SHAREVLA: SHARE WHAT IS COMMON, SPECIALIZE WHAT IS NEEDED FOR MULTI-TASK VISION-LANGUAGE-ACTION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models adapt pretrained vision-language models to robot control and now solve individual manipulation tasks reliably, yet a single policy that executes many skills remains an open problem. Naive multi-task training suffers from interference among skills, and the prevailing alternative, merging independently trained experts, requires task-specific masks and must leave the deepest layers of the action expert unmerged. This fragility is traced to its source: independently trained action experts diverge from their very first block, so that no component of one expert transfers to another skill and every new skill requires a full training run. We therefore propose ShareVLA, which reverses the order of merging and training. A shared trunk covering 96% of the action expert is trained once on demonstrations from all skills and frozen. Lightweight expert heads are then trained on top of it, one per skill. The shared frozen trunk enables all expert heads to be assembled into a single model with zero merge error by construction, without backbone masks and with only a single backbone forward pass at inference. A lightweight router selects the head. On LIBERO, ShareVLA reaches 91.8% average success at the same parameter count as the strongest merging-based method while using 44% of its training compute, and exceeds multi-task training of the same architecture by 14.2 points. These results indicate that multi-skill VLA policies need not merge what was learned apart, and that deciding what to share before specializing is simpler and cheaper.

1 INTRODUCTION

Vision-Language-Action (VLA) models turn pretrained vision-language models into capable robot policies (Kim et al., 2024; Black et al., 2024; Wang et al.). On the LIBERO benchmark (Liu et al., 2023), a policy finetuned on a single task suite now succeeds more than 96% of the time. A deployed robot, however, must perform many skills with one policy and without being told which skill is required. Two routes lead to such a policy. Multi-task training optimizes one network on the union of all skill data and suffers from interference among skills (Yu et al., 2020). Model merging finetunes a separate expert per skill and fuses the experts' weights afterwards, without revisiting the training data (Ilharco et al., 2022; Yadav et al., 2023).

Merging is attractive in that it avoids interference, but VLA experts are particularly difficult to merge: averaging four LIBERO experts yields 0% success (Fu et al., 2026). MergeVLA (Fu et al., 2026), the strongest merge-based VLA to date, recovers 90.2% by gating the merged LoRA update with a binary mask per skill and by keeping the final block of the action expert unmerged, one copy per skill. This paper asks why merging is so brittle in this setting and finds that the cause is structural rather than algorithmic (section 2). Independently trained action experts become mutually uncorrelated from their first block onward (fig. 1). An expert therefore provides no benefit on any other skill (0.07% cross-task success), and each new skill requires a full expert trained from scratch. The merge-oriented architecture also fails to benefit from joint training: trained on all four suites at once, it reaches only 77.6%, substantially below the 90.2% obtained by merging. The benefit of merging thus rests entirely on the per-skill training pipeline, a pipeline that is expensive and, whenever the skill data is available, unnecessary. For practitioners who collect their own demonstrations, the data is available.

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.

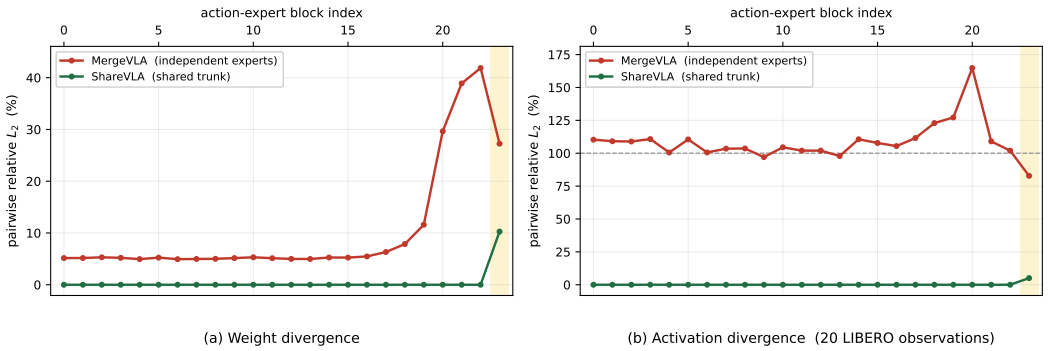


Figure 1: Where do experts diverge? Pairwise relative L_2 distance between the four LIBERO experts at each block of the action expert. (a) Weight space. The independently trained experts of MergeVLA diverge at every block and most strongly at block 22, not at the expert head ($l{=}L$) that MergeVLA leaves unmerged. The experts of ShareVLA are bit-wise identical up to block $L{-}1$ by construction. (b) Activation space, measured on 20 LIBERO observations. The experts of MergeVLA are already mutually uncorrelated ($>100\%$) at the first block, so swapping one expert for another corrupts the entire forward pass. ShareVLA confines all divergence to the final 4% of the action expert.

