VLA-FABRIC: LEARNING INTERACTION FOR COMPOSABLE MULTI-ARM VLA AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models are rapidly expanding what individual robots can accomplish, yet many real-world manipulation settings require multiple arms to operate as a coordinated system. Existing approaches organize multi-arm coordination in different ways, including centralized joint policies, high-level task or role assignment, and locally executed policies that react to teammates. However, how complete action-producing VLAs should explicitly interact with one another during action generation remains largely unexplored. We introduce VLA-Fabric, which organizes one complete VLA per arm while preserving local perception and action ownership, and enables coordination through learned interaction across policy boundaries. We first systematically study where such interaction should occur, how peer information should be aggregated, and how the interaction should be learned, revealing complementary coordination functions spanning shared context, peer-specific latent information, and action-stage interaction. We then re-instantiate these functions in a structurally different VLA architecture, showing that the coordination principle is not tied to a particular tensor layout or backbone implementation. Finally, we compose teams of three and four arms and evaluate them in simulation and physical experiments. On all four physical tasks, Full interaction improves success over independently trained local policies by 16–40 percentage points. Together, these results suggest that explicit interaction among complete VLA policies offers a scalable path toward networked embodied intelligence.

1 INTRODUCTION

Vision-language-action (VLA) models are advancing the capabilities of individual robots by connecting visual and language understanding with learned action generation (Brohan et al., 2023; Kim et al., 2024; Physical Intelligence et al., 2025). Many important robotic applications, however, require several manipulators to contribute to a common physical objective. Cooperative manufacturing, multi-part assembly, shared-load handling, and flexible robotic workcells motivate systems in which multiple arms work together (Dio et al., 2024; Huang et al., 2025; Tian et al., 2025). As pretrained VLAs become more capable, their progress creates an opportunity beyond improving a single robot: organizing multiple action-producing policies into a collaborative system. The ability of each constituent model to perceive and act does not, by itself, specify how those models should work together.

Recent multi-arm VLAs pursue this goal through different organizations. RDT-1B, ET-VLA, Twin-VLA, and Dexora place cross-arm dependencies within a joint policy (Liu et al., 2025; Li & Peng, 2025; Im et al., 2026; Zhang et al., 2026b). Other approaches coordinate at different levels: MA-VLA assigns atomic behaviors to individual arms (Zhang et al., 2026a), while CHORUS demonstrates implicit decentralized coordination through locally executed policies that observe and react to teammates without runtime messages (Doshi et al., 2026). These approaches establish effective collaboration, with coordination inside a joint policy, above the action policy through behavior assignment, or indirectly through observations of teammates. This raises an organizational challenge as the number and roles of manipulators grow. How can a multi-arm system remain modular at the level of individual manipulators while still supporting tightly coupled collaboration?

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.

A natural way to obtain such modularity is to organize the system around complete arm-local policies rather than a growing joint policy. We therefore treat each manipulator and its complete VLA as a self-contained execution unit that perceives its local context and produces its own actions. Local information can remain with the policy that uses it, and larger teams can be formed by composing additional execution units. Crucially, *local does not mean isolated*. Policies participating in the same physical task still need to adapt their actions to one another. The key challenge is to connect these local policies through explicit interaction without giving up their local action ownership.

To realize this local-but-connected organization, we introduce VLA-Fabric[1], a multi-VLA coordination architecture connecting complete arm-local policies through learned internal representation interaction (Figure 1). Each VLA retains local perception and action generation while selected peer representations condition its computation through explicit policy interaction. VLA-Fabric thus makes coordination among complete policies an explicit learning and architecture problem instead of absorbing all cross-arm dependencies into a joint policy.

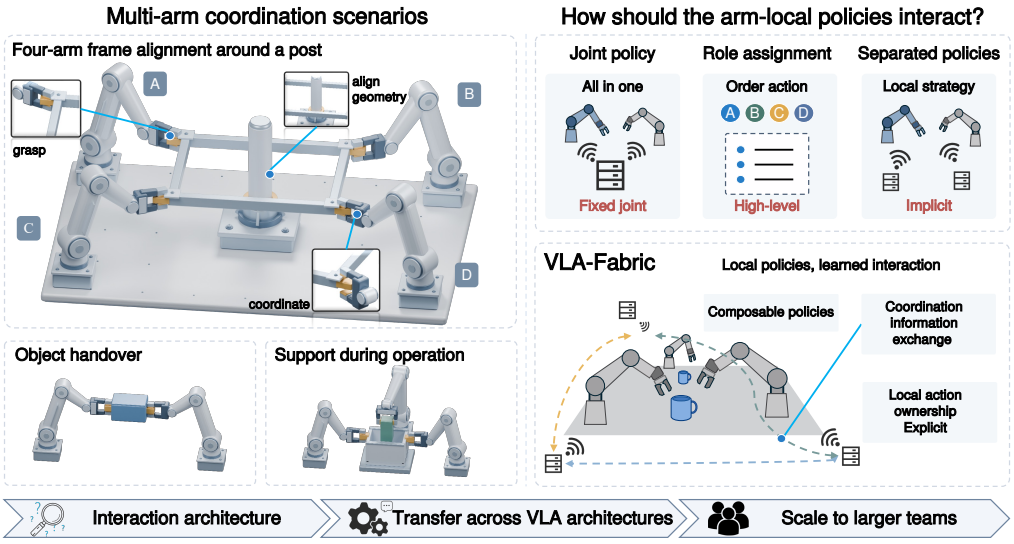


Figure 1: From multi-arm coordination to VLA-Fabric. Complete arm-local VLAs collaborate through learned interaction; we study interaction discovery, cross-architecture transfer, and larger-team scaling.

First, we establish an effective VLA-Fabric interaction architecture through a broad controlled study. Starting from a decomposable bimanual VLA with pretrained single-arm streams (Im et al., 2026), we separate its original cross-arm dependencies. We systematically explore a combinatorial design space covering interaction entry points, aggregation operators, and fine-tuning strategies. The study shows that exposing more peer information does not by itself ensure coordination; interaction structure and adaptation determine how effectively local policies use one another's information. This exploration identifies complementary interaction at shared-context, peer-specific intermediate, and action-generation stages.

Second, we demonstrate that the same VLA-Fabric interaction paradigm can be re-instantiated in substantially different VLA architectures. We use $\pi_{0.5}$ (Physical Intelligence et al., 2025) as a second realization, whose multimodal processing and action generation differ markedly from the Eagle2 VLM-based SingleVLA used in our initial study. Accordingly, the same coordination functions are realized at architecture-appropriate computational boundaries while preserving complete local policies with explicit policy interaction. The resulting $\pi_{0.5}$-Fabric system enables multiple complete $\pi_{0.5}$ policies to interact and achieves strong performance across several bimanual tasks. Its stronger results suggest that advances in the constituent VLA may translate into stronger collaborative systems. This observation is consistent with the value of pretrained single-arm capabilities demonstrated by TwinVLA (Im et al., 2026).

[1]Project website and code: `https://anonymous-0525.github.io/VLA-Fabric/`.

