ARM-WISE COMPOSITIONAL GENERALIZATION IN DUAL-ARM VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalization in multi-arm collaboration can be studied as composing familiar atomic skills in new ways across arms. However, existing evaluations offer limited insight into which training and architectural choices support this ability under different coordination requirements. We introduce **ACG-Bench**, a benchmark for *Arm-wise Compositional Generalization* that provides a common testbed for studying skill recomposition in dual-arm policies. It contains 23 task–condition pairs across 8 task families, with 6 in-domain conditions and 17 unseen compositions covering reordering, synchronization, their combination, and cross-task composition. All methods receive the same per-arm atomic prompts, and success requires achieving the task goal while satisfying physical milestones and specified order or timing constraints. Using $\pi_{0.5}$ as a common vision-language-action backbone, we compare representative data-augmentation and architectural strategies with shared source data and a common evaluation protocol. Our architectural study examines arm-token grouping, skill-specific LoRA adapters (SkillLoRA), and arm-wise attention (AWA), highlighting the complementarity of skill-conditioned parameters and attention structure. Combining these choices yields **AE-VLA**, which achieves 21.53% generalization success in simulation, compared with 2.94% for Single $\pi_{0.5}$, 3.06% for MA-VLA, and 5.53% for two independently controlled $\pi_{0.5}$ policies. On physical SO101 robots, AE-VLA reaches 39.00% mean success across five unseen conditions, compared with 10.00% for the strongest baseline. These findings provide empirical guidance for designing dual-arm policies that generalize beyond fixed training routines.

1 INTRODUCTION

Dual-arm vision-language-action (VLA) models have made progress in learning manipulation tasks (Black et al., 2024; Intelligence et al., 2025a), but executing familiar tasks does not establish whether their component skills can be recombined. The same placing skills may need to run in a different order or synchronize their completion, while skills learned in separate tasks may need

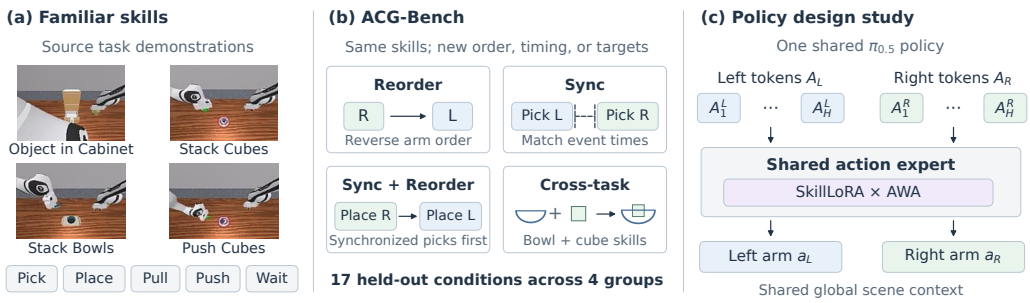


Figure 1: **ACG-Bench and the design study.** We test familiar skills under new order, timing, and cross-task requirements. Using one shared $\pi_{0.5}$ policy, we compare token grouping, skill-specific adapters, and arm-wise attention.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to work together. We call this ability *Arm-wise Compositional Generalization* (ACG), ie., reusing familiar arm-level skills under training-time unseen coordination requirements. Such reuse offers a path beyond fixed collaboration routines without collecting demonstrations for every possible combination.

Existing benchmarks study skill transfer and composition through language grounding, skill chaining, and atomic-to-composite manipulation (Mees et al., 2022; Jiang et al., 2023; Liu et al., 2023; Haresh et al., 2024; Wu et al., 2026), whereas dual-arm benchmarks mainly evaluate coordination patterns seen during training (Mu et al., 2025; Chen et al., 2025; 2026a). Recent methods explore Arm Shuffle in MA-VLA (Zhang et al., 2026) and arm-specific representations in TwinVLA (Im et al., 2026) and SkillVLA (Zhai et al., 2026). However, differences in backbones, datasets, and evaluation settings make their effects on compositional generalization difficult to isolate. This motivates a controlled study: *which training and architectural choices help familiar arm-level skills generalize to unseen compositions?*

We introduce **ACG-Bench**, comprising 23 task–condition pairs across 8 task families: 6 in-domain conditions and 17 unseen compositions covering **reordering**, **synchronization**, **Sync+Reorder**, and **cross-task composition**. All methods receive identical per-arm atomic prompts from a common phase scheduler, isolating execution of a supplied composition from skill planning. Success requires both the task goal and compliance with physical milestones and specified event order or timing, rather than merely reaching the final state.

We first compare three execution baselines. All three use the same pretrained $\pi_{0.5}$ backbone (Intelligence et al., 2025a), source data, and evaluation protocol. The first is the standard $\pi_{0.5}$, where a single shared model jointly controls both arms. The second is MA-VLA (Zhang et al., 2026), which retains this shared-model setup. It introduces Arm Shuffle during training to augment arm-role assignments. The third is an independent-control baseline, which instead deploys a separate $\pi_{0.5}$ policy for each arm. Each policy produces actions only for its assigned arm. These comparisons examine whether arm-role augmentation or independent control improves compositional execution over the standard shared policy. Despite their different designs, all three baselines show limited generalization to unseen compositions.

Starting from the standard shared-model $\pi_{0.5}$, we then investigate three targeted architectural changes: **arm-token grouping** for arm-specific representations, **skill-specific LoRA adapters (SkillLoRA)** for skill-conditioned parameters, and **arm-wise attention (AWA)** for regulating cross-arm information flow. Individual changes yield modest gains, whereas their combination, **AE-VLA**, achieves 21.53% generalization success in simulation and 39.00% mean success across five unseen conditions on physical SO101 robots, compared with the strongest baseline means of 5.53% and 10.00%, respectively. These findings suggest that modest architectural changes can substantially improve arm-wise compositional generalization within a shared policy, building on $\pi_{0.5}$'s pretrained capabilities.

Our contributions are threefold:

- **ACG-Bench:** a benchmark for systematic evaluation of dual-arm skill recomposition across changes in order, synchronization, and cross-task combinations.
- **A unified empirical study:** comparisons of representative augmentation-based and architectural strategies using a common $\pi_{0.5}$ backbone, source dataset, and execution protocol.
- **Design insights and AE-VLA:** evidence of complementary benefits from skill-conditioned parameters and arm-wise attention, validated through their combined configuration in simulation and on physical robots.

2 Related Work

Vision-language-action policies. VLA models map visual observations and language to robot actions, with recent systems improving task, scene, and embodiment transfer through large robot datasets, pretrained representations, and generative action heads Brohan et al. (2022); Zitkovich et al. (2023); Team et al. (2024); Kim et al. (2024); Black et al. (2024); Pertsch et al. (2025); Intelligence et al. (2025a); Wen et al. (2025). Several systems support bimanual control NVIDIA et al. (2025); Liu et al. (2026), and steerable policies study task composition through prompts or sub-

goals Intelligence et al. (2026). We focus on a different source of shift: the local behaviors remain familiar, but the order or timing of skills across the two arms changes. We study this shift on $\pi_{0.5}$.

Bimanual learning and coordination. Bimanual policies have been studied through joint action prediction, cross-arm attention, and explicit coordination modules Grotz et al. (2024); Chernyadev et al. (2024); Lai et al. (2025); Glossop et al. (2025). ACG asks whether the learned coordination can support relations absent from training. We test this with shared scene context and restricted attention between local arm streams.

Arm-wise representation and skill reuse. TwinVLA (Im et al., 2026) represents shared and arm-specific inputs through twin VLM streams with joint attention, sharing its visual encoder and action head. SkillVLA (Zhai et al., 2026) uses hierarchical skill decisions, per-arm action generation, and adaptive communication for cooperative behaviors. MA-VLA (Zhang et al., 2026) studies atomic instruction assignment and arm-role permutation for compositional generalization. We study the related ideas of per-arm token groups, skill-specific weights, and attention access under one training and evaluation setup. Appendix B.4 details the differences from the complete prior systems.