These observations reframe the question. What matters is not whether to combine experts but what to share and when to specialize. If the parameters that all skills must share are learned first, on all the data, and then frozen, the skill-specific parameters can be trained on an identical foundation. Combining the resulting experts is no longer an approximation: their shared parameters are the same tensors, so the merge error is zero by construction, and no masks are needed since no skill-specific backbone weights are ever created.

We instantiate this idea as ShareVLA, a share-then-specialize training scheme. Keeping the MergeVLA architecture unchanged, we split its action expert into a shared trunk, the first $L{-}1$ blocks holding 96% of the parameters, and a lightweight skill-specific expert head. Training proceeds in three stages. The trunk is trained once on mixed multi-skill data and frozen. One head per skill is then trained on top of the frozen trunk. Finally, the heads are assembled around a single copy of the trunk together with a lightweight router that infers the active skill from one backbone forward pass. Adding a skill thus costs a head (4% of an action expert) rather than a full expert, and a routing error degrades performance gracefully, as 96% of the computation is shared. Our contributions are as follows.

- We explain why merging VLA experts fails: independently trained action experts diverge from the first block, which makes specialization all-or-nothing, and the merge-oriented architecture loses 14 points when trained jointly.
- We propose ShareVLA, a share-then-specialize scheme that removes both the post-hoc merge and the task masks, guarantees zero merge error by construction, and reduces skill inference to a single backbone forward pass.
- On LIBERO, ShareVLA reaches 91.7% average success with the same 0.70B parameters as MergeVLA, surpassing its best merged variant (90.2%) with 44% of the training compute.

2 Why Merging Fails for VLA

This section presents two observations about merge-based VLA that motivate the design of ShareVLA. Both use the four LIBERO experts released by MergeVLA (Fu et al., 2026) and its evaluation protocol of 500 episodes per suite.

Experts fail to transfer across skills. MergeVLA reports that its Spatial expert scores 0% on the Object suite but provides no further breakdown. We evaluate all twelve off-diagonal $\langle$expert, suite$\rangle$ pairs, un-normalizing actions with the statistics of the expert under evaluation so that the failure

cannot be attributed to a normalization mismatch. Over $6{,}000$ episodes the mean success rate is 0.07%, and eleven of the twelve pairs are exactly 0.0%. Figure 1 shows why. The experts differ at every block in weight space, and in activation space, measured on real observations, they are mutually uncorrelated ($>100\%$ relative L_2 distance) already at the first of 24 blocks. Replacing one expert with another therefore corrupts the whole forward pass. Specialization is all-or-nothing: no part of an expert can be reused for another skill, and each new skill costs a full finetuning run. Notably, weight divergence peaks at block 22, not at the final block that MergeVLA keeps unmerged, so the unmerged head is not located where the experts differ most.

The architecture fails to benefit from joint training. Merging is needed only when experts must be trained apart. To test whether the merge-oriented architecture helps on its own, we train it on the union of the four suites with the hyperparameters of MergeVLA. Average success falls from 90.2% to 77.6%, and on the Long suite, which composes several sub-skills, from 79.4% to 36.8%. The advantage of merging therefore comes from the per-skill pipeline, not from the architecture. That pipeline trains M full experts, adds skill masks and an M-way routing forward pass, and precludes joint training, all to avoid a training run that could be performed directly whenever the data is available.

Implication. These observations delimit the regime in which merging is the right tool: M experts that already exist and cannot be retrained. Whenever the skill data is available, the same architecture can be filled in differently, by deciding which parameters to share and in what order to train them. The next section develops this alternative.

3 Related Work

Vision-Language-Action models. VLA models finetune pretrained vision-language models into robot policies (Kim et al., 2024; Black et al., 2024). Dual-system designs such as π_0 (Black et al., 2024) and VLA-Adapter (Wang et al.) attach a separate action expert to the backbone and reach strong performance at small scale. However, their multi-task ability comes from joint training over all skills with the task identity given, which is the setting considered in this work.

Model merging. Merging combines independently finetuned models without access to their training data, by adding task vectors to a common initialization (Wortsman et al., 2022; Ilharco et al., 2022) and resolving the conflicts among them (Yadav et al., 2023; Cheng et al., 2025). Merged models have also served as initializations for further training (Choshen et al., 2022; Sukhbaatar et al., 2024). For VLA, MergeVLA (Fu et al., 2026) shows that naive merging yields 0% success and compensates with skill-specific masks, an architecture without self-attention and one unmerged expert head per skill. ShareVLA merges only the backbone, before any skill-specific training, and leaves the skill-specific components unmerged.