Third, we establish the extensibility of VLA-Fabric from bimanual coordination to larger teams of complete VLA policies. Additional manipulators are incorporated by composing additional local policies and extending their interaction. We instantiate three- and four-arm VLA-Fabric teams and evaluate them on simulation tasks with different coordination demands. We further validate the same organization on four physical three- and four-arm tasks, where Full interaction consistently improves over independently trained local policies. Our contributions are:

VLA-Fabric interaction architecture. We introduce VLA-Fabric, which organizes complete arm-local VLAs as the execution units of multi-arm collaboration and connects them through learned representation interaction. A broad controlled architecture study establishes effective interaction structure, aggregation, and adaptation for coordinated action generation.

Cross-architecture transfer of VLA-Fabric. We re-instantiate the same interaction paradigm in Eagle2 VLM-based SingleVLA and $\pi_{0.5}$ despite their substantially different policy computations. The resulting systems demonstrate a common coordination organization across distinct VLA architectures.

Composable multi-arm scalability. We scale VLA-Fabric from two complete VLA policies to teams of three and four while preserving local action generation and explicit policy interaction. We validate the resulting organization across simulation and physical multi-arm tasks with differing coordination demands.

2 Related Work

Cooperative physical manipulation. Multi-arm robotics has studied coordinated transport, cooperative assembly, and learned dual-arm manipulation under shared physical constraints (Dio et al., 2024; Huang et al., 2025; Tian et al., 2025). These settings require compatible arm motions under a shared load or coordinated holding, alignment, and placement during assembly. Planning and control address the resulting task and motion dependencies. Our work studies the complementary policy-level problem of organizing complete learned VLAs into a collaborative system.

Multi-arm VLA coordination. Multi-arm VLAs organize coordination through several mechanisms. RDT-1B, ET-VLA, TwinVLA, and Dexora resolve cross-arm dependencies within joint bimanual policies (Liu et al., 2025; Li & Peng, 2025; Im et al., 2026; Zhang et al., 2026b). MA-VLA uses high-level behavior assignment (Zhang et al., 2026a), while CHORUS coordinates locally executed policies through teammate observations without inference-time communication (Doshi et al., 2026). VLA-Fabric studies explicit learned interaction between complete arm-local VLA policies during action generation.

Learned interaction among embodied policies. LatentToM combines ego and consensus representations in decentralized diffusion policies and supports explicit consensus exchange for cooperative manipulation (He et al., 2025). Its no-message variant instead infers teammate behavior from local observations. CommNet, DIAL, and TarMAC establish broader foundations for learning communication jointly with agent policies (Sukhbaatar et al., 2016; Foerster et al., 2016; Das et al., 2019). Here, each agent is a complete pretrained VLA containing perception, multimodal reasoning, and action generation. We study which internal coordination functions should connect these policies and whether the resulting interaction organization transfers across substantially different VLA architectures.

3 VLA-Fabric: Learning Interaction among Local VLA Agents

3.1 Local VLA Agents and the Interaction Problem

VLA-Fabric organizes multi-arm manipulation around complete local execution units, termed *VLA agents*. Each agent runs a complete VLA policy and owns one arm's perception-action loop, including its policy parameters, inference state, private observations, proprioception, and action output. Selected peer representations condition local action generation through learned interaction. We use *policy* for the learned mapping π_{θ_i} and *agent* for its execution with local ownership and team in-

teraction relations. Task-level context, such as an instruction or shared scene observation, may be available to multiple agents without changing their local action ownership.

For a team of N agents, agent i predicts an H-step action chunk at planning round t:

$$a_i^{t:t+H-1} = \pi_{\theta_i}(o_i^t, q_i^t, \ell, r_i; \mathcal{M}_i^t). \tag{1}$$

Here o_i^t and q_i^t are the agent's visual and proprioceptive inputs, ℓ is the task instruction, and r_i is an optional role descriptor. The context $\mathcal{M}_i^t$ contains peer information received during this round of action generation. Independently trained policies keep $\mathcal{M}_i^t = \varnothing$ throughout fine-tuning and inference.

The central design problem is the construction of $\mathcal{M}_i^t$ without replacing the local action-generating policies. This requires specifying the peer representations to expose, their insertion points in local computation, the aggregation operator, and the adaptation strategy for learning the resulting interaction. We leave these choices open here and investigate them through the controlled architecture study in Section 3.2.

3.2 Discovering the Interaction Architecture

To study interaction between complete VLA agents, we expose the cross-arm dependencies of a centralized policy while preserving its single-arm priors. TwinVLA provides two identifiable pretrained SingleVLA branches (Im et al., 2026), which we separate into local policies and reconnect through explicit representation interfaces.

Figure 2(a) organizes the design space around three decisions: where policies interact, how peer information is combined, and how the resulting interaction is adapted. **Interaction entry.** Candidates range from Independent policies and Private K/V exchange to additional Common and action-side interaction. **Interaction operator.** Common exchange uses one-way transfer, symmetric averaging (AVG), or operator-level synchronization (RAW); action-side interaction uses Linear or Transformer fusion. **Interaction adaptation.** Single-stage enables the candidate architecture from the start of fine-tuning; Common-first initializes it from a Common-only policy before introducing the remaining interactions.

Nine Single-stage designs and seven matched Common-first variants per task yield 32 runs on Handover Box and Shoes Table. Checkpoints are selected on 200 development conditions and evaluated on 200 disjoint paired conditions, as summarized in Figure 2(b).

Where to interact. Private K/V alone improves over Independent policies on both tasks, and richer configurations obtain further gains from Common and action-side interaction. Although additional inputs do not always help, these comparisons support complementary interaction across policy stages rather than one insertion point for every coordination requirement.

How to interact. The tasks impose different coordination demands, against which the operator choices expose different advantages. Symmetric Common exchange improves over one-way transfer in the Common-first comparisons, while action-side fusion provides further gains in several configurations. Neither AVG versus RAW nor Linear versus Transformer yields a consistent winner across both tasks. These observations motivate retaining complementary interaction functions that policies can exploit as coordination demands change, rather than defining VLA-Fabric around one task's peak configuration.

How to adapt. Common-first consistently improves the matched configurations, showing that interaction structure and its adaptation should be considered jointly when learning richer cross-policy coordination. The interaction functions and their adaptation recipe thus play distinct roles. The former specify which peer representations enter local computation, how they are combined, and where they condition action generation. The latter determines how policies learn to exploit this information during task adaptation.

Selecting a reusable realization. The targeted longer-run comparison finds no consistent advantage for Transformer action fusion over the simpler Linear alternative. We therefore carry forward RAW Common, Private K/V, and Linear action interaction as the canonical SingleVLA-Fabric realization. This choice yields a compact realization that retains the three complementary coordination functions used in the subsequent transfer and scaling studies.

