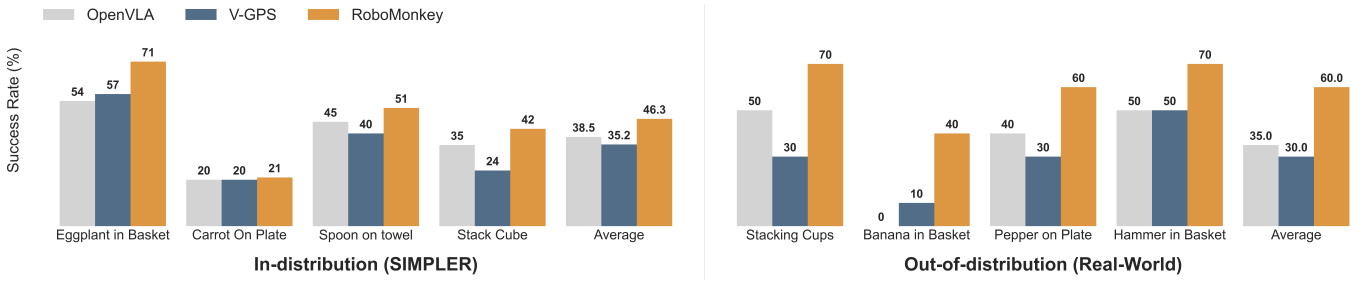

Fig. 3. Scaling test-time compute significantly improves the precision and robustness of generalist robot policies across a wide range of manipulation tasks. We observe an 8% increase in average success rate on in-distribution SIMPLER environments [22], and a 25% improvement in real-world out-of-distribution experiments using the WidowX robot.

Algorithm 1: RoboMonkey Execution

Input: Generic VLA model $\pi_\theta : \mathcal{S} \times \mathcal{I} \to A$, reward model $R_\phi : A \times \mathcal{S} \times \mathcal{I} \to \mathbb{R}$, initial state $s_0 \in \mathcal{S}$, task instruction $I \in \mathcal{I}$, temperature $\mathcal{T} \in \mathbb{R}_{++}$, time horizon $T \in \mathbb{N}_+$, number of VLA samples $\hat{N} \in \mathbb{N}_+$, number of Gaussian samples $\hat{K} \in \mathbb{N}_+$.

for $t = 0, 1, \ldots, T$ **do**

- Sample $\hat{A}_t = \{\hat{a}_t^i\}_{i=1}^{\hat{N}} \sim \pi_\theta(a \mid s_t,\ I;\ \mathcal{T})$
- Compute gripper state $g_t \leftarrow \text{mode}(\{\hat{g}_t^i\}_{i=1}^{\hat{N}})$
- Fit Gaussian distribution $\mathcal{N}(\mu_t,\ \Sigma_t)$ on $\{[\Delta\hat{x}_t^i,\ \Delta\hat{y}_t^i,\ \Delta\hat{z}_t^i,\ \Delta\hat{u}_t^i,\ \Delta\hat{v}_t^i,\ \Delta\hat{w}_t^i]'\}_{i=1}^{\hat{N}}$
- Sample $\tilde{A}_t = \{\tilde{a}_t^i\}_{i=1}^{\hat{K}} \sim \mathcal{N}(\mu_t,\ \Sigma_t)$
- Set $\tilde{a}_t^i \leftarrow [\Delta\tilde{x}_t^i,\ \Delta\tilde{y}_t^i,\ \Delta\tilde{z}_t^i,\ \Delta\tilde{u}_t^i,\ \Delta\tilde{v}_t^i,\ \Delta\tilde{w}_t^i,\ g_t]'$ for all $i \in \{1,\ 2,\ \ldots,\ \hat{K}\}$
- Select action $a_t \leftarrow \arg\max_{\tilde{a}_t^i \in \{\tilde{a}_t^1,\ \ldots,\ \tilde{a}_t^{\hat{K}}\}} R_\phi(\tilde{a}_t^i,\ s_t,\ I)$
- Execute a_t and observe s_{t+1}

environments using two different embodiments, across 4 real-robot tasks and 14 simulation tasks.

A. Implementation Details

We use the Bridge V2 Dataset [46] as our primary training dataset, which includes over 40,000 real-world robotic trajectories collected using the 6-DoF WidowX robot in 24 distinct environments. Following the procedure described in Section V-A, we curated a synthetic action preference dataset consisting of 20 million comparisons. Training was conducted on 8 NVIDIA H100 GPUs with a batch size of 256 using LoRA (r=512, α=128). We use OpenVLA as the base model for all experiments. Both RoboMonkey and V-GPS [29] are evaluated by pairing OpenVLA with their respective verifier checkpoints. SIMPLER results were obtained using two NVIDIA RTX 4090 GPUs, while real-world experiments and latency analysis were conducted on a single NVIDIA H100. LIBERO evaluations used an NVIDIA RTX 6000 Ada. For more details about model training and deployment, please refer to Appendix A.