Multi-task policy learning. A shared backbone with task-specific heads is the classical multi-task recipe (Caruana, 1997; Ruder, 2017). Its main failure mode is gradient interference (Yu et al., 2020; Liu et al., 2021), which recent VLA work counters with mixture-of-experts routing (Xu et al., 2025; Zhang et al., 2026a; Du et al., 2025; Zhang et al., 2026b). ShareVLA demonstrates that learning shared parameters before skill-specific specialization eliminates the need for post-hoc merging and task-specific masks, while maintaining the same parameter count and requiring only 44% of the training compute.

4 ShareVLA

4.1 Problem formulation

Given M manipulation skills with demonstration datasets $\{\mathcal{D}_m\}_{m=1}^{M}$, the goal is to learn a single policy that executes all of them without being told which skill is active. The model-merging formulation (Fu et al., 2026) assumes M pre-trained experts and forbids access to their training data. In contrast, every $\mathcal{D}_m$ is assumed to be available for joint training, which is the situation of any practitioner who collects their own demonstrations. Two design questions then arise: which

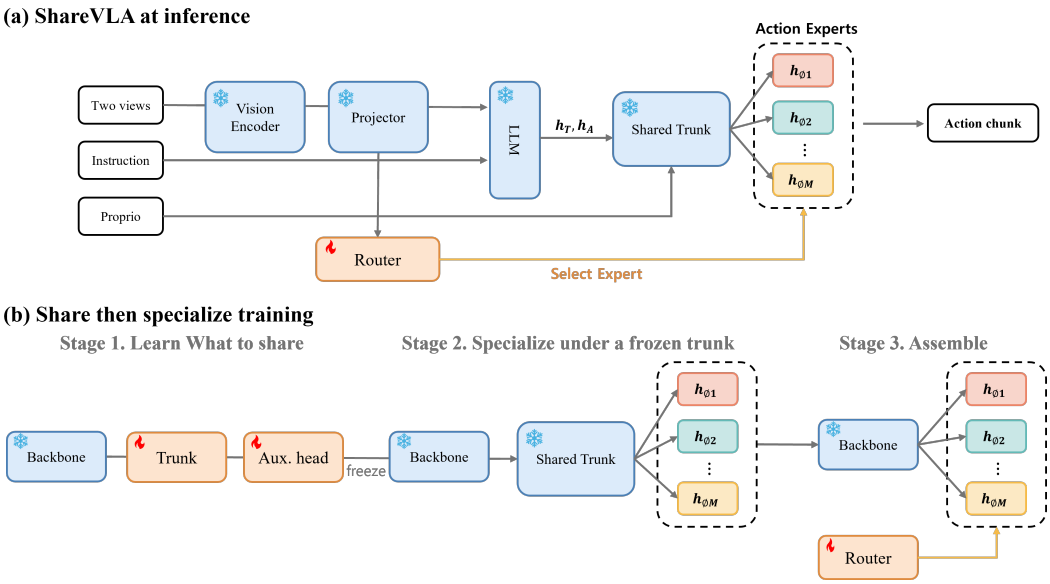


Figure 2: Overview of ShareVLA. (a) At inference, a frozen, TIES-merged backbone encodes two views and the instruction in a single forward pass. The action expert is split into a shared trunk g_θ (blocks $1, \ldots, L{-}1$) and M compact expert heads h_{ϕ_m}. A lightweight router reads the visual tokens and selects one head. (b) Training reverses the order of merging and specialization. Stage 1 trains the trunk once on all skills and freezes it. Stage 2 trains one head per skill from the same frozen trunk. Stage 3 assembles the heads around a single copy of the trunk. Since $\theta^{(1)}{=}\cdots{=}\theta^{(M)}{=}\theta^\star$ by construction, the merge error is zero.

parameters should be shared across skills, and in what order should the shared and skill-specific parameters be learned.

4.2 Architecture

ShareVLA adopts the dual-system architecture of MergeVLA (Fu et al., 2026), in which a frozen vision-language backbone feeds hidden states to a separate action expert (fig. 2a). Two 224×224 camera views and a language instruction are encoded by a frozen DINOv2/SigLIP vision encoder and a Qwen2.5-0.5B language model (Hui et al., 2024), with 64 learnable action queries appended at the action positions. The language model yields task tokens $\mathbf{h}_\mathrm{T}$ (vision patches) and action tokens $\mathbf{h}_\mathrm{A}$, which condition the action expert. The action expert is an MLP-ResNet of $L = 24$ blocks. Each block attends to $\mathbf{h}_\mathrm{T}$ and $\mathbf{h}_\mathrm{A}$ through gated cross-attention and has no self-attention, exactly as in Fu et al. (2026). This architecture is left untouched so that the comparison isolates the training procedure.

