SAME SCENE, DIFFERENT TASK: SKILL ALIGNMENT FOR COMPOSITIONAL GENERALIZATION IN VLAS

Anonymous authors
Paper under double-blind review

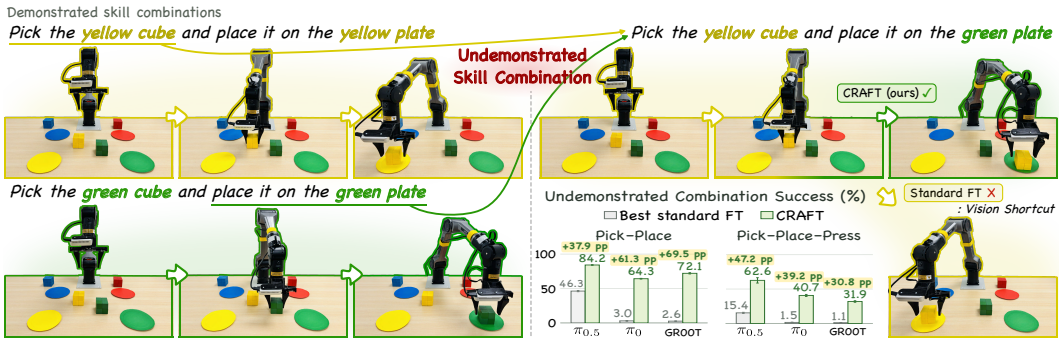

Figure 1: Generalization to undemonstrated skill combinations. Fine-tuned VLAs struggle to execute new combinations of demonstrated skills. CRAFT uses demonstrated skill executions to train the policy to follow instructions for new combinations, improving success on undemonstrated combinations across two benchmarks and three VLA models over standard fine-tuning.

ABSTRACT

Vision-language-action (VLA) models often struggle to generalize to skill combinations absent from their fine-tuning demonstrations, even when every constituent skill has been demonstrated. We focus on a *vision shortcut* as one failure mode: during fine-tuning, visual observations can serve as a proxy for the instruction, so a policy may execute a demonstrated combination associated with similar observations rather than the instructed combination. This motivates counterfactual pairs that hold a demonstration observation fixed while changing the instruction to specify an undemonstrated combination. These pairs lack demonstrated actions for supervision. Crucially, the currently required skill has already been demonstrated, but actions from those executions cannot be transferred directly because the same skill can require different actions across observations. We propose **CRAFT**, which transfers supervision from demonstrated executions of the required skill to counterfactual pairs through reusable skill representations. Across three VLA models and two simulation benchmarks, CRAFT improves success on undemonstrated combinations while maintaining high success on demonstrated ones. CRAFT also improves compositional generalization on a real robot.

1 INTRODUCTION

Vision-language-action (VLA) models map visual observations and natural-language instructions to robot actions (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; 2025; Bjorck et al., 2025). Adapting these models to a target robot embodiment, sensor configuration, or environment often involves fine-tuning on robot demonstrations collected in that setting (Kim et al., 2025). Because collecting demonstrations that cover every task of interest is costly, scalable adaptation requires generalization beyond the demonstrated tasks. Prior work has studied how adapted VLAs retain capabilities from pretraining (Yadav et al., 2026; Guo et al., 2026), generalize across objects and visual conditions (Dey et al., 2025), and remain robust to language variation (Grover et al., 2025). In this work, we study a complementary form of generalization: whether a VLA can *recombine skills demonstrated during adaptation into combinations absent from those demonstrations*.

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.

Many robotic tasks can be viewed as compositions of skills, such as picking an object and then placing it at a target location (Ahn et al., 2022; Shi et al., 2025). We define a *skill* as an operation paired with the entity specified for that operation, such as picking a red cube or placing an object on a blue plate. For example, demonstrations of "*pick the yellow cube and place it on the yellow plate*" and "*pick the green cube and place it on the green plate*" contain all the constituent skills needed for "*pick the yellow cube and place it on the green plate*," even though that combination is absent from the demonstrations (Figure 1). As the number of possible entities grows, collecting demonstrations for every skill combination becomes increasingly costly. This motivates a compositional generalization setting in which all required skills appear in the fine-tuning demonstrations, but their combination does not. Yet even when every constituent skill is demonstrated, we observe that fine-tuned VLAs often perform well on demonstrated combinations but struggle with new combinations (Appendix A.1).

To address this gap, we focus on one failure mode: a *vision shortcut*, in which fine-tuned VLAs use visual observations as a proxy for the instruction (Xu et al., 2025; Lian et al., 2026). As a result, a policy may execute a demonstrated combination associated with similar observations even when instructed to perform a new one (Figure 1). To mitigate this issue, we form *counterfactual pairs* during fine-tuning by keeping a demonstration observation fixed while changing the instruction to specify a combination absent from the demonstrations. The resulting pair, however, has no corresponding demonstrated action target. Crucially, although the combination is absent from the demonstrations, the skill currently required under the changed instruction has already been demonstrated in the dataset. Nonetheless, actions from those demonstrations cannot generally serve as direct targets for the counterfactual pair, because the action required for the same skill can vary across observations. Thus, the key challenge is to *transfer* supervision from demonstrated executions of the required skill to the counterfactual pair while accounting for how the required action varies across observations.

We propose **CRAFT** (**C**ounterfactual Skill-**R**epresentation **A**lignment for **F**ine-**T**uning), which addresses this challenge by learning a representation of the currently required skill that is reusable across executions while conditioning action prediction on the observation. CRAFT introduces a *skill representation* that reflects the currently required skill and a *state representation* that captures visual information; together, they condition action prediction. For each counterfactual pair, CRAFT selects a demonstrated execution of the required skill. It combines the pair's skill representation with that execution's state representation and supervises the resulting prediction using that execution's action target. This supervision trains the counterfactual pair's skill representation to reflect the required skill in action prediction. In this way, CRAFT transfers supervision from demonstrated executions of the required skill to the counterfactual pair without requiring a corresponding demonstrated action target.

We evaluate CRAFT across π-family (Black et al., 2024; 2025) and GR00T (Bjorck et al., 2025) VLA models on two compositional benchmarks in simulation and additionally on a real robot. CRAFT substantially improves success on new skill combinations while maintaining high performance on demonstrated combinations. Our contributions are as follows:

- We propose **CRAFT**, a counterfactual supervision approach that uses existing demonstrations of constituent skills to train VLAs to execute skill combinations absent from the original demonstrations, without collecting demonstrations for those combinations.
- We introduce two compositional benchmarks for two- and three-skill tasks, with demonstration datasets that cover every constituent skill but only a subset of skill combinations; we will release the datasets and simulation environments.
- Across π-family and GR00T VLA models, CRAFT improves success on skill combinations absent from the original demonstrations while maintaining high success on demonstrated combinations; we additionally validate CRAFT on a real robot.

2 Related Work

2.1 Compositional Generalization in VLAs

Compositional generalization concerns generalizing to new combinations of components encountered during training (Lake & Baroni, 2018) and has been studied in computer vision (Uselis et al., 2025) and large language models (Sakai et al., 2025). In robot learning, prior work has studied generalization to new combinations of environmental factors, such as object type and table height (Gao et al., 2024), as well as recombining demonstrated behaviors to execute new tasks (Mandlekar et al., 2020).

More closely related to our setting, recent work has shown that VLAs can struggle to generalize to new combinations of previously learned skills (Xie et al., 2026). One line of work reuses learned behaviors at deployment, either by interpolating task-specific text representations to compose behaviors learned from different tasks (Li, 2025) or by learning reusable skills that can be recombined for new task combinations (Zhai et al., 2026). Another approach expands the training data by generating execution trajectories for new skill combinations and retaining successful ones as synthetic demonstrations (Peng et al., 2026). We instead exploit the constituent-skill coverage already present in the demonstrations, providing supervision during fine-tuning without adding execution trajectories for new combinations.

2.2 Shortcut Learning and Counterfactual Supervision

Shortcut learning occurs when models exploit predictive correlations in the training data that allow accurate predictions with limited reliance on task-relevant information (Geirhos et al., 2020; Si et al., 2022; Elisha et al., 2026). In robot demonstrations, the task being executed can often be inferred from the visual observation, allowing a fine-tuned policy to fit demonstrated actions with limited reliance on the instruction (Fang et al., 2025; Xu et al., 2025; Lian et al., 2026). This can produce a *vision shortcut*, in which the policy uses the visual observation as a proxy for the instruction when predicting actions (Xing et al., 2025). In our compositional setting, such a shortcut can cause the policy to execute a demonstrated combination associated with similar observations rather than the new combination specified by the instruction.

Counterfactual training has been used to reduce shortcut reliance by varying a task-relevant factor while holding other inputs fixed and supervising models on the resulting examples (Kaushik et al., 2020; Wan et al., 2025). The vision shortcut therefore motivates *counterfactual pairs* that keep the observation fixed while changing the instruction. Such a pair, however, may lack a corresponding demonstrated action target. Prior work provides action supervision for such changed instructions by synthesizing counterfactual actions using policies trained on simpler instructions (Glossop et al., 2025) or by constructing action sequences that connect trajectories for the original and changed instructions (Chen et al., 2026). In our compositional setting, the changed instruction specifies a new combination of demonstrated skills, so the skill currently required under that instruction has already been demonstrated. Rather than constructing an action target for the counterfactual pair, we use demonstrated executions of the required skill to train the policy so that its action prediction under the changed instruction reflects that skill.

3 Problem Formulation

We consider a VLA policy $\pi(\mathbf{a} \mid o, \ell)$ that predicts a chunk of H future actions $\mathbf{a} \in \mathbb{R}^{H \times d_a}$ from a visual observation o and a language instruction ℓ, where d_a denotes the action dimension. We fine-tune the policy on robot demonstrations of tasks that share a fixed sequence of operations.

Let $(g_1, \ldots, g_K)$ denote this sequence, such as picking followed by placing. For each operation g_k, a task specifies an entity $e_k \in \mathcal{E}_k$, where $\mathcal{E}_k$ denotes the set of possible entities for that operation, such as objects to manipulate or target locations. We define a skill as an operation–entity pair (g_k, e_k). Because the operation sequence is fixed, the entity tuple $c = (e_1, \ldots, e_K)$ uniquely determines a task and its sequence of constituent skills. The language instruction ℓ specifies c. When the current operation is g_k, we refer to the skill (g_k, e_k) specified by the instruction as the *currently required skill*. We assume that, for a fixed observation and currently required skill, changing entity assignments for future operations does not change the action required for the current operation.

Let $\mathcal{D}$ denote the demonstration dataset used for fine-tuning. For each demonstration $d \in \mathcal{D}$, let c_d denote the entity tuple specified by its instruction. The demonstration records the execution of the task determined by c_d. It therefore contains an execution of each constituent skill in that task. We define the set of demonstrated combinations as

$$\mathcal{C}_{\mathcal{D}} = \{\, c_d : d \in \mathcal{D} \,\} \subseteq \mathcal{E}_1 \times \cdots \times \mathcal{E}_K. \tag{1}$$

We study generalization to tasks with entity tuples $c \notin \mathcal{C}_{\mathcal{D}}$ for which every constituent skill has been demonstrated in $\mathcal{D}$. Our goal is to fine-tune the policy on $\mathcal{D}$ so that it follows instructions for such tasks while maintaining performance on demonstrated combinations. The challenge is that constituent skills are demonstrated in other task combinations and may involve different observations. Because the same skill may require different actions across observations, these demonstrations do not specify how the skills should be executed in a new combination. This motivates reusing supervision from demonstrated skill executions while accounting for how the required action depends on the observation.