B. Can RoboMonkey improve the precision of VLAs on in-distribution tasks?

We first evaluate our model within the SIMPLER [22] environment on in-distribution tasks. This simulation environment is specifically designed to bridge the real-to-sim gap by replicating real-world conditions for WidowX robots. It has demonstrated a strong correlation between performance in SIMPLER and real-world results [22]. We evaluate RoboMonkey and other baselines on four tasks: put eggplant in yellow basket, put carrot on plate, put spoon on towel, and stack green block on yellow block.

Figure 3 presents the evaluation results. RoboMonkey achieves an average success rate of 46.3%, outperforming OpenVLA by 7.8% on average. In the task of placing an eggplant into a basket, RoboMonkey outperforms OpenVLA by 17%. We observe that the base policy frequently collides with the wall when attempting to move the eggplant toward the basket. Similarly, in the block-stacking task, RoboMonkey surpasses OpenVLA by 7%. This task requires accurate grasping and precise placement of small objects. Overall, these results highlight that making local refinements to the base policy's actions can substantially improve precision. Additionally, we observe that pairing V-GPS with OpenVLA results in worse performance than both standalone OpenVLA and RoboMonkey. See Appendices D for details on the task setup and the ablation study on action selection.

C. Can RoboMonkey improve the robustness of VLAs on out-of-distribution tasks?

For a more comprehensive evaluation, we design a set of real-world manipulation tasks on a physical WidowX-250 S robot to evaluate RoboMonkey in OOD settings. These tasks include unseen instructions, objects, distractors, and shapes. We evaluate each approach across 4 task suites with 10 trials each, resulting in a total of 120 rollouts. Figure 3 compares the performance of RoboMonkey, V-GPS, and OpenVLA across a suite of tasks on the WidowX robot. RoboMonkey consistently outperforms both baselines across diverse tasks, including stacking cups, lifting a hammer, placing banana into a yellow basket, and putting a pepper onto a plate. We find that RoboMonkey effectively mitigates issues of imprecise grasp-

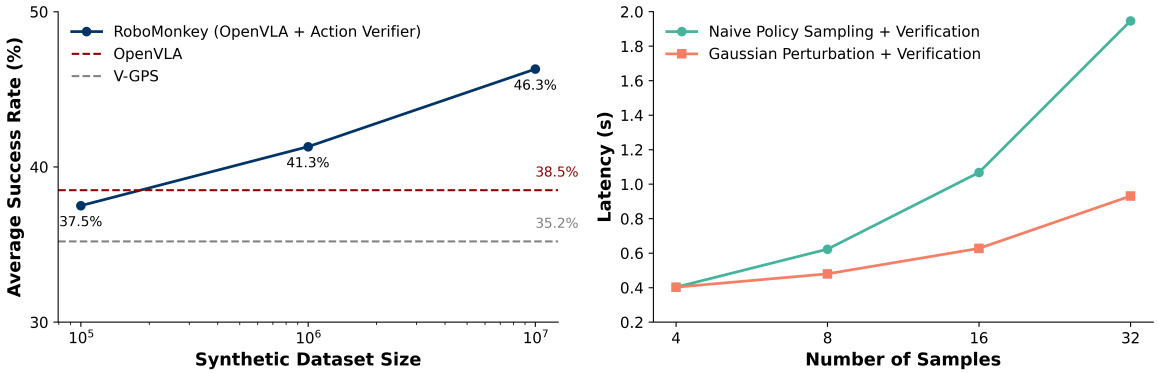

Fig. 4. **Left:** Average success rates across four tasks on SIMPLER as a function of synthetic dataset size. Scaling the dataset size (number of synthetic action comparisons) consistently improves the performance of the RoboMonkey action verifier, leading to higher success rates during robot rollouts. **Right:** Latency comparison between naive policy sampling and Gaussian perturbation across different number of actions.

ing, task progression failures, and collisions at deployment. Detailed task breakdowns and failure analysis are provided on our project page: https://robomonkey-vla.github.io.

Notably, RoboMonkey exhibits substantial improvements on tasks requiring visual and semantic generalization. For example, in the banana-in-basket task, OpenVLA achieved a success rate of 0%, as it lacks the language and visual grounding to differentiate between two yellow objects (banana and yellow basket), thus making no progress in completing the task. Furthermore, RoboMonkey achieves over 20% higher success rates on fine-grained manipulation tasks such as cup stacking and hammer lifting. These tasks require precise reasoning about grasp points, particularly on novel objects and shapes. Overall, RoboMonkey achieved an average success rate of 60%, compared to 35% from OpenVLA and 30% from V-GPS, indicating that our action verifier is significantly less sensitive to distribution shifts than the base policy. As such, these results underscore RoboMonkey's effectiveness in improving the robustness and generalization in OOD scenarios.

