VLA-REPLICA: A Low-Cost, Reproducible Benchmark for Real-World Evaluation of Vision-Language-Action Models

Alex S. Huang[1*] **Jiahui Zhang**[1*] **Shiqing Tang**[2] **Yu Xiang**[1]

[1]Intelligent Robotics and Vision Lab, University of Texas at Dallas
[2]Allen High School

{alex.huang, jiahui.zhang, yu.xiang}@utdallas.edu
shiqing.tang@student.allenisd.org
*Equal contribution

Abstract

Vision-Language-Action (VLA) models have shown strong promise for general-purpose robotic manipulation, but their real-world evaluation remains limited by a lack of accessible, reproducible, and consistent benchmarks. Simulation benchmarks fail to capture real-world complexity, while existing real-world benchmarks often require expensive hardware, centralized evaluation, or are limited in task diversity. We introduce *VLA-REPLICA*, a low-cost, easily reproducible real-world benchmark for evaluating VLA models. Built from off-the-shelf components, our system can be quickly assembled and replicated across laboratories, providing a consistent environment for policy evaluation anywhere in the world. *VLA-REPLICA* includes a diverse suite of manipulation tasks and a small-scale demonstration dataset for target-domain adaptation, with real-world evaluation protocols for both in-distribution and out-of-distribution settings. Experiments with imitation learning and state-of-the-art VLA models reveal model strengths and limitations, while consistent results across independently constructed setups demonstrate the reproducibility of our benchmark.[1]

1 Introduction

Vision-Language-Action (VLA) models [18, 4, 14] have recently emerged as a promising paradigm for building general-purpose robotic systems that can interpret natural language instructions and execute corresponding manipulation behaviors. By leveraging large-scale multimodal data [27, 17], these models demonstrate impressive generalization across objects, scenes, and tasks, bringing robotics closer to the goal of flexible, human-level interaction. However, despite their success in controlled settings, a fundamental challenge remains: *how to reliably evaluate VLA models in the real world*.

Existing evaluation protocols fall short in addressing this challenge. Simulation-based benchmarks such as Meta-World [22] and LIBERO [20] provide scalable and standardized environments, but they suffer from the well-known sim-to-real gap, often leading to overly optimistic estimates of real-world performance. In contrast, real-world benchmarks offer more faithful evaluation but introduce new limitations. Some benchmarks, such as REPLAB [30] and SceneReplica [16], focus primarily on object grasping, limiting task diversity. Others, such as FurnitureBench [12] and ManipulationNet [8],

[1]Data, code, and videos for the project are available at https://irvlutd.github.io/VLAReplica/.

Preprint.

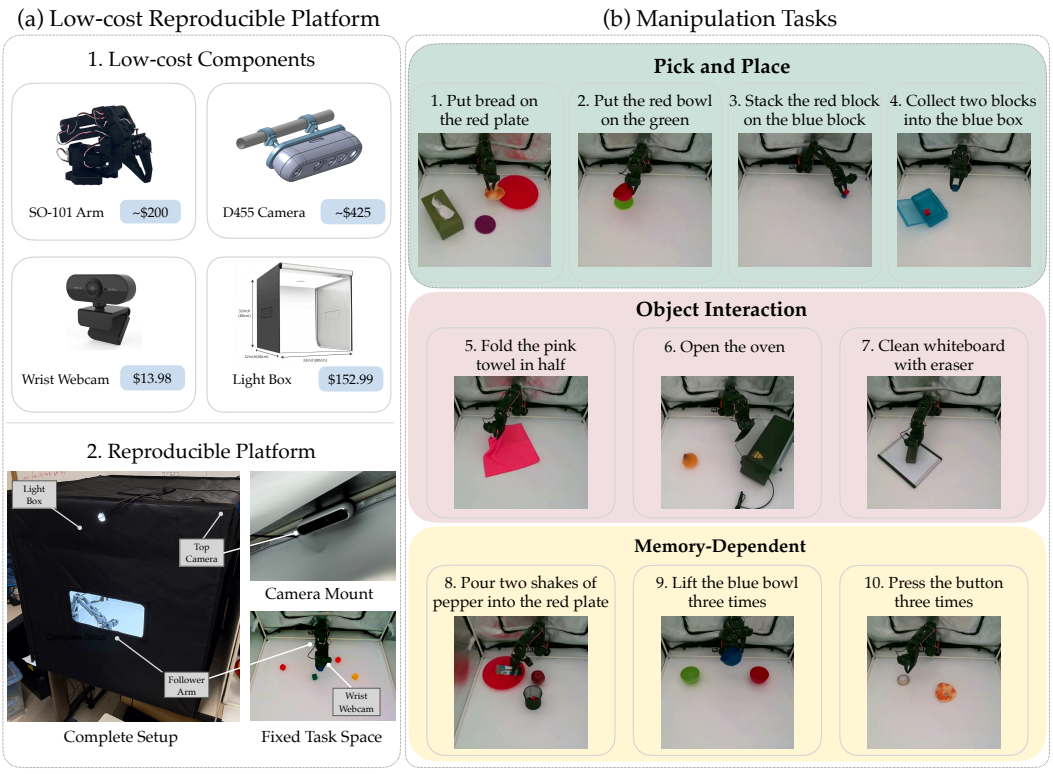


Figure 1: Overview of the VLA-Replica benchmark. (a)1. Hardware components. (a)2. Our assembled platform with the leader arm, the follower arm, the light box, the cameras, and the manipulation workspace. (b) 10 manipulation tasks in the benchmark.

require expensive hardware setups, carefully engineered environments, and substantial human effort to reproduce. Additionally, systems like RoboArena [3] rely on centralized or remote evaluation, where users must submit policies for external execution, restricting accessibility, slowing iteration, and reducing transparency. As a result, there is currently *no widely adopted benchmark that is both real-world grounded and easily reproducible across independent laboratories*.

To address these limitations, we introduce **VLA-REPLICA**, a low-cost, reproducible real-world benchmark for evaluating VLA models. Our key design principle is to make real-world evaluation *accessible, standardized, and locally executable*. Specifically, VLA-REPLICA is built on an affordable robotic platform using off-the-shelf components, including a low-cost SO-101 manipulator, RGB-D cameras, and a controlled light-box environment. The hardware setup is illustrated in Fig. 1(a). The full setup can be assembled in under an hour by non-expert users, and detailed instructions ensure consistent replication across different sites.

Beyond hardware accessibility, VLA-REPLICA provides a structured evaluation protocol tailored to modern VLA models. The benchmark includes a suite of real-world manipulation tasks with diverse interaction patterns (e.g., pick-and-place, tool use, and object manipulation), along with a curated dataset of human demonstrations for target-domain adaptation. Fig. 1(b) shows the 10 manipulation tasks in our benchmark. Crucially, the evaluation is designed to measure both *in-distribution performance*, i.e. how well models learn from limited real-world data, and *out-of-distribution generalization*, including variations in object properties and task requirements. This setup reflects practical deployment scenarios, where pretrained VLA models must adapt to new environments with minimal data.

A central contribution of our work is *reproducibility at scale*. We introduce a standardized environment construction pipeline, including precise workspace configuration, camera placement, background lighting, and object placement protocols, enabling consistent evaluation across independently built setups. This design allows different users to recreate nearly identical physical environments, making it possible to compare results across laboratories without centralized infrastructure.

Table 1: Comparison of robot manipulation benchmarks across environment, reproducibility, hardware cost, task diversity, dataset availability, and evaluation protocol.

Benchmark	Year	Environment	Repro-ducibility	Hardware Cost	Task Diversity	Expert Dataset	Distributed Evaluation
Meta-World [32]	2020	Simulation	High	–	High (50 tasks)	✓	✓
LIBERO [20]	2023	Simulation	High	–	High (100 tasks)	✓	✓
RoboCasa [24]	2024	Simulation	High	–	High (100 tasks)	✓	✓
RoboLab [31]	2026	Simulation	High	–	High (120 tasks)	✗	✓
REPLAB [30]	2019	Real	Low	Low	Low (Grasping)	✓	✓
RAMP [10]	2023	Real	High (April Tag)	High	Low (Assembly)	✗	✓
SceneReplica [16]	2024	Real	High (Overlay)	High	Low (Grasping)	✗	✓
FurnitureBench [12]	2025	Real	High (April Tag)	High	Low (Assembly)	✓	✓
FMB [21]	2025	Real	Low	High	Low (Assembly)	✓	✓
AutoEval [35]	2025	Real	Low	None (Service)	Low (4 tasks)	✓	✗
RoboArena [3]	2025	Real	Low	High	High (User-defined)	✗	✓
RoboChallenge [29]	2025	Real	High (Overay)	None (Service)	High (30 tasks)	✓	✗
VLA-Replica (Ours)	2026	Real	High (Overlay+Lighting)	Low (∼$1050 USD)	High (10 Tasks)	✓	✓

We validate VLA-REPLICA by benchmarking both imitation learning methods [33, 9] and state-of-the-art VLA models [4, 14, 28] under a unified training and evaluation protocol. Our experiments reveal key insights into the performance of current VLA systems, particularly their ability to adapt to new environments and generalize beyond training distributions. We further demonstrate that evaluation results are consistent across independently constructed setups, highlighting the robustness and reproducibility of our benchmark.

In summary, this paper makes the following contributions:

- We introduce *VLA-REPLICA*, a low-cost (∼$1050 USD), fully reproducible real-world benchmark for evaluating VLA models.
- We design a *standardized hardware and environment setup* that can be replicated across laboratories with minimal effort and cost.
- We propose a *comprehensive evaluation protocol* covering both in-distribution learning and out-of-distribution generalization.
- We provide *extensive empirical evaluation* of state-of-the-art VLA models, offering insights into their real-world performance and limitations.

2 Related Work

Vision-Language-Action Models Recent years have seen the emergence of VLA models [6, 5, 18, 4, 14, 26, 13], which are general-purpose robotic manipulation policies trained on diverse robot datasets for language-conditioned tasks. While these models show promising generalization across objects, scenes, and tasks, they often struggle when deployed in new environments or on new robot embodiments. Prior work therefore commonly fine-tunes pretrained VLA policies using a small amount of target-domain data, enabling adaptation to the target environment, embodiment, and tasks.

Simulation Benchmarks for Robot Manipulation Robotic evaluation benchmarks have been widely used to measure the performance and generalization of robot policies. Table 1 provides a comparison of representative robot manipulation benchmarks. A large body of benchmarks are build in simulation [23, 20, 24, 19, 11, 22, 31, 25]. These benchmarks provide scalable and repeatable environments for policy evaluations. However, policies that perform well in simulation may not transfer directly to the real world due to the physical gap between simulation and real world.

Real-World Benchmarks for Robot Manipulation Another line of work evaluates robot policies in the real world [30, 16, 29, 12, 3, 8]. These benchmarks provide realistic measurements of robot performance in physical environments. However, only a few benchmarks offer reproducible test scenes. RAMP [10] and FurnitureBench [12] use April Tags on objects. SceneReplica [16] and RobotChallenge [29] provides reference images for test scenes, where users can check the overlay of real objects with a reference image to place objects. In addition, some recent real-robot evaluation benchmarks rely on remote or centralized evaluation [3, 29], where users submit policy checkpoints or inference APIs and wait for external evaluators to run the experiments. While this improves standardization, it limits immediate, on-site, and reproducible evaluation by individual labs.

Table 2: Bill of materials for the VLA-Replica benchmark

Item	Unit Cost	Quantity	Description
SO-101 Follower Arm	∼$200	1	multiple vendors
Intel RealSense D455 Camera	∼$425	1	multiple vendors
Vinmooog Webcam	$13.98	1	from Amazon
Light Box (see Appendix A)	$152.99	1	from Amazon
Object Set (see Appendix B)	$215.99	1	from Amazon
Total Cost	**∼$1050**		

As shown in Table 1, VLA-REPLICA is a low-cost, reproducible real-world benchmark with diverse manipulation tasks. Built on the SO-101 and commodity objects, it enables reproducible test scenes via reference images and controlled lighting, supports distributed evaluation across sites, and provides in-domain demonstrations for fine-tuning pretrained VLA policies.

3 The VLA-Replica Benchmark

3.1 Physical Platform and Hardware Configuration

We now describe the hardware setup of VLA-REPLICA. Detailed documentation for constructing the complete platform is available at: `https://irvlutd.github.io/VLAReplica`.