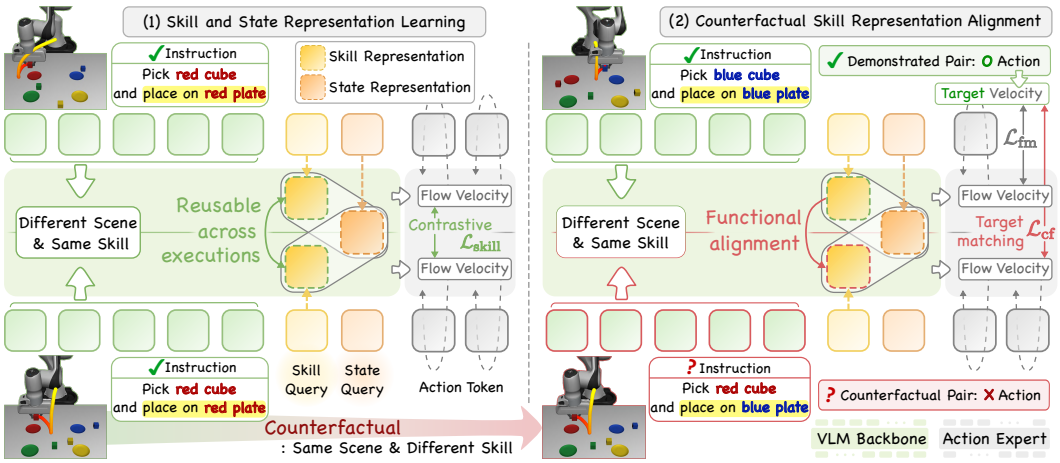

Figure 2: Overview of CRAFT. Yellow instruction highlights mark the currently required skill. (1) We encourage reuse of skill representations across executions by contrasting prediction errors while holding the state and target velocity fixed (target velocity and different-skill replacements omitted for clarity). (2) We combine the counterfactual skill representation with a state representation from a demonstration of the required skill. Its target velocity supervises the prediction, providing functional alignment (skill-changing case shown). "O/X Action" indicates whether the corresponding pair has a demonstrated action.

4 Method: CRAFT

To train the policy on skill combinations absent from the demonstrations, we form counterfactual pairs by holding a demonstration observation fixed while changing the instruction to specify a new combination. Each pair lacks a corresponding demonstrated action target, even though the skill currently required under the changed instruction has already been demonstrated in the dataset. Actions from those demonstrations cannot generally be used as targets for the pair because the same skill can require different actions under different observations. We therefore propose **CRAFT** (**C**ounterfactual Skill-**R**epresentation **A**lignment for **F**ine-**T**uning; Figure 2), which aligns each counterfactual pair's skill representation with the currently required skill under the changed instruction. The skill representation conditions action prediction on the currently required skill and is trained to be reusable across executions of the same skill. Instead of using an action from another execution as the counterfactual pair's target, CRAFT uses the pair's skill representation to condition action prediction for a demonstrated execution of the required skill. We supervise this prediction with that execution's action target, training the pair's skill representation to reflect the required skill.

4.1 Skill and State Representations

To transfer supervision across executions of the same skill, we introduce a *skill representation* z^{skill} that reflects the currently required skill and is trained to be reusable across those executions. Because the same skill may require different actions across observations, we use it with a *state representation* z^{state} that captures relevant visual information; together, they condition action prediction.

For the VLA architectures we consider (Black et al., 2024; 2025; Bjorck et al., 2025), we describe the policy as a pretrained VLM backbone f_ϕ coupled with an action expert p_θ. The backbone produces a vision-language representation $z = f_\phi(o, \ell)$ that conditions the action expert: $\pi(\mathbf{a} \mid o, \ell) = p_\theta(\mathbf{a} \mid z)$.

To obtain the skill and state representations, we add two sets of learnable query tokens, Q^{skill} and Q^{state}, to the VLM token sequence. With distinct attention masks, the skill queries attend to image and text tokens needed to represent the currently required skill in z^{skill}, while state queries attend only to image tokens to produce z^{state} (Appendix B.1). The action expert is then conditioned on both representations:

$$\pi(\mathbf{a} \mid o, \ell) = p_\theta(\mathbf{a} \mid z^{\text{skill}}, z^{\text{state}}), \qquad (z^{\text{skill}}, z^{\text{state}}) = f_\phi(o, \ell; Q^{\text{skill}}, Q^{\text{state}}). \tag{2}$$

4.2 Learning Skill Representations from Demonstrations

To train z^{skill} to reflect the currently required skill, we encourage two complementary properties. A skill representation should be reusable for action prediction across executions of the same skill. At

the same time, representations of different skills should remain distinguishable in action prediction. Together, these properties motivate a contrastive objective that compares action-prediction errors obtained using skill representations from same- and different-skill executions.

Each training sample i consists of an observation o_i, an instruction ℓ_i, and a demonstrated action chunk $\mathbf{a}_i$. For the VLA models we consider, the action expert p_θ is trained with flow matching (Lipman et al., 2022); we denote its velocity prediction function by v_θ. Given a noisy action chunk x_i^t at flow time t, v_θ is trained to predict the target velocity u_i. For fixed i and flow time t, we replace only its skill representation with z_j^{skill} from another execution. Thus, only the skill representation varies across the predictions being compared. We define the resulting velocity prediction $v_{i\leftarrow j}$ and score $s_{i\leftarrow j}$ as

$$v_{i\leftarrow j} = v_\theta\left(x_i^t, t; z_i^{\text{state}}, z_j^{\text{skill}}\right), \qquad s_{i\leftarrow j} = -\frac{\|v_{i\leftarrow j} - u_i\|_2^2}{\sigma_i}, \tag{3}$$

where σ_i normalizes the prediction-error scale for sample i (Appendix B.3.1). A larger $s_{i\leftarrow j}$ therefore indicates lower error in predicting the target velocity u_i.

For a sample i, the score should be higher for a skill representation from another execution of the same skill than for one from an execution of a different skill. This comparison requires a label indicating the skill executed at each training sample. Because the demonstrations do not provide these labels, we use a VLM to segment each demonstration according to the skill being executed and assign the corresponding label to each sample (Appendix B.2). For each i, we sample a positive p from another execution of the same skill and a set of negatives $\mathcal{N}_i$ from executions of different skills. Using the scores in Equation 3, we define the contrastive objective

$$\mathcal{L}_{\text{skill}} = -\mathbb{E}_i\left[\log \frac{\exp(s_{i\leftarrow p}/\tau)}{\exp(s_{i\leftarrow p}/\tau) + \sum_{n\in\mathcal{N}_i}\exp(s_{i\leftarrow n}/\tau)}\right], \tag{4}$$

where τ is a temperature. Thus, $\mathcal{L}_{\text{skill}}$ encourages skill representations to be reusable across same-skill executions and to preserve distinctions between skills in action prediction.

4.3 Counterfactual Supervision from Demonstrated Skills

For sample i, we form a counterfactual pair by keeping its observation o_i fixed and replacing its instruction ℓ_i with a counterfactual instruction $\tilde{\ell}_i$ that specifies a skill combination absent from the demonstrations. We train the counterfactual pair's skill representation to reflect the skill currently required under $\tilde{\ell}_i$ by using it to condition action prediction for a demonstrated execution of that skill.

We construct $\tilde{\ell}_i$ by changing the entities specified for the current or future operations. Assignments for completed operations remain unchanged because their outcomes are already reflected in o_i. We use a reference sample r from the demonstrations to provide the state representation and supervision target. If only the entities specified for future operations change, the currently required skill remains unchanged. For this *skill-preserving* counterfactual, we use the original sample as the reference ($r = i$). If the entity specified for the current operation changes, the currently required skill changes. For this *skill-changing* counterfactual, we select r from a demonstrated execution of the required skill.

For either counterfactual type, we condition the action expert on the counterfactual pair's skill representation and the reference sample's state representation, and supervise the resulting velocity prediction with the reference sample's target velocity u_r. Let $\tilde{i}_{\text{c}}$ and $\tilde{i}_{\text{p}}$ denote the skill-changing and skill-preserving counterfactual pairs of sample i, with skill representations $\tilde{z}_{i,\text{c}}^{\text{skill}}$ and $\tilde{z}_{i,\text{p}}^{\text{skill}}$, respectively. We define the corresponding velocity predictions and counterfactual objective as

$$v_{r\leftarrow\tilde{i}_{\text{c}}} = v_\theta\left(x_r^t, t; z_r^{\text{state}}, \tilde{z}_{i,\text{c}}^{\text{skill}}\right), \qquad v_{i\leftarrow\tilde{i}_{\text{p}}} = v_\theta\left(x_i^t, t; z_i^{\text{state}}, \tilde{z}_{i,\text{p}}^{\text{skill}}\right) \tag{5}$$

$$\mathcal{L}_{\text{cf}} = \underbrace{\mathbb{E}_{i,r}\left\|v_{r\leftarrow\tilde{i}_{\text{c}}} - u_r\right\|_2^2}_{\mathcal{L}_{\text{change}}} + \underbrace{\mathbb{E}_i\left\|v_{i\leftarrow\tilde{i}_{\text{p}}} - u_i\right\|_2^2}_{\mathcal{L}_{\text{preserve}}}. \tag{6}$$

Thus, $\mathcal{L}_{\text{cf}}$ trains the counterfactual pair's skill representation to reflect the skill required under $\tilde{\ell}_i$ in action prediction. In this way, CRAFT extends supervision to combinations absent from the demonstrations without requiring corresponding demonstrated action targets.

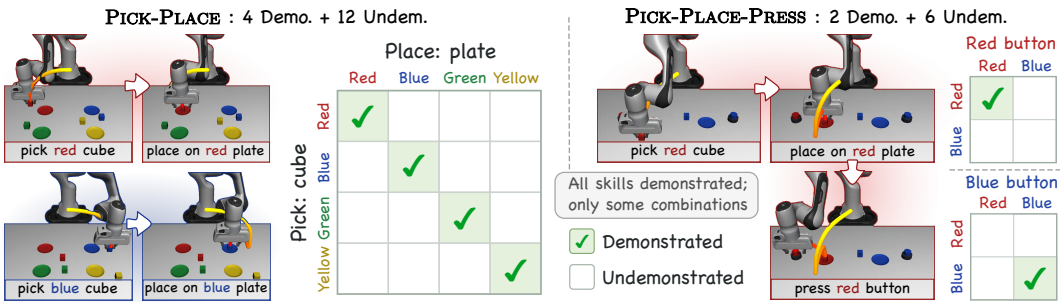

Figure 3: Compositional benchmarks with example demonstrations and task-combination matrices (Appendix C). Demonstrations cover all constituent skills but only a subset of combinations (checked cells); empty cells denote undemonstrated combinations. Rows and columns indicate cube and plate colors, respectively. The two PICK-PLACE-PRESS matrices correspond to red and blue buttons, respectively.

4.4 TRAINING OBJECTIVE

We combine the standard flow-matching loss for demonstrated action chunks with $\mathcal{L}_{\text{skill}}$ and $\mathcal{L}_{\text{cf}}$. Using the notation from Section 4.2, the flow-matching loss is $\mathcal{L}_{\text{fm}} = \mathbb{E}_i \|v_{i \leftarrow i} - u_i\|_2^2$, where $v_{i \leftarrow i}$ denotes the velocity prediction conditioned on sample i's own skill and state representations and u_i is the corresponding target velocity. The total training objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{fm}} + \beta \mathcal{L}_{\text{skill}} + \gamma \mathcal{L}_{\text{cf}}, \tag{7}$$

where β and γ are the weights for $\mathcal{L}_{\text{skill}}$ and $\mathcal{L}_{\text{cf}}$, respectively.

We use loss-specific gradient routing. $\mathcal{L}_{\text{fm}}$ updates all trainable components, whereas $\mathcal{L}_{\text{skill}}$ stops gradients through the state representation and updates the skill-representation pathway and the action expert. For $\mathcal{L}_{\text{cf}}$, we keep the action expert fixed and stop gradients through the reference state representation; gradients still propagate through the expert to the counterfactual skill representation. We optimize all three objectives jointly in a single training stage (Appendix B.3.3).

5 EXPERIMENTS

In simulation, we evaluate whether policies fine-tuned with CRAFT generalize to undemonstrated combinations while maintaining performance on demonstrated ones (Section 5.2). We then examine whether policy responses to instruction changes during execution reflect the currently required skill and whether skill representations are organized by skill across executions (Section 5.3). We assess how the objectives for learning reusable skill representations and counterfactual supervision contribute to compositional generalization (Section 5.4). Finally, we evaluate CRAFT on a real robot (Section 5.5).

5.1 EXPERIMENTAL SETUP

Benchmarks and task splits. We introduce two compositional benchmarks, PICK-PLACE and PICK-PLACE-PRESS (Figure 3). In both benchmarks, the fine-tuning demonstrations cover every constituent skill but only a subset of the possible combinations. PICK-PLACE has two operations—picking a cube and placing it on a plate—with four entities per operation, distinguished by color, yielding 16 combinations. The four same-color combinations are demonstrated; we evaluate the remaining 12. PICK-PLACE-PRESS adds a third operation—pressing a button—and uses two entities per operation, distinguished by color, yielding eight combinations. The two same-color combinations are demonstrated; we evaluate the remaining six. We collect 50 demonstrations per demonstrated combination. We will release both benchmarks and their demonstration datasets.