D. How does scaling the synthetic training dataset impact downstream success rate?

We demonstrate that RoboMonkey's closed-loop success rates on SIMPLER consistently improve as the synthetic dataset size increases, with the average success rate rising from 37.5% to 46.3% as shown in Figure 4. With action verifiers trained on over 10^6 action comparisons, RoboMonkey outperforms both OpenVLA and V-GPS. The improvements are particularly pronounced in fine-grained manipulation tasks like "Stacking Cube", where success rates increase from 27% to 37% to 42% as the synthetic dataset scales. We find that the overall task performance grows nearly log-linearly with synthetic dataset size, highlighting the potential of large-scale synthetic data generation for enhancing action verification.

E. How does RoboMonkey enable practical deployment for test-time scaling?

While RoboMonkey introduces additional computational overhead from action sampling and verification, we mitigate these costs with a practical serving solution. Specifically, we implemented a VLA serving engine on top of SGLang [54] to speed up repeated sampling of initial action candidates and employ Gaussian perturbation to efficiently construct an action proposal distribution, as detailed in Section V-C. With these optimizations, RoboMonkey can sample and verify 16 candidate actions in approximately 650 ms (or 1.5 Hz), achieving a 41.3% lower latency compared to naive policy sampling, as shown in Figure 4 (right). Gaussian perturbation proves more efficient because latency scales only with verification cost, whereas naive policy sampling incurs increasing latency from both sampling and verification as the number of candidate actions grows. See Appendix H for a detailed analysis of the trade-off between action error and compute budget.

Task	**OpenVLA**	**RoboMonkey**
Soup and Sauce in Basket	36%	**59%**
Cheese and Butter in Basket	70%	**79%**
Turn on Stove and Place Moka	58%	**58%**
Black Bowl in Drawer	36%	**37%**
Mugs on Plates	42%	**55%**
Book in Caddy	84%	**86%**
Mug and Pudding on Plate	48%	**59%**
Soup and Cheese in Basket	56%	**62%**
Moka Pots on Stove	26%	**26%**
Mug in Microwave	42%	**44%**
Average Success Rate	49.8%	**56.5%**

TABLE I
COMPARISON OF TASK SUCCESS RATES BETWEEN OPENVLA AND ROBOMONKEY, BOTH FINE-TUNED AND EVALUATED ON THE LIBERO-LONG BENCHMARK.

F. Can we effectively fine-tune the action verifier on new robot setup and task?

We further evaluate RoboMonkey's adaptability on a new robot setup. In this section, we present the fine-tuning evaluation of RoboMonkey on the LIBERO-Long benchmark, which consists of long-horizon tasks with diverse object configurations and layouts in simulation. In our experiments, we curated a new action preference dataset using the procedure described in Section V-A. The OpenVLA-LIBERO checkpoint used for

comparison was trained via behavioral cloning on successful demonstrations. All methods were evaluated across 500 trials.

Table I presents the results and we observe that RoboMonkey can be effectively adapted to tasks in the LIBERO environments. Fine-tuning both OpenVLA and action verifier results in 6.7% improvement in average success rate compared to simply fine-tuning OpenVLA on LIBERO-Long.

VII. Discussion and Limitations

In this paper, we presented RoboMonkey, a novel test-time scaling framework that enhances the precision and robustness of Vision-Language-Action (VLA) models. RoboMonkey achieves significant performance improvements across both in-distribution and out-of-distribution tasks, as well as on new robot setups. Our findings demonstrate that scaling test-time compute through a generate-and-verify paradigm provides a practical and effective path towards building general-purpose robotics foundation models. The current RoboMonkey framework has several limitations that we leave for future work:

a) Computational Overhead: Since it requires sampling multiple candidate actions from a VLA and employs a separate VLM-based action verifier, it incurs increased computational overhead during deployment. Although we mitigate these costs with a practical serving solution—using a VLA serving engine and Gaussian perturbation—the framework may still be less suitable for tasks requiring high-frequency control. Future work could explore more efficient model architectures for action verification and apply system-level optimizations to further reduce the memory footprint and latency of scaling test-time compute.

b) Scaling Synthetic Datasets: Our results show that increasing the size of the synthetic training dataset consistently improves downstream robot performance. However, due to compute constraints and the high cost of fine-tuning, we limited our experiments to 20 million synthetic action comparisons on the Bridge V2 dataset. Scaling synthetic data generation to larger robotics datasets across embodiments, tasks, and environments is a promising direction for future exploration.

c) Evaluation Scope: While our experiments focused on two commonly used robotic arms—WidowX 250S and Franka—future work should evaluate RoboMonkey across a broader range of embodiments.