Low-cost reproducible hardware design. To make the benchmark reproducible, realistic, and consistent across laboratories, we design a low-cost hardware platform for less than **$1100 USD**. As shown in Figure 1(a), the platform consists of a 6-DoF low-cost SO-101 follower arm [1], an RGB web camera, and an Intel RealSense D455 RGB-D camera. For the workspace, we use a 32 inch $\times$ 32 inch $\times$ 32 inch photography light box to reduce environmental variation and provide consistent illumination, at around 5600K. We 3D-print a custom mount to attach the Intel RealSense D455 RGB-D camera to the rear top frame of the light box, where it serves as the top-view camera for observing the workspace. Similarly, we 3D-print a wrist camera mount [2] and attach it onto the end-effector of the SO-101 follower arm. Table 2 shows the bill of materials of our benchmark.

Hardware configuration. To make both camera setups reproducible across laboratories, we implement a camera overlay program to ensure the camera view at any laboratory matches our original setup's view (for both top and wrist cameras). Additionally, to fine-tune the camera position, we place an AprilTag at a fixed position next to the SO-101 arm and record its 6-DoF pose. Combined with the visual overlay program, this AprilTag provides an additional reference metric for aligning the camera with respect to the robot base, helping different users reproduce a consistent camera viewpoint to the workspace. Fig. 2(a) illustrates the process of reproducing the physical platform.

We provide detailed system setup instructions in Appendix A, enabling users to build the benchmark environment. As an initial reproducibility check, a user with no prior knowledge of the benchmark was able to build the setup within one hour.

3.2 SO-101 Action Reproducibility

One advantage of LeRobot's SO-101 robotic arm is its low cost and ease of self-assembly, comprised entirely of 3D-printed parts and Dynamixel STS3215 servo motors. However, this introduces the problem of non-standardized manufacturing, specifically, the differing initial offset of the servo motors across differently constructed SO-101 arms, which constrains each arm to its own specific action space. The LeRobot SO-101 Python library [7] contains useful tools, such as individual robot calibration, to solve this. We ensure action reproducibility by defining a *universal action space* that all SO-101 arms must share.

Arm Calibration. Each action $\hat{\mathbf{a}}$ that controls the SO-101 arm is a 6-dimensional vector $\{a_1, \ldots, a_6\}$, where a_i is a raw encoder value in range $[0, 4095]$ sent to servo motor i on the arm. Every SO-101 arm requires calibration during setup with the `lerobot-setup-motors` package command. During calibration, the arm is extended at a standard "L" shape, which defines the universal "zero position" of all SO-101 arms. Three actions $\{\mathbf{a}^{\text{offset}}, \mathbf{a}^{\text{min}}, \mathbf{a}^{\text{max}}\}$ are captured during calibration: the servos'

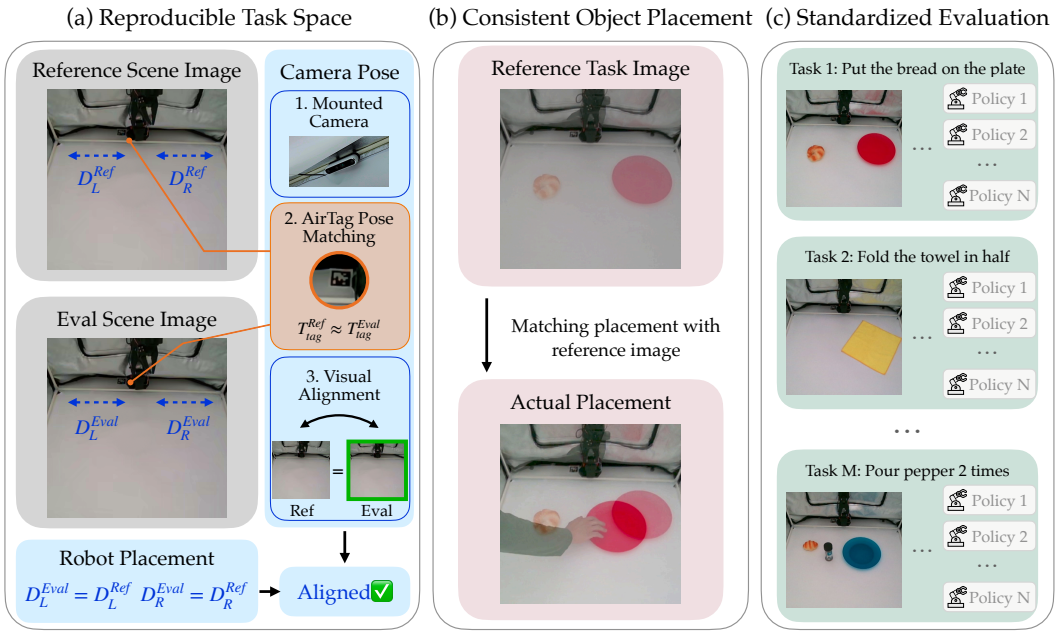


Figure 2: VLA-REPLICA standardized method ensuring reproducibility. (a) Align the task space, robot position, and camera poses via AprilTag calibration and video overlay matching. (b) Use task-specific reference images to ensure consistent object placement during setup. (c) Evaluate different policies on the VLA-REPLICA task suite.

offset values at the "zero position", and the servos' minimum and maximum values swept throughout the joint limits, respectively.

Action Normalization. Let us define a 6×6 diagonal calibration matrix $\mathbf{D}$ of the arm so that its entries r_i are the linear scaling factors calculated from $\mathbf{a}^{\text{min}}$ and $\mathbf{a}^{\text{max}}$ (every arm should have very similar physical joint limits):

$$\mathbf{D} = \frac{1}{180}\begin{bmatrix} r_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & r_6 \end{bmatrix}, \; r_i = \frac{a_i^{\text{max}} - a_i^{\text{min}}}{2}, \; i = 1, \ldots, 6. \tag{1}$$

Now, given an action $\hat{\mathbf{a}}$, we can define the *SO-101 universal action* $\hat{\mathbf{u}}$, the joint angle (in degrees) of each servo joint as

$$\hat{\mathbf{u}} = \mathbf{D}^{-1}(\hat{\mathbf{a}} - \mathbf{a}^{\text{offset}}), \; \hat{\mathbf{u}} \in [-180, 180], \tag{2}$$

where $\mathbf{D}$ and $\mathbf{a}^{\text{offset}}$ are obtained from the arm calibration. $\hat{\mathbf{u}}$ is the action that we save during demonstration collection. Consequently, a policy is also trained to output $\hat{\mathbf{u}}$. Inversely, to convert $\hat{\mathbf{u}}$ into a specific action for the arm, we have

$$\hat{\mathbf{a}} = \mathbf{D}\hat{\mathbf{u}} + \mathbf{a}^{\text{offset}}, \; \hat{\mathbf{a}} \in [0, 4095]. \tag{3}$$

Therefore, as long as an SO-101 arm is calibrated, i.e., $\{\mathbf{a}^{\text{offset}}, \mathbf{a}^{\text{min}}, \mathbf{a}^{\text{max}}\}$ are available, we can use Eq. (3) to convert the policy output $\hat{\mathbf{u}}$ to the actual action $\hat{\mathbf{a}}$ for the arm.

3.3 Task Suite and Evaluation Protocols

Our *VLA-REPLICA* task suite consists of ten tasks as shown in Table 3. The task suite covers a diverse set of robot behaviors, including pick-and-place, pulling, wiping, pouring, other object-centric interactions, and memory capabilities. The full task list, including task success criterion, is provided in Appendix C.

Demonstration and Training Data Collection. We collect demonstrations for the suite of ten real-world robot manipulation tasks by using an SO-101 leader arm to control the SO-101 follower arm. For each task, we collect 50 demonstrations with different scene configurations through an

Table 3: Task definitions and examples for training/ID and OOD evaluation. ID tasks follow the same distribution as training data with varied object configurations, while OOD tasks introduce distribution shifts in object attributes (e.g., color, count) and task requirements to evaluate generalization.

Task #	Task Type	Training & ID Eval Task Example	OOD Eval Task Example
1	Pick-and-Place	Put bread in **red**/**blue** plate	Put bread on **yellow** plate
2		Put **red** bowl on **purple** coaster	Put **green** bowl on **yellow** coaster
3		Stack **blue** cube on **red** cube	Stack **green** cube on **green** cube
4		Put all **2 or 3** blocks in **blue** box	Put all **3 or 4** blocks in **pink** box
5	Object Interaction	Fold **pink**/**yellow** towel in half	Fold **blue** towel in half
6		Open oven	N/A
7		Clean whiteboard with eraser	N/A
8	Memory	Pour pepper **1,2 or 3** times to **red** plate	Pour pepper **4 or 5** times to **blue** plate
9		Lift **green**/**red**/**blue** bowl **1 or 3** times	Lift **yellow** bowl **2, 4 or 5** times
10		Press button **1 or 3** times	Press button **2, 4 or 5** times

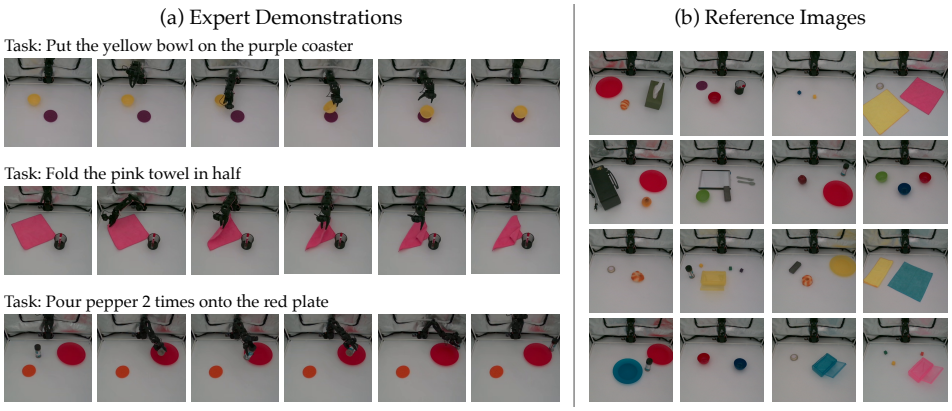


Figure 3: (a) Examples of expert demonstrations collected in our dataset. (b) Examples of reference images for setting up test scenes.

expert teleoperator, resulting in 500 demonstrations total. We ensured consistent robot actions to reduce action divergence, including establishing a "home position" for the robot to return to at the beginning and end of an episode. Fig. 3(a) shows some examples of our collected demonstrations.

Evaluation Scenarios and Test Scene Design. A key challenge in evaluating VLA models is to disentangle *adaptation* from *generalization*. In practical deployments, a policy must both learn effectively from limited target-domain data and generalize to novel variations of the environment. To capture these complementary aspects, we evaluate policies under both *in-distribution* (ID) and *out-of-distribution* (OOD) settings. The ID setting measures how well a model can fit and execute tasks within the target environment after fine-tuning, while OOD setting evaluates robustness to variations that are not observed during training, such as changes in object properties or task configurations.

To enable standardized and reproducible evaluation, we define a *test scene* as the initial configuration of all objects in the workspace, including both target and distractor objects. The *VLA-REPLICA* benchmark provides a total of 90 test scenes (50 ID and 40 OOD), ensuring that every evaluation is performed under identical initial conditions. Fig. 3(b) shows some examples of these test scenes. The design of these test scenes is guided by the need to avoid overfitting to training configurations. During expert data collection, object placements are manually randomized while ensuring diversity across demonstrations. We then analyze the spa-

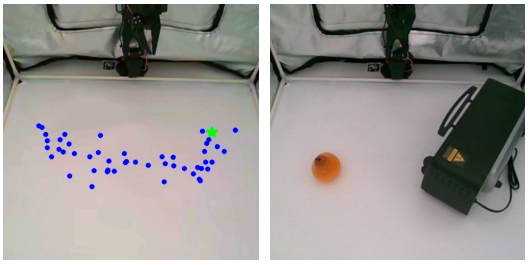


Figure 4: For the task *Open the Oven*, the blue dots indicate the oven center locations in the training set. The green star indicates the selected oven location for a test scene.

tial distribution of object positions by extracting object centers from all 500 demonstrations for each task (Appendix D). This reveals regions with varying densities of object placements (Fig. 4). Based on this distribution, we construct test scenes with controlled deviations from the training data, introducing varying levels of difficulty depending on how far object placements depart from previously seen configurations.

For ID evaluation, we define five test scenes per task that remain within the support of the training distribution while still covering diverse configurations. Across 10 tasks, this results in 50 ID test scenes, each evaluated once per policy. For OOD evaluation, we further design test scenes that introduce systematic variations beyond the training distribution. Specifically, we modify object attributes (e.g., color and shape) and task requirements (e.g., memory or sequencing), and evaluate on eight of the ten tasks, resulting in 40 OOD test scenes. These OOD scenarios are designed to probe whether policies can generalize beyond memorized behaviors and adapt to novel conditions within the same physical environment. Table 3 illustrates the differences between ID and OOD test scenes.