Policies and baselines. We evaluate CRAFT on three VLA models: π_0 (Black et al., 2024), $\pi_{0.5}$ (Black et al., 2025), and GR00T N1.7 (Bjorck et al., 2025). These models use different VLM–action-expert interfaces: the π family uses layer-wise attention to VLM features, whereas GR00T conditions its action expert on the VLM output. For each model, we compare three standard fine-tuning strategies: *Full* updates both the VLM and action expert, *LoRA* (Hu et al., 2021) adapts the VLM with LoRA while fully training the action expert, and *Frozen* freezes the VLM and trains only the action expert. CRAFT uses the *Full* setting. We additionally compare with CAG-VA (Fang et al., 2025), which mitigates vision shortcuts, and a *Skill Composition* baseline inspired by prior

Table 1: Success rates (%) on demonstrated (Demo.) and undemonstrated (Undem.) combinations across three VLA models and two benchmarks. We report means and standard deviations over three evaluation seeds.

	$\pi_{0.5}$		π_0		GR00T N1.7	
Method	**Demo.**	**Undem.**	**Demo.**	**Undem.**	**Demo.**	**Undem.**
PICK–PLACE						
FT (*Full*)	$100_{\pm 0}$	$8.4_{\pm 0.3}$	$100_{\pm 0}$	$2.8_{\pm 0.2}$	$98.8_{\pm 0.3}$	$1.1_{\pm 0.1}$
FT (*LoRA*)	$100_{\pm 0}$	$46.3_{\pm 0.9}$	$97.5_{\pm 0}$	$1.0_{\pm 0.2}$	$97.8_{\pm 0.6}$	$2.6_{\pm 0.3}$
FT (*Frozen*)	$99.0_{\pm 0}$	$21.1_{\pm 0.2}$	$48.2_{\pm 1.4}$	$3.0_{\pm 0.5}$	$92.3_{\pm 1.3}$	$2.2_{\pm 0.1}$
CAG-VA[a]	$96.3_{\pm 1.5}$	$66.5_{\pm 0.5}$	$50.5_{\pm 1.3}$	$6.8_{\pm 0.6}$	$88.7_{\pm 1.3}$	$4.9_{\pm 0.5}$
Skill Composition[b]	$100_{\pm 0}$	$25.7_{\pm 0.4}$	$99.5_{\pm 0.0}$	$12.7_{\pm 0.4}$	$99.7_{\pm 0.6}$	$13.1_{\pm 0.7}$
CRAFT (ours)	$99.0_{\pm 0.9}$	$84.2_{\pm 0.3}$	$94.0_{\pm 0.9}$	$64.3_{\pm 0.5}$	$99.0_{\pm 0.5}$	$72.1_{\pm 0.9}$
PICK–PLACE–PRESS						
FT (*Full*)	$100_{\pm 0}$	$0.7_{\pm 0.0}$	$99.7_{\pm 0.6}$	$0.3_{\pm 0.0}$	$97.7_{\pm 2.3}$	$1.1_{\pm 0.2}$
FT (*LoRA*)	$100_{\pm 0}$	$15.4_{\pm 0.8}$	$99.3_{\pm 0.6}$	$0.7_{\pm 0.0}$	$99.3_{\pm 1.2}$	$1.0_{\pm 0}$
FT (*Frozen*)	$98.0_{\pm 1.0}$	$0.9_{\pm 0.2}$	$82.0_{\pm 3.5}$	$1.5_{\pm 0.4}$	$97.7_{\pm 0.6}$	$1.1_{\pm 0.2}$
CAG-VA[a]	$92.7_{\pm 3.5}$	$21.8_{\pm 1.9}$	$69.0_{\pm 2.6}$	$1.1_{\pm 0.2}$	$88.7_{\pm 2.5}$	$2.1_{\pm 0.2}$
Skill Composition[b]	$99.7_{\pm 0.6}$	$5.8_{\pm 0.2}$	$99.3_{\pm 0.6}$	$0.3_{\pm 0.0}$	$99.3_{\pm 0.6}$	$2.0_{\pm 0.0}$
CRAFT (ours)	$99.7_{\pm 0.6}$	$62.6_{\pm 3.9}$	$95.3_{\pm 0.6}$	$40.7_{\pm 1.5}$	$99.3_{\pm 0.6}$	$31.9_{\pm 1.1}$

For FT, *Full*, *LoRA*, and *Frozen* denote VLM adaptation; the action expert is fully trained in all cases.
[a] For CAG-VA (Fang et al., 2025), we tune the guidance strength for *Full*, *LoRA*, and *Frozen* and report the selected configuration for each VLA model and benchmark; selection details are provided in Appendix D.5.
[b] *Skill Composition* uses the *Full* setting to fine-tune each VLA on constituent skill segments and sequentially executes the required skills at deployment.

work that composes learned behaviors at deployment (Li, 2025; Zhai et al., 2026). *Skill Composition* fine-tunes the VLA on constituent skill segments from the demonstrations and sequentially executes the required skills at deployment. All methods use the same robot demonstrations.

Evaluation protocol and metrics. We use task success rate as the primary metric, counting a rollout as successful if the policy completes all stages specified by the instruction within a fixed horizon. For each combination, 50 initial states are shared across methods and evaluation seeds. For demonstrated and undemonstrated combinations, we report the mean and standard deviation of success rates over three evaluations of a single trained checkpoint with different environment and action-sampling seeds. Training, checkpoint selection, and deployment details are provided in Appendix B.4.

5.2 COMPOSITIONAL GENERALIZATION

Table 1 reports success rates on demonstrated and undemonstrated combinations. Across all VLA models and benchmarks, CRAFT achieves the highest success on undemonstrated combinations while maintaining high performance on demonstrated ones. By contrast, standard fine-tuning often performs well on demonstrated combinations but achieves substantially lower success on the remaining combinations. This gap indicates that high success on demonstrated combinations does not by itself imply the ability to recombine their constituent skills.

Changing the adaptation method alone does not close this gap. The gap persists across *Full*, *LoRA*, and *Frozen*; *Full* has low success on these combinations despite updating the entire VLM. Consistent with our vision-shortcut hypothesis, this may arise because policies can fit demonstrated actions using observation–action correlations that do not generalize to new skill combinations.

CAG-VA and *Skill Composition* represent two alternatives: strengthening language conditioning and composing learned skills at deployment. CAG-VA improves generalization in several settings, but its gains vary substantially across models and benchmarks. *Skill Composition* improves over *Full* on PICK-PLACE for all three models, but its gains are much smaller on PICK-PLACE-PRESS.

In contrast, CRAFT's improvement over the baselines persists across all three VLA models on PICK-PLACE-PRESS, suggesting that its gains are not specific to PICK-PLACE. Nevertheless, a substantial gap remains between demonstrated and undemonstrated performance, particularly on PICK-PLACE-PRESS. We provide detailed compositional-generalization results in Appendix D.1.

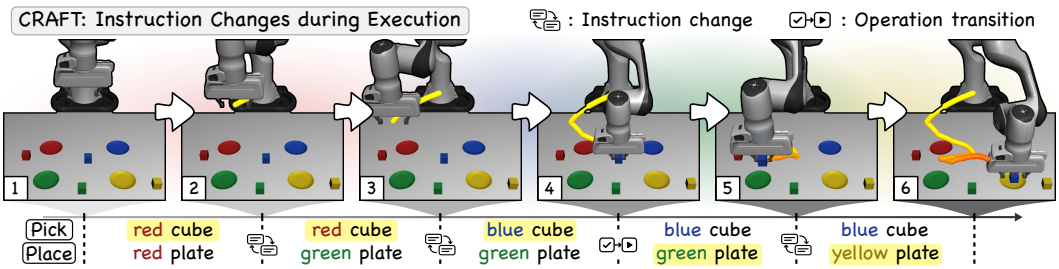

Figure 4: Responses to instruction changes. Yellow highlights identify the entity assigned to the current operation. The robot approaches the red cube and maintains this behavior when only the future placement target changes (1→2→3). It redirects its approach when the picking target changes (3→4), then changes its destination during placement (5→6), placing the blue cube on the yellow plate.

5.3 Behavior and Representation Analysis

We examine whether policy responses to instruction changes during execution reflect the currently required skill (Figure 4; Appendix D.2). When an instruction change leaves the currently required skill unchanged, the policy continues executing that skill. When the required skill changes, the policy begins executing the newly required skill. Together, these responses suggest that the policy remains responsive to the instruction throughout execution. Notably, it does so from a state reached while executing a different skill. This suggests the policy can execute the required skill from states reached during other skill executions, rather than only from states encountered in demonstrations of that skill.

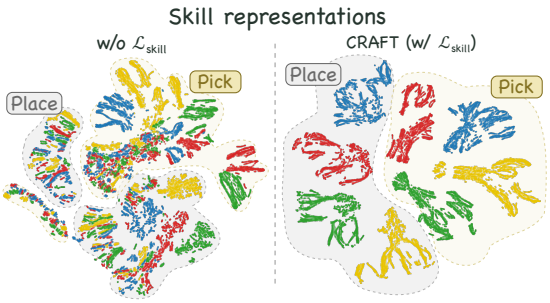

Figure 5: Skill representations learned with and without $\mathcal{L}_{\text{skill}}$ on PICK-PLACE, independently projected with t-SNE. Each point represents a demonstration frame, colored by the entity of the currently required skill. Shaded regions indicate pick and place groups.

Figure 5 examines how $\mathcal{L}_{\text{skill}}$ shapes the learned skill representations. Without $\mathcal{L}_{\text{skill}}$, representations from different entities are more mixed within the pick and place groups. With $\mathcal{L}_{\text{skill}}$, they become more clearly organized by the currently required skill, forming entity-specific groups within each operation. This organization is consistent with $\mathcal{L}_{\text{skill}}$ encouraging skill-dependent structure in the learned representations. We evaluate whether these representations are functionally reusable across executions in Appendix D.3.

5.4 Training Objective Ablations

We compare combinations of auxiliary objectives while keeping the architecture and flow-matching loss unchanged (Table 2). Success on undemonstrated combinations is low for both models with $\mathcal{L}_{\text{fm}}$ alone and remains low when $\mathcal{L}_{\text{skill}}$ is added. This objective encourages skill representations to be reusable across same-skill executions, but does not directly supervise representations induced by counterfactual instructions. Under this setting, adding $\mathcal{L}_{\text{preserve}}$ provides no clear improvement, whereas $\mathcal{L}_{\text{change}}$ improves success on undemonstrated combinations for both models.

Combining $\mathcal{L}_{\text{change}}$ and $\mathcal{L}_{\text{preserve}}$ yields substantially higher success on undemonstrated combinations than using either counterfactual term separately. The combined objective also maintains high success on demonstrated combinations. These results suggest that counterfactual supervision benefits from covering both instruction changes that alter the currently required skill and those that leave it unchanged. We provide additional ablations and sensitivity analyses in Appendix D.4.

Table 2: Success rates (%) on PICK-PLACE for training objective ablations. (mean ± std.; three evaluation seeds)

Auxiliary objectives			$\pi_{0.5}$		GR00T N1.7	
Skill	**Pres.**	**Change**	**Demo.**	**Undem.**	**Demo.**	**Undem.**
			$88.8_{\pm 0.6}$	$1.9_{\pm 0.4}$	$99.3_{\pm 0.3}$	$3.8_{\pm 0.1}$
✓			$99.2_{\pm 0.6}$	$0.3_{\pm 0.4}$	$97.2_{\pm 1.0}$	$6.1_{\pm 0.5}$
✓	✓		$99.8_{\pm 0.3}$	$0.4_{\pm 0.2}$	$97.3_{\pm 0.6}$	$5.3_{\pm 0.3}$
✓		✓	$97.7_{\pm 0.3}$	$14.7_{\pm 0.8}$	$94.5_{\pm 1.3}$	$12.1_{\pm 0.4}$
CRAFT (all)			$99.0_{\pm 0.9}$	$84.2_{\pm 0.3}$	$99.0_{\pm 0.5}$	$72.1_{\pm 0.9}$

All variants include $\mathcal{L}_{\text{fm}}$. **Skill**, **Pres.**, and **Change** denote $\mathcal{L}_{\text{skill}}$, $\mathcal{L}_{\text{preserve}}$, and $\mathcal{L}_{\text{change}}$, respectively. CRAFT uses all three auxiliary objectives.