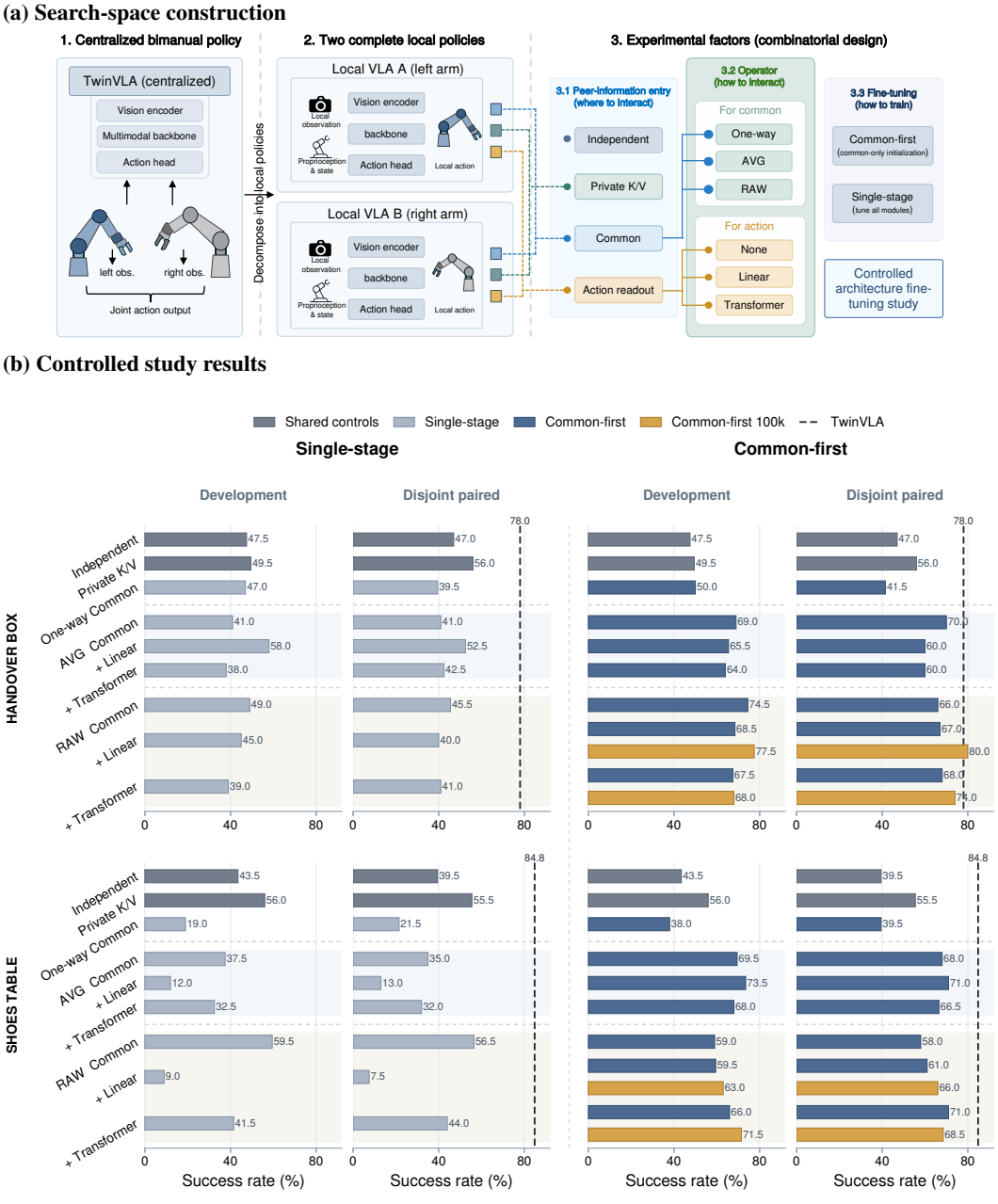


Figure 2: Discovering the VLA-Fabric interaction architecture. (a) Search space over interaction entry, operator, and adaptation. (b) Controlled results on Handover Box and Shoes Table. Gold: 100k RAW follow-up; dashed: TwinVLA.

3.3 From Architecture Search to Functional Interaction

The architecture study motivates three complementary interaction roles. **Shared semantic context** combines Common representations into team-level context. **Peer latent context** exposes peer K/V to local queries, allowing each receiver to select information relevant to its own computation. **Action intent** conditions local action generation on the peer's action-side state. In the selected SingleVLA realization, these roles correspond to RAW Common, Private K/V, and Linear fusion of final readouts. They define coordination functions rather than fixed tensor locations; their formal

operators are given in Appendix A.2. Together, these three roles establish the core VLA-Fabric interaction paradigm in the SingleVLA realization.

4 VLA-FABRIC ACROSS VLA ARCHITECTURES

4.1 RE-INSTANTIATING VLA-FABRIC IN $\pi_{0.5}$

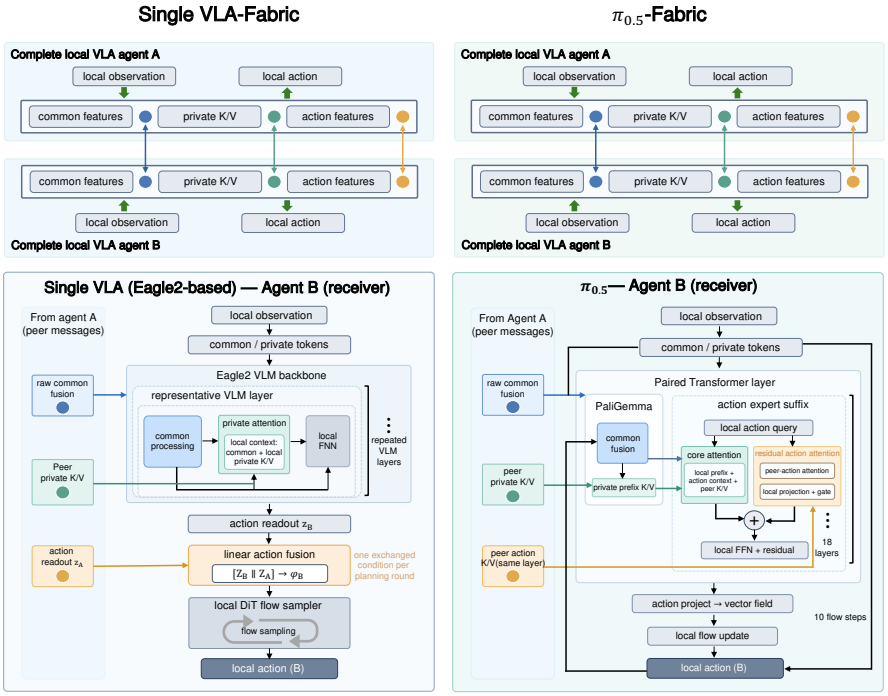


Figure 3: Re-instantiating VLA-Fabric across VLA architectures. The same interaction functions enter different computational boundaries: SingleVLA fuses final readouts, whereas $\pi_{0.5}$ exchanges evolving Action Expert states.