Shared trunk and expert heads. The action expert is decomposed into a trunk g_θ shared by all skills and a skill-specific expert head h_{ϕ_m}:

$$\hat{\mathbf{a}}_m \;=\; h_{\phi_m}\big(g_\theta(\mathbf{x})\big), \qquad m = 1, \ldots, M, \tag{1}$$

where $\mathbf{x}$ denotes the observation (camera views, instruction and proprioception), whose backbone encoding $(\mathbf{h}_\mathrm{T}, \mathbf{h}_\mathrm{A})$ is absorbed into g_θ for brevity, and $\hat{\mathbf{a}}_m \in \mathbb{R}^{H\times 7}$ is the predicted chunk of $H = 8$ actions. The trunk g_θ comprises the input projection and blocks $1, \ldots, L{-}1$, and the head h_{ϕ_m} comprises block L, the output layer normalization and the output projection. The small proprioception projector (0.8M parameters) is treated as part of the head. With $L = 24$ and hidden dimension $d = 896$, the trunk holds $|\theta| = 153{,}557{,}399$ parameters (95.98% of the action expert) and each head $|\phi_m| = 6{,}439{,}560$ (4.02%). A new skill therefore adds 4.02% of an action expert rather than a full one.

4.3 THREE-STAGE TRAINING

The method rests on the order of training (fig. 2b): shared parameters are learned first, on all data, and skill-specific parameters afterwards, with the shared parameters frozen.

Stage 1: Learning the shared trunk. We first fix the backbone by merging the finetuned language models and projectors of the M single-task experts with TIES (Yadav et al., 2023) (top-20% trimming, sign election, disjoint mean, scaling factor $\alpha = 1$). The vision encoder is frozen in every expert and is shared as is. The merged backbone stays frozen for the rest of training. On top of it, a single action expert is trained on the union $\bigcup_m \mathcal{D}_m$ with the ℓ_1 action-chunk loss,

$$\theta^\star,\, \psi^\star \;=\; \arg\min_{\theta,\psi}\; \mathbb{E}_{(\mathbf{x},\mathbf{a})\sim\bigcup_m \mathcal{D}_m} \big\| h_\psi(g_\theta(\mathbf{x})) - \mathbf{a} \big\|_1, \tag{2}$$

where ψ parameterizes an auxiliary head of the same shape as the expert heads. Only the trunk $\theta^\star$ is retained, and $\psi^\star$ serves only as the common initialization of the expert heads in Stage 2. Mixing the data here is deliberate: the trunk is the one component that must serve every skill, so it is the one component trained on every skill.

Stage 2: Learning the expert heads. Every expert head is initialized from $\psi^\star$ and trained on its own skill with the trunk frozen at $\theta^\star$,

$$\phi_m^\star \;=\; \arg\min_{\phi}\; \mathbb{E}_{(\mathbf{x},\mathbf{a})\sim\mathcal{D}_m} \big\| h_\phi(g_{\theta^\star}(\mathbf{x})) - \mathbf{a} \big\|_1, \qquad m = 1, \ldots, M. \tag{3}$$

As θ receives no gradient, the trunk carried by every expert remains bit-wise identical to $\theta^\star$.

Stage 3: Assembly with zero merge error. The M heads are finally assembled into one action expert that keeps a single copy of $\theta^\star$ and switches among $\{\phi_m^\star\}$. Stage 2 guarantees

$$\theta^{(1)} = \theta^{(2)} = \cdots = \theta^{(M)} = \theta^\star, \tag{4}$$

where $\theta^{(m)}$ denotes the trunk carried by the m-th expert after Stage 2, so assembly is a copy rather than an approximation: the merge error is zero by construction. Rather than assuming eq. (4), we verify it element-wise before assembly and abort on any violation. On LIBERO the check passes for all 441 trunk tensors and 153.5M parameters. Consequently, the unmerged-expert-head problem that merge-based methods cannot resolve (Fu et al., 2026) never arises, and task masks are unnecessary since no skill-specific backbone weights are ever created. A further consequence is graceful degradation under routing errors: with 96% of the computation shared, applying the wrong head degrades performance instead of destroying it, unlike the 0.07% cross-task success of independently trained experts (section 2). This effect is quantified in section 5.4.

4.4 SKILL INFERENCE AT TEST TIME

The active skill is unknown at test time, so a router must select ϕ_m. ShareVLA has no task masks and hence no masked backbone variants to score, so routing costs a single backbone forward pass rather than M.