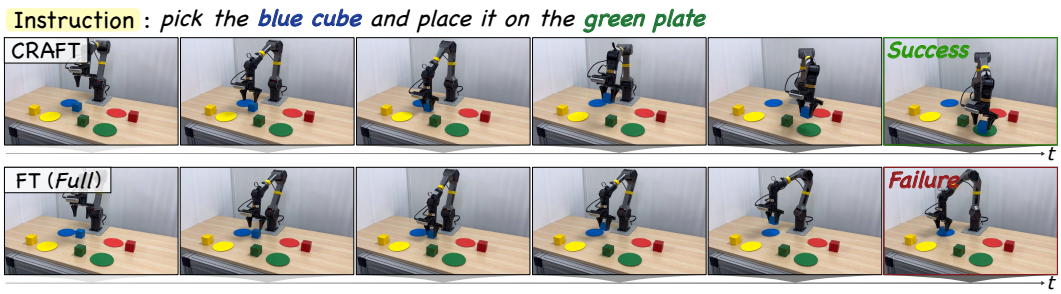

Figure 6: Real-robot execution of an undemonstrated combination. The instruction requires picking the blue cube and placing it on the green plate; this combination is absent from the demonstrations, although both constituent skills have been demonstrated. Both CRAFT and *Full* pick the instructed blue cube, but CRAFT places it on the instructed green plate, whereas *Full* places it on the blue plate, corresponding to the demonstrated blue-to-blue combination.

5.5 Real-Robot Evaluation

We evaluate whether CRAFT improves compositional generalization on a real robot. The real-robot setup follows the task structure of Pick-Place, with four differently colored cubes and four plates of the same four colors. A Piper 6-DoF robot performs the tasks using observations from third-person and wrist cameras. The demonstration dataset contains 20 demonstrations for each of the four same-color combinations, leaving the remaining 12 combinations undemonstrated. We compare CRAFT with FT (*Full*) on $\pi_{0.5}$; both use the same demonstration dataset and are evaluated over five trials per combination.

Figure 6 provides a qualitative comparison of *Full* and CRAFT on an undemonstrated combination. CRAFT achieves higher success on undemonstrated combinations than *Full* while maintaining high success on demonstrated combinations (Table 3). These results suggest that CRAFT can improve real-robot compositional generalization when demonstrations cover the constituent skills but not every combination. We provide additional real-robot rollouts in Appendix D.6.

Table 3: Real-robot task success (successful trials / total trials) on demonstrated (Demo.) and undemonstrated (Undem.) combinations.

Method	Demo.	Undem.
FT (*Full*)	19/20	9/60
CRAFT (ours)	19/20	43/60

6 Discussion

Where the supervision comes from. Our setting lacks demonstrations for some skill combinations, but each constituent skill has already been demonstrated. CRAFT therefore uses demonstrated executions of the currently required skill as the source of supervision, rather than synthesizing trajectories for new combinations or transferring actions directly across observations.

What the results suggest. High success on demonstrated combinations did not consistently translate into high success on undemonstrated combinations. The gains from CRAFT suggest that existing demonstrations can provide supervision beyond the skill combinations they contain.

Limitations. Our formulation assumes a fixed operation sequence and that every constituent skill in an undemonstrated combination appears in the demonstrations. Tasks requiring new skills or operation sequences that vary with the instruction are outside the scope of our formulation.

7 Conclusion

We studied whether VLAs fine-tuned on demonstrations can recombine demonstrated skills into combinations absent from those demonstrations. Motivated by the vision shortcut that can arise during fine-tuning, we proposed CRAFT, which uses counterfactual supervision to train the policy to follow the currently required skill without requiring demonstrations of every skill combination. Across three VLA models and two simulation benchmarks, CRAFT improved success on undemonstrated combinations while maintaining high performance on demonstrated ones. CRAFT also improved compositional generalization on a real robot. These results show that compositional generalization can be improved by making better use of constituent-skill supervision present in demonstrations.

Ethics Statement

This work studies compositional generalization for vision-language-action policies in simulation and on a real robot. Our experiments do not involve human or animal subjects, personally identifiable information, or sensitive personal data.

We use publicly released VLA models (Black et al., 2024; 2025; Pertsch et al., 2025; Bjorck et al., 2025) and construct our simulation benchmarks by adapting the LIBERO environment (Liu et al., 2023) to the compositional setting studied in this work. The demonstration datasets for the simulation benchmarks and real-robot experiments consist of robot manipulation trajectories collected for the corresponding tasks.

The real-robot experiments are conducted in a controlled laboratory setting and evaluate compositional generalization rather than safety for unconstrained real-world deployment.

Reproducibility Statement

To support reproducibility, we provide implementation details in the main paper and additional architecture details, training configurations, and hyperparameters in the appendix. We also specify the evaluation protocols used for the reported experiments. The datasets and evaluation environments developed for this work, together with the corresponding training code, will be publicly released.

AI Use Disclosure

The authors used generative AI tools for feedback on research methodology and experiments, assistance with code debugging, and manuscript editing for clarity. All AI-assisted content was reviewed and finalized by the authors. The authors take full responsibility for the final content of this work.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Zhenyang Chen, Alan Tian, Liquan Wang, Benjamin Joffe, Yingyan Celine Lin, Yuxiao Chen, Siddharth Karamcheti, and Danfei Xu. Resteer: Quantifying and refining the steerability of multitask robot policies. *arXiv preprint arXiv:2603.17300*, 2026.

Sombit Dey, Jan-Nico Zaech, Nikolay Nikolov, Luc Van Gool, and Danda Pani Paudel. ReVLA: Reverting visual domain limitation of robotic foundation models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8679–8686. IEEE, 2025.

Yehonatan Elisha, Oren Barkan, and Noam Koenigstein. Concept-guided fine-tuning: Steering ViTs away from spurious correlations to improve robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Yu Fang, Yuchun Feng, Dong Jing, Jiaqi Liu, Yue Yang, Zhenyu Wei, Daniel Szafir, and Mingyu Ding. When vision overrides language: Evaluating and mitigating counterfactual failures in VLAs. *arXiv preprint arXiv:2505.16875*, 2025.

Jensen Gao, Annie Xie, Ted Xiao, Chelsea Finn, and Dorsa Sadigh. Efficient data collection for robotic manipulation via compositional generalization. *arXiv preprint arXiv:2403.05110*, 2024.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

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.

Shresth Grover, Akshay Gopalkrishnan, Bo Ai, Henrik I Christensen, Hao Su, and Xuanlin Li. Enhancing generalization in vision-language-action models by preserving pretrained representations. *arXiv preprint arXiv:2509.11417*, 2025.

Xinyu Guo, Bin Xie, Wei Chai, Xianchi Deng, Tiancai Wang, Zhengxing Wu, and Xingyu Chen. Priorvla: Prior-preserving adaptation for vision-language-action models. *arXiv preprint arXiv:2605.10925*, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations (ICLR)*, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Artur Kuramshin, Özgür Aslan, Cyrus Neary, and Glen Berseth. Task robustness via re-labelling vision-action robot data. *arXiv preprint arXiv:2606.10918*, 2026.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pp. 2873–2882. PMLR, 2018.

Quanyi Li. Vlas are confined yet capable of generalizing to novel instructions. *arXiv preprint arXiv:2505.03500*, 2025.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action models via instruction-driven routing & sparsification. *Advances in neural information processing systems*, 38:137646–137675, 2026.

Shijie Lian, Bin Yu, Xiaopeng Lin, Laurence T Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, and Kai Chen. Langforce: Bayesian decomposition of vision language action models via latent action queries. *arXiv preprint arXiv:2601.15197*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

Kai Peng, Jie Lu, and Xiaojiang Peng. Unleashing more actions via action compositional training for vla models. *arXiv preprint arXiv:2607.00351*, 2026.

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.

Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. Revisiting compositional generalization capability of large language models considering instruction following ability. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31219–31238, 2025.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Qingyi Si, Fandong Meng, Mingyu Zheng, Zheng Lin, Yuanxin Liu, Peng Fu, Yanan Cao, Weiping Wang, and Jie Zhou. Language prior is not the only shortcut: A benchmark for shortcut learning in VQA. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 3698–3712, 2022.

Stefan Tabakov, Asen Popov, Dimitar Dimitrov, S Ensiye Kiyamousavi, Boris Kraychev, and Vladimir Hristov. Atomic action slicing: Planner-aligned options for generalist vla agents. In *Proceedings of the 41st ACM/SIGAPP Symposium on Applied Computing*, pp. 780–782, 2026.

Arnas Uselis, Andrea Dittadi, and Seong Joon Oh. Does data scaling lead to visual compositional generalization? *arXiv preprint arXiv:2507.07102*, 2025.

Xingyu Wan, Qiaoying Teng, Jun Chen, Yonghan Lu, Deqi Yuan, and Zhe Liu. Eliminating language bias for medical visual question answering with counterfactual contrastive training. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 194–204. Springer, 2025.

Senwei Xie, Yuntian Zhang, Zhenzhou Tan, Ruiping Wang, Pengwei Wang, Shanghang Zhang, and Xilin Chen. Skillnet: Hierarchical skill modeling for compositional generalization in vision-language action models. In *Forty-third International Conference on Machine Learning*, 2026.

Youguang Xing, Xu Luo, Junlin Xie, Lianli Gao, Hengtao Shen, and Jingkuan Song. Shortcut learning in generalist robot policies: The role of dataset diversity and fragmentation. *arXiv preprint arXiv:2505.06530*, 2025.

Kechun Xu, Zhenjie Zhu, Anzhe Chen, Shuqi Zhao, Qing Huang, Yifei Yang, Haojian Lu, Rong Xiong, Masayoshi Tomizuka, and Yue Wang. Seeing to act, prompting to specify: A bayesian factorization of vision language action policy. *arXiv preprint arXiv:2505.13275*, 2025.

Yajat Yadav, Zhiyuan Zhou, Andrew Wagenmaker, Karl Pertsch, and Sergey Levine. Robust fine-tuning of vision-language-action robot policies via parameter merging. In *International Conference on Learning Representations*, volume 2026, pp. 56385–56417, 2026.

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.

A MOTIVATING EXPERIMENTS

A.1 COMPOSITIONAL GENERALIZATION GAP IN FINE-TUNED VLA MODELS

Setup. LIBERO (Liu et al., 2023) is a simulation benchmark for language-conditioned manipulation and has been used to evaluate recent VLA models (Kim et al., 2024; Black et al., 2024; 2025; Bjorck et al., 2025). The LIBERO-GOAL suite contains ten tasks that share the same scene and initial-state distribution but differ in their goals. We focus on its six pick-and-place tasks for compositional evaluation; the remaining four tasks involve other manipulation objectives. Figure A.1 shows an example initial scene and the six pick-and-place tasks.

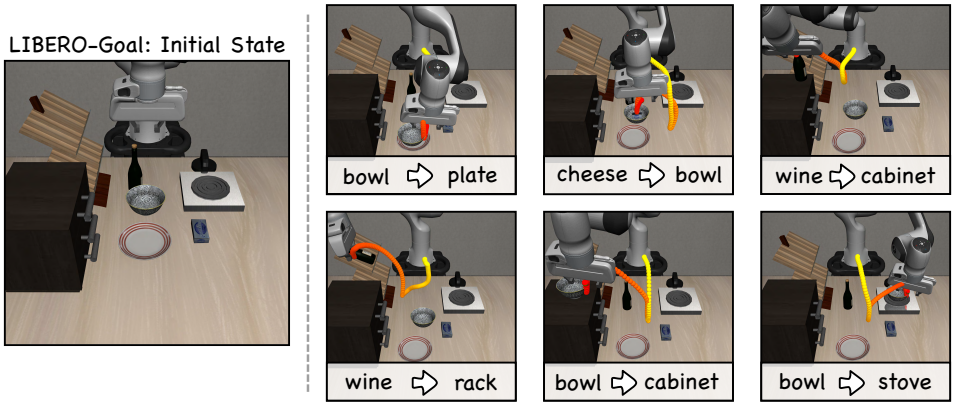

Figure A.1: An example initial scene and the six pick-and-place tasks from the LIBERO-GOAL suite. The first panel shows the initial scene, and the remaining panels show the six tasks with their instructions.

Constructing undemonstrated combinations. We represent each pick-and-place task by its object–target pair, corresponding to a pick skill and a place skill. The six original tasks define the demonstrated combinations marked with ✓ in Table A.1. The remaining nine object–target pairs are undemonstrated combinations, while every constituent pick and place skill appears in at least one demonstrated task. We manually select five of these nine combinations before model evaluation, excluding cases in which task feasibility or manipulation difficulty could confound compositional generalization. To avoid introducing new instruction phrasing, each constructed instruction reuses object and target phrases from the demonstrated pick-and-place tasks (Table A.3). For each constructed task, we define a new Behavior Domain Definition Language (BDDL) specification with the corresponding goal condition while preserving the suite's scene and initial-state distribution.