The next question is whether the interaction functions discovered in SingleVLA remain useful beyond its particular computational structure. We use $\pi_{0.5}$ (Physical Intelligence et al., 2025) because its multimodal processing and iterative Action Expert differ from SingleVLA's VLM-readout-conditioned action head. We re-instantiate VLA-Fabric in this architecture, preserving complete local policies and mapping the three coordination roles to architecture-appropriate computation.

Figure 3 aligns the two realizations. SingleVLA-Fabric combines Common and Private K/V within the Eagle2-based VLM, then fuses final readouts before local action generation. The fused readout conditions each arm's separate local action sampler. In $\pi_{0.5}$, shared context is formed through Common multimodal prefix fusion, while Private Prefix K/V supplies peer information to local Action Expert queries. Action interaction moves inside the Action Expert, where gated residual peer attention conditions the evolving action states. The same three coordination roles are therefore preserved through architecture-specific realizations; detailed implementations are given in Appendix B.1.

4.2 CROSS-ARCHITECTURE RESULTS AND INTERACTION USE

Table 1A presents a four-task bimanual capability comparison spanning Handover Box, Shoes Table, Handover Mic, and Scan Object. Despite the architectural differences between Eagle2-based SingleVLA and $\pi_{0.5}$, both realizations support cooperation through the same interaction organization. The $\pi_{0.5}$ results provide a second effective realization of the architecture discovered in Section 3. Table 1B examines interaction use within each trained policy.

Table 1: Bimanual capability and fixed-weight interaction removals (success, %).

A. Bimanual capability					
Policy	Setting	Box	Shoes	Mic	Scan
RDT-1B (Liu et al., 2025)	ICLR 2025	69.4	66.0	90.0	4.0
π_0 (Black et al., 2025)	RSS 2025	78.8	82.4	**98.0**	18.0
TwinVLA (Im et al., 2026)	ICLR 2026	78.0	84.8	84.0	11.0
SingleVLA-Fabric	Full	80.0	70.0	84.0	5.5
$\pi_{0.5}$-Fabric	Full	**80.5**	**94.5**	97.5	33.0
Shared $\pi_{0.5}$-Fabric	Full	69.0	78.0	97.0	**40.0**
B. Fixed-weight interaction removals					
Policy	Inference setting	Box	Shoes	Mic	Scan
SingleVLA-Fabric	Full	75.0	78.0	81.5	5.5
	w/o Common	38.5	19.0	52.5	2.0
	w/o Private K/V	59.0	64.5	60.5	10.0
	w/o action	43.0	7.5	52.5	2.0
	All-off	20.0	1.5	13.5	0.5
$\pi_{0.5}$-Fabric	Full	79.0	94.5	97.5	33.0
	w/o Common	67.0	93.5	99.0	27.5
	w/o Private K/V	78.5	93.5	11.0	4.0
	w/o action	59.0	87.0	98.0	19.0
	All-off	12.5	85.0	1.0	1.0

Full enables all three interaction functions, while **All-off** removes them at inference from a trained Full policy. **Independent** is instead trained and evaluated without interaction. Fixed-weight removals show substantial use of peer information, with the relative importance of each function varying across coordination demands. Action-side interaction is especially important for $\pi_{0.5}$ on Box, whereas peer latent context plays a larger role on Mic and Scan. Shoes remains comparatively robust to individual removals, while SingleVLA is strongly affected across its evaluated tasks. No single interaction function dominates across all settings, leaving room to optimize when, where, and how strongly peer information is used. The Shared $\pi_{0.5}$-Fabric setting jointly fine-tunes the local policies and their interaction modules across all four tasks, using one checkpoint per role for cross-task reuse.

5 Scaling VLA-Fabric Beyond Bimanual Manipulation

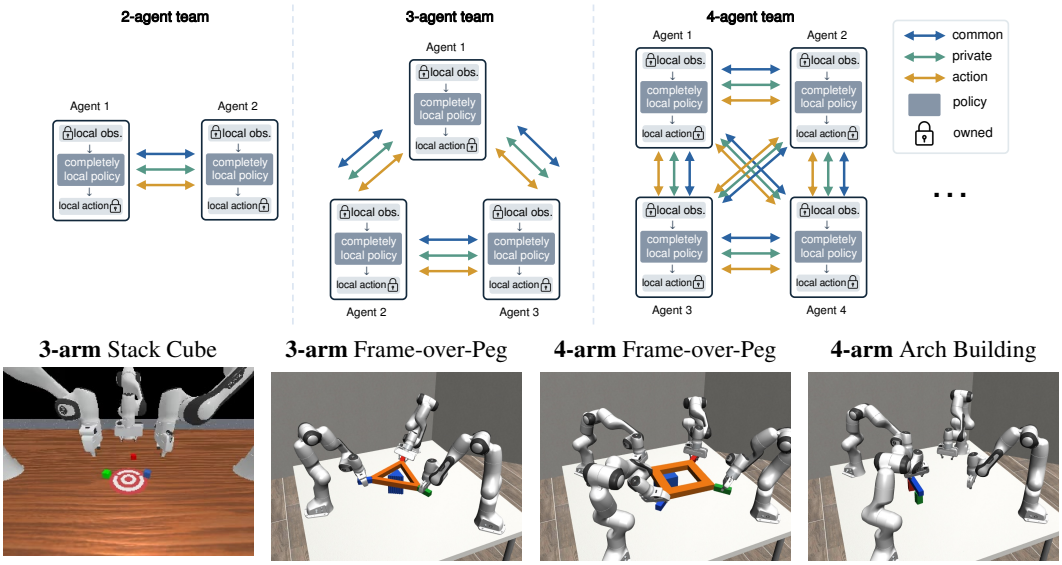


Figure 4: Two- to four-agent VLA-Fabric teams and larger-team simulation tasks (Table 2).

VLA-Fabric aims to move beyond fixed bimanual policies toward multi-arm systems composed of complete VLA agents. Having established the interaction architecture and re-instantiated it across VLA families, we now examine its ability to organize larger collaborative teams. We extend the same peer-interaction structure to three and four policies, each retaining its own local action generation and receiving information from an expanded peer set. The local action heads retain their dimensions as the team gains additional roles. We evaluate three-arm Stack Cube (Stack3) from RoboFactory (Qin et al., 2025) and introduce three- and four-arm Frame-over-Peg (Frame3/4) and four-arm Arch Building (Arch4) for more demanding multi-arm coordination (Figure 4).

Table 2: Larger-team success (%).

Task	Independent	Full
Stack3	90.5	**91.0**
Frame3	67.5	**79.5**
Frame4	9.5	**89.5**
Arch4	0.0	**5.0**

Table 2 suggests that the benefit of explicit interaction depends on how local actions constrain one another. Stack3 largely decomposes into role-specific pick-and-place behaviors, allowing the constituent VLAs to perform well with little measured benefit from interaction. In the frame tasks, several arms continuously support and align a shared object, so each arm's motion changes the conditions under which its peers act. The stronger Full results are consistent with explicit interaction helping policies accommodate these mutual dependencies. VLA-Fabric thus supports both relatively decomposable teams and tightly coupled manipulation, with the value of peer information shaped by the task. Arch4 highlights a complementary challenge: its long-horizon, high-precision assembly combines random object displacement with opportunities for inter-arm collisions. Full achieves some successful completions while Independent achieves none in the reported evaluations, but reliable assembly remains difficult. Organizing policies through interaction can support coordination without automatically resolving the constituent VLAs' planning and precision-control limitations.

