TRAIN TOGETHER OR MERGE LATER? UNIFYING VLA EXPERTS VIA A SHARED ACTION INTERFACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Co-training offers a straightforward way to build a multi-task vision-language-action (VLA) policy, but can fall short of the performance achieved by training each task independently. The challenge is to retain these task-specific gains in a multi-task policy without joint post-training. Combining independently trained experts through model merging is a natural approach, yet strong individual experts do not necessarily yield a strong merged policy. We identify one source of this *incompatibility*: task-specific changes to the *action interface*, comprising action normalization and the action encoder and decoder. We propose POLICYWEAVE, combining merge-compatible post-training with context-guided sparse merging. During post-training, all experts retain the common base policy's action interface, while task adaptation is restricted to LoRA updates in the hidden layers of the action model. This makes the experts more compatible with existing model merging methods. However, merging all experts can still introduce interference from unrelated tasks at deployment. POLICYWEAVE scores each expert's LoRA updates using the initial visual-language context, determines the expert set through leave-one-layer-out ranking stability, and forms a sparse weighted merge of the selected updates that remains fixed for current task. We evaluate POLICYWEAVE with GR00T N1.5 on 18 RoboCasa365 tasks, using only 10% of the target-task demonstrations for supervised fine-tuning (SFT). Preserving the shared action interface raises the average success rate across four static merging methods from 17.0% to 52.8%. POLICYWEAVE achieves 64.7% success with these SFT experts and 74.1% after task-specific reinforcement learning (RL), compared with 60.7% for joint RL. Further evaluations on LIBERO-10 and an AgileX Piper arm support the deployment of independently learned skills in long-horizon and real-world manipulation. These results show that independently learned robot policies can be merged for multi-task deployment without joint post-training across tasks. *Source code and real robot demos are available in the supplementary material.*

1 INTRODUCTION

Vision-language-action (VLA) models learn broadly applicable robot control capabilities through pretraining on diverse robot data (Brohan et al., 2023; Ghosh et al., 2024; Kim et al., 2025; Black et al., 2025b; Bjorck et al., 2025). During downstream post-training, supervised fine-tuning (SFT) adapts VLA policies using task demonstrations, and reinforcement learning (RL) can further improve performance through interaction (Li et al., 2026; Chen et al., 2025). At either stage, co-training offers a straightforward way to obtain a single policy for multiple tasks. However, balancing competing task objectives can limit performance on individual tasks (Yu et al., 2020). Alternatively, task-specific adaptation optimizes each expert independently from the same pretrained model, but leaves the resulting capabilities in separate policies (Figure 1, left). We therefore ask: *can we retain the gains from task-specific training in a multi-task policy without joint post-training?*

Model merging (Ilharco et al., 2023; Yadav et al., 2023) combines parameter updates from independently trained experts without joint retraining and has shown promise in vision and language models. For VLAs, however, strong individual performance does not guarantee that an expert's capabilities will survive merging (Fu et al., 2026). We identify one source of this incompatibility in the *action interface*: the normalization statistics and action encoder and decoder that map between physical actions and model representations. During task-specific post-training, changes to this interface can be compensated for by changes to the expert's hidden-layer parameters. Merging both can

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

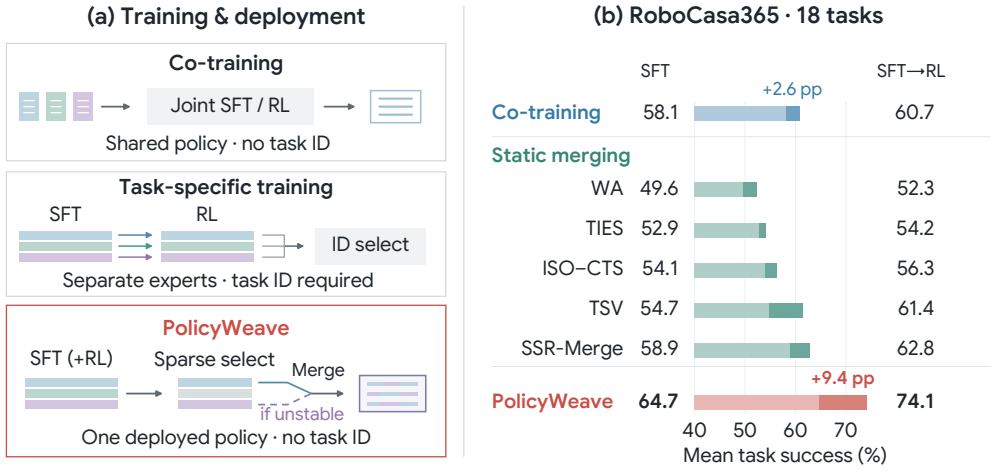


Figure 1: **(a)** Co-training, task-specific adaptation, and PolicyWeave. **(b)** Mean success (%) on 18 RoboCasa365 tasks with SFT and SFT then RL, comparing co-training with model merging.

break this compensation and alter the resulting actions. A simple linear example shows that parameter averaging can change the output even when the original experts produce identical actions (Appendix B.1). This motivates maintaining compatibility while experts learn their task-specific capabilities. To this end, we introduce PolicyWeave, which keeps the base policy's action interface fixed while learning task-specific LoRA updates (Hu et al., 2022) in the hidden layers of the action model (Figure 3(a)). Existing merging methods can then combine these updates without also merging task-specific action interfaces.

A shared action interface improves mergeability, but does not eliminate interference between task updates. Widely used static merging methods combine these updates into one set of parameters for all tasks (Ilharco et al., 2023; Yadav et al., 2023; Gargiulo et al., 2025), so updates unrelated to the current task can still affect its execution. Selecting relevant updates based on the input offers a way to reduce this interference (Tang et al., 2024; Lu et al., 2024). In VLAs, observations and instructions are encoded into visual-language context that conditions the action model, providing a basis for identifying relevant experts. However, tasks sharing objects or goals can produce similar contexts: selecting one expert risks choosing the wrong policy, while merging several can introduce conflicting updates. Our insight is that the experts' own responses to this context can guide expert selection. PolicyWeave uses these responses to rank experts and ranking stability to decide whether to select one expert or merge a sparse set, without a trained router or calibration trajectories (Figure 3(b)).

We evaluate PolicyWeave with GR00T N1.5 (Bjorck et al., 2025) on 18 atomic RoboCasa365 (Nasiriany et al., 2026) tasks, using only 10% of the target-task demonstrations for SFT. Preserving the shared action interface retains individual expert performance (65.6% versus 64.1% with task-specific interfaces), while raising the average success rate across four static merging methods from 17.0% to 52.8% (Figure 2(a)). Using the same shared-interface SFT experts, context-guided sparse merging further improves success to 64.7%. Task-specific RL then raises merged success to 74.1%, while joint RL improves the co-training baseline from 58.1% to 60.7% (Figure 1(b)). Experiments on 10 long-horizon LIBERO-10 (Liu et al., 2023) tasks show retention of multi-stage control capabilities. On four Piper-arm manipulation tasks, PolicyWeave also outperforms the co-trained baselines, supporting multi-task deployment of independently learned skills on a physical robot. These experiments support separating task-specific optimization from multi-task deployment: experts can be trained independently and merged for deployment by preserving their compatibility during post-training and controlling task interference when merging.

2 Method: PolicyWeave

2.1 Merge-Compatible VLA Post-Training

We consider K target tasks indexed by $\mathcal{T} = \{1, \ldots, K\}$, with one LoRA expert independently post-trained for each task from a common base VLA θ_0. The final expert parameters are $\theta_t \in$

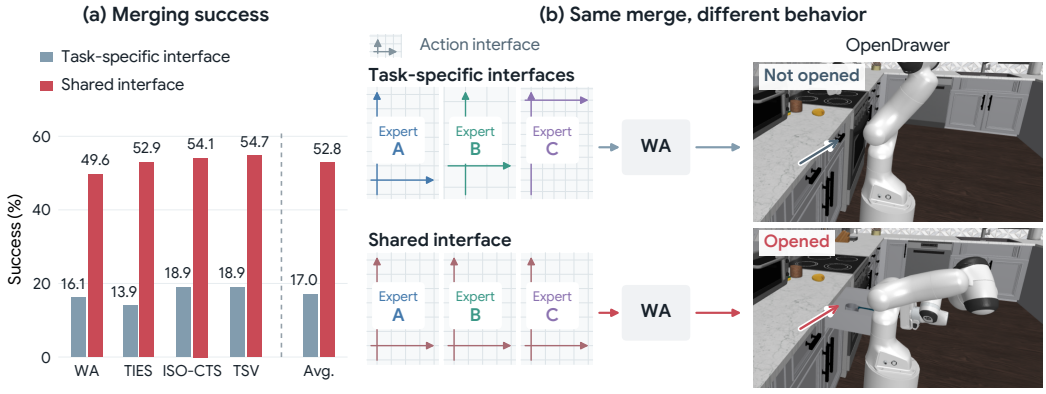


Figure 2: **A shared action interface supports effective merging of experts trained on 18 RoboCasa365 tasks.** (a) Mean SFT success across these tasks for four static merging methods; Avg. denotes their unweighted mean. (b) Different action interfaces, the same weight-averaging (WA) rule. Paired OpenDrawer rollouts show successful opening only with the shared interface.

$\{\theta_t^{\mathrm{SFT}}, \theta_t^{\mathrm{SFT+RL}}\}$, obtained through SFT (Kim et al., 2025) or SFT then RL (Chen et al., 2025). In our setting, RL continues training the same task-specific LoRA parameters. In both cases, $\Delta_t = \theta_t - \theta_0$ collects the induced LoRA weight updates relative to the common base. A merging method $\mathcal{M}$ combines these updates into a deployed policy with parameters

$$\theta_{\mathrm{merge}} = \theta_0 + \mathcal{M}(\{\Delta_t\}_{t\in\mathcal{T}}). \tag{1}$$

Existing LoRA post-training configurations differ in which parts of a VLA they adapt. For example, LoRA can adapt both the VLM and action transformer in π_0 (Black et al., 2025b) and $\pi_{0.5}$ (Black et al., 2025a), whereas the N1.5 configuration of GR00T (Bjorck et al., 2025) adapts the action head while keeping the VLM frozen.

For merging, task adaptation should also preserve a common mapping between model representations and physical actions. We call the normalization statistics and the action head's input-output projectors, namely its action encoder and decoder, the *action interface*. Task-specific normalization changes how physical actions are represented within the model and how its predictions are converted back into physical commands. Updating the encoder and decoder further alters the mappings between actions and hidden representations. An expert can compensate for interface changes through its internal weights, but merging independently adapted weights can break this compensation. A simple example in Appendix B.1 shows that parameter averaging can change the output even when the original experts produce identical actions.

Evidence from static merging. Figure 2(a) compares four static merging methods on 18 RoboCasa365 tasks: their average success rate increases from 17.0% with task-specific interfaces to 52.8% with the shared interface. Figure 2(b) illustrates these interface differences with paired OpenDrawer rollouts: under the same WA rule, the shared-interface policy successfully opens the drawer, while the task-specific-interface policy produces actions that deviate from the intended motion.

Post-training with a shared interface. Motivated by these results, we propose *merge-compatible VLA post-training* (Figure 3,left). Within a shared embodiment and action space, the broadly trained base provides action statistics and input-output mappings that can be reused across downstream tasks. PolicyWeave retains its normalization statistics and freezes its action encoder, action decoder, and VLM. Task adaptation is restricted to LoRA parameters in the hidden action-model layers. This post-training protocol is independent of the merging method: the same expert bank supports existing static methods and the context-guided sparse merging described next.