3.4 Real-World Scene Reconstruction and Reproducibility

Ensuring consistent scene setup across different environments is critical for reliable real-world evaluation. Prior work has explored the use of *image overlay* as a practical tool for improving reproducibility [16]. Building on this idea, we adopt a similar overlay-based approach in *VLA-REPLICA* to standardize scene reconstruction. During setup, a live video stream from the top-view camera is overlaid with predefined test scene images, allowing users to directly match object placements to the target configuration. Fig. 2(b) illustrates the scene setup process for a single evaluation trial. This design enables fast and intuitive alignment of both target and distractor objects, reducing human effort and minimizing deviations across setups. Refer to Appendix E for all scenes.

While image-overlay-based methods have been explored in prior real-world benchmarks (e.g., SceneReplica [16] and RoboChallenge [29]), we integrate this mechanism into a unified evaluation pipeline tailored for VLA models. In combination with our standardized hardware setup and predefined test scenes, this approach provides a simple yet effective solution for achieving consistent and reproducible real-world evaluation across independent laboratories.

4 Benchmarking Experiments

4.1 Algorithms

We evaluate both imitation learning methods and state-of-the-art VLA models on our benchmark. We use all 500 collected demonstrations for training or finetuning the models. For imitation learning, we train ACT [33], a multi-task Diffusion Transformer [15], and a multi-task Diffusion Transformer with a flow-matching expert [15]. These models are trained directly on the collected target-domain demonstrations. For VLA models, we fine-tune several recent state-of-the-art models, including SmolVLA [28], X-VLA [34], π_0 [4] and $\pi_{0.5}$ [14]. All VLA models are fine-tuned using the same 500 expert demonstrations and evaluated on the same held-out evaluation set.

For all methods, we train or fine-tune the policies for 40K steps. For X-VLA, we fine-tune the vision encoder, language encoder, and policy transformer. For π_0, we fine-tune the VLM and keep the vision encoder frozen. At the beginning of each evaluation rollout, all objects are placed to match their corresponding locations in a reference image, ensuring a consistent initial state across trials. This unified evaluation protocol allows us to compare conventional imitation learning methods with pretrained VLA models under the same target-environment adaptation setting. All training and evaluation implementations are based on the official LeRobot [7] implementations.

These models are (More training details can be found in Appendix F):

- **ACT** [33] trains a transformer-based action chunking policy from expert demonstrations using supervised imitation learning.
- **Multi-task Diffusion Transformer (DiT-D)** [15] extends Diffusion Policy [9] with a text- and vision-conditioned transformer policy for language-conditioned robot manipulation.
- **Flow-Matching DiT (DiT-F)** [15] uses the same multi-task DiT backbone but replaces the diffusion objective with a flow-matching objective for action generation.

Table 4: Policy Evaluation Success Rate on 10 In-Distribution (ID) Tasks.

Task #	Task Type	Task (5 runs each)	**ACT** [33]	**DiT-D** [15]	**DiT-F** [15]	**SmolVLA** [28]	**X-VLA** [34]	$\boldsymbol{\pi}_0$ [4]	$\boldsymbol{\pi}_{0.5}$ [14]
1	Pick-and-Place	Put bread on plate	0.4	0.4	0.4	0.6	0.4	0.8	0.8
2		Put bowl on coaster	0	0	0	0.2	0.2	0.6	0.8
3		Stack block on block	0	0	0	0.2	0	0	0.4
4		Put all blocks into box	0	0.2	0	0	0	0	0.4
5	Object Interaction	Fold towel	0.4	0.2	0.2	0.6	0.6	0.8	1.0
6		Open oven	0.4	0.6	0.4	0.4	0	0.2	0.6
7		Erase whiteboard	0.2	0.2	0.2	0.2	0	0.4	0.4
8	Counting / Memory	Shake pepper n times	0.2	0	0	0	0.2	0.2	0.4
9		Lift bowl n times	0.2	0	0	0.2	0	0.2	0.4
10		Press button n times	0	0	0	0.2	0	0.2	0.2
Average Success Rate			**0.18**	**0.16**	**0.12**	**0.26**	**0.14**	**0.34**	**0.54**

- **SmolVLA** [28] is a lightweight VLA model designed for efficient fine-tuning on task-specific demonstration datasets.
- **X-VLA** [34] is a VLA model that conditions action generation on visual observations and language instructions for generalizable robot manipulation.
- $\boldsymbol{\pi_0}$ [4] is a generalist vision-language-action model for language-conditioned robot control.
- $\boldsymbol{\pi_{0.5}}$ [14] extends π_0 with improved generalization to new objects, environments, and tasks.

4.2 In-Distribution Policy Evaluation

We first evaluate all trained policies in the same physical setting (i.e., the same light box) where the demonstrations were collected, and test the polices with the ID tasks as described in Table 3. This setting represents an in-domain evaluation scenario, where the policy is tested on the target environment after being trained or fine-tuned with a limited number of demonstrations collected from that environment. This evaluation measures how effectively policies learn or adapt to the target tasks using the collected demonstrations. Since all policies are trained using the same 500 demonstrations and evaluated under the same robot setup, the results provide a fair comparison of their ability to learn from limited target-domain data.

Table 4 reports the success rates of all evaluated policies on the in-domain evaluation tasks. After training or fine-tuning, the policies perform well on pick-and-place tasks with deformable objects, such as bread and towels. In contrast, rigid objects such as blocks are sensitive to the gripper pose: an inaccurate grasp pose can cause the object to slip or shift. This leads to lower success rates on tasks such as *Stack Blocks* and *Put Blocks in Box*. Under the same training budget, fine-tuned VLA policies generally outperform imitation learning policies trained from scratch, which suggests VLA pretraining provides a useful initialization for adapting to the target tasks and environment. Refer to Appendix G for detailed ID evaluation results.

4.3 Out-of-distribution Policy Evaluation

To evaluate generalization, we test the policies on eight out-of-distribution tasks as shown in Table 3. For pick-and-place and object-interaction tasks, these OOD tasks include new objects shapes or new colors . For memory-counting tasks, we evaluate whether policies can generalize to unseen repetition counts. For example, while the training demonstrations include *Shake the Pepper* once and three times, the OOD evaluation asks the policy to *Shake the Pepper* twice.

We report the OOD evaluation results in Table 5. Compared to the ID results, we notice the performance drop of these polices. We further observe that SmolVLA, π_0, and $\pi_{0.5}$ maintain success rates on OOD pick-and-place and object-interaction tasks that are comparable to their ID task performances. One possible reason is that these OOD tasks mainly test color and object-shape generalization, where pretrained VLA models can reuse learned manipulation skills while adapting to new visual appearances. However, none of the methods, including both imitation learning baselines and VLA models, generalize well to memory-counting tasks, where we observe a large success rate drop. From our qualitative observations, the policies fail to keep track of how many times an action has been executed. For example, when asked to *Shake the Pepper 2 times*, the policies continue shaking repeatedly and fail to place the pepper back on the table. These results show that VLA

Table 5: Policy Evaluation Success Rate on 8 Out-of-Distribution (OOD) Tasks.

Task #	Task Type	Task (5 runs each)	**ACT** [33]	**DiT-D** [15]	**DiT-F** [15]	**SmolVLA** [28]	**X-VLA** [34]	π_0 [4]	$\pi_{0.5}$ [14]
1	Pick-and-Place	Put bread on plate	0.4	0	0.2	0.8	0.6	0.8	1.0
2		Put bowl on coaster	0.2	0.2	0	0.4	0	0.6	0.4
3		Stack block on block	0	0	0	0.2	0	0.2	0
4		Put all blocks into box	0	0	0	0.2	0	0	0.2
5	Object Interaction	Fold towel	0	0.2	0	0.6	0	0.6	0.8
6	Counting / Memory	Shake pepper n times	0	0	0	0	0	0.2	0.4
7		Lift bowl n times	0	0	0	0.2	0	0	0
8		Press button n times	0	0	0	0	0	0	0
Average Success Rate			**0.075**	**0.05**	**0.025**	**0.3**	**0.075**	**0.3**	**0.35**

models can generalize to different colors and object shapes, but still struggle with memory-counting tasks that require tracking repeated actions. Refer to Appendix G for detailed OOD evaluation results.

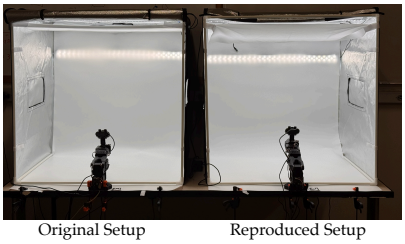
4.4 Reproducibility Analysis

To evaluate the reproducibility of our benchmark, we built a new instance of the real-world setup at a different location as shown in Fig. 5. The new setup includes a separate but identical light box, cameras, and SO-101 follower arm. The setup was assembled by a user without prior experience with the benchmark, following the provided construction and calibration instructions in Appendix A.

We then evaluated the same policies used in Sec. 4.1 on the newly constructed platform. The top and wrist cameras were calibrated to match the viewing angles of the original setup, and the policy outputs, which correspond to the original SO-101 arm, were transformed to the action space of the new arm using Eq. (3) from Sec. 3.2. To focus on reproducibility rather than policy capability, we selected five ID and three OOD tasks with the highest success rates in the original setting, and excluded tasks that already had low success rates on the original platform. This allows us to test whether the same policy achieves comparable performance when deployed on an independently built instance of the benchmark. The results are reported in Table 6. Across all models and tasks, the success rates on the new setup are comparable to those obtained on the original platform, which suggests that our benchmark can provide consistent evaluation results across independently built environments. These results further support that policies fine-tuned on our released demonstrations can be reasonably evaluated on newly constructed instances of the platform.

Table 6: Reproducible success rates of models on a subset of ID and OOD evaluation tasks.

Task (5 runs each)	**ACT** [33]	**SmolVLA** [28]	π_0 [4]	$\pi_{0.5}$ [14]	**Avg.**
Original ID {1,2,5,6,7}	0.28	0.40	0.56	0.72	**0.49**
Reproduced ID {1,2,5,6,7}	0.32	0.44	0.48	0.68	**0.48**
Original OOD {1,2,5}	0.20	0.60	0.67	0.73	**0.55**
Reproduced OOD {1,2,5}	0.20	0.53	0.60	0.67	**0.50**



Original Setup Reproduced Setup

Figure 5: Original and reproduced setups.

5 Conclusion & Limitation

We introduced **VLA-REPLICA**, a low-cost and reproducible real-world benchmark for evaluating vision-language-action (VLA) models. Our benchmark combines an affordable hardware setup, standardized environment design, and a unified evaluation protocol covering both in-distribution adaptation and out-of-distribution generalization.

Experiments with imitation learning and state-of-the-art VLA models reveal their strengths and limitations in real-world settings. We further show that evaluation results are consistent across independently constructed setups, demonstrating the reproducibility of our benchmark. We hope VLA-REPLICA provides a practical foundation for standardized real-world evaluation and accelerates progress toward general-purpose robotic systems.

Limitations. Although *VLA-REPLICA* improves the accessibility and reproducibility of real-world evaluation, several limitations remain. First, the benchmark currently focuses on tabletop manipulation with a single low-cost robot embodiment, which may not capture the diversity of real-world robotic systems and environments. Second, while the benchmark includes multiple manipulation behaviors and OOD evaluation settings, the overall task scale remains smaller than large-scale simulation benchmarks. Finally, although our standardized setup and calibration procedures reduce many sources of variation, small differences in hardware, lighting, and object placement may still introduce inconsistencies across independently constructed environments.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under Grant Nos. 2346528 and 2520553, the NVIDIA Academic Grant Program Award, and a gift funding from XPeng.

References

[1] So101 arm. `https://huggingface.co/docs/lerobot/so101`, .

[2] So101 camera mount. `https://github.com/TheRobotStudio/SO-ARM100/tree/main/Optional/Wrist_Cam_Mount_Vinmooog_Webcam`, .

[3] P. Atreya, K. Pertsch, T. Lee, M. J. Kim, A. Jain, A. Kuramshin, C. Eppner, C. Neary, E. Hu, F. Ramos, et al. Roboarena: Distributed real-world evaluation of generalist robot policies. In *Proceedings of the Conference on Robot Learning (CoRL 2025)*, 2025.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arxiv:2410.24164*, 2024.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023.

[7] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascal, J. Choghari, J. Moss, and T. Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024.