6 VLA-Fabric in Real-World Multi-Arm Systems

Having established the interaction architecture, transferred it across VLA families, and extended it to larger simulated teams, we now evaluate VLA-Fabric in physical multi-arm systems. The study covers four three- and four-arm tasks involving parallel placement, object relay, and shared-object manipulation. We collect 50 demonstrations per task and evaluate Full and independently trained local $\pi_{0.5}$ policies over 25 physical trials per policy setting. **Basket3** has three arms place vegetables into a shared basket. In **Relay3**, one arm positions the receiving target, a second hands over an object, and the third receives and places it in the target. **Relay4** uses two arm pairs, each handing over and depositing vegetables. **Frame4** requires four arms to jointly lift, align, and lower a frame over a peg.

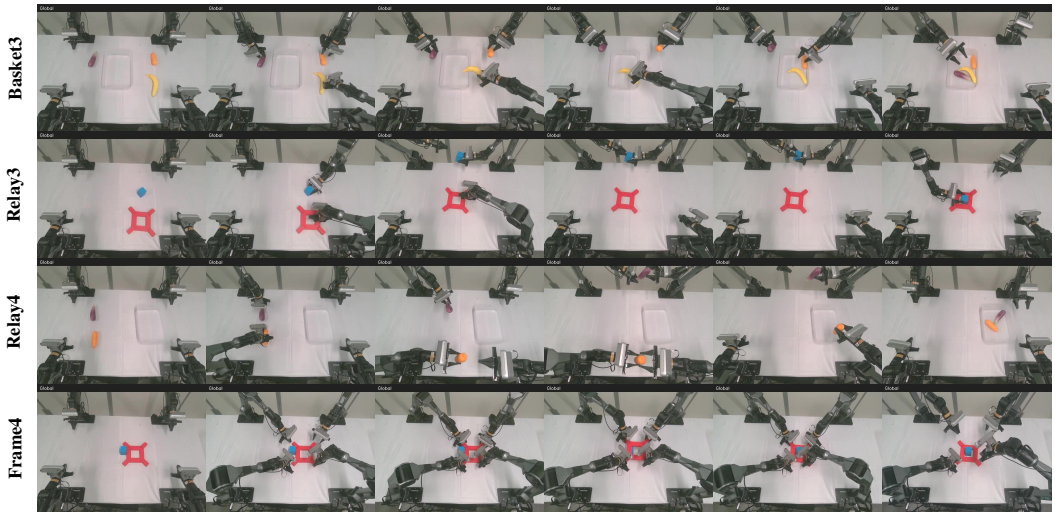


Figure 5: Representative physical executions of the four real-world tasks; each row shows six chronological global-camera frames.

VLA-Fabric Full improves over Independent on all four physical tasks by 16–40 percentage points (Table 3). The pattern is consistent with the simulation study: Basket3 is comparatively decomposable and already supports strong local execution. Relay3, Relay4, and Frame4 impose tighter dependencies through handover timing or shared-object control and show larger benefits from explicit interaction. In a relay, the handing arm must maintain support until the receiving arm can take over. In Frame4, each arm's motion affects the frame supported by all four arms. Such dependencies require coordination during action execution, beyond assigning each arm a role or target. The physical study therefore examines whether complete local policies can accommodate these shared constraints under perception and control uncertainty. The results demonstrate that the multi-VLA organization remains effective under these conditions, while reliable execution of the more coupled behaviors remains an important direction for improvement. Across simulation and hardware, the recurring pattern suggests that interaction benefits reflect physical coupling among agents, beyond arm count alone.

Table 3: Physical success (%).

Task	Independent	Full
Basket3	72.0	**88.0**
Relay3	16.0	**56.0**
Relay4	40.0	**72.0**
Frame4	36.0	**60.0**

7 Discussion and Limitations

VLA-Fabric makes a complete local VLA an explicit organizational unit of multi-arm coordination. Local perception and action ownership are preserved, while collaboration is learned through representation exchange across policy boundaries. The same coordination roles can be re-instantiated across VLA architectures, and additional policies can form larger teams without redefining a global joint action space. This separates progress in constituent models from progress in team organization, allowing the two to advance together. Stronger VLAs can improve the execution units, while interaction mechanisms determine which information is shared, when it enters policy computation, and with which peers. Across the evaluated architectures, tasks, and team sizes, this common organization supports reuse of the coordination principle while allowing the underlying policy computation to change.

Our experiments use homogeneous teams with fixed roles and up to four arms, leaving changing team compositions and heterogeneous VLAs as directions for broader composability. Broader composition will require interfaces that accommodate differences in representation and action generation while maintaining clear local policy boundaries. Such interfaces must align the meaning and timing of exchanged information across policies with different internal computations. The current interaction structure is static; adapting content, topology, and frequency to task phase and agent participation could make coordination more selective. Long-horizon, high-precision manipulation remains difficult, as illustrated by Arch Building, motivating complementary improvements in planning and control of the constituent VLAs. Larger teams also motivate more selective representation exchange and interaction mechanisms that accommodate changing role assignments and available communication resources.

8 Conclusion

In this paper, we introduced VLA-Fabric to address how multiple complete VLA policies can collaborate as a coordinated system for increasingly demanding multi-arm manipulation. Each arm-local VLA remains a complete execution unit, preserving its own perception and action generation while acquiring collaborative capability through learned representation interaction. We established this organization through systematic exploration of interaction entry, aggregation, and adaptation, then re-instantiated its coordination functions in a structurally different VLA. Extending the same principle to three- and four-agent teams in simulation and physical systems completes a framework for discovering, transferring, and scaling inter-VLA coordination. This organization allows individual VLA capability and multi-VLA coordination to advance together, connecting improvements in local models with the mechanisms through which they collaborate. Beyond the operators studied here, it opens directions in coordination-aware model design, adaptive representation exchange, communication-efficient deployment, and heterogeneous VLA teams. VLA-Fabric thereby offers a concrete step from increasingly capable individual embodied models toward networked embodied intelligence.

AI USE STATEMENT

Generative AI tools were used to assist research planning, literature organization, manuscript structuring, language editing, and code-oriented document preparation. The authors verified all cited sources, implementations, experimental protocols, numerical results, analyses, and scientific claims, and take full responsibility for the submission.

REPRODUCIBILITY STATEMENT

Appendices A–C mirror the discovery, transfer, and larger-team studies, documenting implementation, selection, complete results, and evaluation boundaries. The anonymous project website links the code release, including both VLA realizations and 2/3/4-agent workflows. Pretrained weights and datasets are obtained separately as documented in the release.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025. URL https://www.roboticsproceedings.org/rss21/p010.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

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

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *International Conference on Machine Learning*, volume 97, pp. 1538–1546, 2019. URL https://proceedings.mlr.press/v97/das19a.html.