VIII. acknowledgments

This work was supported by DARPA and the National Aeronautics and Space Administration under the University Leadership Initiative program.

References

[1] Christopher Agia, Rohan Sinha, Jingyun Yang, Zi ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress, 2024. URL https://arxiv.org/abs/2410.04640.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[5] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[6] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[7] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*, 2024.

[8] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems. *arXiv preprint arXiv:2403.02419*, 2024.

[9] 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*, page 02783649241273668, 2023.

[10] Jaden Clark, Suvir Mirchandani, Dorsa Sadigh, and Suneel Belkhale. Action-free reasoning for policy generalization, 2025. URL https://arxiv.org/abs/2502.03729.

[11] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[12] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet,

Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023. URL https://arxiv.org/abs/2303.03378.

[13] Ryan Ehrlich, Bradley Brown, Jordan Juravsky, Ronald Clark, Christopher Ré, and Azalia Mirhoseini. Codemonkeys: Scaling test-time compute for software engineering, 2025. URL https://arxiv.org/abs/2501.14723.

[14] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A robotic dataset for learning diverse skills in one-shot. In *RSS 2023 Workshop on Learning for Task and Motion Planning*, 2023.

[15] Joey Hejna, Chethan Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning, 2024. URL https://arxiv.org/abs/2408.14037.

[16] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022.

[17] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[18] 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.

[19] Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E. Gonzalez, and Ion Stoica. S*: Test time scaling for code generation, 2025. URL https://arxiv.org/abs/2502.14382.

[20] Derun Li, Jianwei Ren, Yue Wang, Xin Wen, Pengxiang Li, Leimeng Xu, Kun Zhan, Zhongpu Xia, Peng Jia, Xianpeng Lang, Ningyi Xu, and Hang Zhao. Finetuning generative trajectory model with reinforcement learning from human feedback, 2025. URL https://arxiv.org/abs/2503.10434.

[21] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024. URL https://arxiv.org/abs/2411.19650.

[22] 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. *arXiv preprint arXiv:2405.05941*, 2024.

[23] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control, 2023. URL https://arxiv.org/abs/2209.07753.

[24] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation, 2025. URL https://arxiv.org/abs/2410.18647.

[25] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[26] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[27] Peter Mitrano and Dmitry Berenson. Data augmentation for manipulation, 2022. URL https://arxiv.org/abs/2205.02886.

[28] Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought, 2023. URL https://arxiv.org/abs/2305.15021.

[29] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance, 2025. URL https://arxiv.org/abs/2410.13816.

[30] Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[31] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

[32] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model, 2025. URL https://arxiv.org/abs/2501.15830.

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

[34] Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Guha, E Kelly Buchanan, Mayee Chen, Neel Guha, Christopher Ré, et al. Archon: An architecture search frame-

work for inference-time techniques. *arXiv preprint arXiv:2409.15254*, 2024.

[35] Sebastian Sartor and Neil Thompson. Neural scaling laws in robotics, 2025. URL https://arxiv.org/abs/2405.14005.

[36] Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (laws)?, 2025. URL https://arxiv.org/abs/2502.17578.

[37] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

[38] Rohan Sinha, Apoorva Sharma, Somrita Banerjee, Thomas Lew, Rachel Luo, Spencer M. Richards, Yixiao Sun, Edward Schmerling, and Marco Pavone. A system-level view on out-of-distribution data in robotics, 2023. URL https://arxiv.org/abs/2212.14020.

[39] Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Edward Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models, 2024. URL https://arxiv.org/abs/2407.08735.

[40] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[41] Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. The good, the bad, and the greedy: Evaluation of llms should not ignore non-determinism. *arXiv preprint arXiv:2407.10457*, 2024.

[42] Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, Kurt Keutzer, and Trevor Darrell. Aligning large multimodal models with factually augmented rlhf, 2023. URL https://arxiv.org/abs/2309.14525.

[43] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[44] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. advances in neural information processing systems. *Advances in neural information processing systems*, 30(2017), 2017.

[46] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[47] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Perez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions, 2025. URL https://arxiv.org/abs/2411.16627.

[48] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment, 2025. URL https://arxiv.org/abs/2502.01828.

[49] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2407.08693.

[50] Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Josef Dai, Yuanpei Chen, and Yaodong Yang. Safevla: Towards safety alignment of vision-language-action model via safe reinforcement learning, 2025. URL https://arxiv.org/abs/2503.03480.

[51] Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment, 2025. URL https://arxiv.org/abs/2411.19309.

[52] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025. URL https://arxiv.org/abs/2503.22020.

[53] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

[54] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

[55] Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models, 2024. URL https://arxiv.org/abs/2407.20635.