We attach a lightweight router (268K parameters) to the projector output $\mathbf{V} \in \mathbb{R}^{N\times d}$, i.e., the $N = 512$ visual tokens before the language model, whose i-th row is $\mathbf{v}_i$. The router holds M learnable skill prototypes $\mathbf{P} \in \mathbb{R}^{M\times d}$ with rows $\mathbf{p}_m$ and scores each visual token by its cosine similarity to the closest prototype,

$$s_i \;=\; \max_m\; \frac{\mathbf{v}_i^\top \mathbf{p}_m}{\|\mathbf{v}_i\|\,\|\mathbf{p}_m\|}, \qquad i = 1, \ldots, N. \tag{5}$$

The top-ρN tokens ($\rho = 0.3$) are kept and pooled with weights $\mathrm{softmax}(\mathbf{s})$, $\mathbf{s} = (s_1, \ldots, s_N)$ restricted to the kept tokens, and a three-layer MLP maps the pooled descriptor to a distribution over the M skills. Tokens that match no prototype, such as background and table surface, receive near-zero weight and are discarded, so the decision rests on skill-discriminative regions. The router is trained for 10k steps on the same data with skill labels while the backbone and action expert stay frozen.

Routing schedule. The router can be evaluated at every timestep, but on LIBERO it never switches heads within an episode: a single decision at $t = 0$, as in Fu et al. (2026), yields exactly the same success counts on every suite as per-timestep routing (section 5.4). This is expected, since each LIBERO episode instantiates one task from one suite, so the quantity the router infers is constant within an episode. Temporal routing is therefore not claimed as a contribution, and results are reported under the t=0 protocol of Fu et al. (2026).

4.5 Training cost

Training takes 30k steps for the trunk, 12k per expert head and 10k for the router, i.e., 88k gradient steps for $M = 4$, against $4 \times 50\text{k} = 200\text{k}$ for merge-based training, or 44% of the budget. This count covers the steps ShareVLA performs itself. The TIES-merged backbone of Stage 1 reuses the publicly released single-task backbones and adds no gradient steps. The assembled policy matches the merged baseline in size (0.70B, excluding the vision encoder shared by all methods) while needing neither M task masks nor M backbone forward passes for routing.

5 Experiments

5.1 Experimental setup

Benchmarks. Experiments are conducted on LIBERO (Liu et al., 2023), whose four task suites (Spatial, Object, Goal and Long) contain ten tasks each. Following Fu et al. (2026), the success rate is reported over 50 trials per task, i.e., 500 episodes per suite and 2,000 in total. Robustness under distribution shift is assessed on LIBERO-Plus (Fei et al., 2025), which perturbs the same tasks along seven axes (background textures, camera viewpoints, language instructions, lighting, object layout, robot initial states and sensor noise, 10,030 episodes in total). All models are trained on LIBERO only and evaluated on LIBERO-Plus zero-shot, as in Fu et al. (2026). Following the same protocol, each perturbation category is reported as the success rate averaged over the four task suites, and the overall score averages the suite-level success rates.

Baselines. All baselines share the backbone of ShareVLA (Qwen2.5-0.5B with a frozen DINOv2/SigLIP encoder), inputs and evaluation protocol, and fall into three groups. (i) Single-task experts. MergeVLA$_{\text{FT}}$ denotes the four per-suite experts of Fu et al. (2026). They form an upper bound but require four models and the skill identity at test time. (ii) Model merging. MergeVLA merged with Task Arithmetic (TA) (Ilharco et al., 2022), TIES (Yadav et al., 2023), WUDI (Cheng et al., 2025), KnOTS (Stoica et al., 2025), TSV (Gargiulo et al., 2025) and EMR (Huang et al., 2024), as reported in Fu et al. (2026). (iii) Multi-task training. The MergeVLA architecture trained jointly on the union of the four suites with its own hyperparameters (50k steps, LoRA rank 64, effective batch size 16), and the VLA-Adapter architecture (Wang et al.) trained identically. The two differ only in the action-expert block (VLA-Adapter keeps self-attention and tanh gating), which isolates the effect of the merge-oriented architectural changes of MergeVLA in the multi-task regime.

Implementation details. Stage 1 trains the full action expert on the union of the four suites for 30k steps (learning rate 2×10^{-4}, batch size 16). Stage 2 trains one expert head per suite for 12k steps (learning rate 5×10^{-5}, batch size 16) with the trunk and backbone frozen, and the router is trained for 10k steps. Each run uses one NVIDIA RTX 4090. Following Fu et al. (2026), parameter counts exclude the frozen vision encoder (730.9M) shared by all methods. Unless stated otherwise, ShareVLA results come from a single run.