Manipulation benchmarks and skill composition. Existing benchmarks evaluate generalization over objects, layouts, goals, visual appearance, task sequences, and skill combinations James et al. (2020); Mees et al. (2022); Mu et al. (2021); Liu et al. (2023); Haresh et al. (2024); Zhang et al. (2025). Recent bimanual and multi-robot suites expand task and scene diversity Mu et al. (2025); Chen et al. (2025); Wang et al. (2025); Wu et al. (2025); Qin et al. (2025); Chen et al. (2026b). ACG-Bench controls the low-level skill vocabulary and changes the arm-level event graph itself. Its constraint-aware metric also rejects executions that reach the final state through the wrong order or timing, making coordination structure an explicit evaluation target.

3 Arm-wise Compositional Generalization

3.1 Problem Formulation

We study whether a dual-arm policy can execute familiar skills in combinations absent from training. A bowl routine demonstrated left-arm-first may be tested right-arm-first or with synchronized picks. Cross-task composition combines behaviors from different source tasks, such as bowl handling and cube placement.

We describe a task by an *event graph*. Each node is a skill execution with a specified arm, object, and target, such as "left arm picks bowl A." Order constraints specify which skill events must happen before others. Synchronization constraints require two designated completion milestones to occur within a task-specific time window.

A test condition evaluates ACG when its required skill types occur in the training demonstrations, but their joint event graph does not. The robot setup stays fixed. The policy must achieve the task goal while following the required order and timing. Arm-specific phase instructions supply the target structure, so the question is whether the policy can *execute* a new composition of learned skills.

3.2 ACG-Bench

ACG-Bench builds dual-arm tasks in ManiSkill Mu et al. (2021) using the task and expert-program generation pipeline of RoboTwin 2.0 Chen et al. (2025). The benchmark has six source task families: Stack Bowls, Stack Cubes, Push Cubes, Burger & Fries, Can in Basket, and Object in Cabinet. Cube in Bowl and Bowl in Cabinet are held-out cross-task families. Task programs are run from varied valid initial states and repaired after failed grasps, collisions, or failed task checks. Accepted programs collect 100 clean expert demonstrations per source task. We split them into skills such as Pick, Place, Lift, Pull, Move, Push, Recover, and Wait. State checks on grasp attachment, object location, and cabinet joints define skill completion. Object and target arguments can change without creating a new skill type. Appendix A.3 gives skill definitions, source-task coverage, and example phase prompts.

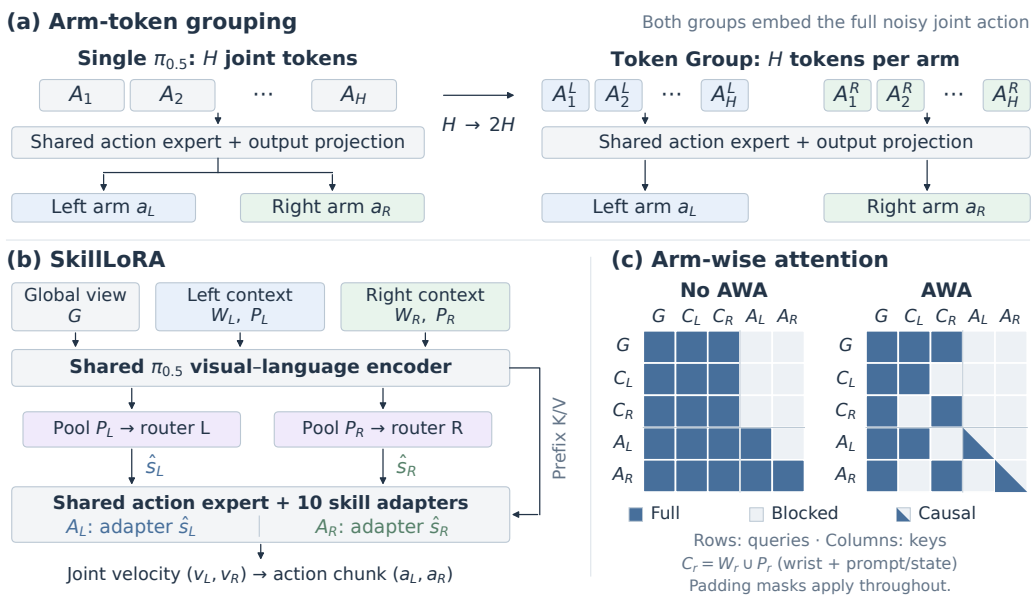


Figure 2: **Three design choices within one $\pi_{0.5}$ policy.** (a) Single uses H joint action tokens to predict both arms. Token Group uses two groups of H tokens, each assigned to one arm, within a shared action expert. (b) SkillLoRA selects a skill adapter for each group. (c) AWA controls attention between groups while retaining global context. Here $H = 50$; arrows in (a) summarize decoding and denoising.

We pool all 600 source demonstrations into one multitask training set. Each method uses the same trained policy across all 23 task–condition pairs, without task-specific fine-tuning; Dual $\pi_{0.5}$ uses a fixed pair of per-arm policies. Held-out compositions contribute no training demonstrations.

At evaluation, a common rule-based scheduler supplies per-arm atomic prompts to *every* method. Each prompt gives the active skill and its arguments. Shared state checks update prompts and record milestone completion times for scoring. The vocabulary and transition rules are identical across methods. This protocol tests execution of a supplied skill plan, not plan inference.

We construct four separate generalization groups:

- **Reorder** reverses or changes event order while retaining the task goal and its skill events.
- **Sync** requires designated events of the two arms to complete within a time window instead of following the training sequence.
- **Sync+Reorder** changes order and timing rules together.
- **Cross-task Combination** combines skills from different source task families, as in Cube in Bowl and Bowl in Cabinet.

Each condition is counted in one reporting group. Cross-task conditions keep that label even if they also change timing. The resulting common set has 23 task–condition pairs: six in-domain, four Reorder, six Sync, four Sync+Reorder, and three Cross-task Combination. The 17 non-in-domain pairs form the generalization set.

Cross-task tests also change the task context, while Sync+Reorder changes both order and timing. The groups can therefore differ in more than one aspect of difficulty. We report each group as well as the mean over all 17 generalization conditions. Table 1 lists every condition in the main text.

4 Design Choices on $\pi_{0.5}$

We study token grouping, skill-specific weights, and attention on pretrained $\pi_{0.5}$ (Intelligence et al., 2025b). Composing familiar skills under new coordination requirements calls for reusing each arm's

behavior while retaining information about the shared scene. Arm-token grouping gives each arm an explicit action representation within one shared backbone. SkillLoRA associates parameter updates with atomic skills so they can be reused across arms and tasks. AWA restricts direct attention to each arm's local inputs and actions while preserving shared global context, aiming to reduce dependence on familiar arm pairings without removing information needed for coordination. We test all four SkillLoRA/AWA settings on Token Group and call the full combination **AE-VLA**.

4.1 Shared Backbone and Arm-token Grouping

Every variant receives a global image I_G, wrist images I_L, I_R, arm states q_L, q_R, and the same benchmark-provided active instructions p_L, p_R. One policy generates a joint action chunk $a = (a_L, a_R)$ over $H = 50$ control steps. We retain the backbone's visual encoder, vision–language model, and flow-matching action expert. At denoising time t, the expert predicts a joint velocity field

$$v_\theta(a_t, t \mid I_G, I_L, I_R, q_L, q_R, p_L, p_R) = (v_L, v_R), \tag{1}$$

which is integrated to obtain both arms' actions.

Single $\pi_{0.5}$ uses H joint action tokens: each token predicts both arms at one chunk position. **Token Group** uses two groups of H tokens, $[A_L, A_R]$, where A_L supplies the left-arm output and A_R the right-arm output (Figure 2a). Both groups pass through one shared action expert and output projection. This gives each arm an explicit token range on which SkillLoRA can select weights and AWA can set attention connections.

The prefix is $[G, W_L, W_R, P_L, P_R]$: G and W_r are global and wrist image tokens, and P_r combines arm r's prompt with its discrete state. Fixed ranges and output assignments identify the arms, without a learned arm-ID embedding. Both action groups still embed the full noisy joint action, so grouping does not make their inputs independent. Token Group alone uses the standard sequence mask; SkillLoRA and AWA are added separately below. Appendix B.3 gives the exact layout and output assignment.