Table A.1: Object–target combinations derived from the six pick-and-place tasks in LIBERO-GOAL. ✓ marks a demonstrated combination, with the corresponding task index from Table A.2; ○ marks an undemonstrated combination used for evaluation. Blank cells denote undemonstrated combinations not evaluated.

	cabinet top	**rack**	**plate**	**stove**	**bowl**
wine bottle	✓ (1)	✓ (4)	○	○	
bowl	✓ (2)		✓ (3)	✓ (6)	
cream cheese	○		○	○	✓ (5)

Models and evaluation. We evaluate publicly released LIBERO-fine-tuned checkpoints of π_0 (Black et al., 2024), $\pi_{0.5}$ (Black et al., 2025), π_0-FAST (Pertsch et al., 2025), and GR00T N1.7 (Bjorck et al., 2025). We use each checkpoint without further training. For each model and task, we evaluate 50 trials for each of three evaluation seeds, sampling initial states according to the LIBERO-GOAL protocol. Each policy executes the first five actions of each predicted chunk before replanning.

Table A.2: Success rate (%) on the ten original LIBERO-GOAL tasks. Released checkpoints are evaluated on 50 trials per task and evaluation seed; we report mean ± std over three seeds. Task indices are used for cross-reference with Tables A.1 and A.3.

	Task	π_0	$\pi_{0.5}$	π_0-FAST	GR00T N1.7
pick-and-place					
1	put the wine bottle on top of the cabinet	$98.0_{\pm 1.6}$	$98.0_{\pm 1.6}$	$97.3_{\pm 0.9}$	$98.7_{\pm 0.9}$
2	put the bowl on top of the cabinet	$97.3_{\pm 1.9}$	$96.7_{\pm 0.9}$	$93.3_{\pm 6.6}$	$94.7_{\pm 1.9}$
3	put the bowl on the plate	$100.0_{\pm 0.0}$	$99.3_{\pm 0.9}$	$98.0_{\pm 0.0}$	$96.0_{\pm 1.6}$
4	put the wine bottle on the rack	$75.3_{\pm 1.9}$	$95.3_{\pm 2.5}$	$46.0_{\pm 1.6}$	$96.0_{\pm 1.6}$
5	put the cream cheese in the bowl	$98.7_{\pm 0.9}$	$98.0_{\pm 0.0}$	$99.3_{\pm 0.9}$	$93.3_{\pm 0.9}$
6	put the bowl on the stove	$98.0_{\pm 1.6}$	$97.3_{\pm 0.9}$	$98.7_{\pm 0.9}$	$98.0_{\pm 1.6}$
	mean	$94.6_{\pm 0.8}$	$97.4_{\pm 0.6}$	$88.8_{\pm 1.0}$	$96.1_{\pm 0.2}$
other					
7	open the top drawer and put the bowl inside	$82.0_{\pm 4.3}$	$93.3_{\pm 2.5}$	$88.7_{\pm 3.4}$	$86.7_{\pm 2.5}$
8	turn on the stove	$99.3_{\pm 0.9}$	$100.0_{\pm 0.0}$	$94.0_{\pm 0.0}$	$100.0_{\pm 0.0}$
9	open the middle drawer of the cabinet	$98.7_{\pm 0.9}$	$96.7_{\pm 0.9}$	$74.7_{\pm 2.5}$	$95.3_{\pm 3.4}$
10	push the plate to the front of the stove	$93.3_{\pm 2.5}$	$94.0_{\pm 2.8}$	$87.3_{\pm 3.4}$	$92.0_{\pm 1.6}$
	mean	$93.3_{\pm 1.4}$	$96.0_{\pm 1.1}$	$86.2_{\pm 1.2}$	$93.5_{\pm 1.1}$

Table A.3: Success rate (%) on the five constructed undemonstrated combinations. We report mean ± std over three evaluation seeds, with 50 trials per task and seed.

Instruction	π_0	$\pi_{0.5}$	π_0-FAST	**GR00T N1.7**
put the wine bottle on the plate	$6.0_{\pm 0.0}$	$52.0_{\pm 3.3}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
put the wine bottle on the stove	$2.0_{\pm 1.6}$	$77.3_{\pm 4.1}$	$0.7_{\pm 0.9}$	$0.0_{\pm 0.0}$
put the cream cheese on top of the cabinet	$0.0_{\pm 0.0}$	$78.7_{\pm 2.5}$	$7.3_{\pm 1.9}$	$6.7_{\pm 2.5}$
put the cream cheese on the plate	$4.0_{\pm 1.6}$	$24.7_{\pm 7.7}$	$88.0_{\pm 1.6}$	$0.7_{\pm 0.9}$
put the cream cheese on the stove	$2.7_{\pm 0.9}$	$22.0_{\pm 2.8}$	$1.3_{\pm 0.9}$	$2.7_{\pm 1.9}$
mean	$2.9_{\pm 0.4}$	$50.9_{\pm 2.4}$	$19.5_{\pm 0.7}$	$2.0_{\pm 1.0}$

Results. Across the four models, mean success on the six demonstrated combinations ranges from 88.8 to 97.4%, compared with 2.0 to 50.9% on the five undemonstrated combinations (Tables A.2 and A.3). The resulting demonstrated-to-undemonstrated gap ranges from 46.5 to 94.1 percentage points. Although the magnitude of the gap varies across models, all four show substantially lower success on undemonstrated combinations despite every constituent skill appearing in the demonstrated tasks. We analyze the associated failure behavior further in Appendix A.2.

A.2 FAILURE BEHAVIOR CONSISTENT WITH A VISION SHORTCUT

Appendix A.1 shows a substantial gap between demonstrated and undemonstrated combinations, even when every constituent skill has been demonstrated. We next examine how the policies fail on undemonstrated combinations and whether their behavior is consistent with the vision-shortcut hypothesis.

A representative failure on an undemonstrated combination. Figure A.2 shows a rollout from the fine-tuned $\pi_{0.5}$ policy under the undemonstrated instruction "put the wine bottle on the stove." Although this object–target combination is absent from the demonstrations, both constituent skills—picking the wine bottle and placing an object on the stove—have been demonstrated. The policy grasps the instructed wine bottle but places it on the rack rather than the stove. This outcome corresponds to the demonstrated task "put the wine bottle on the rack," illustrating a failure in which the policy completes a familiar demonstrated combination instead of the instructed one.

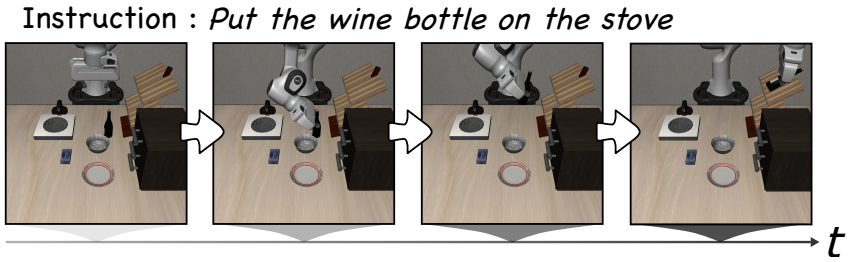

Figure A.2: Failure on an undemonstrated combination in LIBERO-GOAL. Given "put the wine bottle on the stove," the fine-tuned $\pi_{0.5}$ policy grasps the instructed object but places it on the rack, corresponding to the demonstrated "put the wine bottle on the rack" task.

Which demonstrated tasks failed rollouts resemble. To determine whether this behavior extends beyond a single example, we analyze the outcomes of rollouts on the five constructed undemonstrated combinations from Appendix A.1. For each rollout, we evaluate the goal predicates of the ten original LIBERO-GOAL tasks and record whether it completes (i) the instructed undemonstrated task, (ii) a demonstrated task that shares the object with the instruction, (iii) a demonstrated task that shares the target, (iv) another original task, or (v) none of them. If multiple task predicates become satisfied during a rollout, we assign the rollout to the task whose predicate is satisfied first. Table A.4 summarizes the results.

Across models, when a rollout completes one of the original demonstrated tasks instead of the instructed combination, it almost always completes a task that shares either the object or the target with the instruction. The dominant pattern differs across models: $\pi_{0.5}$ and π_0-FAST more often complete object-sharing tasks, whereas π_0 more often completes target-sharing tasks; GR00T N1.7 exhibits both patterns. Rollouts categorized as *None* satisfy none of the evaluated task goals. Manual inspection suggests that many of these cases result from earlier execution failures, such as failing to grasp the instructed object, rather than completion of an alternative task. Thus, when failed rollouts do complete an evaluated task, their outcomes are strongly concentrated on demonstrated combinations that partially match the instruction.

Table A.4: Outcomes of rollouts on the five constructed undemonstrated combinations. Values are percentages of trials pooled across combinations, reported as mean $\pm$ std over three seeds. *None* indicates that the rollout satisfies neither the instructed task goal nor any of the ten original LIBERO-GOAL task predicates.

Model	Instructed task	Shares the object	Shares the target	Other of the ten	None
π_0	$2.9_{\pm 0.4}$	$9.6_{\pm 0.7}$	$48.8_{\pm 0.6}$	$0.5_{\pm 0.4}$	$38.1_{\pm 1.2}$
$\pi_{0.5}$	$50.9_{\pm 2.4}$	$38.4_{\pm 1.2}$	$2.8_{\pm 0.3}$	$0.0_{\pm 0.0}$	$7.9_{\pm 0.9}$
π_0-FAST	$19.5_{\pm 0.7}$	$41.9_{\pm 2.0}$	$4.1_{\pm 0.2}$	$0.8_{\pm 0.7}$	$33.7_{\pm 0.8}$
GR00T N1.7	$2.0_{\pm 1.0}$	$39.3_{\pm 1.0}$	$25.5_{\pm 1.6}$	$1.9_{\pm 0.7}$	$31.3_{\pm 1.0}$

Behavior under an empty instruction. As a complementary qualitative probe, we evaluate the same fine-tuned $\pi_{0.5}$ policy with an empty instruction. Figure A.3 shows two example trials. Despite receiving no task instruction, the policy executes behaviors corresponding to demonstrated tasks: it turns on the stove in one trial and puts the bowl on the plate in the other. Although an empty instruction is outside the standard task setting, these examples show that demonstrated-task behaviors can arise from the visual observation without an explicit task instruction.

Summary. On undemonstrated combinations, failed rollouts that complete an evaluated task are strongly biased toward demonstrated combinations sharing a constituent with the instruction, while demonstrated-task behaviors can also arise under an empty instruction. These observations provide behavioral evidence for the vision-shortcut failure mode, while not excluding other contributing factors.

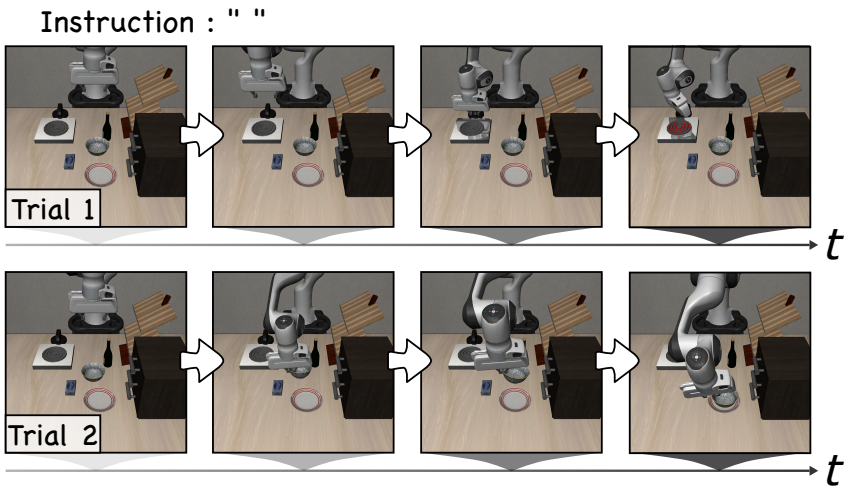

Figure A.3: Behavior of the fine-tuned $\pi_{0.5}$ policy under an empty instruction. Without a task instruction, the policy turns on the stove in one trial (top) and puts the bowl on the plate in another (bottom), both corresponding to demonstrated tasks.

B Method Details

B.1 Architecture

Query representations. CRAFT introduces learnable skill queries Q^{skill} and state queries Q^{state} to obtain the two representations described in Section 4.1. We use the same query-based representation design for the π family (π_0 and $\pi_{0.5}$) and GR00T N1.7, while adapting its implementation to each model's VLM–action expert interface. For the π family, the representations are layer-wise K/V tensors at the query positions; for GR00T, they are the final hidden states at those positions. Figure B.1 compares the original and CRAFT interfaces and shows the corresponding attention masks.