5.2 Main results on LIBERO

Table 1 summarizes the results. ShareVLA attains 91.7% average success, the highest of all single-model policies, every method except the four separate single-task experts. It improves on the strongest merge-based method, MergeVLA$_{\text{TIES}}$ (90.2%), by 1.5 points at the same parameter count and with 44% of the training compute, leads on three of the four suites, and on the Object suite (99.8%) even exceeds the single-task expert (98.6%).

Table 1: Success rate (%) on LIBERO. All methods share the same backbone and evaluation protocol. Steps is the total number of gradient steps required to obtain the multi-skill policy. Params exclude the frozen vision encoder shared by all methods. †Results reported in Fu et al. (2026). (arch.) denotes an architecture trained jointly on all suites in this work. ShareVLA is evaluated as a single assembled checkpoint that stores one trunk, four heads and four proprioception projectors. Best result among single-model policies in **bold**.

Method	Setting	Steps	Params	Spatial	Object	Goal	Long	Avg.
Single-task experts (four models, skill identity given)								
MergeVLA$^{\dagger}_{\text{FT}}$	single-task	200k	2.72B	98.0	98.6	95.0	95.0	96.7
Model merging								
MergeVLA$^{\dagger}_{\text{EMR}}$	merge	200k	0.70B	96.0	63.2	62.0	40.6	65.5
MergeVLA$^{\dagger}_{\text{TSV}}$	merge	200k	0.70B	99.4	97.8	74.4	54.8	81.6
MergeVLA$^{\dagger}_{\text{KnOTS}}$	merge	200k	0.70B	96.8	98.8	84.8	71.4	88.0
MergeVLA$^{\dagger}_{\text{TA}}$	merge	200k	0.70B	98.0	98.8	85.4	76.6	89.7
MergeVLA$^{\dagger}_{\text{WUDI}}$	merge	200k	0.70B	97.6	98.2	85.6	78.2	89.9
MergeVLA$^{\dagger}_{\text{TIES}}$	merge	200k	0.70B	94.8	94.6	91.8	79.4	90.2
Multi-task training								
MergeVLA (arch.)	multi-task	50k	0.68B	93.8	93.0	86.8	36.8	77.6
VLA-Adapter (arch.)	multi-task	50k	0.62B	95.0	96.6	88.0	47.8	81.9
ShareVLA (ours)	share-then-specialize	88k	0.70B	**98.8**	**99.8**	**92.6**	**75.6**	**91.7**

Comparison with multi-task training. The same MergeVLA architecture trained jointly on all four suites reaches 77.6%, 14.1 points below ShareVLA, and collapses on the Long suite (36.8%). The VLA-Adapter architecture trained under identical settings performs better (81.9%): the changes MergeVLA makes to enable merging, removing self-attention and switching to sigmoid gating, cost 4.3 points once merging is no longer needed. ShareVLA uses this same merge-oriented architecture and still recovers 91.7%, so the difference is attributable to the training procedure alone. Separating the shared trunk from the skill-specific heads, and training them in that order, avoids the interference that undermines naive joint training.

Training and inference cost. The Steps column of table 1 summarizes training cost: ShareVLA needs 88k gradient steps, against 200k for the four experts that every merge-based method must first train (section 4.5). At inference, the per-step policy latency of the two models is indistinguishable: measured on real LIBERO observations on one RTX 4090 (100 steps after warm-up), ShareVLA takes 49.4 ms per step (median) against 48.1 ms for MergeVLA, with the router accounting for 0.3 ms. The difference lies in routing. MergeVLA runs $M = 4$ additional backbone forward passes at the start of every episode to select the mask and head (199 ms when only the masks are swapped), whereas the router of ShareVLA reuses the visual tokens of the current forward pass and adds no backbone forward. The advantage is therefore structural rather than a speed-up: no start-up delay per episode and no re-routing cost should the required skill change.

5.3 Robustness under distribution shift

Table 2 reports LIBERO-Plus results, on which ShareVLA underperforms the baselines: 43.0% against 53.7% for multi-task training of the same architecture and 62.5% for MergeVLA$_{\text{TIES}}$. The per-category breakdown localizes the gap. On background textures and language instructions, ShareVLA is within one point of the multi-task baseline, and on object layout within five. On camera viewpoints (10.4% vs. 32.3%), sensor noise (32.2% vs. 54.4%), lighting (63.0% vs. 73.9%) and robot initial states (23.9% vs. 33.4%) it degrades sharply. These perturbations are precisely the ones that alter the low-level statistics of the visual input, and the gap to the merged model is widest on the same categories (camera viewpoints 10.4% vs. 50.7%, sensor noise 32.2% vs. 66.0%).