[8] Y. Chen, K. Kimble, E. H. Adelson, T. Asfour, P. Chanrungmaneekul, S. Chitta, Y. Chitambar, Z. Chen, K. Goldberg, D. Kragic, et al. Manipulationnet: An infrastructure for benchmarking real-world robot manipulation with physical skill challenges and embodied multimodal reasoning. *arXiv preprint arXiv:2603.04363*, 2026.

[9] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[10] J. Collins, M. Robson, J. Yamada, M. Sridharan, K. Janik, and I. Posner. Ramp: A benchmark for evaluating robotic assembly manipulation and planning. *IEEE Robotics and Automation Letters*, 9(1):9–16, 2023.

[11] H. Geng, F. Wang, S. Wei, Y. Li, B. Wang, B. An, C. T. Cheng, H. Lou, P. Li, Y.-J. Wang, Y. Liang, D. Goetting, C. Xu, H. Chen, Y. Qian, Y. Geng, J. Mao, W. Wan, M. Zhang, J. Lyu, S. Zhao, J. Zhang, J. Zhang, C. Zhao, H. Lu, Y. Ding, R. Gong, Y. Wang, Y. Kuang, R. Wu, B. Jia, C. Sferrazza, H. Dong, S. Huang, Y. Wang, J. Malik, and P. Abbeel. Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning, 2025. URL `https://arxiv.org/abs/2504.18904`.

[12] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *The International Journal of Robotics Research*, 44 (10-11):1863–1891, 2025.

[13] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou. $\pi^*_{0.6}$: A vla that learns from experience. *arXiv:2511.14759*, 2025.

[14] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[15] B. Jones. Dissecting and open-sourcing multitask diffusion transformer policy, 2025. URL `https://brysonkjones.substack.com/p/dissecting-and-open-sourcing-multitask-diffusion-transformer-policy`. Blog post.

[16] N. Khargonkar, S. H. Allu, Y. Lu, B. Prabhakaran, and Y. Xiang. Scenereplica: Benchmarking real-world robot manipulation by creating replicable scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8258–8264. IEEE, 2024.

[17] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[18] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

[19] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[20] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[21] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, 44(4):592–606, 2025.

[22] R. McLean, E. Chatzaroulas, L. McCutcheon, F. Röder, T. Yu, Z. He, K. Zentner, R. Julian, J. K. Terry, I. Woungang, N. Farsad, and P. S. Castro. Meta-world+: An improved, standardized, RL benchmark. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL `https://openreview.net/forum?id=1de3azE606`.

[23] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[24] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[25] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

[26] NVIDIA, J. Bjorck, N. C. Fernando Castañeda, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

[27] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[28] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[29] A. Yakefu, B. Xie, C. Xu, E. Zhang, E. Zhou, F. Jia, H. Yang, H. Fan, H. Zhang, H. Peng, et al. Robochallenge: Large-scale real-robot evaluation of embodied policies. *arXiv preprint arXiv:2510.17950*, 2025.

[30] B. Yang, J. Zhang, V. Pong, S. Levine, and D. Jayaraman. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *arXiv preprint arXiv:1905.07447*, 2019.

[31] X. Yang, R. Dagli, A. Zook, H. Hadfield, A. Goyal, S. Birchfield, F. Ramos, and J. Tremblay. Robolab: A high-fidelity simulation benchmark for analysis of task generalist policies, 2026. URL `https://arxiv.org/abs/2604.09860`.

[32] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[33] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[34] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

[35] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. *arXiv preprint arXiv:2503.24278*, 2025.

A VLA-REPLICA Benchmark Setup Instructions

This section provides step-by-step instructions for reliably reproducing our benchmark environment across different laboratories (online version).

Please read and follow these instructions carefully. Careful inspection of the photographs and text is crucial for creating a reproducible and precise environment.

A.1 Environment Overview

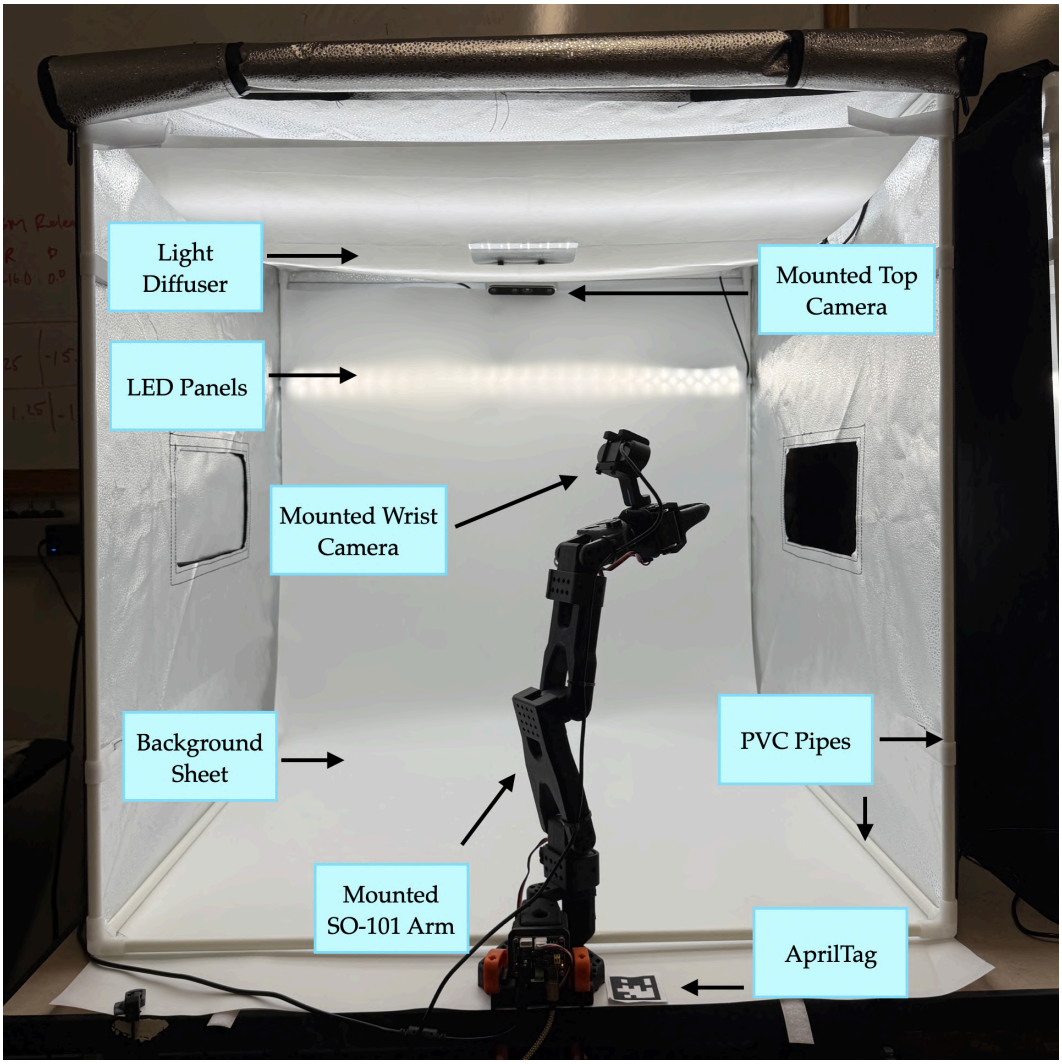


Figure A.1: **Benchmark environment.** Physical workspace showing the SO-101 follower arm, 32×32 in light box, LED panel, white background sheet, and AprilTag.

The benchmark comprises of the components listed in Table A.1. The cameras, after calibration, handle RGB observation, while the SO-101 follower arm executes manipulation tasks.

A.2 Assemble the Camera Mounts

Top Camera: Before mounting hardware, 3-D print the two required parts (PLA/PETG, standard 15% infill, 0.4mm) and assemble the top camera mount (for the D455 Realsense) as follows.

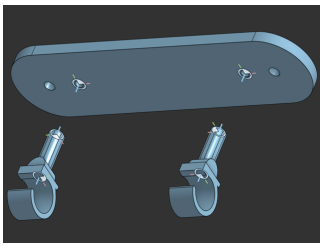
1. Print **two copies** of the snap-hook part (`Part1.stl`).

Table A.1: Parts list for the benchmark setup.

Qty	Item
1	Glendan 32×32 in box set (box tarp, 12× PVC pipes, 8× PVC edge connectors, white PP background sheet, white light diffuser sheet, 3×LED panel set, power cables) (link)
1	Intel RealSense D455
1 set	3-D printed camera mount (1× backplate, 2× snap-hooks, 2× M3×6 mm screws, 1× M3×12 mm screws, 1× M3 nut, 2× M4×6 mm screws) (link)
1	SO-101 follower arm (link)+ 3-D printed wrist camera mount (link) + Vinmooog RGB camera (link)+ 12 V power adapter
1	4 cm tag size *tag36h11* AprilTag (generate it here)
4	Clamps
1 roll	Rubber grip tape (link)
1 (opt.)	SO-101 leader arm + 6 V power adaptor

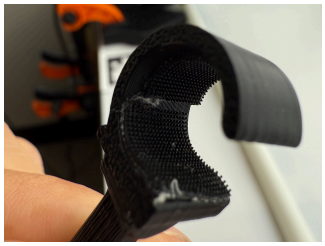
2. Print **one copy** of the camera backplate (`Part2.stl`).
3. Attach one snap-hook (Part 1) to the backplate (Part 2) using one M3×6 mm screw. Repeat for the second hook. The assembled unit is referred to as *Part 3* (Fig. A.2(a)).
4. Screw Part 3 tightly to the **rear mounting holes** of the D455 camera using two M4×6 mm screws (Fig. A.2(b)).
5. To prevent the hooks from sliding on the PVC pipe, attach a small piece of rubber grip tape to the *inside* of each hook (Fig. A.2(c)). **Do not over-tighten the screws or apply excessive force to the hooks, as they may snap.**

Note: the CAD file for the snap-hook has an inner diameter that matches the outer diameter of the PVC pipe for the Glendan light box. Other light boxes may have different PVC diameters and may require manual re-CAD.



(a) Snap-hooks attached to backplate (Part 3).

(b) Part 3 screwed to D455 camera.



(c) Rubber grip tape on inside of hooks.

Figure A.2: **Camera mount assembly.**

SO-101 and Wrist Camera setup: Before mounting hardware, 3-D print the one required part (PLA/PETG, standard 15% infill, 0.4mm) and mount the wrist camera as follows.

1. If applicable (i.e. the SO-101 did not come pre-assembled), follow LeRobot's SO-101 documentation page to assemble the SO-101 Follower arm: (`https://huggingface.co/docs/lerobot/so101`). **Don't calibrate the assembled SO-101 arm yet.**
2. Follow TheRobotStudio's page to print and set up the wrist camera mount with the Vinmooog webcam: (`https://github.com/TheRobotStudio/SO-ARM100/tree/main/Optional/Wrist_Cam_Mount_Vinmooog_Webcam`)
3. Secure the camera mount onto the end-effector of the SO-101 with one M3×12 mm screw and the M3 nut.

Important Checklist:

☐ Both snap-hooks are attached with M3 screws and sit flush against the backplate.

☐ The mount is fastened to the D455 with M4 screws; the camera does not wobble.

☐ Rubber grip tape is applied to the inside of both hooks.

☐ The SO-101 follower arm is assembled, with the wrist camera mounted in place.

A.3 Set Up the Light Box

A controlled environment with consistent lighting and a clutter-free background is essential for consistent visual observations across laboratory settings. To do this, we set up a 32×32 in. Glendan light box at the edge of a work table.

Read the orientation convention below before assembly.

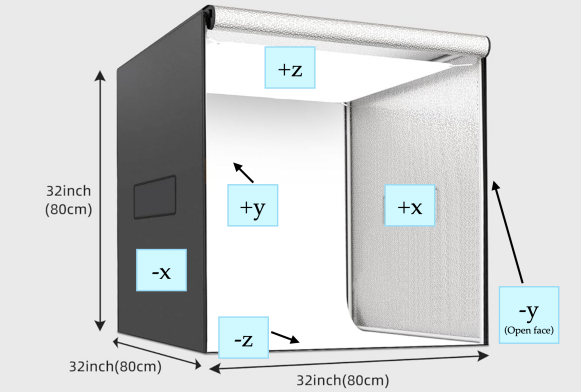


Figure A.3: **Light box coordinate system.** $-y$ is the open face; $+z$ is the top; $-z$ is the workspace floor.