2.2 Context-Guided Sparse Merging

Even with a shared action interface, static merging can introduce interference from task-irrelevant updates (Yadav et al., 2023; Tang et al., 2024). As shown in Figure 3(b), PolicyWeave uses the initial observation and instruction to select and merge relevant expert updates. We consider tasks covered by the expert bank, without a supplied task identity.

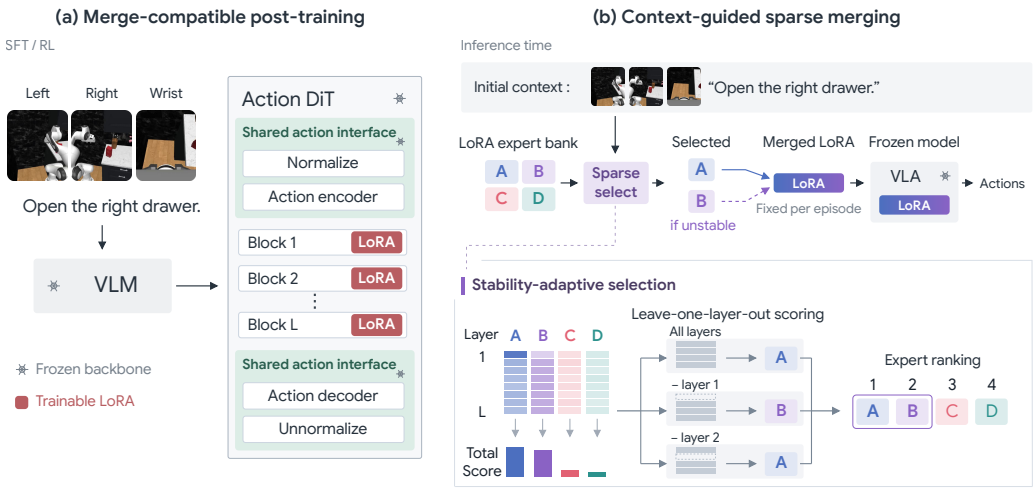


Figure 3: Left: POLICYWEAVE trains task-specific LoRA experts while freezing the VLM and shared action interface. Right: initial context guides sparse expert selection and merging. Leave-one-layer-out stability determines the expert set, and merged parameters remain fixed per episode.

Merging selected expert updates. Let c be the visual-language context encoded from the initial observation and instruction, $\mathcal{I}^\star(c) \subseteq \mathcal{T}$ the selected expert set, and $\gamma_t(c)$ its nonnegative merging weights that sum to one. We merge the selected updates at every adapted hidden action-model layer $u \in \mathcal{H}$:

$$\Delta \mathbf{W}_u(c) = \sum_{t \in \mathcal{I}^\star(c)} \gamma_t(c) \Delta \mathbf{W}_{t,u}. \tag{2}$$

Here $\mathcal{H}$ denotes the adapted layers, $\Delta \mathbf{W}_{t,u}$ is expert t's update, and $\Delta \mathbf{W}_u(c)$ is the merged update at layer u. We add these updates to the base weights, using the same merging weights across layers. With one selected expert, its weight is one and the deployed parameters equal that expert's parameters. The initial context determines the merged parameters, which remain fixed throughout the episode while the policy responds to new observations. We omit the context argument c from subsequent notation.

Expert relevance from context responses. The frozen VLM encodes the instruction and observed scene into a common context c for all experts. This context conditions the action DiT through cross-attention (Bjorck et al., 2025). Task-specific key updates change how action queries match this context, so we use their responses to it as relevance scores. One base-model forward pass with task adapters disabled provides the shared key-projection inputs, avoiding a full forward pass through each expert. Let $\mathbf{H}_\ell \in \mathbb{R}^{d_{\text{in}} \times N}$ contain the N valid context-token representations as columns at the input to layer ℓ's key projection, and let $\Delta \mathbf{W}^{\text{K}}_{t,\ell} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be expert t's LoRA update to that projection. Here d_{in} and d_{out} are the projection's input and output dimensions. We normalize its response by both update and context magnitude to compare experts independently of scale. For the set of scoring layers $\mathcal{C}$, the layer score $e_{t,\ell}$ and aggregate expert score s_t are scalars:

$$e_{t,\ell} = \frac{\|\Delta \mathbf{W}^{\text{K}}_{t,\ell} \mathbf{H}_\ell\|_F^2}{\|\Delta \mathbf{W}^{\text{K}}_{t,\ell}\|_F^2 \|\mathbf{H}_\ell\|_F^2}, \qquad s_t = \sum_{\ell \in \mathcal{C}} e_{t,\ell}. \tag{3}$$

The score measures alignment between the context and the input directions to which the key update is sensitive. All scoring layers contribute equally.

Expert selection by ranking stability. Selecting one expert risks choosing the wrong policy, while selecting several can introduce conflicting updates. We determine the set size by checking whether the highest-scoring expert remains unchanged when individual scoring layers are excluded. Let $\rho = (\rho_1, \ldots, \rho_K)$ be the expert indices sorted by descending aggregate score, so $s_{\rho_1} \geq \cdots \geq s_{\rho_K}$. We remove each layer's score contribution in turn and identify the expert with the highest remaining score, without additional forward passes. For layer ℓ, this gives

$$s_t^{(-\ell)} = s_t - e_{t,\ell}, \qquad t_\ell^\star = \arg\max_{t \in \mathcal{T}} s_t^{(-\ell)}. \tag{4}$$

Here $t^\star_\ell$ identifies the highest-scoring expert after removing layer ℓ. If ρ_1 remains first after every removal, we select it alone. Otherwise, we collect all experts that become first under a layer removal. We select the smallest top-k set in the original ranking that includes all these experts:

$$k^\star = \max_{\ell \in \mathcal{C}} \mathrm{rank}_\rho(t^\star_\ell), \qquad \mathcal{I}^\star = \{\rho_1, \ldots, \rho_{k^\star}\}, \tag{5}$$

where $\mathrm{rank}_\rho(t^\star_\ell)$ is its position in the original ranking, starting at 1. Fixed Top-k variants use a prescribed set size with the same ranking and weighting rule. In both cases, the selected updates are weighted using the original aggregate scores: $\gamma_t = s_t / \sum_{j \in \mathcal{I}^\star} s_j$ for $t \in \mathcal{I}^\star$. Appendix B.3 provides a theoretical justification for the ranking-stability test. Inference details and scorer ablations are given in Appendices B.4 and D.5.

3 Experiments

We organize our evaluation around three questions: (1) Can merging preserve independently acquired task capabilities, including gains from task-specific RL and long-horizon control? (2) How does a shared action interface improve merging? (3) Can context responses guide which experts to select and how many to merge?

3.1 Experimental Setup

Baselines and evaluation. We use two simulation benchmarks to examine complementary aspects of VLA merging. On 18 atomic RoboCasa365 tasks (Nasiriany et al., 2026), SFT and SFT then RL expert banks test both skill retention and whether improvements from task-specific RL survive multi-task deployment. LIBERO-10 (Liu et al., 2023) instead tests retention of long-horizon control using an SFT-only bank, with each expert trained on a complete multi-stage task. We compare PolicyWeave with five representative post-hoc merging baselines: weight averaging (WA) (Wortsman et al., 2022), TIES (Yadav et al., 2023), ISO-CTS (Marczak et al., 2025), TSV-Merge (Gargiulo et al., 2025), and SSR-Merge (Wei et al., 2026). Together, they cover direct averaging, coordinate-level conflict resolution, and subspace-based merging, including methods tailored to LoRA. We evaluate shared-interface post-training across four static merging methods (Figure 2). Our main merging comparisons then hold the shared-interface expert bank fixed within each benchmark. For each task, we report the mean success rate over 50 evaluation episodes. For the challenging long-horizon LIBERO-10 tasks, we additionally report partial success, measured by the maximum fraction of goal predicates simultaneously satisfied during each episode and averaged across episodes.

Implementation details. Our expert banks use GR00T N1.5 (Bjorck et al., 2025), initialized from a benchmark-specific base checkpoint. We train rank-8 LoRA experts in the action model while preserving the PolicyWeave action interface. Each RoboCasa365 SFT expert uses 50 demonstrations, or 10% of the available data, and RL follows the Flow-SDE formulation of π_{RL} (Chen et al., 2025). The RoboCasa365 *Co-training* baseline starts from the official checkpoint co-trained on all 18 tasks using 10% of the training data. We continue this policy with 600 RL steps, comparable to the 610 steps summed across the task-specific continuations. LIBERO-10 uses an SFT expert bank with 38 demonstrations per task on average. PolicyWeave uses eight action cross-attention key projections for scoring and fixes the merged update for the episode. Complete protocols are given in Appendix C.

3.2 Preserving Task Capabilities through Merging

Atomic Skills and RL Gains (RoboCasa365). Table 1 compares PolicyWeave with static merging methods on the same SFT and SFT then RL expert banks. SSR-Merge is the strongest static method in both banks, reaching 58.9% and 62.8% success, respectively. PolicyWeave reaches 64.7% and 74.1%, closer to the corresponding *Individual* results of 65.6% and 76.2%.

Table 1: **RoboCasa365 success (%), averaged over 50 episodes per task.** Each pair reports **SFT → SFT then RL**. All merging methods use the same shared-interface expert bank. Bold marks the best merging result for each task and stage.

	Baselines		Merging methods					
Task	*Indiv.*	*Co-training*	WA	TIES	ISO-CTS	TSV	SSR-Merge	**Policy Weave**
Close blender lid	22→32	8→24	8→8	8→18	10→0	4→10	14→10	**20→30**
Close fridge	68→74	64→70	74→74	66→78	62→68	**78**→68	70→**82**	**78**→76
Close toaster oven	80→80	68→76	46→56	54→40	68→**76**	68→68	46→44	**78→76**
Set up coffee mug	50→76	50→40	26→26	28→32	40→54	32→50	30→52	**56→70**
Navigate kitchen	22→68	28→28	2→6	6→10	8→14	14→22	6→6	**20→62**
Open cabinet	86→94	72→86	60→58	60→66	66→80	68→64	74→78	**80→88**
Open drawer	70→84	80→80	60→66	70→70	66→76	78→72	68→74	**82→82**
Open mixer head	94→96	82→90	76→76	74→74	74→68	68→80	88→84	**92→92**
Counter → cabinet	64→74	64→62	54→68	44→58	40→60	62→64	**74**→64	56→**70**
Counter → stove	70→70	60→48	56→70	68→70	64→56	**72→78**	**72**→76	60→70
Drawer → counter	36→58	44→40	30→18	**34**→36	16→24	28→36	26→40	**34→60**
Sink → counter	76→76	74→74	68→78	76→78	68→70	84→82	**90→88**	72→72
Toaster → counter	84→84	66→64	**82→90**	70→66	78→82	68→78	74→88	78→82
Slide dishwasher rack	86→86	58→66	66→54	66→52	76→70	54→62	72→74	**78→84**
Turn off stove	50→60	22→30	14→22	44→22	**46**→38	26→50	42→40	42→**56**
Turn on kettle	74→90	70→70	**96→98**	88→94	92→90	88→96	90→84	90→86
Turn on microwave	70→82	56→64	40→42	44→60	38→32	44→60	50→60	**64→90**
Turn on sink faucet	78→88	80→80	34→32	52→52	62→56	48→66	74→86	**84→88**
Mean SFT	65.6	58.1	49.6	52.9	54.1	54.7	58.9	**64.7**
Mean SFT then RL	76.2	60.7	52.3	54.2	56.3	61.4	62.8	**74.1**
Gain (pp)	+10.6	+2.6	+2.7	+1.3	+2.2	+6.7	+3.9	+9.4

The change from SFT to SFT then RL tests how much of the experts' improvement survives merging. Task-specific RL raises *Individual* success by 10.6%. All static methods benefit from the improved experts, but retain only part of this gain. PolicyWeave retains a larger gain of 9.4%, whereas joint RL improves *Co-training* by 2.6% with a comparable RL-step budget. Independent RL can therefore improve multi-task deployment, provided the merging method preserves the resulting capabilities. For reference, Figure 4 also includes selected RoboCasa365 (Nasiriany et al., 2026) Atomic-Seen leaderboard results reported under their distinct protocols.