Maximilian Dio, Knut Graichen, and Andreas Völz. Time-optimal path parameterization for cooperative multi-arm robotic systems with third-order constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3043–3048, 2024. doi: 10.1109/IROS58592.2024.10802342. URL https://cris.fau.de/publications/338877293/.

Ria Doshi, Tian Gao, Annie Chen, Chelsea Finn, and Jeannette Bohg. CHORUS: Decentralized multi-embodiment collaboration with one VLA policy. *arXiv preprint arXiv:2606.12352*, 2026. URL https://arxiv.org/abs/2606.12352.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016. URL https://papers.neurips.cc/paper_files/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html.

Chengyang He, Gadiel Sznaier Camps, Xu Liu, Mac Schwager, and Guillaume Sartoretti. Latent theory of mind: A decentralized diffusion architecture for cooperative manipulation. *arXiv preprint arXiv:2505.09144*, 2025. URL https://arxiv.org/abs/2505.09144.

Philip Huang, Ruixuan Liu, Shobhit Aggarwal, Changliu Liu, and Jiaoyang Li. APEX-MR: Multi-robot asynchronous planning and execution for cooperative assembly. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.098. URL https://www.roboticsproceedings.org/rss21/p098.html.

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

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

Chengmeng Li and Yaxin Peng. Embodiment transfer learning for vision-language-action models. *arXiv preprint arXiv:2511.01224*, 2025. URL https://arxiv.org/abs/2511.01224.

Zhiqi Li, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, Yishen Ji, Shiyi Lan, Hao Zhang, Yilin Zhao, Subhashree Radhakrishnan, et al. Eagle 2: Building post-training data strategies from scratch for frontier vision-language models. *arXiv preprint arXiv:2501.14818*, 2025. URL https://arxiv.org/abs/2501.14818.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.07864.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Yiran Qin, Li Kang, Xiufeng Song, Zhenfei Yin, Xiaohong Liu, Xihui Liu, Ruimao Zhang, and Lei Bai. RoboFactory: Exploring embodied agent collaboration with compositional constraints. *arXiv preprint arXiv:2503.16408*, 2025. URL https://arxiv.org/abs/2503.16408.

RoboTwin Team. RoboTwin 2.0 leaderboard, 2026. URL https://robotwin-platform.github.io/leaderboard.html. Single-task, clean2clean results. Board updated September 1; accessed September 23, 2026.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2016. URL https://arxiv.org/abs/1605.07736.

Yunsheng Tian, Joshua Jacob, Yijiang Huang, Jialiang Zhao, Edward Li Gu, Pingchuan Ma, Annan Zhang, Farhad Javid, Branden Romero, Sachin Chitta, Shinjiro Sueda, Hui Li, and Wojciech Matusik. Fabrica: Dual-arm assembly of general multi-part objects via integrated planning and learning. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 406–419. PMLR, 2025. URL https://proceedings.mlr.press/v305/tian25a.html.

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

Zongzheng Zhang, Jingrui Pang, Zhuo Yang, Kun Li, Minwen Liao, Saining Zhang, Guoxuan Chi, Jinbang Guo, Huan-ang Gao, Modi Shi, Dongyun Ge, Yao Mu, Jiayuan Gu, Rui Chen, Hao Dong, Huazhe Xu, Li Yi, Yixin Zhu, Hang Zhao, Pengwei Wang, Shanghang Zhang, Guocai Yao, Jianyu Chen, Hongyang Li, and Hao Zhao. Dexora: Open-source VLA for high-dof bimanual dexterity. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026b. URL https://arxiv.org/abs/2605.18722.

APPENDIX

Appendices A–C provide supporting formulations, experimental settings, and additional results for interaction discovery, cross-architecture transfer, and larger-team scaling, respectively.

A INTERACTION ARCHITECTURE DISCOVERY

A.1 SINGLEVLA DECOMPOSITION AND DESIGN SPACE

TwinVLA's two pretrained SingleVLA branches expose identifiable cross-arm dependencies while preserving each arm's pretrained prior (Im et al., 2026). We separate these branches into complete local policies and vary the representations through which they interact. All candidates originate from the same pretrained SingleVLA policies, using an Eagle2 VLM-based backbone. Table A1 defines the nine interaction designs.

One-way Common shares the Common keys and values from one designated arm. AVG Common averages the two agents' Common keys and values before they condition private attention. RAW Common synchronizes Common computation at operator boundaries, before the agents proceed with their local computation. Both symmetric variants retain Private K/V interaction. The Linear and Transformer variants additionally combine role-ordered action-side readouts before each arm's local action head. Thus aggregation, receiver-specific latent conditioning, and late-action capacity can be varied without changing local observation or action ownership.

Table A1: Interaction design space. Each design preserves complete local VLA policies.

Design	Common interaction	Private K/V	Action interaction
I0	None	Off	None
I1	None	On	None
I2	One-way	On	None
I3-AVG	Averaged K/V	On	None
I3-RAW	Operator-level	On	None
I4-AVG	Averaged K/V	On	Linear
I4-RAW	Operator-level	On	Linear
I5-AVG	Averaged K/V	On	Transformer
I5-RAW	Operator-level	On	Transformer

A.2 FUNCTIONAL FORMULATION OF VLA-FABRIC

Let $\mathcal{N}(i)$ denote agent i's incoming peers and k a layer or interaction stage; the planning-round index is omitted below. The three functions condition local action generation at different computational boundaries.

Shared semantic context. Agent j exposes its Common-owned representation $h^k_{j,\mathrm{sem}}$. The symmetric operator $\mathrm{Agg}^k_{\mathrm{sem}}$ combines these representations into shared context while private observations remain local:

$$\widetilde{h}^k_{i,\mathrm{sem}} = \mathrm{Agg}^k_{\mathrm{sem}}\big(\{h^k_{j,\mathrm{sem}} : j \in \mathcal{N}(i) \cup \{i\}\}\big). \tag{A.1}$$

This expression describes the symmetric Common realizations; the one-way candidate instead restricts the direction of context transfer.

Peer latent context. Agent i retains its local query Q^k_i and local keys and values K^k_i, V^k_i. The role-ordered peer keys and values $\widehat{K}^k_{\mathcal{N}(i)}, \widehat{V}^k_{\mathcal{N}(i)}$ augment this context. Attention produces the receiver-specific latent context c^k_i:

$$c^k_i = \mathrm{Attn}\big(Q^k_i, [K^k_i, \widehat{K}^k_{\mathcal{N}(i)}], [V^k_i, \widehat{V}^k_{\mathcal{N}(i)}]\big). \tag{A.2}$$

Brackets denote concatenation along the token dimension. Each receiver selects relevant peer information through its own local query.