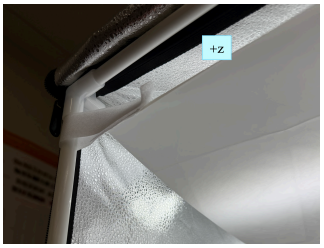
Orientation convention We define the box coordinate system as follows (see Fig. A.3):

- $-y$: open face of the box (with full-length zippers). When unzipped, the box tarp should open upwards. The SO-101 arm is mounted near this face, pointing *into* the box ($+y$ direction).
- $+z$: top face (round hole for the top camera);
- $-z$: bottom face (workspace where objects are placed); SO-101 arm mounted on this face.
- $\pm x$: side faces (removable Velcro panels).
  - During data capture and policy evaluation, the top camera is zoomed in by 1.5x and cropped to 680×480p. so that it cannot view the panels if they are removed (The top camera view that is input to the policy is the same as in Appendix E). Additionally, the box is lit up well enough so that the outside of the panels is completely dark from the camera exposure difference.
  - In our methods, we keep the $-x$ face on the box and remove the $+x$ face Velcro panel in order to switch out objects during data capture and policy evaluation.
- The **front-view camera** is mounted on the internal edge where the $+z$ and $+y$ faces intersect (see Fig. A.1).

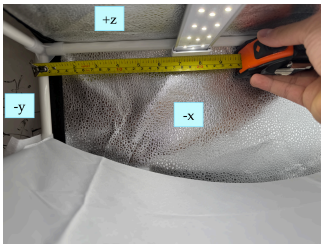
Assembly steps.

1. Construct the cube-shaped **PVC frame** using the 12 pipes and 8 edge connectors supplied with the Glendan kit. **Do not attach the zipper tarp yet**; complete all internal installations first.
2. **Attach the white light diffuser sheet** to the top ($+z$) face of the frame using the supplied velcro strips. Secure the sheet using the velcro strips on both the $-y$ and $+y$ pipes, as close as possible to the $+z$ face.
   (Fig. A.4(a)).

3. **Attach the provided white hooks onto the LED panels** and then **mount the three LED panels** on the PVC frame before fitting the tarp (Fig. A.4(b)):
   (a) *LED Panel 1*: $+z$ face, pointed downward toward $-z$; center $\approx$ 7.5 inches from the $-y$ face.
   (b) *LED Panel 2*: $+z$ face, pointed downward toward $-z$; center $\approx$ 7.5 inches from the $+y$ face (mirror image of Strip 1).
   (c) *LED Panel 3*: $+y$ face, pointed toward $-y$; center $\approx$ 8 inches from the $+z$ face.
   (d) *Note*: Connect the power cables of each LED panel to the LED power supply after putting the tarp on the box frame in the next step. Feed cables through the small hole in the tarp near the top of the $-x$ or $+x$ faces.
4. **Mount the front-view camera** on the PVC frame:
   (a) Connect the USB-C cable to the RealSense camera. (Note: Feed the computer-side of the cable through the small hole in the tarp near the top of the $-x$ or $+x$ faces after the tarp is placed over the frame in the next step).
   (b) Snap the camera mount hooks onto the PVC pipe at the junction of the $+z$ and $+y$ inner faces.
   (c) Position the camera's midpoint $\approx$ 17.5 inches from the $-x$ face. The camera angle should be approximately $-50°$ to $-60°$ from horizontal; fine calibration is performed in Section A.7.
5. Slide the **zipper tarp** over the completed PVC/LED frame, ensuring that the $-y$ face of the frame matches the side of the tarp with the zippers. Ensure that the $-z$ side (the object workspace) is actually on the bottom.
6. **Attach the white PP background sheet** to the inner $+y$ face using the velcro strips on the tarp and the sheet (Fig. A.4(c)):
   (a) Tuck the sheet *under* the $-z$ PVC pipes so that the workspace is as flat as possible. Fold the sides of the sheet under the pipes if necessary to flatten the $-z$ workspace.
   (b) If the sheet bunches near the $-y$ face, cut a small triangular slit near the $-y$ edge to relieve the tension.
   (c) The sheet should extend $\approx$ 3.5 inches beyond the $-y$ open face of the box.
   (d) Clamp the overhanging portion of the sheet to the table on both sides.



(a) Diffuser sheet



(b) LED panel positions.

(c) Background sheet.

Figure A.4: **Light box assembly.**

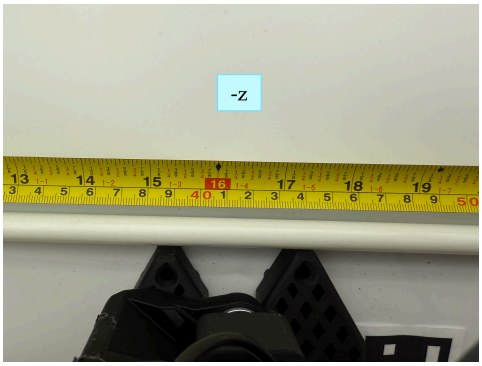
Important Checklist:

- ☐ LED panels are positioned as specified; all power connections are secure; the lights can be turned on to the maximum brightness (around 5600K).
- ☐ The diffuser sheet is taut with no visible sag.
- ☐ The background sheet is flat against the workspace; no curves or wrinkles are visible from the camera view.
- ☐ The zipper tarp fully covers all PVC pipes and LED wiring.

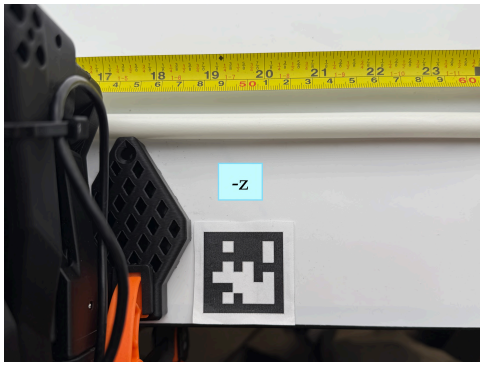
A.4 Mount the SO-101 Follower Arm and Place AprilTag

We use a mixture of AprilTag camera calibration and image overlay matching to ensure consistent camera viewing angles the top view camera (Fig. A.5).

1. Clamp both sides of the **SO-101 follower arm** to the edge of the table so that the front edge of its base touches the PVC pipe running between the $-z$ and $-y$ faces of the box. The center of the SO-101 base should be ≈ 16.5 inches from the $-x$ face (Fig. A.5(a)).
2. Attach the **12 V power adaptor** to the SO-101 arm.
3. Place the **4 cm AprilTag** on the *right* side of the SO-101 base (Fig. A.5(b)):
   (a) The *northwest corner* of the tag must touch the vertical edge of the SO-101 base.
   (b) The *south black border* of the tag must be aligned with the bottom edge of the SO-101 base.
4. Double-check the tag orientation. Wrinkled paper causes unreliable detection; affix it flat using double-sided tape on all four corners.



(a) SO-101 arm clamped to table. Center of base is 16.5 inch from the $-x$ face.



(b) AprilTag aligned with the base edges.

Figure A.5: **SO-101 arm placement and AprilTag positioning.**

Important Checklist:

- ☐ The SO-101 base center is 16.5 inch from the $-x$ box face.
- ☐ The base front edge is flush against the $-z/-y$ PVC pipe.
- ☐ The AprilTag northwest corner touches the vertical edge of the base; the south border aligns with the base bottom edge.
- ☐ The tag is flat with no wrinkles or lifted corners.

A.5 Install Software

Before proceeding to calibration, ensure the following software is installed on the host computer:

1. Clone the benchmark repository, create a new Conda environment, and install dependencies:

```
git clone https://github.com/IRVLUTD/VLAReplica.git
cd VLAReplica
conda env create -f environment.yml
conda activate vlareplica
```

2. Find available cameras indices with the command (note down the numbers):

```
lerobot-find-cameras
```

Record the camera indices for the two cameras.

3. Find USB device serial ports from the following command:

   `lerobot-find-port`

   Then unplug the SO-101 USB cable from the computer, and press `Enter`. The terminal will output something like `/dev/ttyACM1`. Record the serial port for the follwoer arm.

A.6 Calibrate the SO-101 arm

Next, calibrate the SO-101 follower according to the LeRobot Docs (`https://huggingface.co/docs/lerobot/so101?setup_motors=Command#calibrate`). **Follow the video carefully, and ensure each motor is at the middle position before starting the calibration process.** During calibration, thoroughly rotate each of the six motors to their physical joint limits.

(If you can't find the USB port address, use the command `lerobot-find-port`)

1. Locate the calibration file that LeRobot saved to your device. It should be under:

   `~/.cache/huggingface/lerobot/calibration/robots/<your-robot-id>`

   in your root folder.
2. Copy this .json file to: `VLAReplica/calibration/robots/so101_follower`
3. And rename that file to: `so101_follower_arm.json`

A.7 Camera Calibration

We provide a calibration script that detects the AprilTag and reports the camera pose in real time, allowing fine adjustment of the camera mount before locking it in place. The target pose values are listed in Table A.2.

Table A.2: Target front-view camera pose relative to the AprilTag. Adjust until all values match within the specified tolerance.

x (m)	y (m)	z (m)	R (deg)	P (deg)	Y (deg)
-0.06 ± 0.01	-0.39 ± 0.01	1.25 ± 0.01	-18.5 ± 1.0	3.0 ± 1.0	2.5 ± 1.0

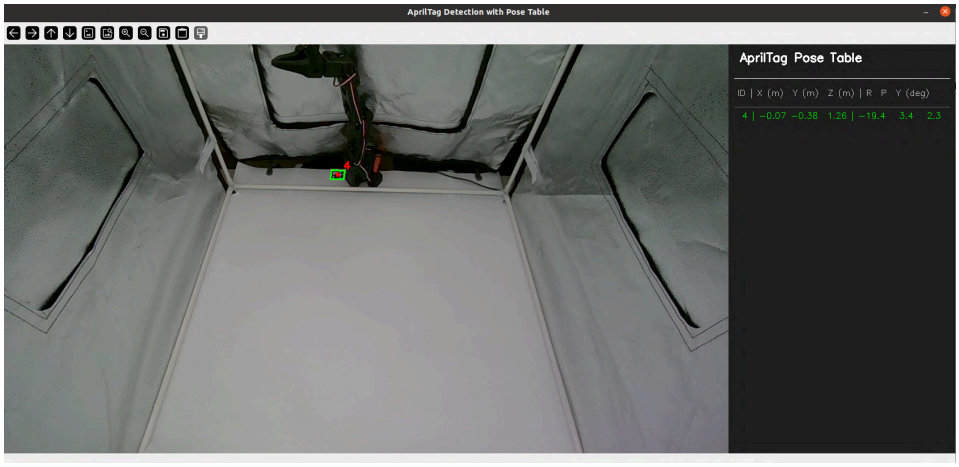


Figure A.6: **AprilTag camera calibration GUI.** The live camera feed (*left*) and the detected AprilTag pose table (*right*) are shown simultaneously. Adjust the camera position until the pose values match Table A.2.

1. In a new terminal inside the virtual environment, run the calibration script (replace `<your-top-camera-index>` with the number you recorded in Appendix A.5):

   ```
   python calibration/camera/detect_apriltag.py --camera-index
   <your-top-camera-index>
   ```

2. A GUI window will display the live camera feed alongside the estimated AprilTag pose (Fig. A.6). Reach into the box and physically slide or tilt the camera mount along the PVC pipe until all reported values match Table A.2 as closely as possible.
3. Some error is acceptable (see Table A.2). Once satisfied, Press `q` to exit the program.
4. Although the AprilTag pose estimator may output values close to Table A.2, there may still be slight camera misalignment. To solve this, we utilize *visual overlay matching* (see Fig. A.7) to ensure the camera view is as close as possible to *VLA-REPLICA's* original view.
   (a) First, calibrate the top camera for the second time. Run the following, replacing `your-top-camera-id` with the number you recorded in Appendix A.5:

   ```
   python calibration/camera/overlay.py --overlay-image-folder
   calibration/camera/referenceImages/top --base-cam
   <your-top-camera-id>
   ```

   (b) Use the provided GUI to match the view of your camera with the reference image by reaching into the box and sliding or tilting the camera mount along the PVC pipe.
   (c) Next, calibrate the wrist camera for the first time. Run the following, replacing `your-wrist-camera-id` with the number you recorded in Appendix A.5:

   ```
   python calibration/camera/overlay.py --overlay-image-folder
   calibration/camera/referenceImages/wrist --base-cam
   <your-wrist-camera-id>
   ```

   (d) Slightly loosen the M3 screw on the wrist camera mount on the SO-101, and use the provided GUI to match the view of your camera with the reference image by rotating the wrist camera on the end-effector.