4.2 Skill-conditioned Action Parameters

Token Group + SkillLoRA adds a bank of $K = 10$ low-rank updates indexed by atomic skill type. For a token h in arm r's action range, an adapted linear transformation is

$$h' = Wh + \frac{\alpha}{k} B_{\hat{s}_r} A_{\hat{s}_r} h, \qquad k = 4, \quad \alpha = 16, \tag{2}$$

where W is shared and $\hat{s}_r$ is the selected skill. The banks are shared across arms and tasks; the arms can select different skills or reuse the same skill parameters.

Selection uses a separate two-layer router for each arm. The router pools the prefix features at that arm's prompt/state positions after attention and predicts a distribution over skill types; its argmax selects the adapter. One selection is used for all H tokens and denoising steps. Training and inference both use the prediction; skill labels only supervise the routers.

SkillLoRA is applied to attention projections and feed-forward layers in all 18 action-expert blocks, and to the action input/output projections. The visual and language prefix has no skill adapters. This choice adds both parameters and a router loss, so its effect cannot be assigned to skill-specific weights alone.

4.3 Arm-wise Attention Connectivity

Arm-wise attention (AWA) applies to grouped policies with or without SkillLoRA, giving **AE-VLA** and **Token Group + AWA**. Let $C_r = W_r \cup P_r$ denote arm r's local prefix tokens, and $A_{r,j}$ its action token at chunk position j. The allowed key sets $\mathcal{V}$ for each query group are

$$\begin{aligned} \mathcal{V}(G) &= G \cup C_L \cup C_R, \\ \mathcal{V}(C_r) &= G \cup C_r, \\ \mathcal{V}(A_{r,j}) &= G \cup C_r \cup \{A_{r,i} : i \le j\}. \end{aligned} \tag{3}$$

The mask applies both within the observation prefix and from action queries to prefix/action keys. Global queries read both local prefixes, so cross-arm observation information can still pass through

shared global features. Prefix tokens do not attend to action tokens. No latent collaboration tokens are used.

Without AWA, the standard sequence mask allows bidirectional attention within each action block and lets a later block read earlier blocks. AWA blocks both cross-arm action directions and makes each block lower triangular (Figure 2c). Padding masks remain in both settings. This comparison tests the whole mask, including its change to temporal attention; Appendix B.3 gives the details.

4.4 Training and Comparison Protocol

We retain flow matching, with $a_t = (1-t)a + t\epsilon$ and $\epsilon \sim \mathcal{N}(0, I)$. SkillLoRA variants additionally optimize the router's cross-entropy loss:

$$\mathcal{L} = \mathbb{E}\Big[\|v_\theta(a_t, t \mid o) - (\epsilon - a)\|^2\Big] + 0.1\,\mathcal{L}_{\text{route}}, \tag{4}$$

where o denotes the common policy inputs and the action error is averaged over chunk positions and action dimensions. The router loss averages cross-entropy over valid per-arm skill labels; missing labels are ignored. The hard choice blocks action-loss gradients through the router, which is trained by the classification loss. We jointly fine-tune the shared backbone, action expert, adapters, and routers. Variants without SkillLoRA use only the action loss.

5 Experiments

5.1 Experimental Setup

Common pretrained backbone. Single $\pi_{0.5}$, both policies in Dual $\pi_{0.5}$, our MA-VLA implementation (Zhang et al., 2026), and all four Token Group settings initialize their backbones from the same pretrained $\pi_{0.5}$ checkpoint (Intelligence et al., 2025b). This design keeps the pretraining source fixed across all comparisons. For MA-VLA, we replace the original π_0 backbone with $\pi_{0.5}$, and all reported MA-VLA results use this adapted implementation. In Dual $\pi_{0.5}$, each arm is controlled by an independent policy that receives its own wrist view together with the shared global view. We omit TwinVLA (Im et al., 2026) from the quantitative comparison because its pretrained weights differ from those of $\pi_{0.5}$, which would confound architectural differences with differences in pretraining. These choices enable a more controlled comparison with respect to pretrained initialization.

Training and methods. Using the pooled source data in Section 3, we train the single-policy variants for 30,000 steps with batch size 32 on two NVIDIA A800 GPUs. Single $\pi_{0.5}$ predicts both arms with the base model. Dual $\pi_{0.5}$ trains a separate policy for each arm. All receive the same per-arm skill prompts at evaluation; SkillLoRA also uses skill labels to train its routers.

Test set and averages. Each method runs 100 episodes per condition with seeds 1000–1099, for 2,300 episodes across the 23 conditions. We average results equally over conditions. The generalization mean includes all 17 non-in-domain conditions, so the four groups have weights $4{:}6{:}4{:}3$.

Metrics. Constraint-compliant success rate (CCSR) counts episodes that reach the goal and meet the required milestones, order, and timing:

$$\text{CCSR} = 100 \times \frac{N_{\text{success}}}{N_{\text{episodes}}}. \tag{5}$$

Here, N_{success} denotes the number of valid successful episodes, while N_{episodes} denotes the total number of evaluated episodes. Normalized progress is defined as the fraction of required milestones completed under these scoring rules and is used to measure partial task execution. For Sync, the allowable event-time gap is 20 s for Can in Basket, 15 s for Object in Cabinet, and 45 s for all other tasks. For Can in Basket, task completion is determined by environment success, while Sync additionally requires satisfaction of the 20 s event-time constraint. If an arm-wise collision occurs, the corresponding episode is considered a failure. We therefore set its success score to zero while retaining the measured progress. Appendix C.4 provides the complete scoring rules.

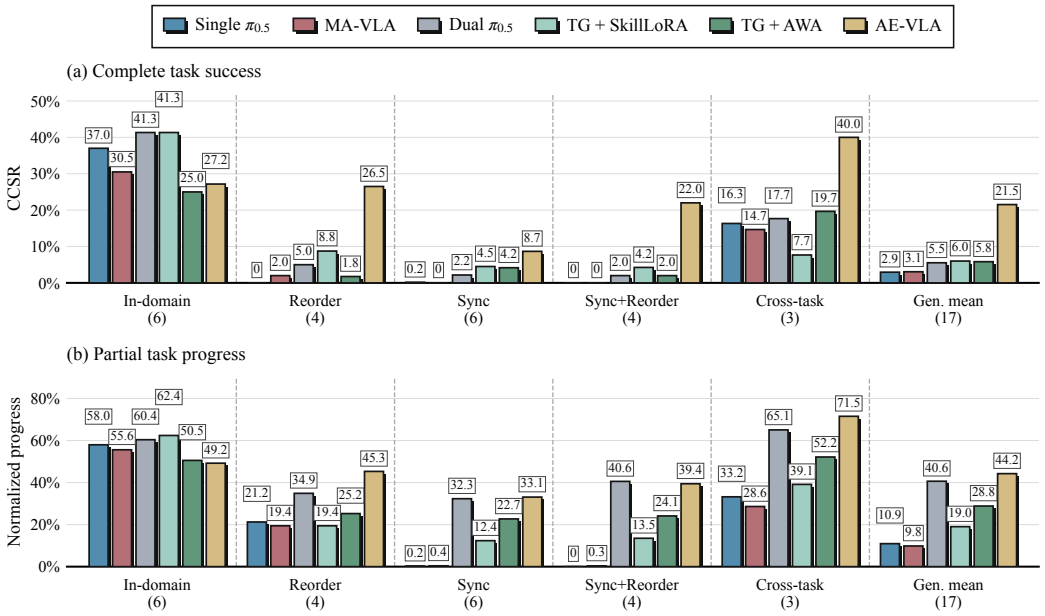


Figure 3: **Success and progress by test group.** Parentheses give condition counts; each condition has 100 episodes per method. TG means Token Group. Bars show means over conditions. Generalization averages all 17 non-in-domain conditions.

5.2 Main Results

Figure 3 shows a large gap between in-domain and compositional success. Single $\pi_{0.5}$ falls from 37.00% in-domain CCSR to 2.94% on the generalization set. MA-VLA reaches 3.06% and Dual $\pi_{0.5}$ reaches 5.53% on this set. The combined AE-VLA setting reaches 21.53%, a gain of 18.59 points over Single and 16.00 over Dual. It reaches 26.50% on Reorder, 22.00% on Sync+Reorder, and 40.00% on Cross-task Combination. Pure Sync remains difficult: AE-VLA reaches 8.67%. Its in-domain success is 27.17%, below all three baselines.