Action intent. Each z_j represents agent j's action-side state, a final readout in the selected SingleVLA implementation. Concatenation follows a fixed role order, and a receiver-specific fusion operator ϕ_i^{act} produces $\widetilde{z}_i$ for local action generation:

$$\widetilde{z}_i = \phi_i^{\text{act}}\big(\text{Concat}^{\text{role}}_{j\in\mathcal{N}(i)\cup\{i\}}\, z_j\big). \tag{A.3}$$

The selected realization uses Linear fusion followed by the local flow-matching action head. Equations A.1–A.3 describe the three functional roles through their respective interaction operators. Section 4 re-instantiates them within a different VLA computation.

A.3 Experimental Setup

The controlled study uses ALOHA Handover Box and Shoes Table. Each task includes nine Single-stage designs and seven Common-first variants, giving 32 model–task runs. Independent and Private K/V lack Common interaction and serve as shared controls across the two adaptation settings.

Single-stage enables the target interaction architecture directly during fine-tuning. Common-first initializes the policies with Common interaction before introducing the remaining interaction functions. The total adaptation budgets are 40k updates for Single-stage and 50k for Common-first, including initialization. With these unequal budgets, the comparison measures the sensitivity of each design to the complete adaptation recipe.

All variants freeze the vision encoder and adapt the language backbone, action heads, and enabled interaction modules. Interaction operators are evaluated in one differentiable training graph, with the native action objective applied to every arm. A normalized description is

$$\mathcal{L}_{\text{act}} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{\text{VLA}}\left(\widehat{a}_i^{t:t+H-1}, a_i^{t:t+H-1}\right). \tag{A.4}$$

Here H denotes the predicted action horizon. The implementation sums the arm-local losses, differing by a constant scale at fixed team size. The prediction target is unchanged across interaction designs.

Models are selected using 200 development conditions and evaluated on 200 disjoint paired conditions per task. The paired conditions are shared across designs, allowing comparison under the same task initializations. Success is the proportion of episodes satisfying the task's completion criterion; episodes reaching the horizon without completion remain failures.

A.4 Complete Architecture-Search Results

Table A2 gives the complete numerical matrix underlying the paired-condition panels in Figure 2. Each row reports one design and adaptation choice on both tasks. Common-first improves all seven matched designs, but the ranking of Common aggregation and action operators varies across tasks. The matrix therefore distinguishes access to peer information from the policy's ability to learn useful conditioning on that information.

Table A2: Complete architecture-search results. Success rates (%) use 200 paired conditions per task, disjoint from model selection.

Design	Adaptation	Handover Box	Shoes Table
I0: Independent	Single-stage	47.0	39.5
I1: Private K/V	Single-stage	56.0	55.5
I2: One-way Common	Single-stage	39.5	21.5
I3-AVG	Single-stage	41.0	35.0
I3-RAW	Single-stage	45.5	56.5
I4-AVG: Linear	Single-stage	52.5	13.0
I4-RAW: Linear	Single-stage	40.0	7.5
I5-AVG: Transformer	Single-stage	42.5	32.0
I5-RAW: Transformer	Single-stage	41.0	44.0
I2: One-way Common	Common-first	41.5	39.5

Table A2 (continued)

Design	Adaptation	Handover Box	Shoes Table
I3-AVG	Common-first	70.0	68.0
I3-RAW	Common-first	66.0	58.0
I4-AVG: Linear	Common-first	60.0	71.0
I4-RAW: Linear	Common-first	67.0	61.0
I5-AVG: Transformer	Common-first	60.0	66.5
I5-RAW: Transformer	Common-first	68.0	71.0

A.5 Architecture Selection and SingleVLA Controls

The extended comparison considers two preselected architecture families, RAW Common with Linear or Transformer action interaction. Table A3 compares their terminal models at the same total budget. Added Transformer capacity does not provide a consistent advantage across tasks, supporting Linear fusion as a compact common realization for subsequent studies. This targeted follow-up reuses the architecture-analysis conditions.

Table A3: Extended-budget Common-first comparison, with 110k total updates including initialization. Success rates (%) use 200 paired conditions per task.

Design	Handover Box	Shoes Table
I4-RAW: Linear	80.0	66.0
I5-RAW: Transformer	74.0	68.5

Independent in Table A2 is trained without all three interaction functions. In contrast, the SingleVLA removals in Table 1 retain the trained Full weights and suppress selected information only at inference. All-off disables Common aggregation, Private K/V, and action fusion together. These removal results come from a separate fixed-weight removal batch. They test how trained policies use interaction; Independent tests what policies learn without an interaction interface.

B Cross-Architecture Transfer of VLA-Fabric

B.1 From SingleVLA to $\pi_{0.5}$

Action-generation boundary. Both VLA families generate continuous actions with flow matching (Lipman et al., 2022). SingleVLA uses an Eagle2-based VLM and a separate DiT action head (Li et al., 2025; Im et al., 2026). Its backbone readout is fused once before the two local action heads run their iterative samplers. In $\pi_{0.5}$, peer-action interaction instead operates on the evolving Action Expert suffix within each flow step.

Common and Private Prefix interaction. Each arm retains a complete $\pi_{0.5}$ policy with its own parameters, proprioception, action-generation state, and output head. Common interaction synchronizes the Common-owned multimodal PaliGemma prefix at seven operator boundaries in each of 18 layers. Private Prefix K/V remain associated with their sending role and augment the context available to the receiver's Action Expert. The receiver keeps its query local, so peer context is selected according to its own evolving action computation. Raw private images are not exchanged.

Receiver-local action interaction. A separate peer-action attention branch adds a gated residual before the Action Expert feed-forward block. For receiver i and peer j, let k index the expert layer and s the flow step. Peer action-side keys and values condition the local action-side query:

$$r_i^{k,s} = \alpha_i^k W_i^k \,\mathrm{Attn}\Big(Q_{i,\mathrm{act}}^{k,s}, K_{j,\mathrm{act}}^{k,s}, V_{j,\mathrm{act}}^{k,s}\Big), \qquad h_i^{k,s} = h_{i,\mathrm{core}}^{k,s} + r_i^{k,s}. \tag{B.1}$$

Here $h_{i,\mathrm{core}}^{k,s}$ is the receiver's core attention state; W_i^k and α_i^k are its output projection and scalar gate. The gate is zero-initialized and learned without a hard bound, initially preserving the core computation while allowing the interaction to develop. Peer attention has a separate softmax from core attention and is refreshed at every Action Expert layer and flow step. The update conditions the receiver's evolving action representations throughout the iterative generation process.

The correspondence with Appendix A.2 is functional: Common forms shared context, Private Prefix K/V provide receiver-specific context, and the residual branch supplies action-stage interaction. Each homogeneous team uses interaction parameters fine-tuned within its own VLA architecture.

B.2 Cross-Architecture Experimental Setup

The transfer study uses ALOHA Handover Box and Shoes Table, together with RoboTwin 2.0 Handover Mic and Scan Object (Chen et al., 2025). Models are fine-tuned from pre-trained $\pi_{0.5}$ policies using the native flow-matching action objective. Table B1 summarizes the demonstrations, total adaptation, and evaluation size. The Box and Shoes Full models use Common-first initialization; Mic and Scan enable Full interaction directly. Each task uses fixed model weights for evaluation and interaction removal. In particular, Scan's Full and removal results use the same model with 60k total adaptation updates, trained on 50 demonstrations.