The pattern has a direct explanation. ShareVLA trains no backbone parameters: the vision encoder and language model stay frozen throughout, and the expert heads occupy only 4% of the action expert. Every baseline in table 2 adapts the backbone with LoRA and thereby acquires some invariance to visual perturbations during training. ShareVLA has no parameters through which such invariance

Table 2: Success rate (%) on LIBERO-Plus. All models are trained on LIBERO and evaluated zero-shot under seven perturbation categories. Following Fu et al. (2026), each entry is averaged over the four task suites and Avg. averages the suite-level success rates. †Results reported in Fu et al. (2026). All baselines adapt the backbone with LoRA, whereas ShareVLA keeps the backbone frozen. Best result among single-model policies in **bold**.

Method	BG	Camera	Lang.	Light	Layout	Init.	Noise	Avg.
MergeVLA$^{\dagger}_{\text{FT}}$ (single-task)	92.7	62.4	75.7	92.7	73.7	46.4	74.7	72.4
MergeVLA$^{\dagger}_{\text{TSV}}$ (merge)	64.3	44.8	58.8	72.9	59.4	30.4	52.7	53.5
MergeVLA$^{\dagger}_{\text{TA}}$ (merge)	78.2	**53.1**	**68.4**	79.0	65.0	**34.6**	62.7	61.6
MergeVLA$^{\dagger}_{\text{TIES}}$ (merge)	**85.7**	50.7	66.0	**84.2**	**68.1**	30.3	**66.0**	**62.5**
MergeVLA (arch., multi-task)	75.5	32.3	56.1	73.9	62.9	33.4	54.4	53.7
ShareVLA (ours)	75.4	10.4	55.6	63.0	58.1	23.9	32.2	43.0

Table 3: Ablations on LIBERO (success rate, %). Rows in gray denote the default ShareVLA configuration. Independent experts trains a full action expert per suite on a frozen backbone, either the original pretrained one (base) or the TIES-merged one used by ShareVLA (merged). Merged heads averages the four expert heads into one and removes the router. Oracle routes with the ground-truth suite, and Single head applies the Spatial head to every suite. All rows use the multi-expert evaluation path.

	Variant	Params	Spatial	Object	Goal	Long	Avg.
Sharing	Independent experts, base backbone	1.16B	78.0	93.2	76.8	15.0	65.8
	Independent experts, merged backbone	1.16B	95.2	90.8	89.6	**81.6**	89.3
	Shared trunk (ShareVLA)	0.70B	98.8	99.8	92.6	75.8	**91.8**
Steps	Head training 6k steps	0.70B	98.0	97.0	**93.8**	74.8	90.9
	Head training 12k steps	0.70B	98.8	99.8	92.6	75.8	**91.8**
Head	Head = block L (4.0%)	0.70B	98.8	99.8	92.6	75.8	**91.8**
	Head = blocks $L-1, L$ (8.0%)	0.72B	97.2	99.8	93.4	70.2	90.2
Routing	Learned router	0.70B	98.8	99.8	92.6	75.8	**91.8**
	Oracle routing	0.70B	98.8	99.8	92.6	75.4	91.7
	Merged heads, no router	0.68B	96.6	99.4	91.0	69.6	89.2
	Single head (Spatial)	0.68B	98.8	95.4	83.6	53.6	82.9

could be learned. Training budget is not the cause: doubling the head training steps from 6k to 12k raises the overall LIBERO-Plus score by only 1.0 point and leaves camera viewpoints unchanged (10.9% → 10.4%), while improving in-distribution LIBERO by 0.9 points. Freezing the backbone is what makes ShareVLA parameter-efficient and merge-free, and it is also what limits its robustness to visual shift. We regard this as the main trade-off of the current design. Adapting the backbone during Stage 1, e.g., with low-rank adapters, is a natural extension that leaves the share-then-specialize recipe intact.

5.4 Ablation studies

Sharing the trunk. Training four independent action experts on the same frozen backbone, i.e., removing the shared trunk while leaving everything else unchanged, yields 89.3% with 1.16B parameters. Sharing the trunk raises this to 91.8% with 0.70B parameters: 2.5 points better with 40% fewer parameters. The shared trunk therefore does more than save parameters. Training it on all skills jointly yields a better representation than training it on each skill alone. The exception is the Long suite, where independent experts lead by 5.8 points, as discussed below. The backbone matters as well: with the original pretrained backbone in place of the merged one, the same independent-expert pipeline reaches only 65.8%, so a backbone adapted to the skill distribution is a prerequisite for training the action expert on top of a frozen one.