Progress gives a different view. AE-VLA and Dual $\pi_{0.5}$ complete similar fractions of required milestones on the generalization set (44.25% and 40.64%), yet their full success rates differ widely. Thus, local progress does not imply successful completion of the required combination. For example, on synchronization tasks, Dual $\pi_{0.5}$ can grasp the objects but lacks the waiting and coordination needed between arms, causing collisions between arms and task failure. Single and MA-VLA also show low progress (10.94% and 9.83%).

Gains also vary across tasks. AE-VLA reaches 79% on Stack Bowls/Reorder versus 4% for Dual, and 54% on Cube in Bowl/Cross-task versus 0%. Dual is stronger on several other conditions, including Object in Cabinet/In-domain and Burger & Fries/Sync. Single is best on Push Cubes/In-domain; MA-VLA is best on Stack Cubes/In-domain. Table 1 reports all 23 conditions, including cases where the baselines do better.

5.3 Design Choices and Their Interaction

Table 2 shows the complete comparison. Token Group raises generalization CCSR from 2.94% to 3.82%, with almost no change in progress. Adding SkillLoRA alone reaches 6.00%; adding AWA alone reaches 5.82%. Their combination reaches 21.53%. AWA alone has higher progress than SkillLoRA alone (28.84% versus 19.02%), but not higher full success.

The gain from AWA is 2.00 points without SkillLoRA and 15.53 with it. The difference, 13.53 points, measures their interaction. The two choices work better together for these trained models. Appendix C.2 gives the results for each condition.

Table 1: **All simulation conditions: CCSR (%).** Each condition has 100 episodes per method. Only Dual $\pi_{0.5}$ uses two policies. Parentheses in cross-task rows retain the timing setting. Bold marks the best result.

Task	Condition	Single $\pi_{0.5}$	MA-VLA	Dual $\pi_{0.5}$	AE-VLA
Stack Bowls	In-domain	74.00	77.00	**83.00**	**83.00**
	Reorder	0.00	7.00	4.00	**79.00**
	Sync	0.00	0.00	0.00	**26.00**
	Sync+Reorder	0.00	0.00	1.00	**67.00**
Stack Cubes	In-domain	17.00	**28.00**	27.00	20.00
	Reorder	0.00	1.00	9.00	**19.00**
	Sync	0.00	0.00	**4.00**	3.00
	Sync+Reorder	0.00	0.00	0.00	**21.00**
Push Cubes	In-domain	**50.00**	10.00	12.00	8.00
	Reorder	0.00	0.00	**6.00**	3.00
	Sync	0.00	0.00	2.00	**4.00**
	Sync+Reorder	0.00	0.00	**1.00**	0.00
Burger & Fries	In-domain	9.00	0.00	**17.00**	6.00
	Reorder	0.00	0.00	1.00	**5.00**
	Sync	0.00	0.00	**6.00**	1.00
	Sync+Reorder	0.00	0.00	**6.00**	0.00
Can in Basket	In-domain	**58.00**	29.00	50.00	25.00
	Sync	1.00	0.00	0.00	**18.00**
Cube in Bowl	Cross-task (Reference)	0.00	3.00	0.00	**54.00**
	Cross-task (Sync)	0.00	0.00	0.00	**11.00**
Object in Cabinet	In-domain	14.00	39.00	**59.00**	21.00
	Sync	0.00	0.00	**1.00**	0.00
Bowl in Cabinet	Cross-task (Reference)	49.00	41.00	53.00	**55.00**

Table 2: **Design choices on the same $\pi_{0.5}$ backbone.** The four grouped variants form a 2×2 comparison of SkillLoRA and AWA. Values are percentages averaged over conditions.

Variant	Token Group	SkillLoRA	Arm-wise Attn	ID CCSR	Gen. CCSR	Gen. Progress
Single $\pi_{0.5}$	–	–	–	37.00	2.94	10.94
Token Group	✓	–	–	**45.00**	3.82	10.98
Token Group + SkillLoRA	✓	✓	–	41.33	6.00	19.02
Token Group + AWA	✓	–	✓	25.00	5.82	28.84
Token Group + Both (AE-VLA)	✓	✓	✓	27.17	**21.53**	**44.25**

This improvement has an in-domain cost. AWA lowers Token Group from 45.00% to 25.00% in-domain CCSR, and Token Group + SkillLoRA from 41.33% to 27.17%. The comparison also has limits: SkillLoRA adds both weights and router supervision; AWA changes both cross-arm and within-chunk attention. The four settings separate the effects of these two design choices, but do not isolate each change within them.

5.4 Real-world SO101 Results

We use two 6-DoF SO101 arms with one global and two wrist cameras. Training demonstrations cover Stack Bowls, Stack Cubes, and Push Cubes, collected with LeRobot teleoperation Cadene et al. (2024). Cube in Bowl is held out: its cube pick-and-place skill is learned from Stack Cubes and tested with a bowl target. We evaluate three in-domain and five unseen conditions with distractor objects. Each method has 20 trials per condition, giving 640 trials in total. Appendix C.3 describes the setup and scoring.



Figure 4: **SO101 examples.** Rows show Stack Bowls/In-domain, Stack Bowls/Reorder, Stack Bowls/Sync, and Cube in Bowl/Cross-task. Table 3 reports all evaluation trials.

Table 3: **SO101 results with distractor objects.** Each task row shows successes out of 20 trials. Mean rows average rates over three in-domain or five unseen conditions. Bold marks the best result.

Setting	Task	Single $\pi_{0.5}$	MA-VLA	Dual $\pi_{0.5}$	AE-VLA
In-domain	Stack Bowls	15/20	15/20	**18/20**	17/20
	Stack Cubes	5/20	3/20	**10/20**	9/20
	Push Cubes	10/20	9/20	**14/20**	10/20
	ID mean (%)	50.00	45.00	**70.00**	60.00
Reorder	Stack Bowls	0/20	4/20	0/20	**9/20**
Reorder	Push Cubes	0/20	2/20	0/20	**6/20**
Sync	Stack Bowls	0/20	0/20	0/20	**12/20**
Sync	Push Cubes	0/20	0/20	**6/20**	2/20
Cross-task	Cube in Bowl	2/20	0/20	4/20	**10/20**
	OOD mean (%)	2.00	6.00	10.00	**39.00**

AE-VLA reaches 39.00% mean success on the five unseen conditions, compared with 2.00% for Single, 6.00% for MA-VLA, and 10.00% for Dual (Table 3). It leads on four of the five conditions, including Stack Bowls/Sync (12/20 versus 0/20 for every baseline). Push Cubes/Sync is the exception: Dual reaches 6/20 and AE-VLA reaches 2/20. In-domain, AE-VLA reaches 60.00%, below Dual's 70.00%. The real-robot results thus also show improved generalization with some loss of in-domain success. Figure 4 shows example phases.

6 Conclusion

We introduced **ACG-Bench**, a testbed for evaluating dual-arm skill reuse under unseen ordering, synchronization, and cross-task compositions. Using a common $\pi_{0.5}$ backbone, we compare representative data-augmentation and architectural strategies, highlighting the complementarity of skill-conditioned parameters and arm-wise attention. Their combination, **AE-VLA**, achieves 21.53% generalization CCSR in simulation and 39.00% mean success across five unseen SO101 conditions, versus 5.53% and 10.00% for Dual $\pi_{0.5}$. Our study focuses on execution under supplied skill structures, while synchronization and in-domain performance remain challenging. A key next step is a high-level planner that decomposes novel tasks into familiar skills, assigns them to arms, determines order and synchronization, and revises plans from execution feedback. Coupling planning with compositional execution would enable evaluation of the full planning–execution loop and test whether systems can construct and execute new skill combinations for unseen tasks.