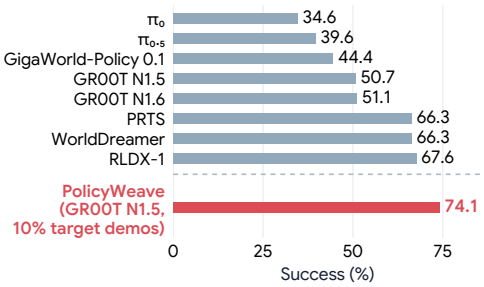


Figure 4: RoboCasa365 Atomic-Seen leaderboard references.

Long-Horizon Skill Retention (LIBERO-10). LIBERO-10 tests whether merging preserves complete long-horizon behaviors. Without a shared action interface, all four static methods yield 0% final success (Appendix D.1); with it, they reach 14.0–28.2%. Figure 5 also reports partial success, which reaches 53.9–61.2% for these methods. The shared interface improves mergeability, but much of the intermediate progress still fails to translate into full task completion, consistent with accumulated control errors over longer sequences. PolicyWeave achieves 75.4% final success and 86.4% partial success, largely preserving both intermediate progress and long-horizon task completion. Its final success also exceeds the selected cotrained VLA results, including GR00T N1.5 at 72.4%.

Table 2: LIBERO-10 merging retention.

Method	Indiv. → Merged	Ret. ↑
MergeVLA	89.6 → 70.8	79.0
PolicyWeave	80.8 → 75.4	**93.3**

To further assess skill retention, we compare PolicyWeave with our MergeVLA reproduction on LIBERO-10. Since their VLA backbones and post-training pipelines differ, we report merged success as a fraction of each method's *Individual* expert success. PolicyWeave retains 93.3%, compared with 79.0% for MergeVLA (Table 2).

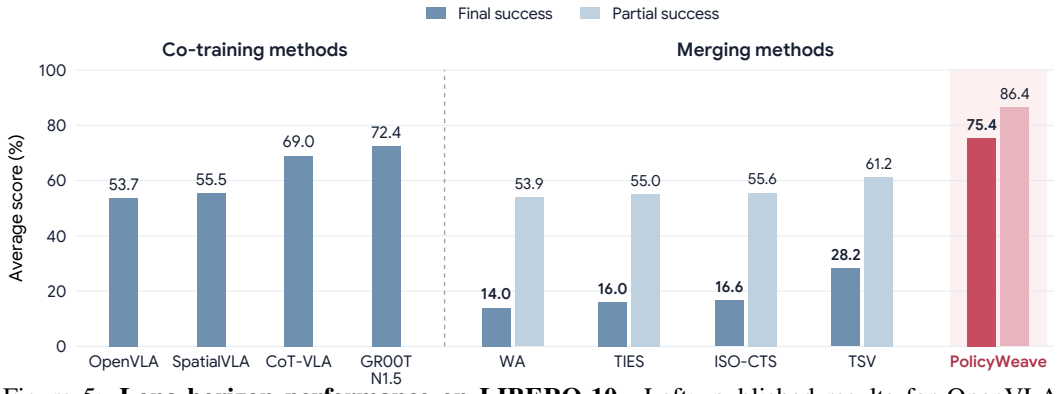


Figure 5: **Long-horizon performance on LIBERO-10.** Left: published results for OpenVLA (Kim et al., 2025), SpatialVLA (Qu et al., 2025), CoT-VLA (Zhao et al., 2025a), and co-trained GR00T N1.5 baseline. Right: success rates across merging methods using the same expert bank.

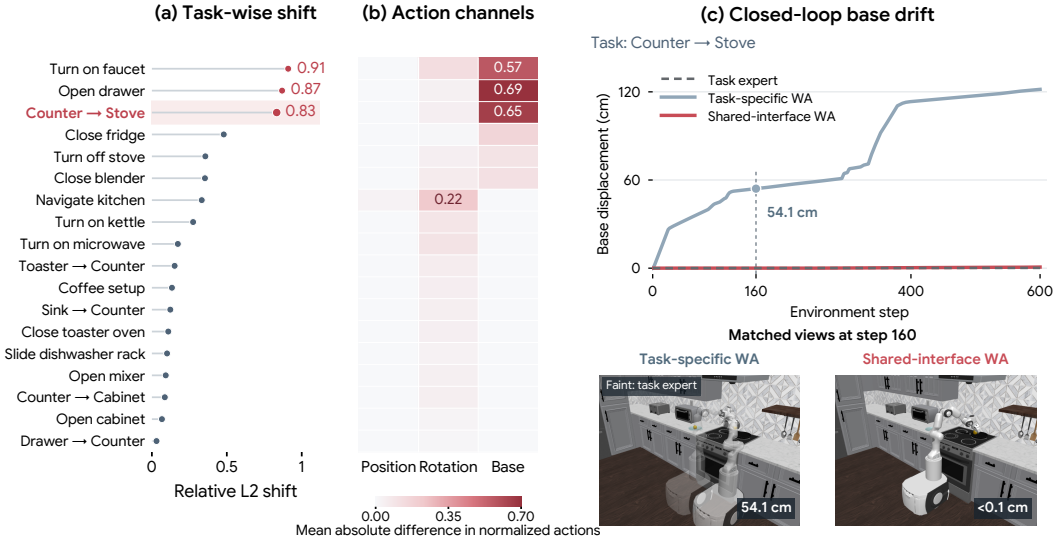


Figure 6: **Action normalization and closed-loop base drift on RoboCasa365.** (a) Differences between task-specific and shared normalization of the same raw actions. (b) The largest differences occur in base-motion channels; rows follow (a). (c) Actual base displacement from a matched initial scene on PickPlaceCounterToStove. The views show step 160.

3.3 The Role of a Shared Action Interface

The merging results on RoboCasa365 (Figure 2) and LIBERO-10 motivate examining how action-interface differences affect physical execution. We first apply task-specific and shared normalization to the same 1,800 raw actions from 18 RoboCasa365 tasks. Figure 6(a,b) shows that differences in normalized actions vary across tasks and are largest in the base-motion channels. Guided by this observation, we examine base motion during closed-loop execution on PickPlaceCounterToStove, comparing a task expert with WA policies constructed from the task-specific-interface and shared-interface banks. The policies start from matched scenes with paired denoising noise. In the scene shown in Figure 6(c), task-specific WA produces substantial base drift, reaching 54.1 cm by step 160, whereas shared-interface WA stays within 0.1 cm of its initial position.

To examine how individual interface components affect control, we use experts trained with task-specific action interfaces and replace each expert's action encoder or decoder with its average across experts, or replace its action normalization with the base statistics, keeping all other components unchanged. Encoder averaging and normal-

Table 3: Sensitivity of individual experts to action-interface changes.

Setting	Success (%)
Individual	64.1
Encoder WA	17.1 (−47.0%)
Decoder WA	60.4 (−3.7%)
Base Normalization	12.3 (−51.8%)

Table 4: **LIBERO-10 selection and merging (500 episodes).** Accuracy and success are in %; $\bar{k}$ is the mean number of selected experts. The three shared-interface POLICYWEAVE variants use the same context-response scorer. Subset success uses cases where adaptive selection merges multiple experts, defined separately for each expert bank.

Method	Top-1 Acc.	Mean k	Success (all)	Subset success
Trainable MLP	89.4	1	69.4	–
POLICYWEAVE (Top-1)	93.6	1	73.2	36.4
POLICYWEAVE (Top-2)	93.6	2	66.2	57.6
POLICYWEAVE	93.6	1.076	**75.4**	**63.6**
POLICYWEAVE w/o shared interface	90.2	1.112	70.0	0.0



Figure 7: **Real-world evaluation setup.** Experiments use an AgileX Piper arm and two cameras across 4 manipulation tasks.

Table 5: **Real-robot success (%) over 20 rollouts per task and method.** Mean is the four-task average.

Task	$\pi_{0.5}$	GR00T N1.5	POLICYWEAVE
Upright bottle	30	10	**40**
Stack blocks	15	10	**55**
Press button	80	85	**95**
Place cup	70	65	**80**
Mean	48.8	42.5	**67.5**

ization replacement reduce success more than decoder averaging (Table 3). While these components differ in sensitivity, keeping the full interface shared ensures that merged hidden-layer updates use the same action input and output mappings.

3.4 EXPERT SELECTION AND SPARSE MERGING

Expert identification. We focus on LIBERO-10, where long-horizon tasks share objects and intermediate goals, to test expert scoring and the choice of which updates to merge. Table 4 compares three shared-interface POLICYWEAVE variants with a trainable MLP selector on the same expert bank. The MLP learns a context-to-expert mapping, analogous to the learned routing in Twin-Merging (Lu et al., 2024). Context-response scoring reaches 93.6% Top-1 accuracy versus 89.4% for the MLP. With single-expert selection in both cases, POLICYWEAVE (Top-1) also achieves higher closed-loop success than the MLP. The experts' own responses thus provide effective selection signals without training a separate selection model.

Sparse merging. The shared-interface POLICYWEAVE variants share expert scores and differ only in the selection rule. POLICYWEAVE (Top-2) lowers success relative to POLICYWEAVE (Top-1), showing that including another expert can introduce harmful interference. Stability-adaptive selection instead reaches 75.4% success versus 73.2% for Top-1, with only 1.076 experts selected on average. On the same episodes where adaptive selection uses $k > 1$, it achieves 63.6% success, compared with only 36.4% for Top-1. These results support adapting the number of merged updates to ranking stability rather than always selecting one or two experts. Applying the same selection and merging rule to experts trained without a shared action interface yields 90.2% Top-1 accuracy and 70.0% overall success. Expert identification remains effective, yet no rollout succeeds in that variant's own multi-expert subset. Identifying relevant experts therefore does not by itself ensure that their updates can be merged successfully. These results support preserving the shared interface not only for static merging, but also when sparse selection calls for merging multiple experts. Further ablations are provided in Appendix D.

3.5 REAL ROBOT DEPLOYMENT

We test independent post-training and merging on an AgileX Piper arm with six revolute joints and a parallel gripper (Figure 7). The four tasks are placing a bottle upright, stacking two blocks, pressing a button to turn on a light, and placing a cup on a coaster. We compare POLICYWEAVE with $\pi_{0.5}$ (Black et al., 2025a) and GR00T N1.5 (Bjorck et al., 2025) co-trained on all four tasks, using 20 rollouts per task and method. Table 5 shows that POLICYWEAVE outperforms GR00T co-training on all four tasks, with the largest gains on placing the bottle upright and stacking blocks. These

tasks require controlled object reorientation or precise placement and are the most difficult for the co-trained policy. With the same shared base and action interface, POLICYWEAVE allows each expert to learn its task independently. It achieves 100% Top-1 expert-selection accuracy across the four tasks, identifying the corresponding expert without a supplied task identity. Together, these results support independent adaptation followed by context-guided sparse merging as a way to deploy specialized skills that are harder to acquire through co-training. Data and training settings are given in Appendix E.