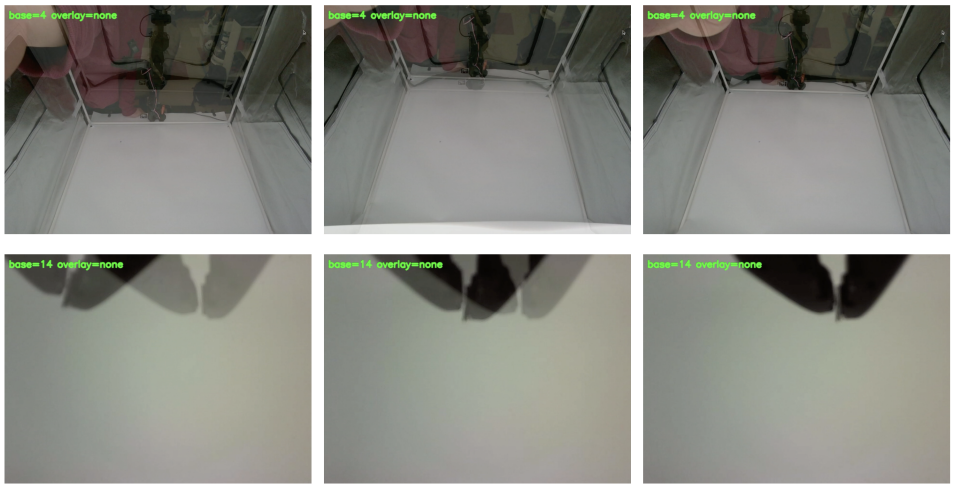


Figure A.7: **Visual calibration GUI.** Top camera (*top*) and wrist camera (*bottom*) calibration over time. The cameras are adjusted physically until the overlay match the reference image.

Important Checklist:

- ☐ All six pose values (x, y, z, R, P, Y) match the targets in Table A.2.
- ☐ Reference images and camera views match almost identically for both top and wrist cameras.

The environment setup is complete.

A.8 Policy evaluation script `benchmark.py`

Table A.3: Command-line flags for `benchmark.py`.

Flag	Description
`--policy-type <model>`	Selects the policy family to evaluate. Currently supported models: `{act,smolvla,dit,xvla,pi0,pi05}`
`--policy-path <path>`	Hugging Face repo ID or local path for the policy checkpoint.
`--policy-from-hub`	If called, loads policy from Hugging Face Hub instead of local directory.
`--run-all-tasks`	Runs evaluation across all tasks from task config, instead of single task.
`--task-subset <ID or OOD>`	When using `--run-all-tasks`, restricts evaluation to ID or OOD task subset.
`--iterations <number>`	Number of evaluation iterations per task.
`--eval-follower-calib-dirs <path>`	Follower calibration directory. (default: `calibration/robots/so101_follower`).
`--eval-follower-ports <serial port>`	Serial port for the follower robot.
`--eval-follower-ids <id>`	Robot ID for the follower arm. (default: `so101_follower_arm`
`--eval-top-indexes <index>`	Top-camera index for the active arm.
`--eval-wrist-indexes <index>`	Wrist-camera index for the active arm.
`--reset-mode fixed`	Uses a fixed reset action instead of teleoperated leader reset.
`--reset-action-file <path>`	JSON file containing the normalized reset action vector required when `--reset-mode fixed` is used. (default: `arm_reset.json`)

Now you are ready to start policy evaluations. The python script `benchmark.py` allows model evaluations, with an assortment of CLI flags (Table A.3). Example command:

```
python benchmark.py \
  --policy-type pi0 \
  --policy-path lerobot/pi0_base \
  --policy-from-hub \
  --run-all-tasks \
  --task-subset ID \
  --iterations 5 \
  --eval-follower-calib-dirs calibration/robots/so101_follower \
  --eval-follower-ports /dev/ttyACM1 \
  --eval-follower-ids so101_follower_arm \
  --eval-top-indexes 4 \
  --eval-wrist-indexes 14 \
  --reset-mode fixed \
  --reset-action-file arm_reset.json
```

After the script loads the corresponding policy and connects successfully to the followers, the follower arm will move to a start position (predefined in `arm_reset.json`). An openCV GUI will pop up, overlaying the live video feed from the top camera with the proper test scene (i.e. object placements) for that task (Fig. A.8).

When the live video feed and the overlay image match almost exactly, press `enter` on the keyboard to start the policy inference. The policy is given 90 seconds to complete the task before the iteration ends. If the policy completes the task before the 90 seconds, press `right arrow` to skip to the setup phase of the next iteration. The SO-101 arm will reset back to the start position. Evaluate each rollout as **success** or **failure**. Refer to success criterion in Appendix C.

Important Checklist:

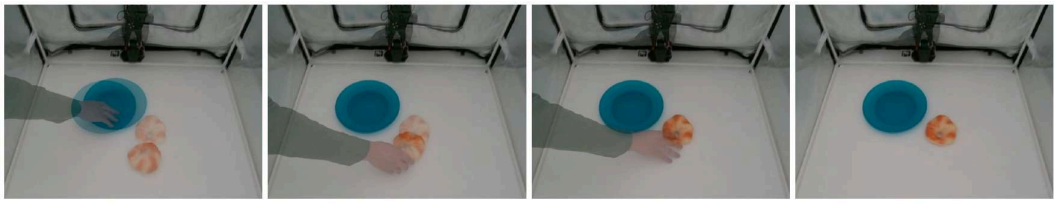


Figure A.8: `benchmark.py` **live video evaluation GUI.** The user is currently setting up the scene for the `Put bread on plate` task.

- ☐ Ensure SO-101 arm works properly.
- ☐ Run a policy and ensure proper behavior.

B Dataset Objects

Table B.1: Bill of Materials for the objects used in policy evaluation.

Object	**Price (USD)**	**Variations of object used**	**Link**
Button	9.99	N/A	Amazon
Pen cup	5.99	N/A	Amazon
Colored towels	7.49	Pink, yellow, blue	Amazon
Oven	34.98	N/A	Amazon
Mug coasters	13.98	Green, orange, purple, yellow	Amazon
Colored bowls	12.99	Red, blue, yellow, green	Amazon
White board & marker	9.75	N/A	Amazon
Toy fruit set	15.99	Apple, yellow pear	Amazon
Colored plates	19.99	Red, blue	Amazon
Spoon & fork set	7.99	N/A	Amazon
Toy bread set	27.99	Croissant (A), small bun (B), donut (C)	Amazon
Tissue boxes	7.49	Green box	Amazon
Black pepper	2.49	N/A	Amazon
Colored pencil box	14.99	Blue, yellow, pink	Amazon
Colored blocks	16.99	Red, blue, yellow, green, orange, purple	Amazon
Whiteboard eraser	3.12	N/A	Amazon
Total:	**212.21**		

C Task Variant List & Success Criterion

Each task has 5 in-distribution (ID) and 5 out-of-distribution (OOD) variants (except Task 5 & 6, which do not have OOD tasks). All 90 tasks variants are listed below. *Level difficulty* refers to the number of distractor objects in frame, and the target objects' positional deviations from the training set (Appendix D). The success criteria for each task (i.e. what counts as success) is also listed.

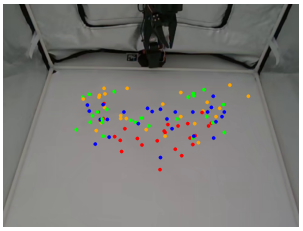
Table C.1: In-Distribution (ID) and Out-of-Distribution (OOD) Task Variants

ID#	Level	Task Variant	Dist.	Success Criterion
Task 1: Put Bread on Plate				
1	Easy	Put A bread on the red plate	ID	Bread is resting on the correct colored plate, and SO-101 moved back to its home position.
2	Easy	Put B bread on the blue plate	ID	
3	Medium	Put A bread on the blue plate	ID	
4	Medium	Put B bread on the red plate (1 distractor)	ID	
5	Hard	Put A bread on the red plate (2 distractors)	ID	
6	Easy	Put C bread on the red plate	OOD	
7	Easy	Put C bread on the blue plate	OOD	
8	Medium	Put A bread on the yellow plate	OOD	
9	Medium	Put B bread on the yellow plate	OOD	
10	Hard	Put C bread on the yellow plate	OOD	
Task 2: Put Bowl on Coaster				
11	Easy	Put red bowl on green coaster	ID	Correct bowl is touching/resting on correct colored coaster, and SO-101 moved back to its home position.
12	Easy	Put blue bowl on orange coaster	ID	
13	Medium	Put blue bowl on green coaster	ID	
14	Medium	Put red bowl on purple coaster (1 distractor)	ID	
15	Hard	Put yellow bowl on purple coaster (2 distractors)	ID	
16	Easy	Put blue bowl on purple coaster	OOD	
17	Easy	Put red bowl on orange coaster	OOD	
18	Medium	Put yellow bowl on green coaster	OOD	
19	Medium	Put yellow bowl on orange coaster	OOD	
20	Hard	Put green bowl on yellow coaster	OOD	
Task 3: Stack Blocks				
21	Easy	Stack red block on blue block	ID	Correctly colored top block has touched/rested on correctly colored bottom block for more than 2 seconds.
22	Easy	Stack yellow block on blue block	ID	
23	Medium	Stack blue block on yellow block	ID	
24	Medium	Stack blue block on red block (1 distractor)	ID	
25	Hard	Stack red block on yellow block (2 distractors)	ID	
26	Easy	Stack yellow block on red block	OOD	
27	Easy	Stack blue block on blue block	OOD	
28	Medium	Stack red block on green block	OOD	
29	Medium	Stack green block on yellow block	OOD	
30	Hard	Stack green block on green block	OOD	
Task 4: Fold Towel				
31	Easy	Fold pink towel in half	ID	Correctly colored towel's edges are lifted and folded on itself by more than 50%, and SO-101 moved back to its home position.
32	Easy	Fold yellow towel in half	ID	
33	Medium	Fold pink towel in half (1 distractor)	ID	
34	Medium	Fold yellow towel in half (1 distractor)	ID	
35	Hard	Fold pink towel in half (2 distractors)	ID	
36	Easy	Fold blue towel in half	OOD	
37	Easy	Fold blue towel in half	OOD	
38	Medium	Fold blue towel in half (1 distractor)	OOD	
39	Medium	Fold blue towel in half (2 distractors)	OOD	
40	Hard	Fold blue towel in half (2 distractors)	OOD	
Task 5: Open Oven (No OOD Variants)				
41	Easy	Open the oven	ID	

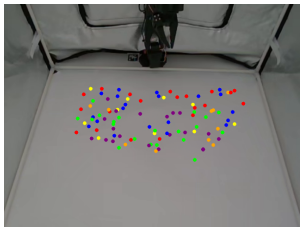
ID#	Level	Task Description	Dist.	Success Criterion
42	Easy	Open the oven	ID	Oven door opened for 2+ seconds, SO-101 back to home position.
43	Medium	Open the oven (1 distractor)	ID	
44	Medium	Open the oven (2 distractors)	ID	
45	Hard	Open the oven (2 distractors)	ID	
Task 6: Clean Whiteboard (No OOD Variants)				
46	Easy	Clean whiteboard with eraser	ID	Eraser wiped whiteboard 2 or more times, and then placed next to whiteboard.
47	Easy	Clean whiteboard with eraser	ID	
48	Medium	Clean whiteboard with eraser (1 distractor)	ID	
49	Medium	Clean whiteboard with eraser (2 distractors)	ID	
50	Hard	Clean whiteboard with eraser (2 distractors)	ID	
Task 7: Pour Pepper				
51	Easy	Pour 1 shake of pepper into the red plate	ID	The exact number of pepper shakes is poured into the correct colored plate, the pepper is placed to the left of the plate, and SO-101 moved back to its home position.
52	Easy	Pour 3 shakes of pepper into the red plate	ID	
53	Medium	Pour 2 shakes of pepper (1 distractor)	ID	
54	Medium	Pour 1 shake of pepper (1 distractor)	ID	
55	Hard	Pour 3 shakes of pepper (2 distractors)	ID	
56	Easy	Pour 4 shakes of pepper into red plate	OOD	
57	Easy	Pour 5 shakes of pepper into red plate	OOD	
58	Medium	Pour 1 shake of pepper into blue plate	OOD	
59	Medium	Pour 3 shakes of pepper into blue plate	OOD	
60	Hard	Pour 5 shakes of pepper into blue plate	OOD	
Task 8: Lift Bowl				
61	Easy	Lift green bowl one time	ID	Correct colored bowl lifted up and down the correct number of times, and SO-101 moved back to its home position.
62	Easy	Lift blue bowl one time	ID	
63	Medium	Lift red bowl three times	ID	
64	Medium	Lift green bowl three times (1 distractor)	ID	
65	Hard	Lift blue bowl three times (2 distractors)	ID	
66	Easy	Lift green bowl two times	OOD	
67	Easy	Lift blue bowl two times	OOD	
68	Medium	Lift red bowl two times	OOD	
69	Medium	Lift yellow bowl two times	OOD	
70	Hard	Lift yellow bowl four times	OOD	
Task 9: Press Button				
71	Easy	Press button one time	ID	Button is touched the correct number of times, and SO-101 moved back to its home position.
72	Easy	Press button three times	ID	
73	Medium	Press button one time (1 distractor)	ID	
74	Medium	Press button three times (1 distractor)	ID	
75	Hard	Press button three times (2 distractors)	ID	
76	Easy	Press button two times	OOD	
77	Easy	Press button four times (1 distractor)	OOD	
78	Medium	Press button two times (2 distractors)	OOD	
79	Medium	Press button four times (2 distractors)	OOD	
80	Hard	Press button five times (2 distractors)	OOD	
Task 10: Collect Blocks				
81	Easy	Collect 2 blocks into blue box	ID	All the blocks on the workspace are placed into the correct colored box, and SO-101 moved back to its home position.
82	Easy	Collect 2 blocks into yellow box	ID	
83	Medium	Collect 3 blocks into blue box (1 distractor)	ID	
84	Medium	Collect 3 blocks into yellow box	ID	
85	Hard	Collect 4 blocks into blue box (2 distractors)	ID	
86	Easy	Collect 2 blocks into pink box	OOD	
87	Easy	Collect 3 blocks into pink box	OOD	
88	Medium	Collect 5 blocks into blue box	OOD	
89	Medium	Collect 5 blocks into yellow box	OOD	
90	Hard	Collect 5 blocks into pink box	OOD	