AI USE STATEMENT

Generative AI tools were used solely to polish the language and presentation of this manuscript, including grammar, phrasing, and readability improvements. They were not used to generate experimental data, design the benchmark or evaluation protocol, implement the evaluation methods, or replace the authors' scientific judgment. The authors reviewed all AI-assisted text and take full responsibility for the accuracy, claims, results, and final content of this pape

ETHICS STATEMENT

This work studies compositional generalization in simulated and physical dual-arm manipulation. We do not anticipate specific ethical concerns from the reported research beyond the standard safety risks of physical robot operation, which require appropriate precautions in deployment.

REPRODUCIBILITY STATEMENT

Appendices A–C describe the benchmark conditions and atomic skills, policy implementation and training configuration, and evaluation protocols. They also provide per-condition architectural results and the real-world experimental setup to support reproduction and comparison.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026a.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026b.

Nikita Chernyadev, Nicholas Backshall, Xiao Ma, Yunfan Lu, Younggyo Seo, and Stephen James. Bigym: A demo-driven mobile bi-manual manipulation benchmark. *arXiv preprint arXiv:2407.07788*, 2024.

Catherine Glossop, William Chen, Arjun Bhorkar, Dhruv Shah, and Sergey Levine. Cast: Counterfactual labels improve instruction following in vision-language-action models. *arXiv preprint arXiv:2508.13446*, 2025.

Markus Grotz, Mohit Shridhar, Yu-Wei Chao, Tamim Asfour, and Dieter Fox. Peract2: Benchmarking and learning for robotic bimanual manipulation tasks. In *CoRL 2024 Workshop on Whole-body Control and Bimanual Manipulation: Applications in Humanoids and Beyond*, 2024.

Sanjay Haresh, Daniel Dijkman, Apratim Bhattacharyya, and Roland Memisevic. Clevrskills: Compositional language and visual reasoning in robotics. *Advances in Neural Information Processing Systems*, 37:38235–38266, 2024.

Hokyun Im, Euijin Jeong, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. TwinVLA: Data-efficient bimanual manipulation with twin single-arm vision-language-action models. In *International Conference on Learning Representations*, 2026. URL `https://arxiv.org/abs/2511.05275`.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. pi06: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi05: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. pi07: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

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. In *International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2210.03094.

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.

Matthew Lai, Keegan Go, Zhibin Li, Torsten Kröger, Stefan Schaal, Kelsey Allen, and Jonathan Scholz. Roboballet: Planning for multirobot reaching with graph neural networks and reinforcement learning. *Science Robotics*, 10(106):eads1204, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Songming Liu, Bangguo Li, Kai Ma, Lingxuan Wu, Hengkai Tan, Xiao Ouyang, Hang Su, and Jun Zhu. Rdt2: Exploring the scaling limit of umi data towards zero-shot cross-embodiment generalization. *arXiv preprint arXiv:2602.03310*, 2026.

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

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the computer vision and pattern recognition conference*, pp. 27649–27660, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Yiran Qin, Li Kang, Xiufeng Song, Zhenfei Yin, Xiaohong Liu, Xihui Liu, Ruimao Zhang, and Lei Bai. Robofactory: Exploring embodied agent collaboration with compositional constraints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10075–10085, 2025.

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.

Yi Ru Wang, Carter Ung, Grant Tannert, Jiafei Duan, Josephine Li, Amy Le, Rishabh Oswal, Markus Grotz, Wilbert Pumacay, Yuquan Deng, et al. Roboeval: Where robotic manipulation meets structured and scalable evaluation. *arXiv preprint arXiv:2507.00435*, 2025.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, et al. Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

Zenan Wu, Bingqing Wei, Lu Liu, Zheqi He, Xi Wang, Jiakang Liu, Zehui Li, Guocai Yao, Jing-Shu Zheng, Xi Yang, and Yongtao Wang. ATOM-Bench: A real-world benchmark for atomic skills and compositional generalization in manipulation policies. *arXiv preprint arXiv:2606.16826*, 2026. URL https://arxiv.org/abs/2606.16826.

Xuanran Zhai, Zekai Huang, Longyan Wu, Qianyou Zhao, Qiaojun Yu, Jieji Ren, Ce Hao, and Harold Soh. SkillVLA: Tackling combinatorial diversity in dual-arm manipulation via skill reuse. *arXiv preprint arXiv:2603.03836*, 2026. URL https://arxiv.org/abs/2603.03836.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11142–11152, 2025.

Zaibin Zhang, Junlan Xiao, Zhongbo Zhang, Yifan Wang, Li Kang, Yiran Qin, Changxing Xia, Heng Zhou, Talas Fu, Enshen Zhou, Ruimao Zhang, Zhenfei Yin, Huchuan Lu, and Lijun Wang. MA-VLA: Multi-arm vision-language-action model for collaboration and compositional generalization. *arXiv preprint arXiv:2608.25864*, 2026. URL https://arxiv.org/abs/2608.25864.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A BENCHMARK AND ATOMIC SKILLS

We first describe the conditions, atomic operations, and phase prompts. We then give the policy implementation, complete results, and scoring rules.

A.1 CONDITION INVENTORY

Table 4 lists the 23 shared conditions. Each cross-task condition is counted in the Cross-task Combination group, even when it also changes timing.

Table 4: ACG-Bench condition inventory. Reference and Sync distinguish the execution requirements within the Cross-task group.

Task family	ID	Reorder	Sync	Sync+R	Cross-task
Stack Bowls	✓	✓	✓	✓	–
Stack Cubes	✓	✓	✓	✓	–
Push Cubes	✓	✓	✓	✓	–
Burger & Fries	✓	✓	✓	✓	–
Can in Basket	✓	–	✓	–	–
Cube in Bowl	–	–	–	–	Reference, Sync
Object in Cabinet	✓	–	✓	–	–
Bowl in Cabinet	–	–	–	–	Reference
Number of conditions	6	4	6	4	3

A.2 TASK-PROGRAM AND DEMONSTRATION GENERATION

Each task starts with a goal, scene, object list, allowed motion-planning calls, event graph, and success checks. A code-generation agent writes the simulator task and expert program. We run the program from varied valid initial states. A checking agent identifies collisions, failed grasps, and failed task checks, and the program is revised. Accepted programs collect 100 clean expert demonstrations per source task. We split the trajectories into skills using gripper state, object attachment, object and end-effector poses, and joint positions.

A.3 ATOMIC SKILLS AND THEIR ARGUMENTS

An atomic skill describes what one arm should do during a phase. It is smaller than a complete dual-arm task, but can include several control steps: Pick includes approaching and grasping an object; Place includes carrying it to a target and releasing it. We describe a skill execution as (r, s, o, g), where r is the arm, s the operation, o the object or articulated part, and g an optional target or dependency. Pick(bowl) and Pick(cube) share an operation while keeping different object arguments.

Table 5 describes the operations in the included simulation tasks. The prompts are examples from the demonstration conversion scripts. They specify a behavior, rather than a fixed trajectory: the policy still has to use the current scene and arm state to execute it.

Manipulation and waiting. Wait can mean remaining still with an empty gripper or continuing to hold an object. In Can in Basket, the basket arm waits while the other arm places the can. In Object in Cabinet, the drawer arm waits after opening the drawer. Recover instead requests an arm motion after manipulation. Neither word implies a new task goal. In the Push Cubes conversion, recovery motion is included in the Push segment rather than given a separate Recover prompt. Thus, a skill label describes the demonstrated phase, not every motion that occurs within it.

A.4 SKILL COVERAGE ACROSS TASKS

Table 6 connects the atomic operations to the six source families and two held-out cross-task families. Reorder, Sync, and Sync+Reorder keep a family's manipulation operations and change their

Table 5: Atomic operations and example prompts. Object and target arguments vary across tasks. These semantic names do not specify numerical adapter IDs.

Skill	Behavior and arguments	Example atomic prompt
Pick	Approach and grasp the specified object.	Pick up the bowl.
Place	Carry a held object to the target and release it.	Place the can into the basket.
Lift	Raise a grasped object or container to the required pose.	Lift the basket.
Pull	Move a grasped articulated part to open it.	Pull the drawer open.
Move	Position the end effector for the next manipulation.	Move to the blue cube.
Push	Move an object toward a target through contact.	Push the blue cube to the target.
Recover	Return the arm after its manipulation phase.	Recover.
Wait	Maintain the current configuration until the other arm or a dependency allows progress.	Wait for the other robots.