Head training budget and capacity. Two ablations test whether the Long deficit can be closed on the head side. Doubling the head training steps from 6k to 12k raises the average from 90.9% to 91.8%, but the Long suite gains only 1.0 point. Enlarging the expert head from one block (4.0% of the action expert) to two (8.0%) is likewise ineffective: the average drops by 1.6 points and the Long suite by 5.6. With twice the skill-specific parameters and the same per-suite data, the larger head overfits, and the Long suite, which has the fewest demonstrations per unit of horizon, suffers most. Neither more training nor more capacity in the heads helps, which identifies the Long deficit as structural: the long-horizon skill would need to adapt the trunk itself, which the shared design precludes. The single-block head trained for 12k steps is therefore retained.

Routing. The learned router (268K parameters) reproduces oracle routing on three suites with identical success counts and exceeds it by two episodes on the Long suite (91.8% vs. 91.7%). Routing is not a bottleneck. Two further variants are informative. First, averaging the four expert heads into one and removing the router still yields 89.2%, only 2.6 points below ShareVLA and close to MergeVLA$_{\text{TIES}}$ (90.2%), with no masks, no router and a single dense model. This is possible only since the heads share a trunk: their weights differ by 10% and their activations by 5%, whereas the heads of MergeVLA differ so much that they must be kept apart. Second, applying the Spatial head to every suite still yields 82.9% on average and 53.6% on the Long suite, whereas the same cross-task evaluation of the experts of MergeVLA yields 0.07% (section 2). Misrouting degrades ShareVLA gracefully, as 96% of the computation is shared.

Routing schedule. Evaluating the router at every timestep produces exactly the same success counts as a single decision at $t = 0$ on every suite. The router never switches heads within an episode (zero switches over all routing calls, argmax probability 1.0 after warm-up), as expected on LIBERO, where each episode instantiates one task from one suite. The comparison with MergeVLA is therefore not confounded by the routing protocol.

6 Conclusion

This work has shown that merging independently trained VLA experts is fragile for a structural reason: the experts diverge from the first block of the action expert, so that no component transfers across skills and every new skill requires a full training run. ShareVLA reverses the order of merging and training. A shared trunk is learned once on all skills and frozen, compact expert heads are specialized on top of it, and the experts are assembled with zero merge error, as their shared parameters are identical by construction. The result needs neither task masks nor a post-hoc merge and infers the skill with a single backbone forward pass. On LIBERO, ShareVLA reaches 91.7% at the same parameter count as MergeVLA with 44% of its training compute, and its heads remain mergeable into a single dense model (89.2%). These results suggest that deciding what to share before specializing is preferable to merging what was learned apart.

Limitations. The evaluation is limited to simulation, a single embodiment and a single backbone scale (0.5B). ShareVLA also assumes a fixed, labeled set of skills during training, and adding a skill requires a new head and a router update, both of which are lightweight.

AI use statement

In this work, we used generative AI tools to assist with language editing, sentence-level rewriting, improving the clarity of the manuscript, and identifying relevant literature. All technical content, including the proposed method, mathematical formulations, experimental design, results, and AI-assisted code, was carefully reviewed and verified. We take full responsibility for the final content of this work.

References

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al.

\

pi_0 : $A vision-language-action flow model for general robot control. arXiv$ *preprint arXiv:2410.24164*, 2024.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. *arXiv preprint arXiv:2503.08099*, 2025.

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*, 2022.

Zhiying Du, Bei Liu, Yaobo Liang, Yichao Shen, Haidong Cao, Xiangyu Zheng, Zhiyuan Feng, Zuxuan Wu, Jiaolong Yang, and Yu-Gang Jiang. Himoe-vla: Hierarchical mixture-of-experts for generalist vision-language-action policies. *arXiv preprint arXiv:2512.05693*, 2025.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

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 Rodola. Task singular vectors: Reducing task interference in model merging. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18695–18705. IEEE, 2025.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. *Advances in Neural Information Processing Systems*, 37: 122741–122769, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model, 2024. *URL https://arxiv. org/abs/2406.09246*, 1(2):4, 2024.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34:18878–18890, 2021.

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

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

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*, volume 2025, pp. 4501–4519, 2025.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, et al. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *arXiv preprint arXiv:2403.07816*, 2024.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model, 2025. *URL https://arxiv. org/abs/2509.09372*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Lu Xu, Jiaqian Yu, Xiongfeng Peng, Yiwei Chen, Weiming Li, Jaewook Yoo, Sunghyun Chunag, Dongwook Lee, Daehyun Ji, and Chao Zhang. Mose: skill-by-skill mixture-of-experts learning for embodied autonomous machines. *arXiv preprint arXiv:2507.07818*, 2025.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Haozhe Zhang, Sixian Li, Yifei Zhang, Zezheng Huai, Hao Chen, Chunhua Shen, Jingjing Gong, and Xipeng Qiu. Core-vla: Towards scalable and robust vision-language-action modeling via conditional routing of experts. *arXiv preprint arXiv:2607.03693*, 2026a.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18764–18772, 2026b.