4 RELATED WORK

4.1 MULTI-TASK VISION-LANGUAGE-ACTION MODELS

Pretraining and post-training. Generalist vision-language-action (VLAs) (Brohan et al., 2023; Ghosh et al., 2024; Kim et al., 2025; Black et al., 2025b;a; Bjorck et al., 2025; Shukor et al., 2025) combine heterogeneous robot data with vision-language pretraining to support diverse tasks and embodiments. Supervised fine-tuning (SFT) adapts these policies to downstream tasks using expert demonstrations (Kim et al., 2025; Black et al., 2025b). RL further improves task performance through interaction (Li et al., 2026; Chen et al., 2025; Lyu et al., 2026; Wang et al., 2026; Cao et al., 2026).

Structured multi-task skill learning. AtomicVLA (Zhang et al., 2026) uses skill-guided action experts. DyGRO-VLA (Lin et al., 2026) routes grouped RL residuals across tasks. DiTEA (Li & Wang, 2026) uses language instructions to select experts within the action model. KinRT (Yang et al., 2026) uses kinematic trajectory clusters to supervise routing in an MoE action model. These methods modify the training architecture, whereas POLICYWEAVE merges independently post-trained updates without architectural changes.

4.2 MODEL MERGING

Model merging combines task-adapted models without joint retraining. Static methods produce one input-independent parameterization (Wortsman et al., 2022; Matena & Raffel, 2022; Ilharco et al., 2023). They address conflicts through sign resolution, sparsification, and masking (Yadav et al., 2023; Yu et al., 2024; Davari & Belilovsky, 2024; Wang et al., 2024b), or subspace decomposition and projection (Sun et al., 2025; Gargiulo et al., 2025; Marczak et al., 2025). Methods tailored to LoRA search over adapter combinations or align low-rank updates (Huang et al., 2024; Stoica et al., 2025; Zhao et al., 2025c; Panariello et al., 2025; Wei et al., 2026). Dynamic methods condition expert selection or merging on the input (Tang et al., 2024; Lu et al., 2024; Oh et al., 2025; Wang et al., 2024a). Policy perturbations can compound during closed-loop control, making VLA merging particularly challenging (Ross et al., 2011; Fu et al., 2026). CORAL (Luo et al., 2026) independently trains LoRA experts and folds one into the base policy using a client-side instruction-to-expert function. MergeVLA (Fu et al., 2026) redesigns the action expert and requires task-specific masks even with static merging methods. POLICYWEAVE instead preserves a shared action interface during post-training, making experts compatible with existing static merging methods. An extended comparison appears in Appendix A.

5 CONCLUSION

Independent post-training allows task-specific optimization, but deploying the resulting capabilities together requires compatible experts. POLICYWEAVE preserves the action interface during post-training, improving compatibility with existing static merging methods while retaining individual task performance. Context-guided sparse merging limits the remaining interference at deployment. RoboCasa365 experiments show that gains from independent RL carry into the merged policy, while LIBERO-10 and Piper arm experiments support long-horizon skill retention and real-world multi-task manipulation. Together, these results show how maintaining compatibility during post-training and controlling interference at deployment enable independently learned skills to support a multi-task policy. Separating expert post-training from merging also provides a basis for developing model merging methods for VLAs without redesigning the policy.

AI USE STATEMENT

We used OpenAI generative AI tools, including Codex, in three areas: (i) *writing*, to draft and revise manuscript text and improve clarity; (ii) *retrieval and discovery*, to find related work and relevant technical documentation; and (iii) *research execution*, to assist with method implementation, mathematical proof drafting, and result analysis and interpretation. The authors directed the research and take responsibility for the final content, including AI-assisted text and artifacts.

ETHICS STATEMENT

Simulation experiments use publicly released benchmarks and initialization checkpoints under their respective licenses. We do not redistribute third-party datasets or pretrained weights. The real-robot study concerns tabletop object manipulation. Because merged policies can change closed-loop behavior, physical deployment should include task-specific validation, workspace and collision constraints, emergency stopping, and appropriate human supervision.

REPRODUCIBILITY STATEMENT

- **Method.** Section 2 describes shared-interface post-training and context-guided sparse merging. Appendix B.4 provides the inference algorithm.
- **Training and evaluation.** Appendix C documents base checkpoint sources, training data, SFT and RL settings, merging hyperparameter selection, and evaluation protocols for both simulation benchmarks.
- **Real-robot experiments.** Appendix E specifies the robot platform, observations and actions, task definitions, demonstration collection, training configurations, and evaluation procedure.
- **Supplementary materials.** We provide anonymized code for expert loading, sparse and static merging, and simulation evaluation, together with real-robot demonstration videos.

REFERENCES

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. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Lang Cao, Renhong Chen, Luyi Li, Peng Wang, Mofan Peng, and Yitong Li. Z-1: Efficient reinforcement learning for vision-language-action models. *arXiv preprint arXiv:2606.31846*, 2026.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pp. 270–287, 2024.

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*, 2025. URL https://arxiv.org/abs/2409.15250.

Yuxia Fu, Zhizhen Zhang, Yuqi Zhang, Zijian Wang, Zi Huang, and Yadan Luo. MergeVLA: Cross-skill model merging toward a generalist vision-language-action agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22335–22347, 2026.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18695–18705, 2025. doi: 10.1109/CVPR52734.2025.01742.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

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. In *International Conference on Learning Representations*, 2022.

Jiaheng Hu, Jay Shim, Chen Tang, Yoonchang Sung, Bo Liu, Peter Stone, and Roberto Martin-Martin. Simple recipe works: Vision-language-action models are natural continual learners with reinforcement learning. In *Reinforcement Learning Conference*, 2026. URL https://arxiv.org/abs/2603.11653.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. In *First Conference on Language Modeling*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, pp. 2679–2713, 2025.

Chengxuan Li and Xingwan Wang. DiTEA: Mixture-of-experts for vision-language-action model in robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18379–18387, 2026. doi: 10.1609/aaai.v40i22.38902.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *International Conference on Learning Representations*, 2026.

Sixu Lin, Yunpeng Qing, Litao Liu, Ming Zhou, Ruixing Jin, Xiaoyi Fan, and Guiliang Liu. DyGRO-VLA: Cross-task scaling of vision-language-action models via dynamic grouped residual optimization. *arXiv preprint arXiv:2605.17486*, 2026.

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*, volume 36, 2023.

Huihan Liu, Changyeon Kim, Bo Liu, Minghuan Liu, and Yuke Zhu. Pretrained vision-language-action models are surprisingly resistant to forgetting in continual learning. *arXiv preprint arXiv:2603.03818*, 2026a.

Yuan Liu, Haoran Li, Shuai Tian, Yuxing Qin, Yuhui Chen, Yupeng Zheng, Yongzhen Huang, and Dongbin Zhao. Towards long-lived robots: Continual learning VLA models via reinforcement fine-tuning. In *Proceedings of Robotics: Science and Systems*, 2026b. doi: 10.15607/RSS.2026.XXII.086.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-Merging: Dynamic integration of modular expertise in model merging. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2505.

Yuankai Luo, Woping Chen, Tong Liang, and Zhenguo Li. CORAL: Scalable multi-task robot learning via LoRA experts. *arXiv preprint arXiv:2603.09298*, 2026.

Mingyang Lyu, Yinqian Sun, Erliang Lin, Huangrui Li, Ruolin Chen, Feifei Zhao, and Yi Zeng. Reinforcement fine-tuning of flow-matching policies for vision-language-action models. In *IEEE International Conference on Robotics and Automation*, 2026.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost Van De Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 43177–43199, 2025.

Michael S. Matena and Colin A. Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-1287.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, 2026.

Changdae Oh, Yixuan Li, Kyungwoo Song, Sangdoo Yun, and Dongyoon Han. DaWin: Training-free dynamic weight interpolation for robust adaptation. In *International Conference on Learning Representations*, 2025.

Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023.

Aniello Panariello, Daniel Marczak, Simone Magistri, Angelo Porrello, Bartłomiej Twardowski, Andrew D. Bagdanov, Simone Calderara, and Joost van de Weijer. Accurate and efficient low-rank model merging in core space. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2066.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.011. URL https://www.roboticsproceedings.org/rss21/p011.html.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

Ralf Römer, Yi Zhang, Yuming Li, and Angela P. Schoellig. CLARE: Continual learning for vision-language-action models via autonomous adapter routing and expansion. *IEEE Robotics and Automation Letters*, 2026. doi: 10.1109/LRA.2026.3693992.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 627–635, 2011.

Weijie Shen, Yitian Liu, Yuhao Wu, Zhixuan Liang, Sijia Gu, Dehui Wang, Tian Nian, Lei Xu, Yusen Qin, Jiangmiao Pang, Xinping Guan, Xiaokang Yang, and Yao Mu. Expertise need not monopolize: Action-specialized mixture of experts for vision-language-action learning. *arXiv preprint arXiv:2510.14300*, 2025.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with SVD to tie the KnOTS. In *International Conference on Learning Representations*, 2025.

Wenju Sun, Qingyong Li, Yangliao Geng, and Boyang Li. CAT merging: A training-free approach for resolving conflicts in model merging. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 57523–57543, 2025.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts, 2024. URL https://arxiv.org/abs/2402.00433.

Hanqing Wang, Bowen Ping, Shuo Wang, Xu Han, Yun Chen, Zhiyuan Liu, and Maosong Sun. LoRA-Flow: Dynamic LoRA fusion for large language models in generative tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 12871–12882, 2024a. doi: 10.18653/v1/2024.acl-long.695.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 50268–50287, 2024b.

Ziqian Wang, Rui Zhang, Yitian Liu, Xingjian Mao, Minqian Wang, and Yao Mu. Q-VGM: Q-guided value-gradient matching for offline-to-online RL of flow-matching VLA policies. *arXiv preprint arXiv:2606.08015*, 2026.

Zhengxuan Wei, Yi Dong, Zonghui Li, Xianhui Lin, Xing Liu, Hong Gu, Shaofeng Zhang, Wenbin Li, and Qi Fan. SSR-Merge: Subspace signal routing for training-free LoRA merging in diffusion models. In *International Conference on Machine Learning*, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 23965–23998, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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*, 2026.

Tianhang Yang, Yanze Zheng, Junjie Wang, Wei-Bin Kou, Ruotong Li, and Yujiu Yang. Route by kinematics, act by observation: Kinematics-supervised expert routing in MoE-augmented VLA. *arXiv preprint arXiv:2607.26807*, 2026. URL https://arxiv.org/abs/2607.26807.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 57755–57775, 2024.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.neurips.cc/paper_files/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf.

Qixin Zeng, Shuo Zhang, Hongyin Zhang, Renjie Wang, Han Zhao, Libang Zhao, Runze Li, Donglin Wang, and Chao Huang. CRL-VLA: Continual vision-language-action learning. *arXiv preprint arXiv:2602.03445*, 2026.

Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, Liang Lin, and Xiaodan Liang. AtomicVLA: Unlocking the potential of atomic skill learning in robots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20743–20754, 2026.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Tsung-Yi Lin, Gordon Wetzstein, Ming-Yu Liu, and Donglai Xiang. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1702–1713, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zhao_CoT-VLA_Visual_Chain-of-Thought_Reasoning_for_Vision-Language-Action_Models_CVPR_2025_paper.html.

Ruihan Zhao, Tyler Ingebrand, Sandeep Chinchali, and Ufuk Topcu. MoS-VLA: A vision-language-action model with one-shot skill adaptation. *arXiv preprint arXiv:2510.16617*, 2025b.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, and Fei Wu. Merging LoRAs like playing LEGO: Pushing the modularity of LoRA to extremes through rank-wise clustering. In *International Conference on Learning Representations*, 2025c.

APPENDIX OVERVIEW

The appendix is organized as follows:

- Appendix A: extended related work and comparisons with CORAL and MergeVLA.
- Appendix B: action-interface analysis, context-response scoring, ranking stability, and the inference algorithm.
- Appendix C: checkpoints, training, evaluation protocols, and baseline configurations.
- Appendix D: interface diagnostics, task-level results, expert selection, robustness, and scaling.
- Appendix E: the Piper setup, training, and successful rollout sequences.
- Appendix F: limitations and directions for future work.