Table 6: Manipulation operations and their roles in ACG-Bench. Source rows summarize demonstration prompts; the final two rows describe held-out compositions. Wait and recovery behavior are discussed in Appendix A.3.

Task family	Manipulation operations	Source behavior or held-out combination
Stack Bowls	Pick, Place	Each arm handles a bowl and places it at its target.
Stack Cubes	Pick, Place	Each arm handles a cube and places it at its target.
Push Cubes	Move, Push	Each arm approaches a cube and pushes it to a target.
Burger & Fries	Pick, Place	The arms place the burger and fries at their respective targets.
Can in Basket	Pick, Lift, Place	One arm holds and lifts the basket; the other places the can inside.
Object in Cabinet	Handle grasp, Pull, Pick, Place	One arm grasps and opens the drawer; the other places the box inside.
Cube in Bowl	Pick, Place	Combine bowl handling and cube placement with a bowl target.
Bowl in Cabinet	Handle grasp, Pull, Pick, Place	Combine drawer opening with bowl placement.

relations across arms. Cross-task composition combines source behaviors with new partners or targets. Sharing an operation name alone does not establish that an entire manipulation is familiar: the object, target, and partner arm also determine what changes at test time.

The cabinet prompt "Grab the drawer handle" specifies the grasp that precedes Pull. Its wording is retained here to distinguish handle grasping from picking up a free object. The skill-label file determines how prompt variants map to the ten adapter indices used by SkillLoRA (Appendix B.2).

A.5 From Demonstrations to Per-arm Prompts

The source trajectories are segmented before training. For Stack Bowls, Stack Cubes, and Burger & Fries, the conversion scripts use each arm's gripper closing and opening events to separate picking from placement. They then assign a pair of atomic prompts to each segment. Table 7 gives the Stack Cubes example. A segment can contain two active behaviors, such as the left arm recovering while the right arm picks. Its phase index identifies the prompt pair; it is not a single atomic-skill ID.

Gripper transitions do not identify every behavior. Push Cubes keeps its grippers closed, so its segmentation uses end-effector height and changes in joint motion to separate approach, pushing, and the handoff between arms. The cabinet script uses handle-grasp and joint-motion cues to locate the end of pulling. These offline boundaries label the demonstrations; at evaluation, the shared scheduler uses the task's state checks to update prompts. The evaluator separately checks the required physical milestones and final goal. A gripper closure in a demonstration is therefore an annotation cue, not sufficient evidence of a successful test-time grasp.

Table 7: The five prompt pairs in the Stack Cubes source conversion. The two prompt strings are stored together and separated by arm at policy input. This is a training annotation, not a fixed test-time clock.

Phase	Left-arm prompt	Right-arm prompt
1	Pick up the blue cube.	Wait for the other robots.
2	Put the blue cube on the target.	Wait for the other robots.
3	Recover.	Pick up the green cube.
4	Wait for the other robots.	Place the green cube on the target.
5	Wait for the other robots.	Recover.

Table 8: How a familiar routine can be recomposed. The first three rows illustrate changes in order and timing; the last illustrates source-skill reuse across tasks. Exact milestones remain task-specific.

Test group	Change to the source routine
Reorder	Require $k_R \prec d_R \prec k_L \prec d_L$: the right arm completes its manipulation first.
Sync	Require $\lvert\tau(k_L) - \tau(k_R)\rvert \leq \delta$, followed by the source placement order $d_L \prec d_R$.
Sync+Reorder	Use the same synchronized pick pair, then reverse the placement order to $d_R \prec d_L$.
Cross-task	Combine bowl handling from Stack Bowls with cube placement from Stack Cubes; the cube's target is now a bowl.

A.6 Examples of Unseen Compositions

Consider a source routine in which the left arm picks and places its object before the right arm does the same. Write k_L, k_R for the two pick milestones and d_L, d_R for the two placement milestones. The source order is $k_L \prec d_L \prec k_R \prec d_R$, where $\prec$ denotes a required predecessor. Recovery and waiting prompts are omitted from this example so that the manipulation events are easy to compare.

Here $\tau(e)$ is the recorded completion time of event e, and δ is the task's synchronization window. Synchronization is checked on these event times; it does not require the arms to have identical trajectories or continuously overlap their motions. A cross-task condition remains in the Cross-task group even when it also synchronizes events. Thus, Cube in Bowl/Sync contributes once to the 17-condition generalization mean.

B Policy Implementation

B.1 Training Configuration

Table 9 lists the multitask AE-VLA training configuration. The four grouped variants turn Skill-LoRA and AWA on or off. AWA enables both arm-wise attention switches together. Token Group + AWA uses the multi-agent attention model without skill banks. All four use the same source dataset, batch size, training steps, and learning-rate settings.

Because the cosine schedule has equal peak and final rates, the learning rate is constant after warmup. Joint actions are offsets from the current joint state; gripper actions remain absolute. These settings disable extra image masking, added Wait examples, and shuffling of arm roles. This does not disable the backbone's standard training image preprocessing. Single $\pi_{0.5}$ and every AE-VLA variant produce both arms from one model instance. Dual $\pi_{0.5}$ trains and runs an independent instance for each arm.

Flow-matching action loss. Let a be a demonstrated joint action chunk, o the policy inputs, and ϵ Gaussian noise. At noise level t, training uses

$$a_t = (1-t)a + t\epsilon, \qquad \mathcal{L}_{\text{action}} = \mathbb{E}\left[\|v_\theta(a_t, t \mid o) - (\epsilon - a)\|^2\right]. \tag{6}$$

The expectation samples a demonstration, a noise vector $\epsilon \sim \mathcal{N}(0, I)$, and $t \sim \text{Beta}(1.5, 1)$ scaled to $[0.001, 1]$. The squared error is averaged over action dimensions and chunk positions. The joint

Table 9: Multitask AE-VLA training configuration.

Component	Setting
Initialization	$\pi_{0.5}$ base checkpoint
Backbone / action expert variants	`gemma_2b` / `gemma_300m`
Control dimensions / model dimensions	2×8 / 32 (zero-padded)
Action horizon / grouped action tokens	50 / 2×50
Prompt/state token budget	200 per arm; 400 in total
Optimizer	AdamW; $(\beta_1, \beta_2) = (0.9, 0.95)$
Optimizer ϵ / weight decay	10^{-8} / 10^{-10}
Global gradient-norm clipping	1.0
Learning-rate warmup	1,000 steps
Peak / final learning rate	5×10^{-5} / 5×10^{-5}
Configured cosine schedule horizon	200,000 steps
Training steps / global batch size	30,000 / 32
Checkpoint interval	30,000 steps
Frozen parameters	None
EMA decay	0.99
Image-mask / wait-augmentation probability	0 / 0
Arm-shuffle probability	0
Latent collaboration tokens	Disabled

prediction contains both arms' velocities. At inference, ten Euler steps denoise the joint action chunk.

B.2 Shared Instruction Interface and SkillLoRA

Single $\pi_{0.5}$, MA-VLA, Dual $\pi_{0.5}$, and every AE-VLA variant use the same per-arm atomic prompts at evaluation. Each prompt identifies the active skill and its object, target, or dependency arguments. The scheduler uses the same phases and state checks for every method; the time at which a prompt changes can differ because the policies produce different trajectories.

Skill labels and routing. The ten adapter IDs are atomic skill types, not full tasks or skill–object pairs. During training, the data loader maps dataset task IDs to per-arm skill IDs using `skill_labels.jsonl`. These labels train the routers. All methods receive the same skill information in their evaluation prompts, but SkillLoRA adds a skill classification loss during training.

The configuration enables `router_use_text_only` and `router_per_arm`. Each arm has its own two-layer MLP ($2048 \to 1024 \to 10$, with a Swish hidden activation). The router averages its arm's prompt/state features after prefix attention, ignoring padding. The text-only option selects the positions to average; those features include discrete state tokens and have already attended to allowed image tokens. The router can therefore use visual and state information. It selects the highest-scoring skill and encodes that choice as a one-hot vector with no gradient through the choice. Training and inference both use this prediction. Ground-truth labels are used only in the training loss. One selection per arm is reused for all 50 action tokens and all ten Euler denoising steps.