Table B1: Task-specific $\pi_{0.5}$ Full settings. Update counts include all adaptation, including initialization where used.

Task	Demos	Total updates	Trials
Handover Box	50	50k	200
Shoes Table	50	40k	200
Handover Mic	50	30k	200
Scan Object	50	60k	200

Evaluation conditions fix the task initialization and instruction across compared settings. Observation assignment is unchanged between Full and its controls: each policy receives its private wrist observation and state, as well as the shared external view where supplied by the task. Independent retains these inputs while removing all representation exchange during both learning and inference. All-off instead disables the three paths in an already trained Full policy. These interventions retain different learning histories even when their inference-time information interfaces coincide.

B.3 Reference Settings and Additional Tasks

Table B2: Reference evaluation settings. Trial counts are per task.

Methods	Tasks	Source / setting	Trials
RDT-1B, π_0, TwinVLA	Box, Shoes	Im et al. (2026), Table 9 / Easy	500
TwinVLA	Mic, Scan	Im et al. (2026), Table 10 / Easy	100
RDT-1B, π_0	Mic, Scan	RoboTwin Team (2026) / Single, clean2clean	100

Exploratory $\pi_{0.5}$ Full policies also reach 31.5% on Handover Block and 23.0% on Hanging Mug, each over 200 development episodes. Handover Block is distinct from the ALOHA Handover Box task. These additional outcomes indicate remaining difficulty in multi-stage manipulation.

B.4 Additional Interaction Controls

Table 1 reports the individual-path removals and All-off results. In additional 200-episode evaluations, a separately trained $\pi_{0.5}$ Independent policy achieves 63.5% on Handover Box. Disabling both Private Prefix K/V and action interaction in the Full checkpoint, while retaining Common interaction, yields 53.5% on Handover Box and 82.5% on Shoes Table. This is a joint inference-time removal without retraining. The Independent Box model uses 40k total updates, compared with 50k for Full, so the training comparison has unequal budgets.

C Scaling VLA-Fabric to Larger Teams

C.1 From Two to N Local VLA Agents

The interaction functions in Appendix A.2 extend to a team by expanding each receiver's peer set. For the evaluated fully connected teams, $\mathcal{N}(i) = \{1, \ldots, N\} \setminus \{i\}$. Each added agent retains its complete policy, local proprioception, action-generation state, and local action head. Common

aggregation includes the additional agent, and Private Prefix K/V are concatenated in the team's fixed role order. The action branch attends to each peer separately, then concatenates the resulting contexts before the receiver-local projection:

$$u_{ij}^{k,s} = \mathrm{Attn}\Big(Q_{i,\mathrm{act}}^{k,s}, K_{j,\mathrm{act}}^{k,s}, V_{j,\mathrm{act}}^{k,s}\Big), \quad r_i^{k,s} = \alpha_i^k W_i^k \,\mathrm{Concat}_{j\in\mathcal{N}(i)}^{\mathrm{role}}\, u_{ij}^{k,s}. \tag{C.1}$$

This extends Equation B.1 while keeping both the receiver's query and the attention normalization for each peer local. The local observation and action dimensions remain unchanged as arms are added, although peer context and role-dependent modules can grow. Each task and team configuration is fine-tuned separately.

C.2 Simulation Tasks and Setup

Table C1 specifies the team sizes and completion criteria for the larger-team tasks. Stack3 uses RoboFactory; Frame3, Frame4, and Arch4 use our MuJoCo task environments. Agents receive local wrist images and proprioception, together with the global scene view supplied as shared context. Full and Independent use the same task data and observation assignment. Independent disables Common, Private K/V, and action interaction throughout fine-tuning and inference.

Table C1: Larger-team simulation settings. Each policy setting uses 200 evaluation trials; completion requires the full task outcome.

Task	Arms	Success criterion
Stack3	3	Complete the ordered cube stack with all assigned arms released.
Frame3	3	Place the triangular frame over the peg after joint lifting and transport; release and retreat with the frame stable.
Frame4	4	Place the quadrilateral frame over the peg after joint lifting and transport; release and retreat with the frame stable.
Arch4	4	Position the supports and place the beam to form a stable completed arch.

The total Full adaptation budgets are 20k updates for Stack3, 30k each for Frame3 and Frame4, and 50k for Arch4. The Stack3 and frame Independent controls match the corresponding total budgets. Frame4 nevertheless compares different adaptation recipes, so its training comparison is distinct from a fixed-weight path removal. Evaluations share task initializations, although rendered observations can differ across policy runs. Each setting uses 200 trials, and episodes that reach the horizon without completion remain failures. Arch4 combines a longer assembly sequence, randomized object displacement, and inter-arm collision risk.

C.3 Additional Scaling Evidence

Beyond the final comparisons in Table 2, a direct-Full Stack3 policy reaches 87.5% over 200 development episodes after 10k total updates. Frame3 also uses direct-Full adaptation. These examples show that direct-Full adaptation can also learn the transferred interaction functions.

Additional SingleVLA-Fabric teams, based on the Eagle2 VLM, perform Camera Alignment and Take Photo in RoboFactory (Qin et al., 2025). Camera Alignment requires two arms to lift a camera while a third holds an object in front of it. Take Photo adds a fourth arm that presses the camera button while the other arms maintain the camera–object arrangement. Full achieves 100.0% on three-arm Camera Alignment and 96.5% on four-arm Take Photo, each over 200 evaluation episodes. On the same 200 Take Photo conditions, All-off reduces success to 17.5% by disabling Common, Private K/V, and action interaction in the Full checkpoint without retraining. These results extend the larger-team evidence to the Eagle2 VLM-based realization and to coordinated holding and actuation, complementing the $\pi_{0.5}$ tasks in the main text.

C.4 Physical-Robot Setup and Evaluation

Hardware and observations. The platform uses AgileX PiPER robot arms, each with six actuated joints and a gripper, and Intel RealSense D455 cameras. Each policy receives its local wrist RGB

image, joint positions, and gripper state. One global RGB view provides shared scene context in both Full and Independent settings.

Demonstrations and action interface. Each VLA generates local chunks of six joint-position targets and one gripper command per arm. Each task uses 50 physical demonstrations for task-specific $\pi_{0.5}$ fine-tuning.

Tasks and evaluation. Table C2 defines the four tasks. Each Full and Independent setting uses 25 physical trials; Independent learns without interaction, whereas Full retains all three functions. Success percentages and coordination sequences appear in Table 3 and Figure 5.

Table C2: Physical task definitions and successful completion.

Task	Arms	Role assignment and completion criterion
Basket3	3	Each arm loads an assigned vegetable; success requires all vegetables to be placed in the shared basket.
Relay3	3	One arm positions the receiving target, while two arms transfer the object; success requires handover and placement within the target.
Relay4	4	Two sender–receiver pairs relay vegetables to the basket; both pairs must complete their handovers and basket placements.
Frame4	4	Four arms jointly lift, align, and lower the frame over the peg; success requires stable placement after release.