The π family. In the original π family, the action expert attends to the keys and values (K/V) associated with image and instruction tokens at each VLM backbone layer. In CRAFT, the expert instead attends to the K/V at the skill and state query positions, which constitute the skill and state representations, respectively.

GR00T. GR00T conditions its action expert on the final VLM features through cross-attention. In CRAFT, this conditioning uses only the final hidden states at the skill and state query positions.

Attention structure. Skill queries attend to image and instruction tokens, whereas state queries attend to visual tokens without directly attending to instruction tokens. Queries attend bidirectionally within their own group. The skill and state query groups do not attend to each other. Figure B.1(b) shows the attention masks used for each architecture.

Initialization. Each query token is initialized with the mean of the pretrained backbone's token embeddings plus independent Gaussian noise with standard deviation 0.02. The queries are learned during fine-tuning, with the same attention structure used at training and inference.

B.2 Skill Annotation and Counterfactual Instructions

Skill annotation. CRAFT uses operation–entity labels to identify the skill executed at each timestep and to select samples across demonstrations according to skill. In our tasks, the operation order is fixed and the instruction specifies the entity assigned to each operation. Thus, the VLM does not infer skill identities; it only locates transitions between the known operations. Following prior work on VLM-based demonstration segmentation (Kuramshin et al., 2026; Tabakov et al., 2026), we use a VLM to annotate demonstrations for both pick–place and pick–place–press tasks. We provide

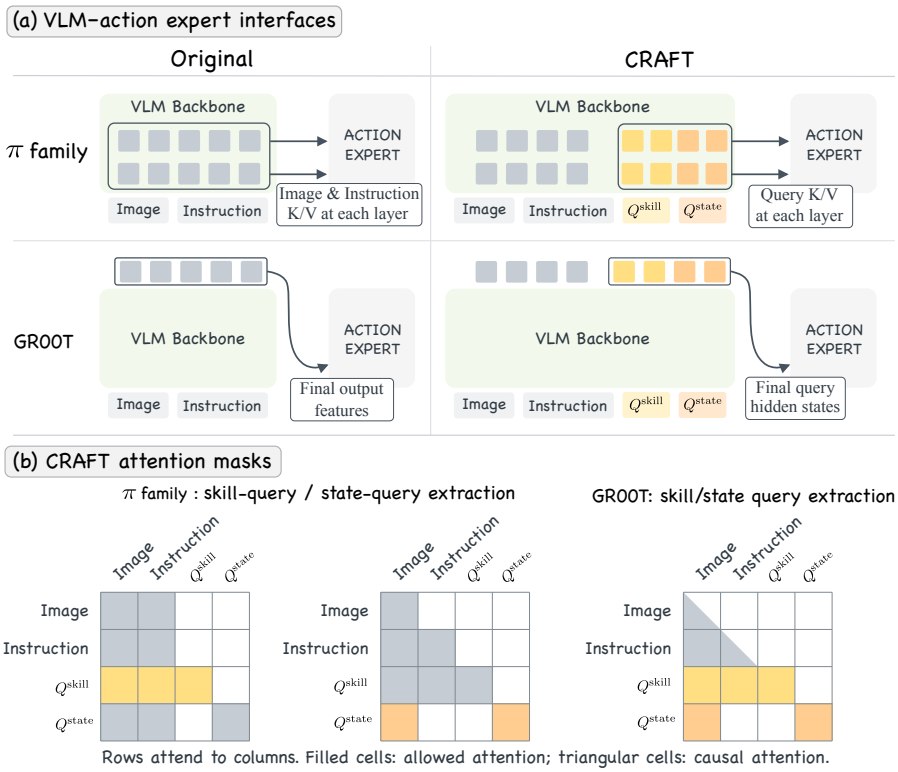

Figure B.1: CRAFT architecture and attention masks. (a) Original and CRAFT VLM–action expert interfaces. CRAFT conditions the π-family action expert on layer-wise K/V from skill and state queries, and GR00T on their final hidden states. (b) Attention masks used to construct the two representations. Rows attend to columns; filled cells permit attention, white cells block it, and triangular cells denote causal attention. The π family extracts the two query representations in separate shared-backbone passes, whereas GR00T extracts both in a single pass. Token counts are schematic; formatting and padding tokens are omitted.

Demonstration: `<time-indexed demonstration frames>`
Instruction: Pick the blue cube and place it on the blue plate.
Ordered skills: pick blue cube → place blue plate.

Locate the first frame in which the grasped cube begins moving with the gripper, counting the initial lift as part of placing. Use the following frames to check that the motion continues. Return the frame ID and note any uncertainty.

Figure B.2: VLM prompt for pick–place skill annotation. The transition is defined as the first frame in which the grasped object begins moving with the gripper, with the initial lift assigned to placing.

time-indexed demonstration frames, the instruction, and the ordered skills, and ask the VLM to return the transition frame indices. We assign the corresponding operation–entity label to each frame within a segment. Figure B.2 shows the annotation prompt for pick–place demonstrations. These labels are used only during training and are not provided to the policy at inference.

Constructing counterfactual instructions. Given the annotated current operation, we construct counterfactual instructions by changing one or more entity assignments for the current or future operations while keeping the observation and operation order fixed. Assignments for completed operations remain unchanged because their outcomes are already reflected in the observation. For each edited operation, we select a replacement entity that has been used for that operation in the demonstrations. We exclude combinations of entity assignments already present in the demonstration set.

Changing or preserving the current skill. We distinguish two cases according to whether the counterfactual instruction changes the currently required skill. A *skill-changing* counterfactual changes the entity assigned to the current operation and may also change future assignments. A *skill-preserving* counterfactual keeps the current-operation entity unchanged and changes one or more future assignments. Table B.1 illustrates both cases during placing in a pick–place–press task. The counterfactual type determines the reference sample used for supervision (Appendix B.3.2).

Table B.1: Counterfactual instructions during placing in the pick–place–press task, whose demonstrations contain only all-blue and all-red combinations. Bold entries indicate changed entity assignments. Both counterfactuals retain the completed pick assignment and specify an undemonstrated combination.

Variant	Pick (completed)	Place (current)	Press (future)
Original	Blue cube	Blue plate	Blue button
Skill-changing	Blue cube	**Red plate**	Blue button
Skill-preserving	Blue cube	Blue plate	**Red button**

B.3 Training

We detail the demonstrated-skill comparisons, counterfactual supervision, and joint optimization used by CRAFT.

B.3.1 Learning Skill Representations from Demonstrations

Selecting comparison samples. We form batches with multiple samples per skill using the operation–entity labels from Appendix B.2. For each sample, we select a positive from another execution with the same label and negatives that execute the same operation on different entities. This focuses the comparison on entity differences within the same operation.

Controlling the comparison. For all comparisons involving a given sample, we hold its state representation, noisy action chunk, flow time, and target fixed. We vary only the skill representation, using the selected positive and negative samples, and evaluate the resulting predictions with the score in Equation 3.

Normalizing the prediction error. To account for differences in target magnitude across samples, we normalize the squared prediction error by the expected squared magnitude of the target velocity. Let D_i denote the number of action elements used to compute the score. For a fixed demonstrated action chunk and independent standard Gaussian noise, this scale is

$$\sigma_i = \mathbb{E}_{\epsilon_i}\left[\|u_i\|_2^2 \mid \mathbf{a}_i\right] = D_i\left(1 + \mathrm{mean}(\mathbf{a}_i^2)\right). \tag{8}$$

The term 1 corresponds to the unit variance of the Gaussian noise, and the second term is the mean squared action magnitude. In implementation, we equivalently divide the mean squared prediction error by $1 + \mathrm{mean}(\mathbf{a}_i^2)$. The prediction error and normalization scale are computed over the same action dimensions and timesteps.

B.3.2 Counterfactual Skill-Representation Alignment

Matching counterfactuals to demonstrations. Within each batch, we match skill-changing counterfactuals with demonstrated reference samples sharing the required operation–entity label. The counterfactual supplies the skill representation, while the reference supplies its state representation, noisy action chunk, flow time, and demonstrated velocity target. For skill-preserving supervision, the reference is the original sample itself. Both cases use the predictions and targets in Equations 5–6.

Supervising the current operation. The counterfactual and reference are matched by the currently required skill, while their subsequent operations may differ. Because an action chunk can cross operation boundaries, we restrict each counterfactual loss to the timesteps of the reference action chunk that remain within the matched operation. The demonstrated flow-matching loss continues to supervise the complete valid action chunk.

Aggregating the losses. We compute the changing and preserving terms separately, averaging squared prediction error over action dimensions and the retained timesteps, and add them as in Equation 6. The preserving term applies only when a future entity assignment can be changed.

B.3.3 JOINT OPTIMIZATION

Assigning updates to each loss. The skill objective jointly learns reusable skill representations and the action expert that interprets them. For counterfactual alignment, we hold the reference state representation and action expert parameters fixed for this loss, training the counterfactual skill representation to elicit the required behavior through the fixed expert. Table B.2 summarizes the gradient paths.

Table B.2: Gradient paths for the training objectives. Each row describes the contribution of one loss to the joint update.

Loss	Skill representation	State representation	Action expert
$\mathcal{L}_{\text{fm}}$	Gradient passes	Gradient passes	Updated
$\mathcal{L}_{\text{skill}}$	Gradient passes	Gradient stopped	Updated
$\mathcal{L}_{\text{cf}}$	Gradient passes	Gradient stopped	Fixed for this loss

Stopping gradients through the state representation does not freeze the shared backbone, which remains trainable through the skill pathway and the demonstrated flow-matching loss.

Joint training. We optimize the pretrained backbone, action expert, and queries jointly in a single training stage using Equation 7, retaining each model's action normalization and flow-matching convention. We gradually introduce the auxiliary objectives by increasing β and γ from zero to their target values during warmup, while keeping the demonstrated flow-matching loss active throughout. Both auxiliary weights remain fixed after warmup. Skill labels, comparison samples, and counterfactual instructions are used only during training.

B.4 IMPLEMENTATION DETAILS

Training configuration. For the simulation experiments, we train each method for 20k optimization steps using NVIDIA H200 GPUs. For CRAFT, we construct skill-balanced batches with four samples per skill on PICK-PLACE and three samples per skill on PICK-PLACE-PRESS, giving batch sizes of 32 and 18, respectively. CRAFT uses three negative samples per anchor, a contrastive temperature of $\tau = 0.1$, and loss weights $\beta = 0.03$ and $\gamma = 1.0$, with equal weight on the skill-changing and skill-preserving terms. The auxiliary-loss weights increase from zero to their target values over the first 3k steps and remain fixed thereafter. We use 64 queries in total: 48 skill and 16 state queries for $\pi_{0.5}$ on PICK-PLACE, and 32 of each for all other model–benchmark configurations.

Optimization and baselines. We use AdamW with a 1k-step learning-rate warmup followed by cosine decay. The peak learning rate is 2.5×10^{-5} for π_0 and $\pi_{0.5}$, and 5×10^{-5} for GR00T. For the π models, we clip the gradient norm at 1.0 and use an EMA decay of 0.999. The LoRA baselines use rank 16, with $\alpha = 16$ for the π models and $\alpha = 32$ for GR00T. For *Skill Composition*, we use a rule based on the observed gripper state to determine when to advance to the next constituent skill.

Checkpoint selection. Following prior VLA evaluations that periodically evaluate training checkpoints and report the best-performing checkpoint (Kim et al., 2025; Li et al., 2026), we evaluate checkpoints every 2k steps from 10k to 20k. For each method, model, and benchmark, we select the checkpoint with the highest mean success on undemonstrated combinations across the three evaluation seeds, using these evaluation rollouts for both selection and reporting.

Deployment. Policies predict 20-step action chunks and execute the first 10 actions before replanning. We retain each model's default number of flow-integration steps: 10 for π_0 and $\pi_{0.5}$, and 4 for GR00T.