Adapted layers and sharing. Each adapted layer has ten pairs of rank-4 factors, with scale $16/4 = 4$. The banks are applied to every one of the action expert's 18 transformer layers: query, key/value, and attention-output projections, both feed-forward gating branches, and the feed-forward output projection. The action input and output projections also use skill banks. The visual encoder and vision–language prefix do not use skill banks. Adapters are shared across arms; router MLPs are arm-specific. The shared model weights and the added parameters are jointly trained. For a matrix with d_{in} inputs and d_{out} outputs, the ten rank-4 adapters add $40(d_{\text{in}} + d_{\text{out}})$ parameters. This expression applies to each adapted attention projection and feed-forward branch.

Loss and interpretation. The router loss is cross-entropy averaged over valid arm labels, with weight 0.1; missing labels are represented by -1 and excluded. The hard choice blocks action-loss gradients through the router. The classification loss can also update the prefix features used by the router. SkillLoRA changes the action layers, number of weights, and training loss. A shared adapter

with the same parameter count or a shuffled-label control could help separate these effects. We test execution with supplied skill prompts; we do not test incorrect instructions or skill labels.

B.3 Token Layout and Exact Attention Masks

Fixed token ranges and output assignments identify each arm. There are no learned arm-ID embeddings. Each arm's instruction is tokenized together with its own discretized state. The $\pi_{0.5}$ suffix contains two action-token blocks and no separate continuous-state tokens. Both blocks embed the full noisy joint action vector, and their decoded outputs are concatenated into the joint prediction. Thus, the mask restricts attention connections, but it does not make the two action streams independent.

From joint tokens to arm-specific tokens. In Single $\pi_{0.5}$, each action token represents one step of the joint action: decoding that token produces both arms' outputs. Its suffix has $H = 50$ tokens. Token Group instead uses $[A_L, A_R]$, with H tokens in each block, for $2H = 100$ action tokens in total. The left block supplies the left-arm output and the right block supplies the right-arm output at each horizon position. Both blocks pass through the same action expert and output projection. Their outputs are assembled into one joint velocity chunk, which is integrated during denoising.

For simulation, the noisy input and velocity prediction both have shape $H \times 32$: 16 physical control dimensions and 16 padding dimensions. Grouping changes the action-token sequence length, not the control horizon or number of controlled joints. The action input projection receives the full 32-dimensional noisy vector in both groups. The per-arm organization therefore refers to token ranges and output assignment, not to separate noisy inputs or independent policy instances.

Why grouping supports the two design choices. The prefix layout is $[G, W_L, W_R, P_L, P_R]$, where each P_r contains the arm's instruction and discretized state. The local group $C_r = W_r \cup P_r$ links that observation to action block A_r. SkillLoRA applies the selected update $\Delta W_{\hat{s}_r}$ to the tokens in A_r. AWA uses the same group boundaries to determine which keys each query may read. Grouping supplies the arm assignment; the two mechanisms separately change weights and attention connections. All four combinations use this layout. The Token Group comparison with Single $\pi_{0.5}$ also changes suffix length from H to $2H$.

With AWA, global prefix tokens can attend to every valid prefix token. Each arm's wrist and prompt/state tokens can attend to the global tokens and their own local group. An action token can read the same prefix group and its own arm's action tokens at the same or earlier chunk positions. Prefix queries never attend to the action suffix.

Equation 3 in the main text gives the complete AWA key sets. Padding masks apply to all three cases.

The `full_attn` variants disable both arm-wise switches and use the standard block mask. Their prefix is bidirectional; action queries can access every valid prefix token. Within the suffix, tokens attend bidirectionally inside each arm block, and a later block can attend to earlier blocks. With the stored $[A_L, A_R]$ order, A_R can attend to A_L, but A_L cannot attend to A_R. With AWA, both cross-arm action directions are blocked and each block becomes lower triangular. The comparison therefore changes both cross-arm and within-chunk attention. It tests the whole mask, rather than blocking cross-arm attention alone while keeping the connections within each chunk fixed.

B.4 Relationship to Prior Design Principles

Our quantitative study holds the pretrained backbone initialization fixed at $\pi_{0.5}$. We adapt MA-VLA (Zhang et al., 2026) from its original π_0 backbone to $\pi_{0.5}$, and the MA-VLA label throughout the paper refers to this adaptation. TwinVLA uses different pretrained weights, so including it would mix differences in pretraining with differences in architecture. We therefore omit its results from the quantitative comparison. Below, we clarify architectural differences from prior work.

TwinVLA (Im et al., 2026) starts from its pretrained single-arm SingleVLA and copies the VLM into two arm-specific branches. The branches share the visual encoder and a diffusion-transformer action head, which jointly decodes their readout tokens. Joint attention combines queries, keys, and

values from both branches, applies a mask permitting cross-arm access, and splits the outputs back into the branches. The projection and feed-forward weights remain arm-specific.

TwinVLA also processes language and ego-view tokens as a shared sequence. A learned gate mixes the two arms' feed-forward experts for these tokens; attention reweighting preserves the contribution of shared modalities. Our Token Group keeps global context and separate arm token ranges in one $\pi_{0.5}$ backbone, without duplicating the VLM. Its two action-token groups create places to apply arm-specific routing and attention rules. SkillLoRA selects atomic-skill adapters on action tokens, rather than mixing arm experts on shared observation tokens. AWA blocks direct cross-arm action attention while retaining a path through global context, whereas TwinVLA's joint attention permits direct exchange between branches. These variants therefore test related design choices within $\pi_{0.5}$; they are not a reproduction of TwinVLA's full architecture.

SkillVLA (Zhai et al., 2026) studies skill reuse and communication between arms. Our variants test per-arm token groups, skill-specific action weights, and attention access within one backbone. They do not include SkillVLA's high-level skill selector or adaptive cooperation gate. A released SkillVLA implementation was unavailable for these experiments. The study compares design ideas under the same training and evaluation setup; it does not rank the full TwinVLA and SkillVLA systems.

C Additional Results and Evaluation Protocol

C.1 Task Results

Table 1 lists all 23 conditions for the three baseline policies and AE-VLA.

C.2 Complete Architectural Results

Table 10 reports the full 2×2 comparison on Token Group. Token Group + AWA was scored from its 2,300 episode records using the same set of conditions, task-specific synchronization windows, and scoring rules as the other variants. Each condition uses seeds 1000–1099.

C.3 Real-world SO101 Setting

Figure 5 shows the real-world platform, which consists of two SO101 arms with 6 degrees of freedom per arm. One fixed RGB camera observes the shared workspace and one wrist RGB camera is attached to each arm; all three streams run at 30 Hz.

We collect 50 demonstrations for each of Stack Bowls, Stack Cubes, and Push Cubes through LeRobot teleoperation Cadene et al. (2024), giving 150 demonstrations in total. Figure 6 shows example camera observations. We annotate the demonstrations with the same arm-wise phase vocabulary used in simulation and pool all three tasks when training each real-world policy. The training hyperparameters match those used for the simulation benchmark; the grouped-policy configuration is given in Appendix B.1. No Cube in Bowl demonstrations are used for training. The cube pick-and-place behavior is learned from Stack Cubes; placing the cube in a bowl is a held-out composition evaluated only at test time.

We evaluate eight conditions with 20 rollouts each. The three In-domain conditions are Stack Bowls, Stack Cubes, and Push Cubes. The generalization conditions are Reorder and Sync variants of Stack Bowls and Push Cubes, plus Cube in Bowl/Cross-task Combination. Each rollout randomizes valid object poses and contains distractor objects. In-domain success requires the task goal and all physical milestones. Generalization success additionally requires the target order or synchronization relation; Cross-task Combination requires completion of the recombined bowl-and-cube phase graph. The table reports success counts for individual conditions and mean rates across the three In-domain and five generalization conditions. Each method has 160 evaluated rollouts, for 640 rollouts in total.