A EXTENDED RELATED WORK

A.1 MULTI-TASK AND MODULAR VLA LEARNING

Generalist VLAs acquire broad robot capabilities by sharing parameters across heterogeneous datasets during pretraining or joint finetuning. This line spans autoregressive action tokenization, continuous action generation, open cross-embodiment datasets, and resource-efficient policies (Brohan et al., 2023; Open X-Embodiment Collaboration, 2023; Ghosh et al., 2024; Kim et al., 2025; Black et al., 2025b;a; Bjorck et al., 2025; Shukor et al., 2025). Although scale and data diversity improve transfer, adding a new deployment task generally entails updating a shared policy or repeating multi-task training with the enlarged task set.

Modular approaches introduce explicit structure for specialization. MoS-VLA (Zhao et al., 2025b) learns a basis of skills during pretraining and infers their combination from a new demonstration, while AtomicVLA (Zhang et al., 2026) organizes action generation around skill-guided experts. DiTEA (Li & Wang, 2026) and AdaMoE (Shen et al., 2025) place sparsely activated experts inside the action model, with learned mechanisms for expert selection and contribution. DyGRO-VLA (Lin et al., 2026) instead learns and routes grouped reinforcement-learning residuals for cross-task policy optimization. These systems learn their modular structure or routing mechanism as part of a prescribed training pipeline. POLICYWEAVE addresses a different point in the workflow: task policies are post-trained independently under a common action interface and are merged afterward.

Continual VLA learning updates one policy as tasks arrive sequentially. Recent methods expand and route adapters, use replay, or continue supervised and reinforcement learning while measuring retention of earlier skills (Römer et al., 2026; Liu et al., 2026a;b; Zeng et al., 2026; Hu et al., 2026). This setting is related through its goal of extending a deployed policy, but differs from post-hoc model merging because the earlier and new skills participate in a sequential update process.

A.2 VLA POLICY MERGING AND EXPERT SELECTION

Policy parameters participate in closed-loop control, so a small action change can alter later observations and compound over a trajectory (Ross et al., 2011). RETAIN (Yadav et al., 2026) merges a pretrained VLA with one task-adapted descendant to preserve general capabilities during robust finetuning. ReVLA (Dey et al., 2025) uses gradual model merging to move OpenVLA's adapted visual backbones toward their pretrained states and recover visual out-of-domain generalization. This is a within-policy repair of visual representations rather than consolidation of independently post-trained task experts. CORAL (Luo et al., 2026) and MergeVLA (Fu et al., 2026) also maintain independently specialized components and select among them at deployment without a supplied expert index.

CORAL. CORAL (Luo et al., 2026) freezes a pretrained VLA and learns an isolated LoRA expert for each task. Its manager delegates expert choice to a client-side instruction-to-expert function, then folds the selected adapter into the base weights for standard inference. This design avoids cross-task parameter interference by executing one isolated expert at a time. The routing function is part of the deployment interface rather than a specified model-derived scorer, and CORAL does not

Table 6: Comparison with CORAL and MergeVLA. "Task unknown" means that an expert index is not provided at inference time.

Aspect	CORAL	MergeVLA	POLICYWEAVE
Expert construction	Isolated task LoRAs on a frozen base	Experts trained with a merge-oriented design	Independent task LoRAs with a shared action interface
Action model	Original action head retained and adapted by LoRA	Cross-attention-only expert; deeper task heads retained	Original architecture retained; action encoder and decoder fixed
Task-unknown selection	Externally specified instruction-to-expert function	Model-response-based task selection	Expert-update responses with stability-adaptive selection
Deployment	One selected LoRA folded into the base	Merged shared blocks with a selected mask and task head	Static policy from the shared-interface bank, or context-guided sparse merging
Static merging	Not part of the proposed method	Coupled with task-specific masks and heads	Existing methods produce one policy without task-specific selection

combine multiple task updates within one deployed policy. In contrast, POLICYWEAVE derives its decision from the VLA's context responses and can merge a sparse set of compatible task updates. To compare this deployment choice on a common expert bank, Table 15 evaluates two client-side hard selectors, a trained context MLP and Semantic-NN, against PolicyWeave.

MergeVLA. MergeVLA (Fu et al., 2026) makes the VLA architecture merge-oriented. It applies task-specific masks to merged VLM updates, replaces the action expert with cross-attention-only blocks, and retains a task-specific deeper action head. Its reported TA, WUDI, and TIES variants pair each underlying operator with the task mask; applying standard merging methods directly to the original VLA specialists gives zero success on LIBERO. When no task index is supplied, its training-free router evaluates task-masked VLM variants and selects the mask and action head whose representations best match value-projection subspaces in the action expert. Thus, shared components are merged, while the selected mask and action head remain task-specific at execution.

Position of POLICYWEAVE. MergeVLA addresses mergeability through architecture design, whereas POLICYWEAVE addresses it during independent post-training. Its shared action interface allows existing static methods to produce a single policy without the task-specific masks and heads used by MergeVLA. Expert compatibility can thus be studied separately from the merging algorithm. For context-guided sparse merging, POLICYWEAVE specifies both the expert-response scorer and stability-based set selection, in contrast to CORAL's externally specified single-expert selection.

B ADDITIONAL POLICYWEAVE DETAILS

B.1 SHARED ACTION INTERFACE

We use a scalar linear policy to isolate the effect of a varying action interface. Let z be a common input, h_t expert t's hidden-layer coefficient, and d_t its output-interface coefficient. Its physical action is $a_t = d_t h_t z$. For K experts, define the parameter averages $\bar{d} = \frac{1}{K}\sum_{t=1}^{K} d_t$ and $\bar{h} = \frac{1}{K}\sum_{t=1}^{K} h_t$. The parameter-averaged policy produces $\bar{a} = \bar{d}\bar{h}z$. Its difference from the average expert action is

$$\bar{a} - \frac{1}{K}\sum_{t=1}^{K} a_t = -\frac{1}{K}\sum_{t=1}^{K}(d_t - \bar{d})(h_t - \bar{h})z. \tag{6}$$

Expanding the centered product gives $\frac{1}{K}\sum_t (d_t - \bar{d})(h_t - \bar{h}) = \frac{1}{K}\sum_t d_t h_t - \bar{d}\bar{h}$, establishing the identity. Thus, joint variation in the interface and hidden-layer coefficients can make parameter averaging differ from action averaging. When all experts share $d_t = d_0$, the right-hand side is zero and $\bar{a} = d_0\bar{h}z = \frac{1}{K}\sum_t a_t$.

For a concrete example, take $(d_1, h_1) = (1, 1)$ and $(d_2, h_2) = (2, \frac{1}{2})$. Both experts produce $a_t = z$. Averaging their parameters instead gives

$$\bar{a} = \left(\frac{d_1 + d_2}{2}\right)\left(\frac{h_1 + h_2}{2}\right) z = \frac{9}{8} z. \tag{7}$$

The second expert halves its hidden prediction and doubles the output scale, preserving its action before merging. Averaging both coefficients breaks this compensation. A shared interface eliminates this discrepancy in the linear model; whether averaging different task behaviors yields useful control remains a separate question.

B.2 Context-Response Scoring

Score interpretation. For one expert and scoring layer, write $\mathbf{U} = \Delta\mathbf{W}^{\mathrm{K}}_{t,\ell}$ and $\mathbf{H} = \mathbf{H}_\ell$, and define $\mathbf{Q} = \mathbf{U}^\top\mathbf{U}$ and $\mathbf{S} = \mathbf{H}\mathbf{H}^\top$. Both are positive semidefinite matrices in $\mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{in}}}$. The matrix $\mathbf{Q}$ describes the input directions to which the update responds, while $\mathbf{S}$ is the unnormalized second-moment matrix of the context tokens. For nonzero $\mathbf{U}$ and $\mathbf{H}$, cyclicity of the trace gives

$$e_{t,\ell} = \frac{\operatorname{tr}(\mathbf{U}\mathbf{H}\mathbf{H}^\top\mathbf{U}^\top)}{\operatorname{tr}(\mathbf{U}^\top\mathbf{U})\operatorname{tr}(\mathbf{H}\mathbf{H}^\top)} = \frac{\operatorname{tr}(\mathbf{Q}\mathbf{S})}{\operatorname{tr}(\mathbf{Q})\operatorname{tr}(\mathbf{S})}. \tag{8}$$

Thus, the score measures directional overlap between the update's sensitivity and the context's second moment. Since $\operatorname{tr}(\mathbf{Q}\mathbf{S}) \geq 0$ and $\operatorname{tr}(\mathbf{Q}\mathbf{S}) \leq \lambda_{\max}(\mathbf{Q})\operatorname{tr}(\mathbf{S}) \leq \operatorname{tr}(\mathbf{Q})\operatorname{tr}(\mathbf{S})$, where $\lambda_{\max}$ denotes the largest eigenvalue, we have $0 \leq e_{t,\ell} \leq 1$. Multiplying either $\mathbf{U}$ or $\mathbf{H}$ by any nonzero scalar leaves the score unchanged. This removes overall magnitude as a source of score differences; expert identification depends on how the learned update directions align with the observed context.

Tokenwise form. In Equation (3), $\mathbf{H}_\ell$ stacks only valid context-token representations. For a padded sequence of P tokens, let $\mathbf{h}_{\ell,p}$ be the input of token p to the key projection and let $\mu_p \in \{0, 1\}$ indicate whether that token is valid. The equivalent tokenwise score is

$$e_{t,\ell} = \frac{\sum_{p=1}^{P} \mu_p \|\Delta\mathbf{W}^{\mathrm{K}}_{t,\ell}\mathbf{h}_{\ell,p}\|_2^2}{\|\Delta\mathbf{W}^{\mathrm{K}}_{t,\ell}\|_F^2 \sum_{p=1}^{P} \mu_p \|\mathbf{h}_{\ell,p}\|_2^2}. \tag{9}$$

The key-projection update includes all attention heads. Summing squared column norms gives the squared matrix norm in the main-text expression; padding tokens contribute to neither the numerator nor the denominator.

Efficient low-rank computation. The response score in Equation (3) can be evaluated without materializing the dense update $\Delta\mathbf{W}^{\mathrm{K}}_{t,\ell}$. The LoRA parameterization at an adapted layer is

$$\Delta\mathbf{W}_{t,\ell} = \frac{\alpha_\ell}{r_\ell}\mathbf{B}_{t,\ell}\mathbf{A}_{t,\ell}, \tag{10}$$

where r_ℓ is the rank, α_ℓ is the scaling factor, and $\mathbf{A}_{t,\ell} \in \mathbb{R}^{r_\ell \times d_{\mathrm{in}}}$ and $\mathbf{B}_{t,\ell} \in \mathbb{R}^{d_{\mathrm{out}} \times r_\ell}$ are the LoRA factors for layer input and output dimensions d_{in} and d_{out}. The response to a context token $\mathbf{h}$ satisfies

$$\|\Delta\mathbf{W}_{t,\ell}\mathbf{h}\|_2^2 = \left(\frac{\alpha_\ell}{r_\ell}\right)^2 (\mathbf{A}_{t,\ell}\mathbf{h})^\top(\mathbf{B}_{t,\ell}^\top\mathbf{B}_{t,\ell})(\mathbf{A}_{t,\ell}\mathbf{h}), \tag{11}$$

while the update norm is

$$\|\Delta\mathbf{W}_{t,\ell}\|_F^2 = \left(\frac{\alpha_\ell}{r_\ell}\right)^2 \operatorname{tr}\big(\mathbf{A}_{t,\ell}\mathbf{A}_{t,\ell}^\top\mathbf{B}_{t,\ell}^\top\mathbf{B}_{t,\ell}\big). \tag{12}$$

The LoRA scaling cancels in the normalized response. A single base-context pass therefore suffices to score all experts using low-rank products. With $\mathbf{B}_{t,\ell}^\top\mathbf{B}_{t,\ell}$ and update norms cached, scoring one expert at one layer costs $\mathcal{O}\big(P(r_\ell d_{\mathrm{in}} + r_\ell^2)\big)$, so the full scorer grows linearly with the number of experts and scoring layers.