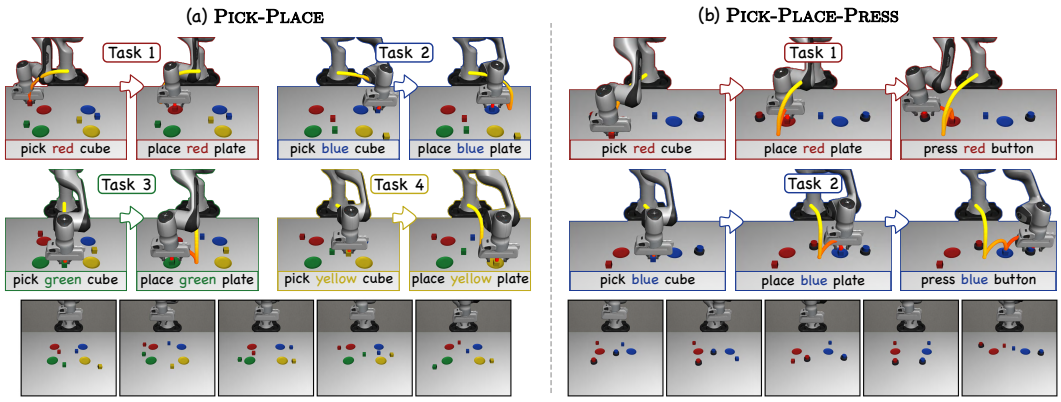

Figure C.1: Benchmark construction for (a) PICK-PLACE and (b) PICK-PLACE-PRESS. Top: demonstrated same-color combinations used for data collection. Bottom: example initial scenes from each benchmark. The remaining combinations form the undemonstrated evaluation set.

C BENCHMARK CONSTRUCTION

We construct two tabletop benchmarks in LIBERO (Liu et al., 2023) to evaluate compositional generalization to undemonstrated combinations of demonstrated skills. In both benchmarks, the operation sequence is fixed, and each instruction specifies the entity assigned to each operation. Scenes contain the same set of objects across task combinations, while object positions vary across episodes. Demonstrations cover only same-color combinations, shown at the top of Figure C.1; the remaining combinations require recombining demonstrated operation–entity skills.

For each benchmark, we generate a shared set of 50 initial scenes and collect one demonstration from each scene for every demonstrated combination using SpaceMouse teleoperation. Evaluation uses a separate set of initial scenes sampled from the same initialization distribution. We allow up to 300 environment steps per rollout for PICK-PLACE and 500 for PICK-PLACE-PRESS, which contains one additional operation.

C.1 PICK-PLACE

The scene contains one cube and one plate in each of four colors: red, blue, green, and yellow. Each instruction specifies which cube to pick and which plate to place it on. The four same-color combinations cover all eight constituent skills—four picking skills and four placing skills—while leaving the other 12 of the 16 combinations undemonstrated.

The plates occupy fixed positions in a four-quadrant layout. Each cube is initialized at a sampled position within the quadrant containing its same-color plate, subject to spacing constraints that separate the objects. This produces variation in the cube positions and corresponding manipulation trajectories across initial scenes. Task success requires the specified cube to be on the specified plate, as determined by a predicate based on contact and relative position.

C.2 PICK-PLACE-PRESS

This benchmark extends the operation sequence to picking a cube, placing it on a plate, and pressing a button. The scene contains one red and one blue instance of each object type. Demonstrations cover the all-red and all-blue combinations, providing all six constituent skills while leaving the other six of the eight combinations undemonstrated.

The two plates occupy fixed positions, while the cube and button of each color are initialized around the corresponding plate. Task success requires the specified cube to be on the specified plate and the end-effector to be within a horizontal tolerance of the specified button center and below a height threshold.



Figure D.1: CRAFT success rates (%) across PICK-PLACE task combinations, averaged over three evaluation seeds. Rows and columns specify cube and plate colors, respectively. Hatched cells denote demonstrated combinations; all panels use the same color scale.

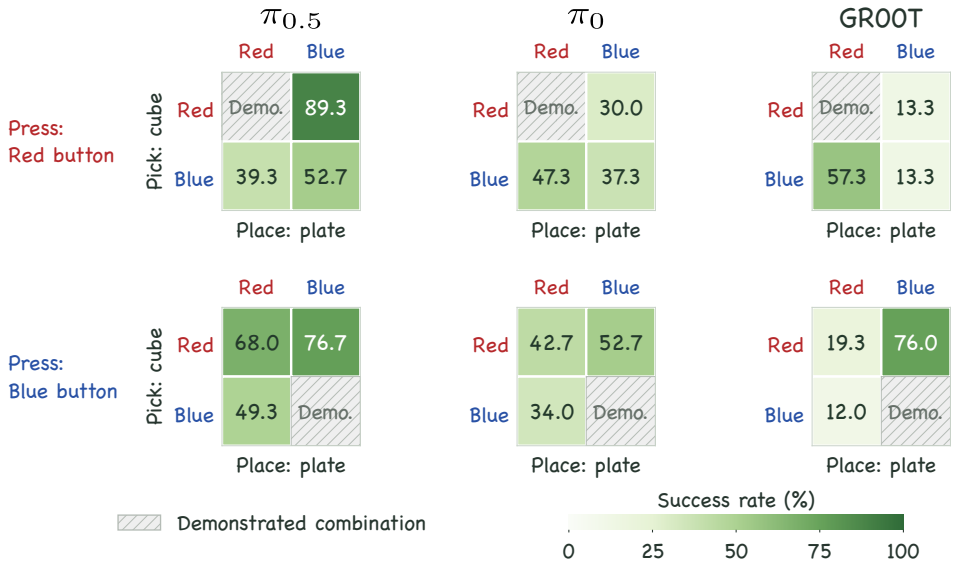

Figure D.2: CRAFT success rates (%) across PICK-PLACE-PRESS task combinations, averaged over three evaluation seeds. Rows and columns within each matrix specify cube and plate colors, respectively; the upper and lower matrices correspond to red and blue buttons. Hatched cells denote demonstrated combinations; all panels use the same color scale.

D ADDITIONAL RESULTS

D.1 PERFORMANCE ACROSS TASK COMBINATIONS

We break down CRAFT's success across individual undemonstrated combinations to examine variation within each benchmark. Figures D.1 and D.2 report success for each task combination, averaged over three evaluation seeds with 50 trials per combination and seed. Hatched cells identify demonstrated combinations, whose aggregate success rates are reported in Table 1.

On PICK-PLACE, success varies substantially across combinations even though every constituent skill is demonstrated. The blue-cube/green-plate combination is among the lowest-success combinations for all three models. Other patterns are model-dependent: the red-cube/green-plate combination reaches 90.7% success for $\pi_{0.5}$ but only 30.7% for GR00T N1.7. Thus, the combinations that remain difficult after CRAFT differ across VLA models.

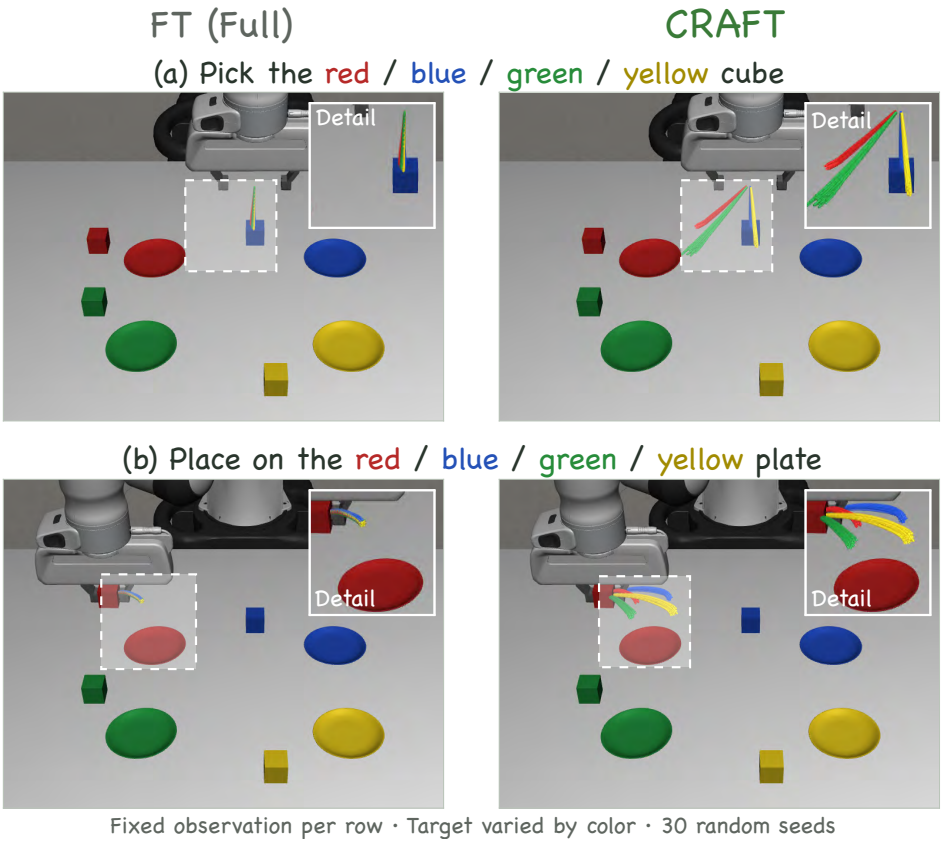

Figure D.3: Action predictions under instruction changes with a fixed observation. Top: the observation is taken while picking the blue cube, and the instructed cube is varied across four colors. Bottom: the observation is taken while placing on the red plate, and the instructed plate is varied across four colors. The blue target in the top row and red target in the bottom row correspond to the original executions. Trajectory colors indicate the instructed entity, with predicted action chunks shown for 30 random seeds. Insets enlarge the marked regions at the same scale within each row.

On PICK-PLACE-PRESS, GR00T N1.7's success is concentrated in two combinations: red–blue–blue and blue–red–red, where the colors specify the cube, plate, and button, respectively. The other four undemonstrated combinations have substantially lower success. In contrast, $\pi_{0.5}$ distributes its success more broadly across the six undemonstrated combinations.

D.2 ACTION PREDICTIONS UNDER INSTRUCTION CHANGES

Using $\pi_{0.5}$ on PICK-PLACE, we examine how action predictions respond to instruction changes while holding the visual observation fixed. For the picking example, we fix an observation from an execution of picking the blue cube and vary the instructed cube among the four colors. For the placing example, we fix an observation from an execution of placing on the red plate and similarly vary the instructed plate. Thus, the blue instruction in the first row and the red instruction in the second correspond to the original execution, while the other colors change the currently required skill without changing the observation. For each instruction, we visualize predicted action chunks from 30 random seeds (Figure D.3).

Under CRAFT, the predicted trajectories separate according to the instructed cube or plate. In contrast, predictions from FT (*Full*) largely overlap across instruction changes in both examples. This qualitative difference suggests that CRAFT's action predictions are more responsive to the currently required skill under a fixed observation.

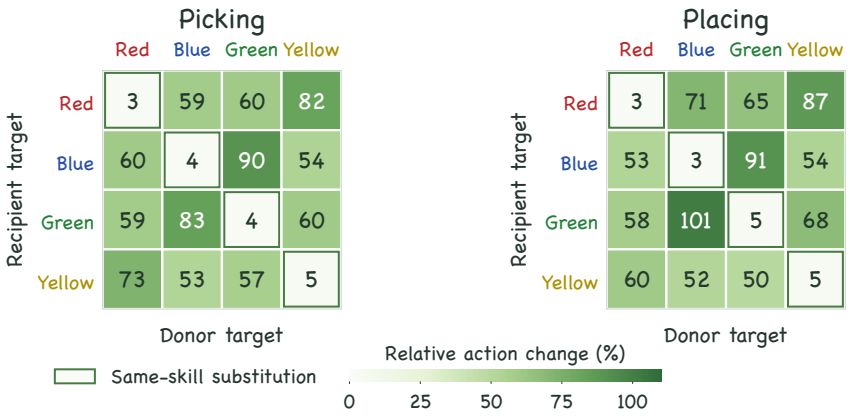

Figure D.4: Skill-representation reuse across executions on PICK-PLACE with $\pi_{0.5}$. For each recipient observation, skill-query K/V are replaced with those from another execution while the recipient's state representation and sampling noise remain fixed. Rows and columns indicate recipient and donor entities, respectively. Each cell reports the mean relative change (%) in the predicted action chunk. Outlined diagonal cells use donors executing the same skill as the recipient; off-diagonal cells use donors executing the same operation with a different entity. Both panels use the same color scale.

D.3 REUSING SKILL REPRESENTATIONS ACROSS EXECUTIONS

We test whether CRAFT's skill representations can be reused across executions using $\pi_{0.5}$ on PICK-PLACE. For each recipient observation, we replace its skill-query K/V with those from another execution while keeping the recipient's state representation and sampling noise fixed. Donors execute the same operation as the recipient, with either the same or a different entity. We evaluate 120 recipient frames, with 15 per skill, and sample 25 donors for each candidate donor skill. For each substitution, we measure the relative change in the predicted action chunk, normalized by the norm of the original prediction.