D Demonstration Dataset Object Placements

For all 500 expert demonstrations of the 10 tasks, we manually segment the target object(s) and record the position of each target object in its initial position by calculating the center of its bounding box (represented by a colored dot). We superimpose all the dots together by task to create 10 *placement images*, which we utilized to aid us in creating the test scene images (see Appendix E). This way, we could ensure that no initial object positions in the test scenes were too close to the initial object positions in the expert demonstration dataset. We provide a key next to each placement image to help identify the objects by color.

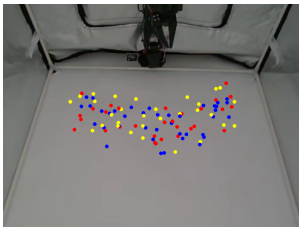


Object(s)
Red plate
Blue plate
A bread
B bread

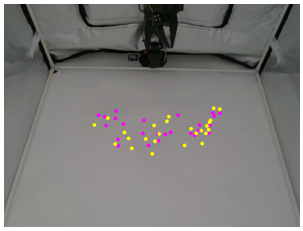


Object(s)
Red bowl
Blue bowl
Yellow bowl
Green coaster
Orange coaster
Purple coaster

Figure D.1: Tasks 1–2: object center annotations.

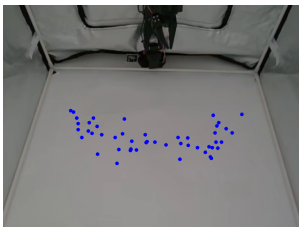


Object(s)
Red block
Blue block
Yellow block

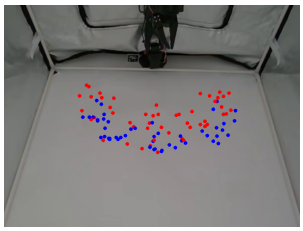


Object(s)
Pink towel
Yellow towel

Figure D.2: Tasks 3–4: object center annotations.

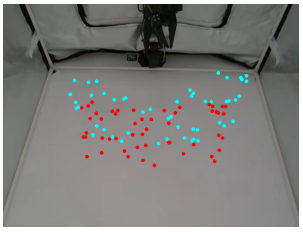


Object(s)
Oven

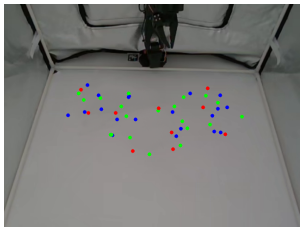


Object(s)
Eraser
Whiteboard

Figure D.3: Tasks 5–6: object center annotations.

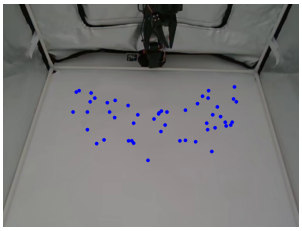


Object(s)
Red plate
Pepper

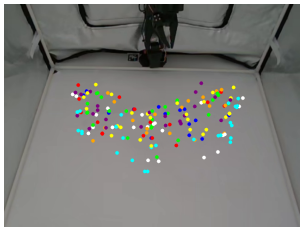


Object(s)
Red bowl
Blue bowl
Green bowl

Figure D.4: Tasks 7–8: object center annotations.



Object(s)
Button



Object(s)
Red block
Blue block
Yellow block
Green block
Orange block
Purple block
Yellow box
Blue box

Figure D.5: Tasks 9–10: object center annotations.

E Test Scene Reference Images

All 90 test scene reference images are provided below. For all tasks, the first row includes ID (in-distribution) tasks, and the second row includes OOD (out-of-distribution) tasks (except 5 & 6).

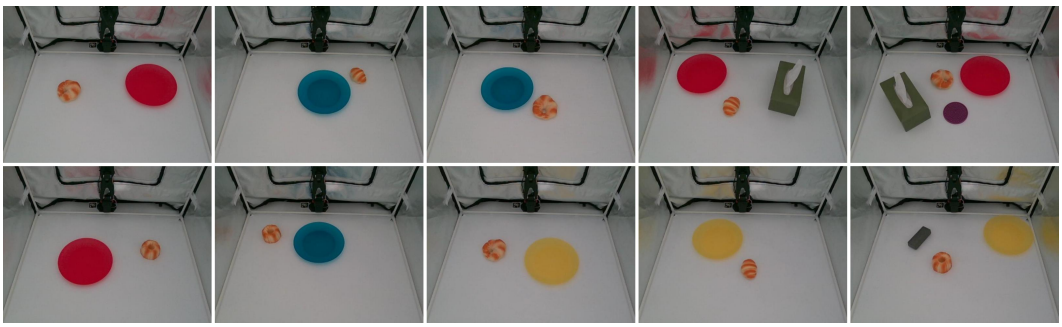


Figure E.1: Reference images for Task 1: Put Bread on Plate (ID# 1-10).

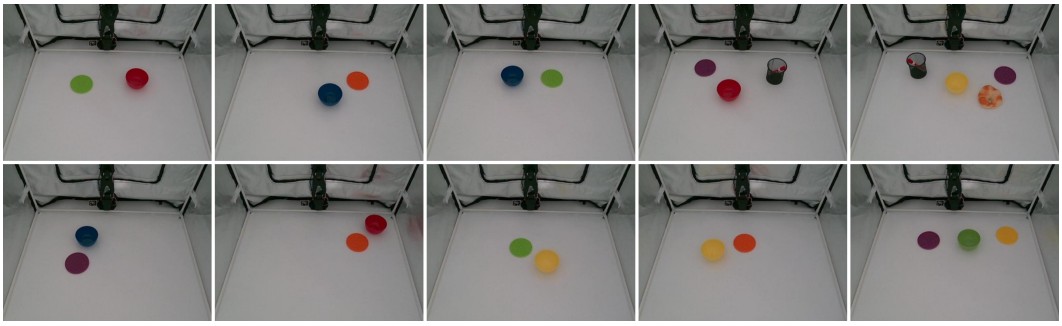


Figure E.2: Reference images for Task 2: Put Bowl on Coaster (ID# 11-20).

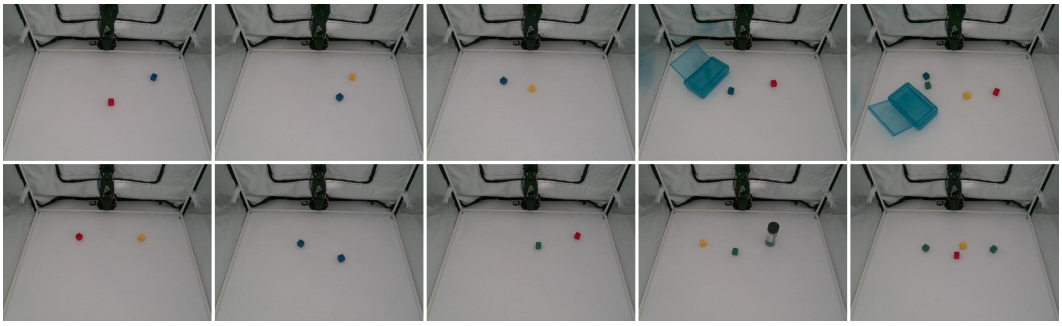


Figure E.3: Reference images for Task 3: Stack Blocks (ID# 21-30).

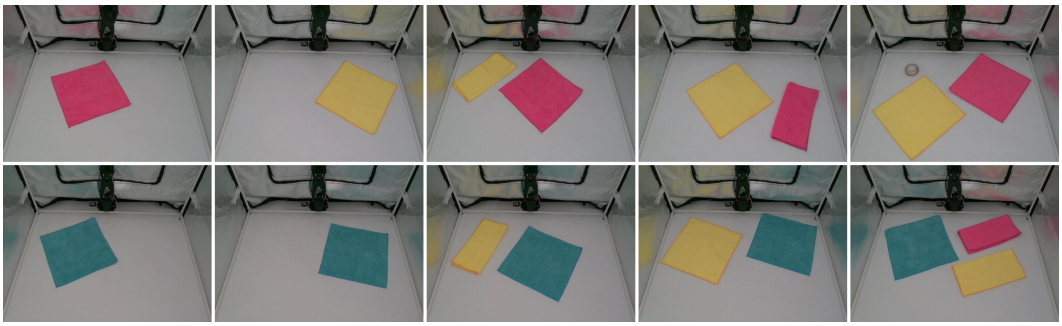


Figure E.4: Reference images for Task 4: Fold Towel (ID# 31-40).

Figure E.5: Reference images for Task 5: Open Oven (ID# 41-45).

Figure E.6: Reference images for Task 6: Clean Whiteboard (ID# 46-50).

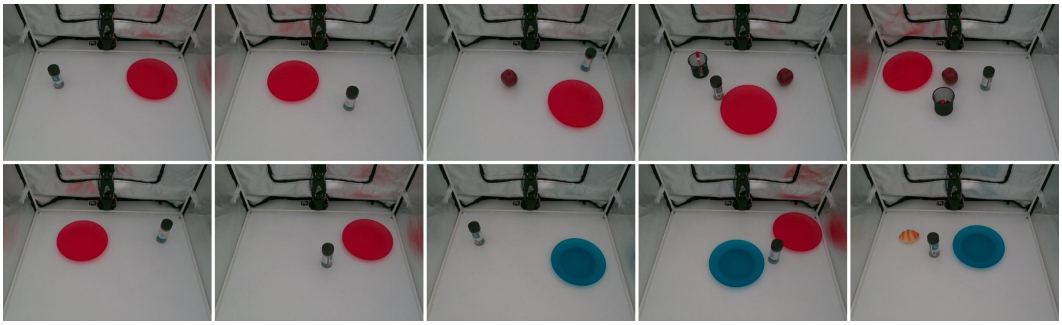


Figure E.7: Reference images for Task 7: Pour Pepper (ID# 51-60).

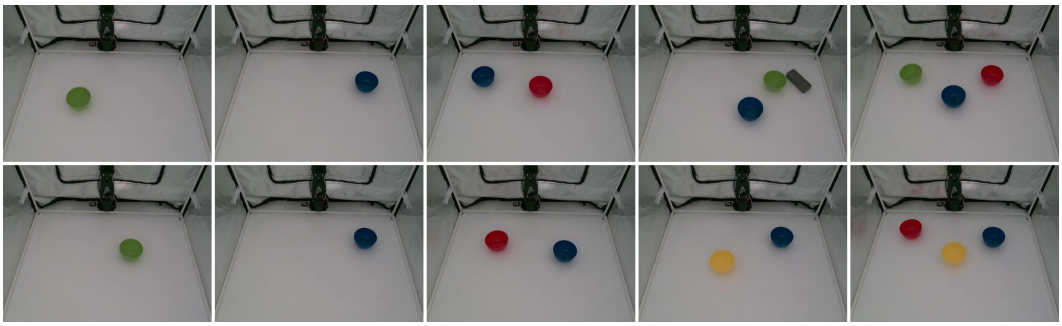


Figure E.8: Reference images for Task 8: Lift Bowl (ID# 61-70).