B.3 Stability of Expert Selection

The single-layer removal test also controls sensitivity to partial reductions in layer contributions. We analyze the score ranking for a fixed context.

Proposition: stability under layer downweighting. Let $a = \rho_1$ be the unique highest-scoring expert. Suppose a remains the unique highest-scoring expert after removing any one scoring layer. For any layer-weight reductions $\delta_\ell \geq 0$ satisfying $\sum_{\ell \in \mathcal{C}} \delta_\ell \leq 1$, define

$$\widetilde{s}_t = \sum_{\ell \in \mathcal{C}} (1 - \delta_\ell) e_{t,\ell}. \tag{13}$$

Then a remains the unique highest-scoring expert under $\widetilde{s}_t$. Thus, the test covers not only removal of one layer, but also partial downweighting spread across several layers, such as halving the contributions of two layers.

Proof. For a competing expert $j \neq a$, let $m_j = s_a - s_j$ denote the original score margin. After removing layer ℓ, this margin is $m_j - (e_{a,\ell} - e_{j,\ell})$. The assumed strict stability is therefore equivalent to

$$m_j > \max_{\ell \in \mathcal{C}} (e_{a,\ell} - e_{j,\ell}), \qquad \text{for every } j \neq a. \tag{14}$$

Write $D = \sum_{\ell \in \mathcal{C}} \delta_\ell$. Since $s_t^{(-\ell)} = s_t - e_{t,\ell}$, the downweighted scores satisfy

$$\widetilde{s}_t = (1 - D) s_t + \sum_{\ell \in \mathcal{C}} \delta_\ell s_t^{(-\ell)}. \tag{15}$$

The coefficients are nonnegative and sum to one. Expert a has a strictly positive margin over every competitor in each score vector on the right, so it also has a strictly positive margin in their convex combination. Hence the leading expert is unchanged.

Adaptive set size. With a fixed tie-breaking rule, Equation (5) returns the smallest prefix of the aggregate ranking that contains every leave-one-layer-out top-ranked expert. Any shorter prefix excludes an expert whose rank attains $k^\star$, whereas the prefix of length $k^\star$ contains them all. In particular, $k^\star = 1$ exactly when the original leading expert remains top-ranked under every layer removal; in this case, Equation (2) recovers that expert's parameter update exactly.

B.4 Episode-Level Inference

Algorithm 1 gives the complete inference procedure. Here $\mathbf{x}_\tau$ contains the observation and instruction at control step τ, and $\widehat{\mathbf{a}}_\tau$ is the predicted action chunk. We use π_{θ, m_0} for the policy with parameters θ and the shared action normalization metadata m_0. Encoding $\mathbf{x}_0$ supplies the initial visual-language context c used in Equation (2). The base pass supplies routing scores; the merged update is then applied to all 120 adapted action-model modules and held fixed for the episode. In practice, we apply the merged update in low-rank form without materializing a dense weight matrix. Hard Top-1 and Hard Top-2 use the same expert scores and score-normalized weights but fix the selected set size.

C Experimental Setup and Protocols

C.1 Expert Banks

For RoboCasa365 (Nasiriany et al., 2026), experts start from the official GR00T N1.5 checkpoint (Bjorck et al., 2025) trained on the benchmark's pretraining split. LIBERO-10 experts start from a public GR00T N1.5 checkpoint trained on LIBERO-90. Within each expert bank, every checkpoint is derived from the same base model and retains the shared PolicyWeave action normalization, encoder, and decoder. Only rank-8 LoRA adapters on the 120 action-model linear modules are updated. RoboCasa365 contains 18 SFT experts, each trained with 50 demonstrations, which corresponds to 10% of the available training data. Its second bank continues each SFT expert with Flow-SDE reinforcement learning (Chen et al., 2025). Thus, every RL-adapted expert is initialized

Algorithm 1 Episode-level inference with POLICYWEAVE

Require: Base policy π_{θ_0,m_0}; PolicyWeave updates $\{\Delta_t\}_{t\in\mathcal{T}}$; initial context $\mathbf{x}_0$; scoring layers $\mathcal{C}$; adapted layers $\mathcal{H}$

Disable all task adapters and encode $\mathbf{x}_0$ with the base policy to obtain the valid key-projection inputs $\{\mathbf{H}_\ell\}_{\ell\in\mathcal{C}}$
for all $(t,\ell)\in\mathcal{T}\times\mathcal{C}$ **do**
    Compute $e_{t,\ell}$ from Equation (3), using Equations (11) and (12)
end for
$s_t \leftarrow \sum_{\ell\in\mathcal{C}} e_{t,\ell}$ for every $t\in\mathcal{T}$
Order the experts as $\rho=(\rho_1,\dots,\rho_K)$ such that $s_{\rho_1}\geq\cdots\geq s_{\rho_K}$
for all $\ell\in\mathcal{C}$ **do**
    $t^\star_\ell \leftarrow \arg\max_{t\in\mathcal{T}}(s_t - e_{t,\ell})$
end for
$k^\star \leftarrow \max_{\ell\in\mathcal{C}} \mathrm{rank}_\rho(t^\star_\ell)$
$\mathcal{I}^\star \leftarrow \{\rho_1,\dots,\rho_{k^\star}\}$
$\gamma_t \leftarrow s_t / \sum_{j\in\mathcal{I}^\star} s_j$ for every $t\in\mathcal{I}^\star$
for all $u\in\mathcal{H}$ **do**
    $\Delta\mathbf{W}_u \leftarrow \sum_{t\in\mathcal{I}^\star}\gamma_t\Delta\mathbf{W}_{t,u}$
end for
Add $\{\Delta\mathbf{W}_u\}_{u\in\mathcal{H}}$ to the base weights, keeping the shared action interface fixed
$\tau\leftarrow 0$
while the episode has not terminated **do**
    Predict $\widehat{\mathbf{a}}_\tau$ from $\mathbf{x}_\tau$ with the merged policy
    Execute $\widehat{\mathbf{a}}_\tau$ and observe $\mathbf{x}_{\tau+1}$
    $\tau\leftarrow\tau+1$
end while

from its corresponding SFT checkpoint. LIBERO-10 contains a single bank of 10 SFT experts trained with all available demonstrations for their respective tasks. The number of demonstrations ranges from 29 to 49, with a mean of 37.9 per task.

C.2 TRAINING DETAILS

Each task-specific job uses one NVIDIA RTX PRO 6000 Blackwell GPU. SFT uses the flow-matching objective in bfloat16, AdamW with $(\beta_1,\beta_2)=(0.95,0.999)$ and $\epsilon=10^{-8}$, a cosine schedule with 5% warmup, and no gradient accumulation. Both benchmarks use LoRA rank 8, scaling factor 16, and dropout 0; Table 7 gives the remaining settings.

Table 7: SFT configuration for each task-specific expert.

Benchmark	Data per task	Steps	Batch	LR	Weight decay
RoboCasa365	50 demos (10%)	10,000	128	3×10^{-5}	10^{-5}
LIBERO-10	all (37.9 avg.)	10,000	128	3×10^{-5}	10^{-5}

Co-training baselines. The RoboCasa365 *Co-training* SFT baseline uses the official checkpoint co-trained on all 18 target tasks with 10% of the training data (Nasiriany et al., 2026). We initialize our joint RL continuation from this checkpoint. For LIBERO-10, we re-evaluate the GR00T N1.5 co-training checkpoint provided in the official LIBERO example of NVIDIA's Isaac-GR00T repository (`n1.5-release`).

Reinforcement learning. RoboCasa365 RL uses PPO actor-critic optimization with Flow-SDE action log-probabilities, four rollout epochs, two update epochs, $\gamma=0.99$, GAE 0.95, policy and value clipping at 0.2, weight decay 0.01, and gradient-norm clipping at 10. LoRA optimization uses float32 and policy forward passes use bfloat16. The selected task-specific RL checkpoints sum to 610 steps; the *Co-training* SFT policy undergoes 600 joint RL steps. Task-specific checkpoints are selected by per-task validation success, while the cotrained checkpoint is selected by mean validation

success across the 18 tasks. Checkpoint selection and final evaluation use separate rollout sets. Tables 8 and 9 give the configurations and validation curve; the selected joint checkpoint is at step 600.

Table 8: RoboCasa365 SFT then RL configurations at the selected checkpoints. M denotes million; batch sizes are micro/global.

Training	RL steps	Envs	Env. steps (M)	Batch	Actor LR	Value LR
Individual	33.9 avg. (610 total)	16	23.74	8/304	10^{-5}	10^{-4}
Co-training	600 total	18	26.27	8/304	10^{-5}	10^{-4}

Table 9: RoboCasa365 co-training RL checkpoint-selection curve. Values are mean validation success rates (%) over the 18 tasks.

RL step	0	100	200	300	400	500	600
Success	58.1	57.2	59.8	59.6	60.1	59.8	**60.7**

C.3 Rollout Protocol and Evaluation Metrics

RoboCasa365 evaluation uses the official version 1.0.1 target split, four Euler integration steps for flow sampling, and an action horizon of 16. LIBERO-10 episodes contain at most 480 simulator steps and execute action chunks of length 10. These settings are held fixed across the methods compared within each benchmark. RoboCasa365 results are averaged over 50 episodes per task. On LIBERO-10, each method is evaluated in separate closed-loop rollouts using the same reset schedule and policy-noise seeds.

For LIBERO-10, let $\mathcal{G}_e$ be the set of goal predicates for episode e, and let $y_{e,\tau,j}$ indicate whether predicate j is satisfied at timestep τ. We define partial success as

$$\mathrm{PS} = \frac{1}{N_{\mathrm{eval}}} \sum_{e=1}^{N_{\mathrm{eval}}} \max_{1 \le \tau \le T_e} \frac{1}{|\mathcal{G}_e|} \sum_{j \in \mathcal{G}_e} y_{e,\tau,j}, \tag{16}$$

where T_e is the episode length and N_{eval} is the number of evaluation episodes. This metric measures peak simultaneous goal completion, capturing partial progress on long-horizon tasks.

C.4 Merging Baselines

All merging methods operate on the induced weight updates $\Delta \mathbf{W}_{t,u}$. WA assigns a coefficient of $1/K$ to each of the K task updates. TIES, TSV-Merge, and ISO-CTS operate in the shared low-rank Core Space (Panariello et al., 2025) before reconstruction in weight space. SSR-Merge uses its LoRA subspace signal-routing formulation. Let λ denote the global coefficient applied to the merged task update. For TIES, TSV-Merge, and ISO-CTS, we select $\lambda \in \{0.1, 0.2, \ldots, 1.0\}$ using validation rollouts across all target tasks, with seeds disjoint from the final evaluation, and hold it fixed during testing. The selected values are 0.8 for TIES, 0.3 for TSV-Merge, and 0.4 for ISO-CTS across all three expert banks. We set $\lambda = 1.0$ for SSR-Merge. All other hyperparameters follow the defaults of the corresponding methods (Yadav et al., 2023; Gargiulo et al., 2025; Marczak et al., 2025; Wei et al., 2026). For the RoboCasa365 SSR-Merge baseline, the routing transform is estimated from five successful trajectories per task, with ten states sampled from each trajectory. In the main comparisons, all static merged checkpoints are materialized on the same base policy and retain the shared action interface.