C.4 Constraint-compliant Scoring

For conditions without synchronization, we check milestones in the required order. An episode succeeds only if it reaches the final environment goal and completes all milestones under the task

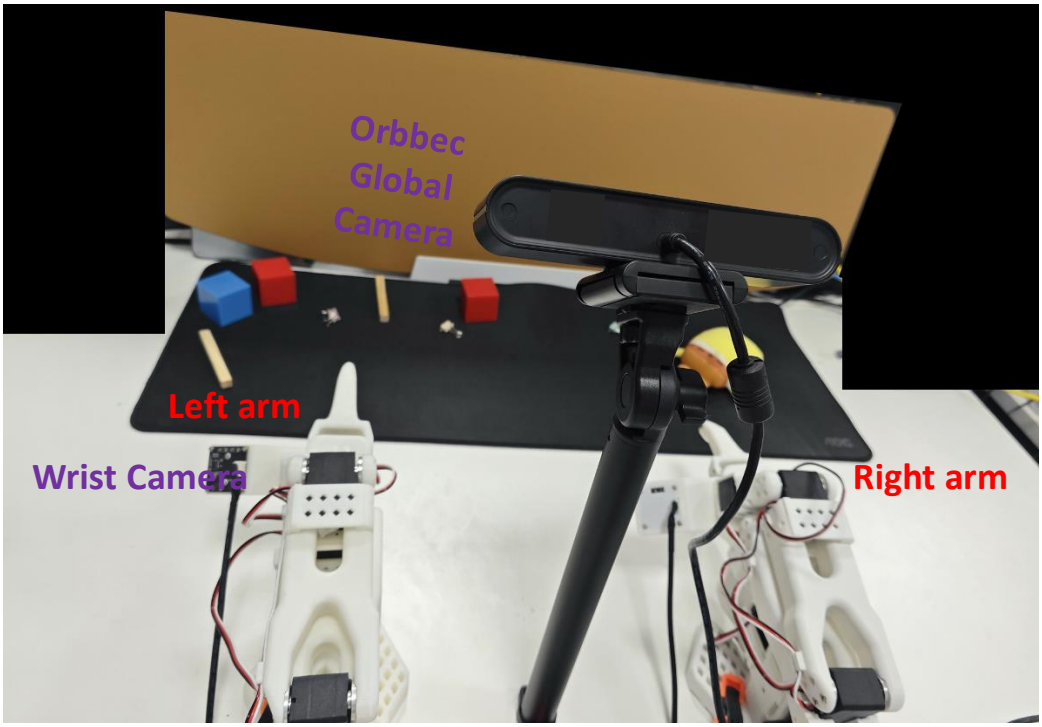


Figure 5: **SO101 setup.** Two arms share a workspace observed by one fixed global camera and one wrist camera on each arm.

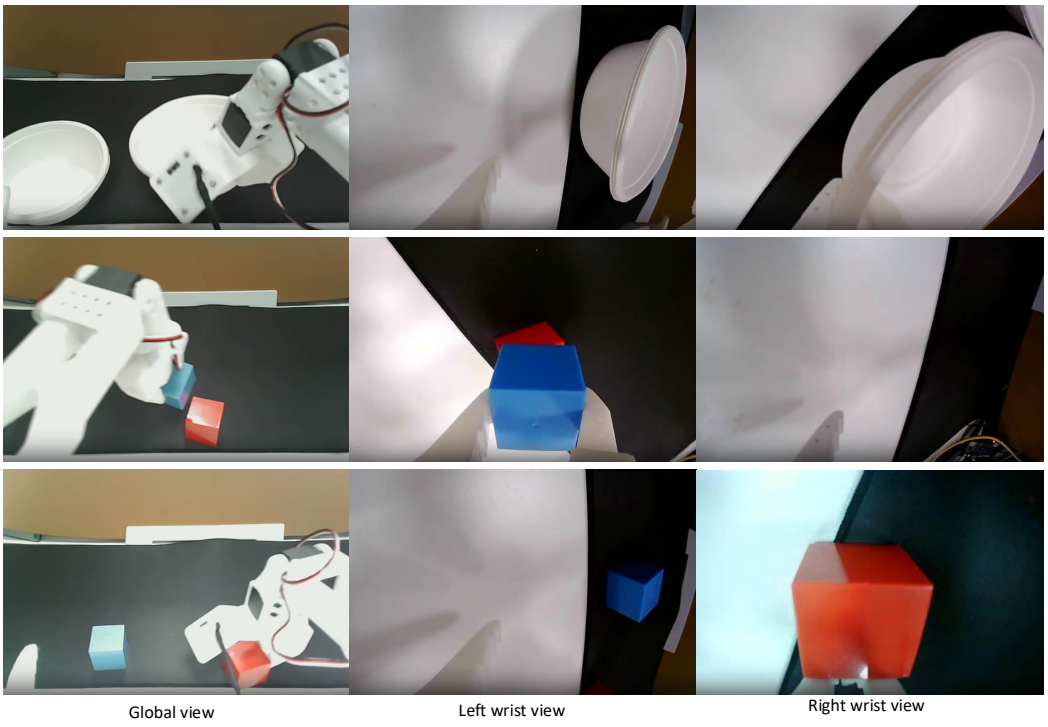


Figure 6: **SO101 training demonstrations.** Example observations from the global, left-wrist, and right-wrist cameras (left to right). We collect 50 demonstrations per training task through LeRobot teleoperation.

Table 10: **Four design settings on all conditions: CCSR (%).** TG means Token Group; SL means SkillLoRA. Each setting uses one policy and the same 23 conditions, with 100 episodes per condition.

Task	Condition	TG	TG + SL	TG + AWA	AE-VLA
Stack Bowls	In-domain	**88.00**	83.00	43.00	83.00
	Reorder	0.00	32.00	0.00	**79.00**
	Sync	0.00	19.00	0.00	**26.00**
	Sync+Reorder	0.00	17.00	7.00	**67.00**
Stack Cubes	In-domain	**34.00**	**34.00**	8.00	20.00
	Reorder	0.00	3.00	1.00	**19.00**
	Sync	0.00	0.00	1.00	**3.00**
	Sync+Reorder	0.00	0.00	0.00	**21.00**
Push Cubes	In-domain	3.00	**21.00**	17.00	8.00
	Reorder	0.00	0.00	**6.00**	3.00
	Sync	**4.00**	0.00	1.00	**4.00**
	Sync+Reorder	**1.00**	0.00	0.00	0.00
Burger & Fries	In-domain	**20.00**	10.00	1.00	6.00
	Reorder	0.00	0.00	0.00	**5.00**
	Sync	0.00	**2.00**	0.00	1.00
	Sync+Reorder	0.00	0.00	**1.00**	0.00
Can in Basket	In-domain	**74.00**	49.00	40.00	25.00
	Sync	1.00	6.00	**21.00**	18.00
Cube in Bowl	Cross-task (Reference)	0.00	2.00	3.00	**54.00**
	Cross-task (Sync)	1.00	0.00	0.00	**11.00**
Object in Cabinet	In-domain	**51.00**	**51.00**	41.00	21.00
	Sync	0.00	0.00	**2.00**	0.00
Bowl in Cabinet	Cross-task (Reference)	**58.00**	21.00	56.00	55.00

rules. For Sync and Sync+Reorder, the evaluator records the first unconditional completion time of each milestone. Both designated synchronized milestones must occur, with a time gap no larger than the task window. Later milestones must follow this pair in the required order; tied completion times are allowed. Normalized progress is the number of milestones completed under these rules divided by the number required.

Can in Basket is an exception: in-domain success uses the environment's `task_success` directly. Its Sync condition also requires the two pick events to occur within 20 s. Object in Cabinet/Sync uses a 15 s window; all other included Sync and Sync+Reorder conditions use 45 s. These checks concern event times, not the overlap of whole action intervals.

Manual review found arm collisions in all apparent Dual $\pi_{0.5}$ successes on Stack Bowls/Sync and Cube in Bowl/Sync. Their CCSR is set to zero while their milestone progress is retained. Progress is therefore not a measure of collision-free execution.

The Single $\pi_{0.5}$ and MA-VLA logs were generated before the final synchronization windows were fixed, so we rescored them from their recorded milestone timelines. As a consistency check, the same scorer reproduces all cached condition-level flags and progress values for the other five variants, except for the two documented manual collision overrides. Single $\pi_{0.5}$ and MA-VLA each have 2,300 evaluated episodes, with seeds 1000–1099 in every condition.