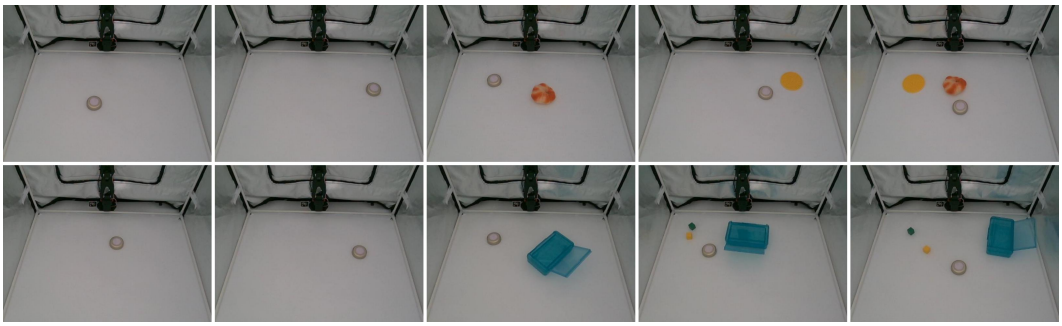


Figure E.9: Reference images for Task 9: Press Button (ID# 71-80).

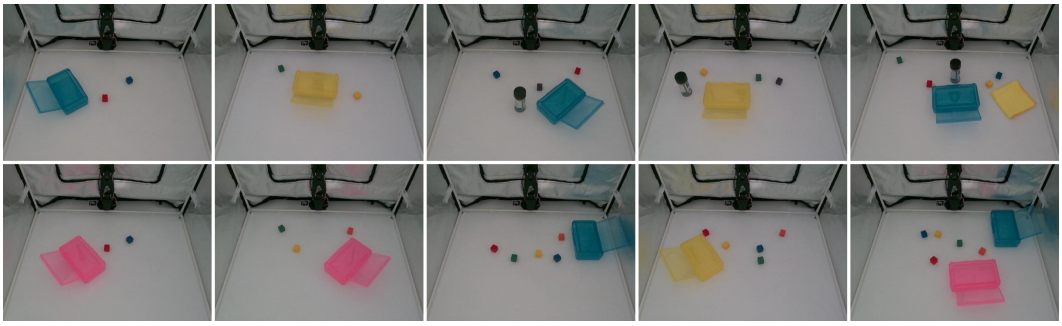


Figure E.10: Reference images for Task 10: Collect Blocks (ID# 81-90).

F Training Details

All imitation learning and VLA fine-tuning experiments are implemented using the official LeRobot codebase [7]. Detailed training configurations are reported in Table F.1. For all methods, we keep the default optimizer, learning rate, and weight decay settings from the official implementation.

Table F.1: Training and fine-tuning details for the evaluated policies.

Parameter	**ACT**	**DiT-D**	**DiT-F**	**SmolVLA**	**X-VLA**	π_0	$\pi_{0.5}$
Training type	From scratch	From scratch	From scratch	Fine-tuning	Fine-tuning	Fine-tuning	Fine-tuning
Dataset size	500 demos	500 demos	500 demos	500 demos	500 demos	500 demos	500 demos
Batch size	128	128	128	128	128	16	16
Training steps	40K	40K	40K	40K	40K	40K	40K
GPUs	2	1	1	4	4	1	1
GPU type	A6000 Ada	H200	H200	A6000 Ada	A6000 Ada	H200	H200
Action chunk size	32	32	32	32	32	32	32
Number of action steps	32	24	24	32	32	32	32
Vision encoder	Default	Default	Default	Default	Trainable	Frozen	Default
Implementation	LeRobot	LeRobot	LeRobot	LeRobot	LeRobot	LeRobot	LeRobot
Learning Rate	Default	Default	Default	Default	Default	Default	Default

G Detailed Evaluation Results

We provide success rates for every single policy rollout that was used to calculate Tables 4, 5, and 6. We denote a successful trial for that task variant with a 1, and a failed trial with a 0.

Including setup and test runs, we evaluated over 800 policy rollout episodes at 90 seconds each, which totals to over 20 hours of inference and evaluation for 7 different models across two light-box setups. Due to time and physical constraints, we evaluate each variant for each task for each policy only once.

Videos of policy rollouts are available online at `https://irvlutd.github.io/VLAReplica/`

Table G.1: ACT — per-trial success rates across all evaluations.

(a) ID, original env. (10 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	0	1	0	0	0.40
Bowl on Coaster	0	0	0	0	0	0.00
Stack Block	0	0	0	0	0	0.00
Fold Towel	1	1	0	0	0	0.40
Open Oven	0	0	1	0	1	0.40
Erase Whiteboard	0	0	1	0	0	0.20
Pepper N Times	0	1	0	0	0	0.20
Lift Bowl N Times	0	1	0	0	0	0.20
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.18**

(b) OOD, original env. (8 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	0	0	0	1	0.40
Bowl on Coaster	1	0	0	0	0	0.20
Stack Block	0	0	0	0	0	0.00
Fold Towel	0	0	0	0	0	0.00
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.08**

(c) ID, reproduced env. (5 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	0	0	0	1	0.40
Bowl on Coaster	0	0	0	0	0	0.00
Fold Towel	1	1	1	0	0	0.60
Open Oven	0	0	1	0	1	0.40
Erase Whiteboard	0	0	1	0	0	0.20
Overall						**0.32**

(d) OOD, reproduced env. (3 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	0	1	0	0	0	0.20
Bowl on Coaster	1	0	0	0	1	0.40
Fold Towel	0	0	0	0	0	0.00
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Overall						**0.20**

Table G.2: π_0 — per-trial success rates across all evaluations.

(a) ID, original env. (10 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	1	1	0	0.80
Bowl on Coaster	0	1	1	1	0	0.60
Stack Block	0	0	0	0	0	0.00
Fold Towel	1	1	1	1	0	0.80
Open Oven	0	1	0	0	0	0.20
Erase Whiteboard	1	1	0	0	0	0.40
Pepper N Times	1	0	0	0	0	0.20
Lift Bowl N Times	0	0	1	0	0	0.20
Press Button N Times	0	1	0	0	0	0.20
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.34**

(b) OOD, original env. (8 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	0	1	1	1	1	0.80
Bowl on Coaster	1	1	0	0	1	0.60
Stack Block	0	1	0	0	0	0.20
Fold Towel	1	1	0	1	0	0.60
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Pepper N Times	0	0	1	0	0	0.20
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.30**

(c) ID, reproduced env. (5 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	0	1	1	0.80
Bowl on Coaster	1	0	1	1	0	0.60
Fold Towel	1	1	1	0	0	0.60
Open Oven	0	0	0	0	1	0.20
Erase Whiteboard	0	1	0	0	0	0.20
Overall						**0.48**

(d) OOD, reproduced env. (3 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	0	1	1	0.80
Bowl on Coaster	0	1	0	0	1	0.40
Fold Towel	0	0	1	1	1	0.60
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Overall						**0.60**

Table G.3: $\pi_{0.5}$ — per-trial success rates across all evaluations.

(a) ID, original env. (10 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	0	1	1	0.80
Bowl on Coaster	1	1	1	1	0	0.80
Stack Block	1	0	0	1	0	0.40
Fold Towel	1	1	1	1	1	1.00
Open Oven	0	1	0	1	1	0.60
Erase Whiteboard	1	1	0	0	0	0.40
Pepper N Times	0	0	0	1	1	0.40
Lift Bowl N Times	0	1	0	0	1	0.40
Press Button N Times	1	0	0	0	0	0.20
N Blocks into Box	1	0	1	0	0	0.40
Overall						**0.54**

(b) OOD, original env. (8 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	1	1	1	1.00
Bowl on Coaster	1	0	1	0	0	0.40
Stack Block	0	0	0	0	0	0.00
Fold Towel	1	1	1	1	0	0.80
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Pepper N Times	0	0	1	1	0	0.40
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	1	0	0	0	0	0.20
Overall						**0.35**

(c) ID, reproduced env. (5 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	0	1	1	1	1	0.80
Bowl on Coaster	1	1	1	1	1	1.00
Fold Towel	1	1	0	1	0	0.60
Open Oven	0	0	1	1	1	0.60
Erase Whiteboard	1	1	0	0	0	0.40
Overall						**0.68**

(d) OOD, reproduced env. (3 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	0	1	1	0.80
Bowl on Coaster	0	0	1	1	0	0.40
Fold Towel	1	1	1	1	0	0.80
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Overall						**0.67**

Table G.4: SmolVLA — per-trial success rates across all evaluations.

(a) ID, original env. (10 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	1	0	0	0.60
Bowl on Coaster	0	0	1	0	0	0.20
Stack Block	1	0	0	0	0	0.20
Fold Towel	1	1	1	0	0	0.60
Open Oven	1	0	1	0	0	0.40
Erase Whiteboard	0	1	0	0	0	0.20
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	0	0	0	1	0	0.20
Press Button N Times	0	1	0	0	0	0.20
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.26**

(b) OOD, original env. (8 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	1	0	1	0.80
Bowl on Coaster	1	0	1	0	0	0.40
Stack Block	0	1	0	0	0	0.20
Fold Towel	1	1	0	1	0	0.60
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	1	0	0	0	0	0.20
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	1	0	0	0	0	0.20
Overall						**0.30**

(c) ID, reproduced env. (5 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	0	1	1	0	0	0.40
Bowl on Coaster	0	1	1	0	0	0.40
Fold Towel	1	1	1	0	1	0.80
Open Oven	0	0	1	0	1	0.40
Erase Whiteboard	0	1	0	0	0	0.20
Overall						**0.44**

(d) OOD, reproduced env. (3 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	0	1	1	0	1	0.60
Bowl on Coaster	1	0	0	1	0	0.40
Fold Towel	1	0	1	1	0	0.60
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Overall						**0.53**

Table G.5: DiT-Multitask — per-trial success rates, original environment.

(a) ID, original env. (10 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	0	0	1	0	0.40
Bowl on Coaster	0	0	0	0	0	0.00
Stack Block	0	0	0	0	0	0.00
Fold Towel	0	1	0	0	0	0.20
Open Oven	1	0	0	1	1	0.60
Erase Whiteboard	1	0	0	0	0	0.20
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	1	0	0	0	0.20
Overall						**0.16**

(b) OOD, original env. (8 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	0	0	0	0	0	0.00
Bowl on Coaster	1	0	0	0	0	0.20
Stack Block	0	0	0	–	0	0.00
Fold Towel	0	0	1	0	0	0.20
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.05**

Table G.6: DiT-FlowMatching — per-trial success rates, original environment.

(a) ID, original env. (10 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	0	1	0	0	0.40
Bowl on Coaster	0	0	0	0	0	0.00
Stack Block	0	0	0	0	0	0.00
Fold Towel	0	0	1	0	0	0.20
Open Oven	1	0	1	0	0	0.40
Erase Whiteboard	0	1	0	0	0	0.20
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.12**

(b) OOD, original env. (8 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	0	0	0	1	0	0.20
Bowl on Coaster	0	0	0	0	0	0.00
Stack Block	0	0	0	0	0	0.00
Fold Towel	0	0	0	0	0	0.00
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.03**

Table G.7: X-VLA — per-trial success rates, original environment.

(a) ID, original env. (10 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	0	0	0	0.40
Bowl on Coaster	1	0	0	0	0	0.20
Stack Block	0	0	0	0	0	0.00
Fold Towel	1	0	1	1	0	0.60
Open Oven	0	0	0	0	0	0.00
Erase Whiteboard	0	0	0	0	0	0.00
Pepper N Times	0	1	0	0	0	0.20
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.14**

(b) OOD, original env. (8 tasks)

Task	#1	#2	#3	#4	#5	Avg.
Bread on Plate	1	1	1	0	0	0.60
Bowl on Coaster	0	0	0	0	0	0.00
Stack Block	0	0	0	0	0	0.00
Fold Towel	0	0	0	0	0	0.00
Open Oven	–	–	–	–	–	–
Erase Whiteboard	–	–	–	–	–	–
Pepper N Times	0	0	0	0	0	0.00
Lift Bowl N Times	0	0	0	0	0	0.00
Press Button N Times	0	0	0	0	0	0.00
N Blocks into Box	0	0	0	0	0	0.00
Overall						**0.07**

H License

Our dataset is released under the Creative Commons Attribution 4.0 International (CC BY 4.0) license, which permits use, sharing, adaptation, and redistribution with appropriate attribution.

The SO-101 arm design and associated resources are released under the Apache License 2.0, which permits use, modification, and distribution, including for academic research purposes.