C.5 Expert Selection Baselines

Trainable MLP. The MLP learns a context-to-expert mapping, analogous to the learned routing in Twin-Merging (Lu et al., 2024), and selects one expert. On both benchmarks, it receives a masked mean of the 2,048-dimensional frozen-base visual-language context and has two hidden layers of

width 128. Its output contains one score per expert: 18 on RoboCasa365 and 10 on LIBERO-10. The highest-scoring expert is selected for the episode. We train it on 10 demonstrations per task, using balanced cross-entropy and 500 AdamW steps with batch size 2 and learning rate 10^{-3} on both benchmarks. We use the final training checkpoint.

Semantic-NN. Semantic-NN (Sem-NN) uses the frozen `all-mpnet-base-v2` text encoder (Reimers & Gurevych, 2019; Song et al., 2020). Each expert is represented by one lowercased, word-separated task-name prototype. Selection uses cosine similarity to the instruction embedding, without demonstration calibration.

C.6 Published Leaderboard References

Figure 4 includes selected results from the official RoboCasa365 leaderboard. The selected Atomic-Seen scores cover 18 tasks in pretraining kitchens under the multi-task learning protocol. Our results use target kitchens under the foundation-model learning protocol, with 50 SFT demonstrations per task (10% of target demonstrations) followed by RL. This fraction refers to SFT data, separately from base pretraining and RL interaction. Published entries use their original configurations; the leaderboard's GR00T N1.5 entry uses RoboCasa 1.0.1 with a rollout horizon 1.5 times that of the original paper.

D Additional Results and Analysis

D.1 Action-Interface Compatibility

Task-specific action-interface baseline. This rank-8 LoRA bank uses the same GR00T N1.5 base and 18-task evaluation protocol. It additionally adapts the action encoder and decoder and stores task-specific normalization. Each merging method combines the learned action modules, while evaluation manually loads the target task's normalization for the merged checkpoint and thus assumes task identity.

Table 10: Static merging under task-specific and shared action interfaces on RoboCasa365. Retention is the success rate of each merging method divided by Individual success. All entries are percentages.

Action interface	Individual	WA	TIES	ISO-CTS	TSV-Merge	Retention range
Task-specific	64.1	16.1	13.9	18.9	18.9	21.7–29.5
PolicyWeave	65.6	49.6	52.9	54.1	54.7	75.6–83.4

LIBERO-10 without a shared action interface. For the PolicyWeave variant in Table 4, single-expert selection uses the selected expert's normalization, encoder, and decoder. When multiple experts are selected, their normalization statistics are combined, and their encoder and decoder parameters are merged with the same weights as their hidden-layer updates. The interface therefore follows the selected experts rather than a supplied task identity. Table 11 reports additional aggregate results for the four static merging methods without a shared action interface. All four methods have zero final success and 8.3–13.2% partial success (Equation (16)), showing limited intermediate-goal progress.

Table 11: LIBERO-10 aggregate success rates without a shared action interface. Partial success measures the maximum fraction of goal predicates satisfied simultaneously during an episode, averaged across episodes. All entries are percentages.

Metric	WA	TIES	ISO-CTS	TSV-Merge
Final success	0.0	0.0	0.0	0.0
Partial success	10.6	9.8	8.3	13.2

Normalization and closed-loop diagnostics. For the paired normalization diagnostic, relative ℓ_2 shift is $\|\mathbf{A}_{\text{shared}} - \mathbf{A}_{\text{task}}\|_F / \|\mathbf{A}_{\text{task}}\|_F$, where each matrix contains the same sampled raw actions transformed by the indicated normalization. We use 1,800 raw actions from 18 RoboCasa365 tasks, sampling 20 timesteps from five demonstrations per task. The heatmap averages absolute differences in normalized actions over samples and dimensions within each action group.

Figure 6(c) compares a task expert, task-specific WA, and shared-interface WA across three matched scenes, with paired denoising noise. Rollouts use a 600-step horizon, action chunks of length 16, and four Euler steps. Base displacement is $100\|\mathbf{p}_\tau^{xy} - \mathbf{p}_0^{xy}\|_2$ in centimetres, where $\mathbf{p}_\tau^{xy}$ is the mobile base's planar world position in metres at timestep τ. The figure shows the first scene at step 160, with the expert trajectory as a visual reference. Across the three scenes, maximum base displacement is 88.3–122.0 cm for task-specific WA and 0.6–1.6 cm for shared-interface WA.

D.2 LIBERO-10 Per-Task Results

Published co-training references. Figure 5 uses published LIBERO-Long success rates for jointly fine-tuned OpenVLA (Kim et al., 2025), SpatialVLA (Qu et al., 2025), and CoT-VLA (Zhao et al., 2025a), under their original configurations. These reports provide final success only. The GR00T N1.5 result is our re-evaluation of the official checkpoint described in Appendix C.2.

Task-level results. Table 12 reports final and partial success for each task underlying the averages in Figure 5. Each task is evaluated over 50 episodes; the Average row gives the equally weighted mean across the 10 tasks. Partial success is defined in Equation 16.

Table 12: LIBERO-10 success (%), averaged over 50 episodes per task. Each cell reports final success above partial success, the maximum fraction of goal predicates satisfied simultaneously during an episode. Bold marks the best merging result for each task and metric.

	Baselines		Merging methods				
Task	*Indiv.*	GR00T N1.5	WA	TIES	ISO-CTS	TSV	**Policy Weave**
SOUP + SAUCE → BASKET	88 91	72 80	24 62	34 67	28 63	48 73	**84** **89**
CREAM CHEESE + BUTTER → BASKET	86 92	84 92	10 51	12 52	24 58	34 65	**70** **84**
TURN ON STOVE + MOKA POT	94 96	78 88	0 50	0 50	0 50	0 50	**90** **94**
BLACK BOWL → BOTTOM DRAWER + CLOSE	90 95	90 94	0 50	2 51	2 51	18 58	**74** **87**
TWO MUGS → TWO PLATES	66 79	82 89	6 49	2 46	10 50	58 76	**66** **78**
BOOK → CADDY BACK COMPARTMENT	98 98	100 100	92 92	90 90	92 92	92 92	**100** **100**
MUG → PLATE + PUDDING RIGHT	88 93	64 79	2 45	4 49	2 49	10 48	**78** **88**
SOUP + CREAM CHEESE → BASKET	88 91	68 79	6 49	14 52	8 54	18 59	**88** **92**
TWO MOKA POTS → STOVE	32 70	14 62	0 65	2 62	0 65	4 64	**32** **73**
MUG → MICROWAVE + CLOSE	78 84	72 78	0 26	0 31	0 24	0 27	**72** **79**
Average	80.8 88.9	72.4 84.1	14.0 53.9	16.0 55.0	16.6 55.6	28.2 61.2	**75.4** **86.4**

MergeVLA reproduction. We reproduce MergeVLA (Fu et al., 2026) on LIBERO-10 using its own backbone and fine-tuning pipeline. Table 13 reports its individual and merged results.

D.3 Expert Selection Statistics

Table 14 reports the selection-size distributions for the shared-interface expert banks.

Table 13: MergeVLA reproduction on LIBERO-10. FT evaluates the task-specific fine-tuned policies and Merged evaluates the resulting merged policy.

Task index	FT (%)	Merged (%)
0	86	76
1	98	98
2	100	94
3	100	74
4	84	78
5	90	82
6	80	30
7	94	68
8	72	32
9	92	76
Average	**89.6**	**70.8**

Table 14: Expert selection sizes under POLICYWEAVE. Columns $k = 1$ through $k = 4$ give episode counts; $k > 1$ gives the percentage of episodes selecting multiple experts. No episode selects more than four experts.

Expert bank	Episodes	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k > 1$ (%)	Mean k
RoboCasa365 SFT	900	887	13	0	0	1.44	1.014
RoboCasa365 SFT then RL	900	889	11	0	0	1.22	1.012
LIBERO-10 SFT	500	467	29	3	1	6.60	1.076

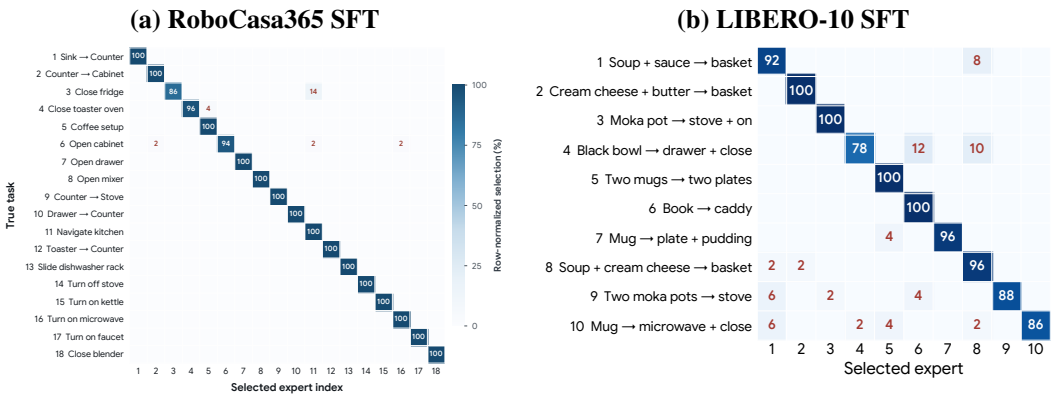


Figure 8: Row-normalized Top-1 routing confusion matrices (%) for the SFT banks: (a) RoboCasa365 (900 episodes); (b) LIBERO-10 (500 episodes). Rows indicate the requested task; columns indicate the selected expert.

Figure 8 compares expert selection on the RoboCasa365 and LIBERO-10 SFT banks. RoboCasa365 is nearly diagonal, whereas LIBERO-10 includes confusions among tasks sharing objects or intermediate goals, such as basket placement and manipulation of mugs or moka pots.

D.4 INSTRUCTION AND OBSERVATION ROBUSTNESS

The instruction comparison in Table 15 uses the RoboCasa365 SFT expert bank and 900 demonstration-first observations per condition, with 50 observations from each task. The lexical/syntactic templates change command wording and structure while preserving task-critical slots; the goal-state templates describe the same desired outcomes. All selectors remain fixed during evaluation, and test task identities are used only to compute metrics.

RoboCasa365 comparison. Table 15 reports instruction robustness and native-instruction closed-loop success. POLICYWEAVE leads on native and lexical/syntactic instructions, whereas Semantic-NN leads on goal-state rephrasings. The closed-loop results follow the native-instruction ranking.

Table 15: RoboCasa365 SFT selection baselines (%). Routing accuracy uses 900 recorded contexts per condition; closed-loop success uses 50 episodes per task. Sem-NN denotes task-name-based semantic retrieval.

Condition	Trainable MLP	Sem-NN	POLICYWEAVE
Top-1 routing accuracy			
Native	94.33	80.56	**99.44**
Lexical/syntactic	79.67	83.78	**91.33**
Goal-state	67.78	**82.56**	81.78
Closed-loop success			
Native	63.1	62.0	**64.7**

On initial rollout observations, Top-1 accuracy is 92.33% for the MLP, 81.44% for Semantic-NN, and 98.67% for POLICYWEAVE.

Observation timing. We evaluate the RoboCasa365 SFT then RL bank on recorded observations at timesteps 0, 16, 32, and 48 from the same 50 demonstrations per task. Instructions remain unchanged, and camera views and robot states are aligned at each timestep. Across these 900 contexts per timestep, Top-1 accuracy remains above 99% (Table 16), indicating that selection does not require the exact initial frame.