[56] Eric Zhu, Mara Levy, Matthew Gwilliam, and Abhinav Shrivastava. Nerf-aug: Data augmentation for robotics with neural radiance fields, 2025. URL https://arxiv.org/abs/2411.02482.

APPENDIX

A. Training

Our action verifier uses LLaVA-7B [26, 42] as the backbone and replaces its final unembedding layer with a reward head. The architecture integrates ViT-Large [33] as the vision encoder and uses a MLP to map the visual features into the same dimensionality as the word embedding space of the language model. We discretize each dimension of a continuous action into 256 bins, following Brohan et al. [3], and overwrite the 256 least frequent tokens in the LLaMA tokenizer with these discrete action tokens.

Training was conducted on 8 NVIDIA H100 GPUs using LoRA (rank=512, $\alpha = 128$), and our codebase builds on top of LLaVA-RLHF [42]. We use a batch size of 256 and train on a synthetic preference dataset comprising 20 million comparisons derived from the Bridge V2 dataset. The model is trained using the Adam optimizer with a learning rate of 2e-5. Training is conducted for a single epoch. A margin weight of 0.1 is applied to the modified Bradley-Terry loss. Example prompt to action verifier is shown below:

```
USER: <image> shows the current
observation from the robot's
wrist-mounted camera. The robot
manipulation arm is attempting to
[instruction].
What action should the robot take
to effectively accomplish the task?
ASSISTANT: The robot should take
the action [Discrete Action Tokens]
USER: Please evaluate the quality
of the robot action.
ASSISTANT: The quality score of the
robot action is
```

B. Deployment

For real-world evaluation, we first sample 5 initial actions from OpenVLA with temperature 1.0. We fit a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ to the translation and rotation components, and use majority voting to determine the gripper state. This creates an action proposal distribution from which we sample 16 candidate actions. We then use the fine-tuned VLM-based verifier to select the optimal action for execution. We conduct 10 trials per task and report the average success rate. In simulation, we follow the same initial sampling procedure and vary the number of candidate actions $\hat{K} \in \{8, 16, 32\}$. We report the best results for each task. All simulated experiments are conducted using a machine equipped with two NVIDIA RTX 4090 GPUs over three random seeds.

C. Baselines

We use the publicly released OpenVLA checkpoint from ht tps://huggingface.co/openvla/openvla-7b, and the V-GPS value function checkpoint from https://github.com/nakamotoo/V-G PS. In simulation, we follow the evaluation procedure outlined in the V-GPS implementation, sweeping over the number of samples $\{10, 50\}$ and softmax temperatures $\{0, 0.1, 1.0\}$, and report the best result for each task. For real-world evaluations, we fix the number of samples to 10 and the temperature to 1.0. It is worth noting that we use our VLA serving engine to enable efficient batch inference for all experiments.

D. Ablation over Action Selection Methods and Number of Samples

To evaluate the effectiveness of our verifier, we adopt a setup similar to that described in Section IV. Specifically, we uniformly sample 1,000 (s, a^*, I) tuples from the auxiliary dataset $\mathcal{D}_{\text{buf}}$, curated from the BridgeV2 [46]. For each tuple, we generate 64 candidate actions using the reference policy, OpenVLA, and apply various selection techniques—including RoboMonkey, V-GPS, majority voting, and random selection—to identify the optimal action among the samples. We report the normalized RMSE between the ground-truth action and the selected action for each method. As shown in Figure 5, RoboMonkey consistently achieves the lowest action error across different sample sizes. When generating 64 samples, RoboMonkey reduces the action error by 21% relative to the greedy decoding baseline, highlighting the effectiveness of our verifier in improving action precision. While prior work such as V-GPS trained with offline RL also improves over greedy decoding, its value function achieves only a 6% reduction in action error. Furthermore, we observe that increasing the number of samples leads to exploitation of the V-GPS value function, resulting in performance degradation when sampling more than 8 actions. In contrast, RoboMonkey remains robust to reward hacking and demonstrates scalability with increased test-time compute.

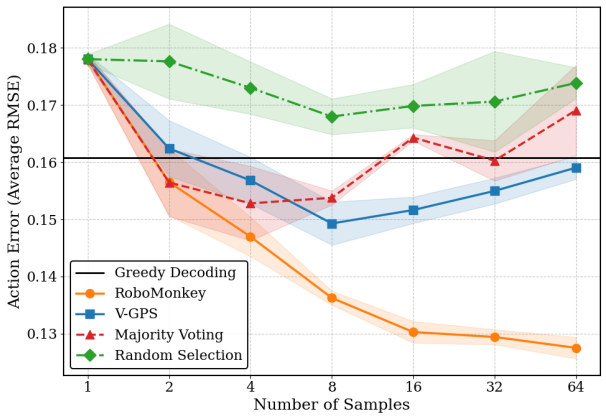

Fig. 5. Comparison of action error (average RMSE) across different selection methods as the number of generated samples increases. RoboMonkey consistently outperforms other baselines and scales effectively with additional compute.