Figure D.4 shows a clear contrast between same-skill and different-skill substitutions for both picking and placing. Replacing the skill representation with one from another execution of the same skill changes the prediction by only $4.0 \pm 3.1\%$ on average across recipients. In contrast, substitutions from different skills produce substantially larger changes across donor–recipient pairs. These results support the intended reuse of skill representations across executions: representations from different executions of the same skill yield similar action predictions under a fixed recipient state, while representations of different skills produce distinct predictions.

D.4 QUERY COUNT AND ALLOCATION

We examine the sensitivity of compositional generalization to the number and allocation of skill and state queries on PICK-PLACE (Figure D.5). For the query-count sweep, skill and state queries are allocated equally. For the allocation sweep, we fix the total at 64 queries and vary the skill/state split for $\pi_{0.5}$.

Total query count. For both $\pi_{0.5}$ and GR00T N1.7, success on undemonstrated combinations increases from 32 to 64 total queries and decreases at 128. Thus, increasing the number of queries does not monotonically improve compositional generalization. Success on demonstrated combinations remains high across the sweep, indicating that the degradation at 128 queries is concentrated on undemonstrated combinations.

Skill/state allocation. With 64 total queries, the state-heavy 16/48 skill/state allocation yields substantially lower success on undemonstrated combinations than the 32/32 and 48/16 allocations, while demonstrated success remains high across all three settings. Within this sweep, performance is therefore lower when a larger share of the query budget is allocated to the state representation than to the skill representation. The balanced 32/32 and skill-heavy 48/16 allocations perform similarly well.

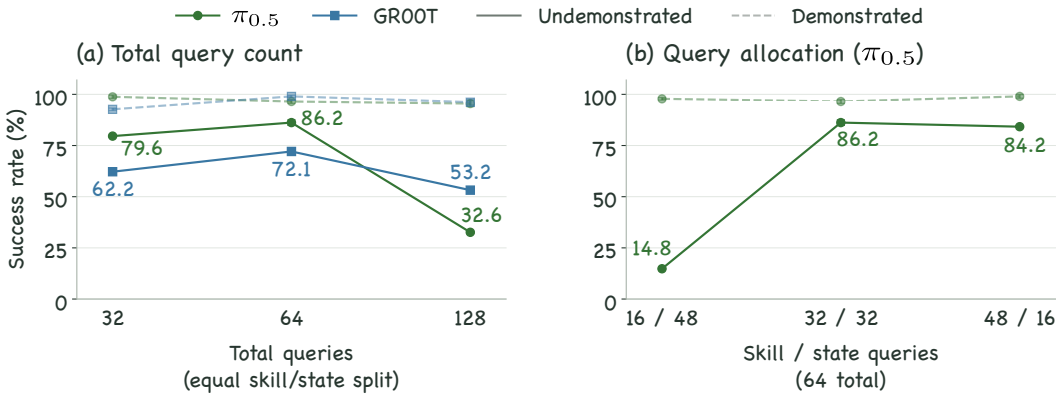

Figure D.5: Query-count and allocation ablations on PICK-PLACE. (a) Total query count for $\pi_{0.5}$ and GR00T N1.7, with an equal number of skill and state queries. (b) Skill/state query allocation for $\pi_{0.5}$ with 64 total queries. Solid and dashed lines denote success on undemonstrated and demonstrated combinations, respectively.

The main $\pi_{0.5}$ configuration uses 48/16 skill/state queries, while the main GR00T N1.7 configuration uses 32/32.

D.5 GUIDANCE-STRENGTH SELECTION FOR CAG-VA

CAG-VA implementation. We implement the vision-action (VA) variant of CAG-VA (Fang et al., 2025). For each backbone and adaptation setting, we fine-tune a separate VA policy on the same demonstrations while masking the language input. At inference, CAG-VA combines the language-conditioned VLA policy $\pi_{\text{cond}}(a \mid o, \ell)$ with the corresponding vision-only policy $\pi_{\text{VA}}(a \mid o)$ using

$$\pi_{\text{CAG}}(a \mid o, \ell) = \pi_{\text{VA}}(a \mid o) + \omega \left(\pi_{\text{cond}}(a \mid o, \ell) - \pi_{\text{VA}}(a \mid o)\right), \tag{9}$$

where ω controls the guidance strength. For the flow-based VLAs, we apply this guidance rule to the velocity predictions at each flow-integration step. Thus, $\omega = 1$ recovers the original language-conditioned VLA, while larger values amplify the difference between the language-conditioned and vision-only predictions. Both branches are used at every inference step.

Guidance-strength selection. We sweep $\omega \in \{1, 1.5, 2, 3, 4, 5\}$ separately for each model–adaptation–benchmark configuration. For each model and benchmark, we first sweep adaptation settings and guidance strengths using seed 7. We select the configuration with the highest undemonstrated-combination success and evaluate that configuration additionally with seeds 0 and 42. Figure D.6 and Table D.1 show the sweeps for the π models. Points evaluated with all three seeds are reported as three-seed means; all remaining points show the original seed-7 sweep result. For GR00T N1.7, we follow the same selection procedure; the selected configurations are Frozen with $\omega = 1.5$ on PICK-PLACE and LoRA with $\omega = 3$ on PICK-PLACE-PRESS. The main results report three-seed performance for the selected CAG-VA configuration for each model and benchmark.

Effect of guidance strength. The effect of guidance varies substantially across models and adaptation settings. For $\pi_{0.5}$ with LoRA, stronger guidance improves success on undemonstrated combinations on both benchmarks, although high guidance strengths also reduce success on demonstrated combinations. For π_0, the gains are considerably smaller, and stronger guidance can reduce demonstrated success without a corresponding improvement on undemonstrated combinations. These results motivate selecting ω separately for each configuration rather than using a single guidance strength across models and adaptation settings.

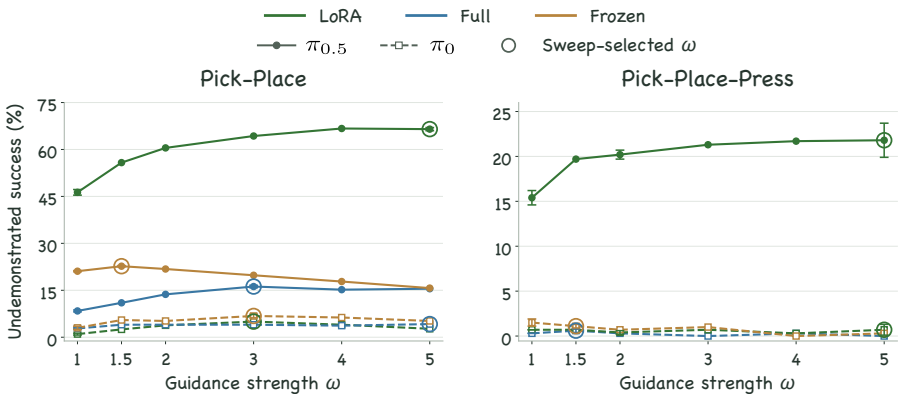

Figure D.6: CAG-VA guidance-strength sweeps on PICK-PLACE and PICK-PLACE-PRESS. Each configuration uses a separately fine-tuned vision-only VA policy together with the corresponding language-conditioned VLA policy. Curves show success on undemonstrated combinations as a function of guidance strength ω. The initial sweep uses seed 7; selected settings additionally evaluated with seeds 0 and 42 are shown as three-seed means with standard-deviation error bars, while the remaining points show seed-7 results. Outlined markers denote the best guided setting ($\omega > 1$) in each seed-7 sweep; $\omega = 1$ recovers the unguided language-conditioned policy.

Table D.1: CAG-VA guidance-strength sweep. Each cell reports undemonstrated success with demonstrated success in parentheses (%). The initial sweep uses seed 7; selected settings additionally evaluated with seeds 0 and 42 are reported using their three-seed means. Bold entries indicate the final CAG-VA configurations reported in Table 1 and include mean ± standard deviation over the three evaluation seeds. The setting $\omega = 1$ corresponds to the unguided language-conditioned policy.

Configuration	$\omega = 1$	1.5	2	3	4	5
PICK-PLACE						
$\pi_{0.5}$ LoRA	46.3 (100)	55.8 (100)	60.5 (100)	64.3 (100)	65.7 (99.5)	**66.5 ± 0.5** (**96.3 ± 1.5**)
$\pi_{0.5}$ Full	8.4 (100)	11.0 (100)	13.7 (99.5)	16.2 (98.5)	15.2 (95)	15.5 (94.5)
$\pi_{0.5}$ Frozen	21.1 (99)	22.7 (98.7)	21.8 (97.5)	19.8 (95.5)	17.8 (86)	15.7 (83)
π_0 LoRA	1.0 (97.5)	2.5 (96.5)	3.8 (93)	5.0 (81)	4.0 (71)	2.7 (57)
π_0 Full	2.8 (100)	4.0 (100)	4.0 (100)	4.0 (97.5)	3.7 (93)	4.2 (87)
π_0 Frozen	3.0 (48.2)	5.5 (57)	5.2 (53.5)	**6.8 ± 0.6** (**50.5 ± 1.3**)	6.3 (40)	5.2 (36.5)
PICK-PLACE-PRESS						
$\pi_{0.5}$ LoRA	15.4 (100)	19.7 (100)	20.2 (99)	21.3 (96)	21.7 (94)	**21.8 ± 1.9** (**92.7 ± 3.5**)
π_0 LoRA	0.7 (99.3)	0.7 (96)	0.4 (97.7)	0.7 (87)	0.3 (88)	0.7 (81)
π_0 Full	0.3 (99.7)	0.6 (98.3)	0.3 (97)	0.0 (95)	0.3 (90)	0.0 (91)
π_0 Frozen	1.5 (82)	**1.1 ± 0.2** (**69 ± 2.6**)	0.7 (64)	1.0 (40)	0.0 (24)	0.3 (6)

D.6 Additional Real-Robot Rollouts

Figure D.7 shows additional real-robot rollouts of CRAFT on undemonstrated Pick-Place combinations. Across these examples, CRAFT picks the cube specified by the instruction and places it on the specified plate, despite the corresponding cube–plate combinations being absent from the demonstrations.

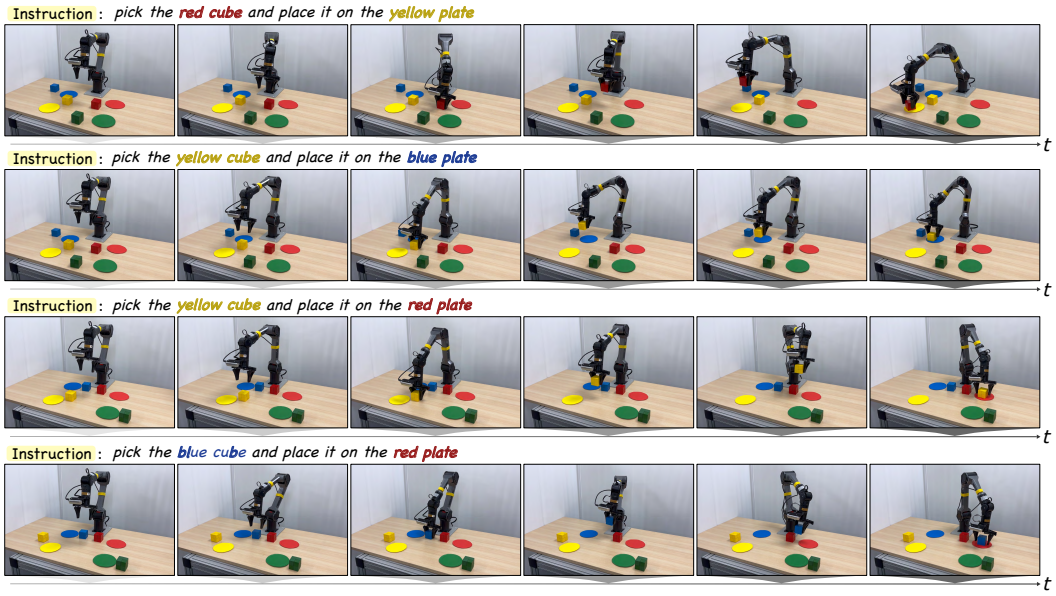

Figure D.7: Additional real-robot rollouts of CRAFT on undemonstrated Pick-Place combinations. Each row shows a different instruction and its execution over time. CRAFT picks the instructed cube and places it on the instructed plate, although the corresponding cube–plate combination is absent from the demonstrations.