Table 16: POLICYWEAVE expert identification from early-trajectory RoboCasa365 observations (900 contexts per timestep).

Timestep	0	16	32	48
Top-1 accuracy (%)	99.44	99.56	99.89	99.89

D.5 POLICYWEAVE SCORER ABLATION

We compare scorer variants on 900 initial demonstration observations from the RoboCasa365 SFT bank, with 50 per task and native instructions. A base pass supplies context at eight cross-attention layers using a zero action latent and timestep zero. The 66 configurations compare key (K), value (V), and combined key-and-value (K + V) update responses, with raw or normalized scores across individual layers, the first or last four layers, and all eight layers. Cross 0–7 index the cross-attention layers in network order. Normalized scores follow Equation (3); raw scores use its numerator. The K + V variant adds the key and value scores within each layer before aggregating across layers.

Table 17 shows that accuracy varies with depth. Cross 3 gives the strongest single-layer K result. Although V scores often perform better at individual layers, aggregating all eight layers favors K over both V and K + V, with either raw or normalized scores.

Across the 18 multi-layer variants, adaptive selection gives 98.56–100% expert recall with mean k between 1.006 and 1.042. Using all eight layers, normalized K scores achieve 99.44% recall at mean $k = 1.006$, while raw K scores achieve 99.78% at 1.007.

D.6 SCALING THE EXPERT BANK

We vary bank size over $K \in \{1, 2, 4, 8, 12, 18\}$ on NavigateKitchen, PickPlaceSinkToCounter, CloseFridge, OpenCabinet, and TurnOnMicrowave, using 50 initial demonstration observations per

Table 17: Top-1 routing accuracy (%) for all 66 scorer configurations on native instructions.

	Normalized response			Raw response		
Scoring layers	K	V	K + V	K	V	K + V
Cross 0	83.33	83.89	89.00	76.89	77.89	85.89
Cross 1	95.78	97.56	98.00	96.33	98.00	98.67
Cross 2	93.11	95.56	94.78	93.11	96.33	95.67
Cross 3	**99.56**	98.33	99.22	**99.56**	96.89	**99.56**
Cross 4	96.67	**99.11**	99.22	96.11	97.22	98.11
Cross 5	97.56	99.00	**99.44**	97.56	**98.89**	99.00
Cross 6	97.56	97.78	98.89	97.44	95.44	98.11
Cross 7	94.67	95.11	96.67	94.11	83.33	90.89
Early 4	98.89	97.56	98.56	99.00	98.11	99.33
Late 4	99.33	98.89	99.33	99.11	97.44	98.56
All 8	99.44	98.56	99.11	99.44	97.89	99.00

task with native instructions. Each bank includes the target expert and adds distractors in descending order of mean key-response score measured in a prior routing study. Thus, $K = 2$ includes the strongest measured distractor. Task identities are used to construct and evaluate the subbanks, not as routing inputs.

Table 18: PolicyWeave statistics as the available expert bank grows. Each bank size is evaluated on 250 native-instruction observations. Routing time reports p50 / p95 latency for a batch of five with the model and expert bank resident on the GPU.

Experts	Top-1 (%) ↑	Expert recall (%) ↑	Mean k ↓	Storage (10^6 bytes)	Route p50 / p95 (ms)
1	100.00	100.00	1.000	19.17	80.67 / 146.15
2	98.00	98.00	1.020	38.35	80.22 / 143.20
4	98.00	98.00	1.020	76.70	79.89 / 140.64
8	98.00	98.00	1.020	153.39	84.73 / 128.61
12	98.00	98.00	1.020	230.09	83.45 / 142.28
18	98.00	98.00	1.020	345.14	90.61 / 139.86

After the strongest distractor is introduced at $K = 2$, Top-1 accuracy and expert recall remain 98.00% through $K = 18$. Exact adapter storage grows linearly, while routing is performed once when the context-conditioned merged update is formed.

E Real-Robot Experiments

E.1 Setup and Training

Platform and tasks. The policy controls one AgileX Piper arm with six joints and a parallel gripper. Observations contain two RGB camera views, the instruction, and seven-dimensional joint-and-gripper state. Actions specify six absolute joint targets in radians and gripper opening in meters. Demonstrations are recorded at 30 Hz. The four tasks are *Upright bottle*, which places a bottle lying on the table upright; *Stack blocks*, which stacks two blocks; *Press button*, which activates a light using the black button; and *Place cup*, which places an empty cup on the white coaster.

Shared adaptation and task experts. Each task has 51 demonstrations. Starting from pretrained GR00T N1.5, we adapt the action model and its input-output projectors jointly on one demonstration per task for 3,000 steps, keeping the VLM frozen. This produces the Piper shared base. State and action normalization use pooled 1st and 99th percentiles from all 204 demonstrations. We then preserve this normalization and the adapted interface while training one rank-8, scale-16 LoRA expert on the remaining 50 demonstrations of each task for 10,000 steps. LoRA updates cover hidden action-transformer linear layers, excluding the input-output projectors. Table 19 summarizes the training settings.

Co-training. Both co-training baselines jointly use all 51 demonstrations per task and train for up to 40,000 steps. GR00T co-training uses the same adapted base, frozen action interface, LoRA configuration, and optimizer settings as the single-task experts; only the task mixture and maximum training steps change. For $\pi_{0.5}$, we use uniform task sampling, freeze the VLM, train rank-8, scale-16 LoRA updates in action attention and feed-forward layers, and update the action input-output projections, timestep embedding, and adaptive normalization. Both the GR00T and $\pi_{0.5}$ training pipelines use AdamW with weight decay 10^{-5}, 5% learning-rate warmup followed by cosine decay, and bfloat16 computation.

Table 19: Piper training configurations. Batch sizes are effective totals; the expert configuration applies independently to each of the four tasks.

Setting	Shared GR00T adaptation	POLICYWEAVE experts	GR00T co-training	$\pi_{0.5}$ co-training
Demonstrations	1 per task	50 per expert	51 per task	51 per task
Maximum steps	3,000	10,000	40,000	40,000
Batch size	128	256	256	256
Gradient accumulation	8	1	1	1
Peak learning rate	3×10^{-5}	10^{-4}	10^{-4}	10^{-4}
LoRA rank / scale	–	8 / 16	8 / 16	8 / 16
VLM	Frozen	Frozen	Frozen	Frozen

Inference and evaluation. Both model families predict 16-step action chunks. GR00T uses four flow integration steps and $\pi_{0.5}$ uses ten. POLICYWEAVE uses the same eight cross-attention key projections and stability-adaptive selection rule as in the simulation experiments, fixing the merged update for each episode. Each method is evaluated over 20 rollouts per task. We report the fraction of successful rollouts for each task and the unweighted mean across the four tasks.

E.2 REAL-ROBOT DEMONSTRATIONS

Figure 9 presents one successful POLICYWEAVE rollout for each Piper task. The sequences show bottle reorientation, block stacking, button activation, and cup placement from the initial scene to task completion. Full videos are included in the supplementary material.

F LIMITATIONS

Our experiments use GR00T N1.5, with real-world evaluation on a Piper arm; other VLA architectures and broader deployment settings remain to be studied. POLICYWEAVE merges experts derived from a common base policy with a shared embodiment and action interface. Extending it across embodiments or to tasks requiring additional action channels would require further interface alignment. The merged parameters remain fixed throughout each episode, so the method does not revise expert selection as new observations arrive. Tasks whose identity becomes clear only through interaction, or whose goals change during execution, would require a mechanism for updating the selection and merged policy online. Our real-world experiments evaluate POLICYWEAVE, while static merging is evaluated only in simulation. Extending static merging to physical robots calls for reliable coefficient selection and safety validation. Developing static merging methods better suited to VLA control and real-world deployment remains an important direction.

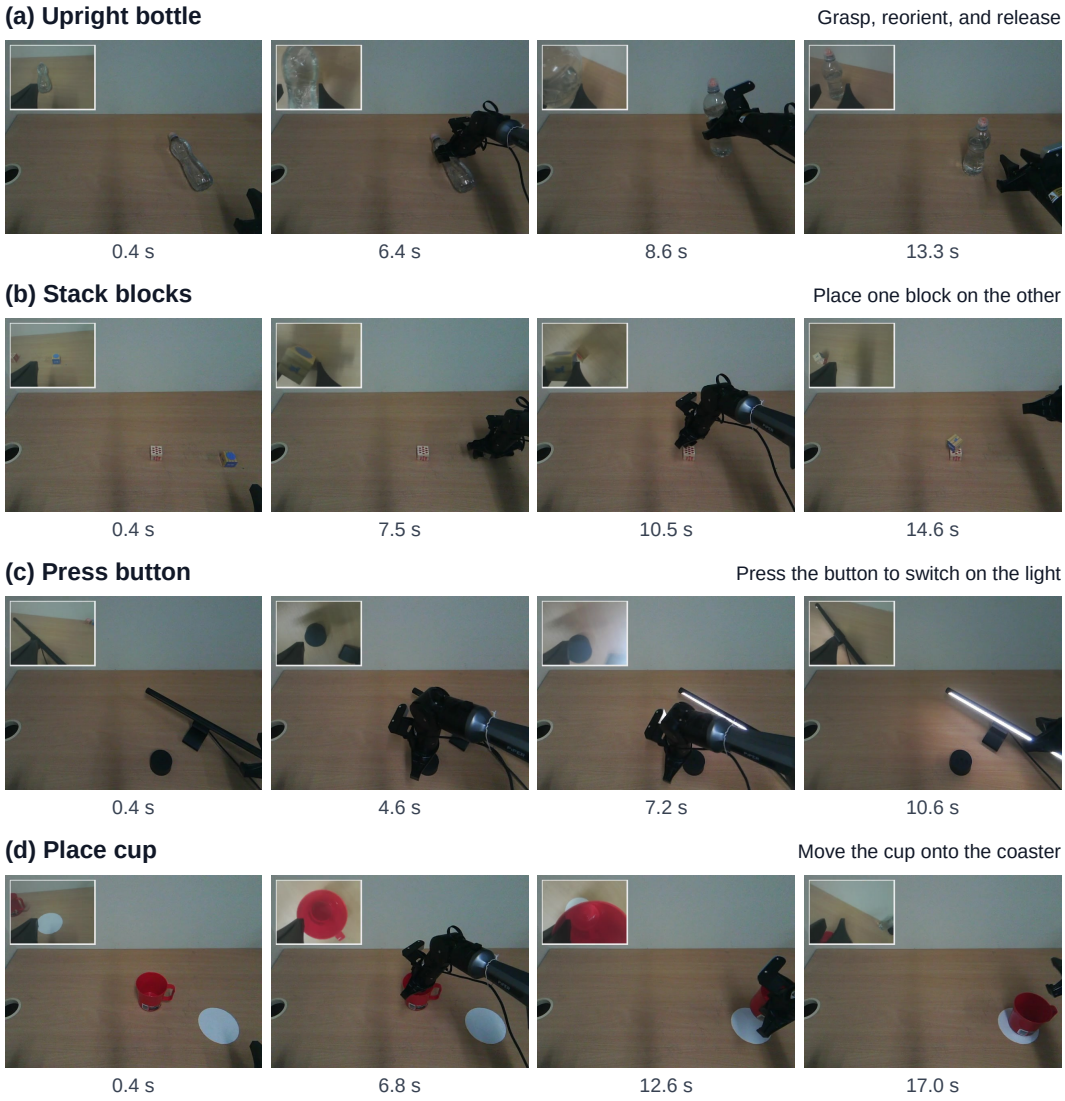


Figure 9: Successful POLICYWEAVE rollouts on the Piper arm. Each row follows one video from left to right; timestamps indicate elapsed video time. Insets show the wrist-camera view.