E. Ablation over Generalist Robot Policies

We conducted additional experiments to ablate the performance of RoboMonkey when paired with different VLA models. Following a similar evaluation setup to Appendix D,

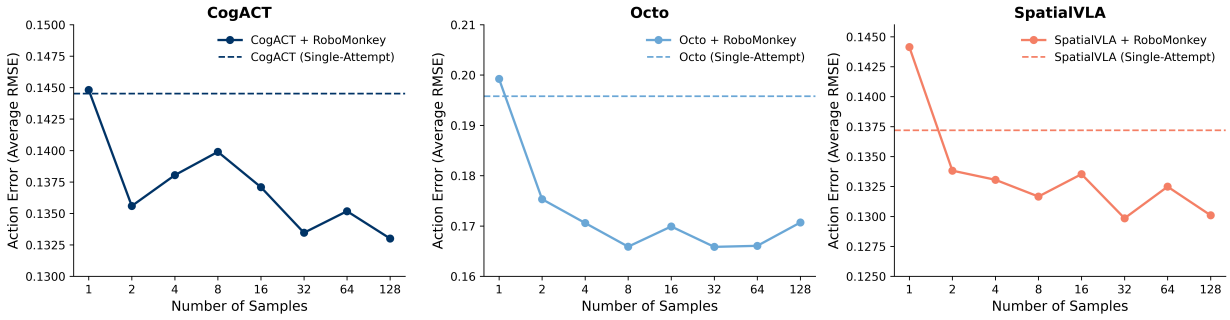

Fig. 6. Effect of scaling test-time compute with RoboMonkey across different generalist robot policies. Action error (average RMSE) decreases as the number of samples increases. Dashed lines denote the action error of each base policy when only generating a single action.

our ablation considers three generalist robot policies: CogACT, Octo, and SpatialVLA.

CogACT is a 7B VLA model with a modular architecture that separates cognitive reasoning from motor control. Built on top of the Prismatic VLM (DINOv2 + SigLIP for vision and LLaMA-2 for language), CogACT introduces a specialized action module implemented as a diffusion transformer. We use the CogACT-base variant in our experiments. Pairing RoboMonkey with CogACT achieves RMSE of 0.133, reflecting an 8% reduction from its single-attempt baseline of 0.145.

Octo is a generalist policy trained on 800K demonstrations from the Open X-Embodiment (OXE) dataset. The policy includes a CNN encoder and a ViT-style transformer backbone with a diffusion-based action head that predicts action sequences. We use Octo-small for evaluation. Integrating RoboMonkey with Octo achieves RMSE of 0.166, representing a 15.3% reduction from its greedy baseline (0.196).

SpatialVLA is a 3.5B parameter spatially grounded VLA model trained on 1.1M robot episodes from OXE and RH20T. The model uses Ego3D Position Encoding to integrate 3D spatial context from depth estimates into visual features. It is pre-trained on a PaLI-Gemma-2 backbone. For deployment, we use a temperature of 0.5 for sampling. SpatialVLA + RoboMonkey achieves RMSE of 0.1298, a 5.3% reduction from its baseline of 0.137.

F. Ablation on Margin for Reward Modeling

α	Precision	Recall	F1 Score
0	0.81	0.85	0.83
0.1	**0.84**	**0.87**	**0.85**
1.0	0.79	0.83	0.81

TABLE II
COMPARISON OF ACTION VERIFIER PERFORMANCE ACROSS DIFFERENT MARGIN WEIGHTS α IN THE LOSS FUNCTION. WE FIND THAT INCORPORATING A SMALL MARGIN ($\alpha = 0.1$) IMPROVES PRECISION, RECALL, AND F1 SCORE

As illustrated in Section V-B, the loss function for training the action verifier follows the Bradley-Terry model [44] with an additional margin component to account for preference levels. To evaluate the effectiveness of this margin component, we generated 10,000 synthetic action comparison pairs from the Bridge V2 dataset following the procedure outlined in Section 5.1. Each action pair consists of distinctly different actions. We trained two variants of the action verifier with margin terms ($\alpha \in 0.1,\ 1.0$) and compared their performance to a baseline model without a margin term ($\alpha = 0$). Table II reports the precision, recall, and F1 score for each setting. The variant with $\alpha = 0.1$ achieved the highest F1 score (0.85), outperforming both the baseline without a margin ($\alpha = 0$, F1 = 0.83) and the large-margin variant ($\alpha = 1.0$, F1 = 0.81). These results suggest that incorporating a margin term can improve the action verifier's performance, but an excessively large margin may negatively impact verification accuracy. Based on this analysis, we adopt the $\alpha = 0.1$ variant for deployment.

G. Latency and Throughput Analysis for Sampling and Verification

Repeated sampling can exploit KV Cache optimizations and batch processing to achieve higher throughput than greedy decoding. However, most VLA models, including OpenVLA, are built on top of Prismatic VLM and do not support batching [18]. SGLang provides efficient serving with prefix caching, overhead-free CPU scheduling, and paged attention. Therefore, to make RoboMonkey practical for deployment, we extended SGLang's capabilities to properly support Prismatic VLM [18] models, enabling us to achieve higher throughput during repeated sampling. Users can easily port their Prismatic VLM models to SGLang using our provided template.

We conducted experiments on a single H100 to measure the latency and throughput of OpenVLA inference across varying batch sizes. As shown in Table III, our optimized implementation significantly outperforms the naive version [18]. For instance, at a batch size of 32, our serving engine reduces latency by 74% and increases throughput by over 120x. Even at smaller batch sizes (e.g. 4), our VLA serving engine achieves a 54% reduction in latency.

Our action verifier also benefits significantly from batch inference, achieving a throughput of 46 actions/s at a batch size of 16. It is notably faster than OpenVLA, as it only requires computation during the prefill stage, which can fully leverage GPU parallelism. In contrast, OpenVLA involves

Batch Sizes	OpenVLA		OpenVLA + SGLang		Action Verifier	
	Latency (s)	Throughput	Latency (s)	Throughput	Latency (s)	Throughput
1	0.15	6.5	0.13	7.6	0.092	11
2	0.31	3.3	0.21	9.6	0.099	20
4	0.61	1.6	0.28	15	0.13	32
8	1.2	0.82	0.42	19	0.20	39
16	2.4	0.41	0.72	22	0.35	46
32	4.9	0.20	1.3	25	0.65	49
64	9.8	0.10	2.4	27	1.3	50
128	20	0.050	4.6	28	2.5	51

TABLE III
LATENCY (SECONDS) AND THROUGHPUT (SAMPLES/SECOND) COMPARISON ACROSS BATCH SIZES FOR OPENVLA, OPTIMIZED OPENVLA, AND 7B ACTION VERIFIER.

both the prefill and decode stages—where action tokens must be generated autoregressively—resulting in lower throughput.

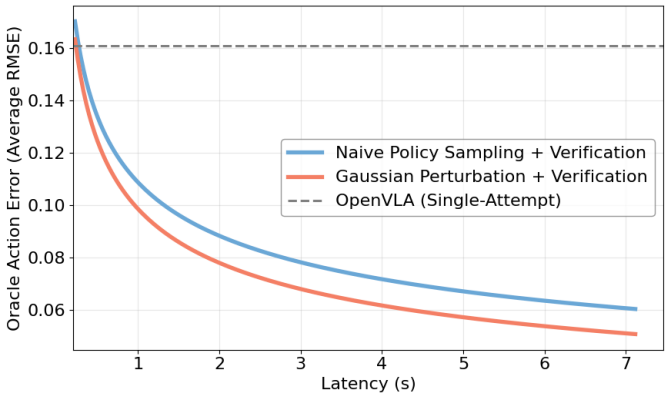

Fig. 7. Action error (average RMSE) as a function of computational overhead for policy sampling and Gaussian perturbation. Gaussian perturbation consistently achieves lower action error under equivalent computational budgets

H. Trade-off between Action Error and Computational Overhead

In this ablation study, we examine the trade-off between action error and computational overhead. In Figure 7, we reproduce the scaling curve from Section IV, but plotting action error against latency. Under equivalent computation budgets, Gaussian perturbation achieves significantly lower oracle action error compared to policy sampling. We observe that the gap between the two methods widens as compute budgets increase. This growing performance gap reflects the fact that Gaussian perturbation scales only with the verifier, while policy sampling incurs additional overhead from both the policy and verifier. These results highlight Gaussian perturbation as a more practical choice for deployment.

I. Evaluation Tasks

As described in Section VI-C, we evaluate RoboMonkey, OpenVLA, and V-GPS on a physical WidowX-250 S robot across four out-of-distribution (OOD) generalization tasks. Real-world evaluations inherently introduce distribution shifts, as we cannot exactly replicate the Bridge V2 setup. In particular, slight variations in camera placement, robot positioning, lighting conditions, and background are unavoidable. We describe the four representative generalization tasks as follows:

- **Stack Blue Cup on Pink Cup**: The goal is to grasp the blue cup and stack it on top of the pink cup. The language instruction of this task was not included in the Bridge V2 dataset.
- **Put Hammer into Yellow Basket**: The robot must lift the hammer and place it inside a yellow basket. Importantly, the hammer represents an unseen object with a novel shape not present in any Bridge V2 demonstration.
- **Put Pepper onto Plate**: This language grounding task requires the robot to identify and approach the pepper while ignoring distractors (e.g., sushi). The robot must then differentiate between the yellow basket and a plate before correctly placing the pepper.
- **Put Banana into Yellow Basket**: The objective of this task is to place a banana from the sink into the yellow basket. This task presents a particular challenge as the system must differentiate between two yellow objects (banana and yellow basket), requiring visual grounding to complete the task successfully.

For details on the task setup for in-distribution and fine-tuning evaluation, please refer to the SIMPLER [22] and LIBERO-LONG [25] benchmark. We include task execution examples in Figure 8.

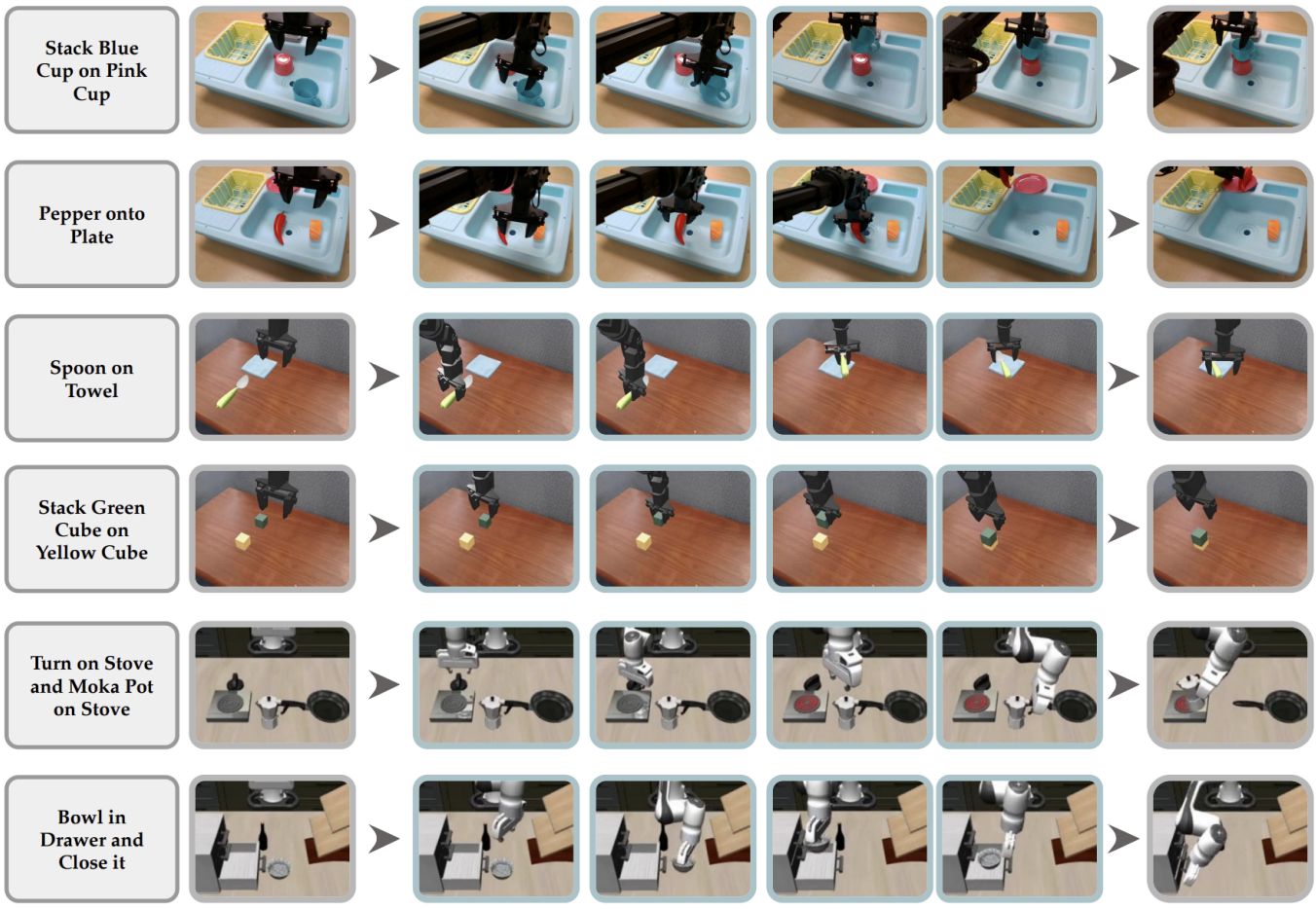

Fig. 8. Representative task executions in real-world, SIMPLER, and LIBERO environments.