LESS FUSION, SAME ACTION: REVISITING REDUNDANT CONNECTIVITY IN MIXTURE-OF-TRANSFORMER BASED ROBOT POLICY MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Transformers (MoT) robot policies typically maintain dense expert connectivity across network depth, yet it remains unclear how much of this connectivity is actually necessary for downstream adaptation. We conduct a systematic empirical study of expert connectivity redundancy by sparsifying pretrained MoT policies during fine-tuning across multiple architectures and robot manipulation benchmarks. We find that substantial portions of dense expert connectivity can be removed while largely preserving downstream performance, and that different sparse masks with matched budgets achieve similar results, suggesting that no particular subset of expert connections is uniquely indispensable. Motivated by this flexibility, we introduce an early-half fusion pattern that concentrates cross-expert connectivity in the first half of the network and applies structured pruning to the remaining layers. This design maintains comparable task success while enabling practical deployment simplifications, including truncation of unused conditioning layers and reduction of historical KV-cache storage in autoregressive policies. Deployment measurements further show reductions in loaded parameters, GPU memory, and per-chunk latency. Together, our results suggest that dense expert connectivity in pretrained MoT robot policies is substantially redundant, and that structured pruning provides a simple path toward more efficient deployment without training a new model from scratch. Our code will be released.

1 INTRODUCTION

Recent advances in vision-language-action models and world-action models have increasingly adopted Mixture-of-Transformers (MoT) architectures (Liang et al., 2024) to integrate heterogeneous capabilities for robotic control (Black et al., 2024; Intelligence et al., 2025; Cai et al., 2026; Bi et al., 2026). In these models, functionally specialized expert streams are responsible for different roles, such as visual-language understanding, future-state prediction, and action generation, while exchanging information through joint attention across Transformer layers. This layer-wise expert fusion provides a flexible interface for combining pretrained representations and task-specific prediction modules, and has become a common design choice in recent robot foundation policies.

Despite its widespread use, the connectivity structure of expert fusion remains poorly understood. Existing MoT policies typically preserve their native expert interactions throughout most or all of the network depth (Intelligence et al., 2025; Agarwal et al., 2026; Luo et al., 2026; Li et al., 2026b;c), implicitly assuming that dense communication across layers is necessary for effective downstream control. However, it is unclear whether every expert interaction must be retained after pretraining, or whether substantial portions of this connectivity are redundant during downstream adaptation. This raises a fundamental question: *how much expert communication does a pretrained MoT robot policy actually need, and at which layers is such communication necessary?*

To answer this question, we conduct an empirical study on expert connectivity redundancy by systematically sparsifying the native expert-fusion connections of pretrained MoT policies during downstream fine-tuning. Across multiple representative architectures and robot manipulation benchmarks, we find that substantial portions of expert connectivity can be removed while retaining performance close to their dense counterparts. More importantly, different sparse masks with

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.

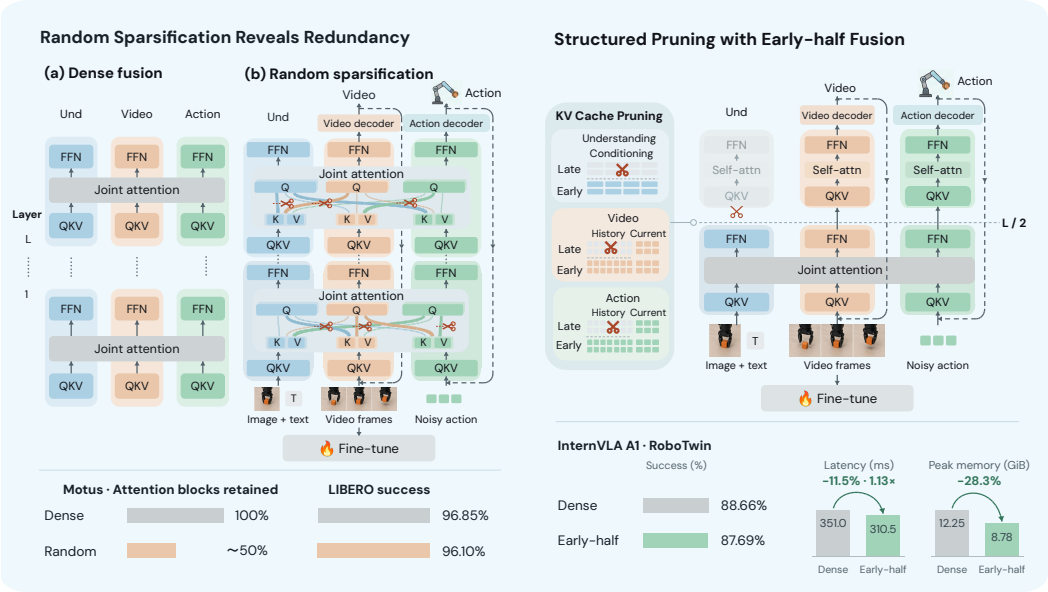

Figure 1: **Overview of our study. Left:** Random sparsification reveals substantial redundancy in dense expert connectivity while largely preserving downstream performance. **Right:** Early-half fusion concentrates cross-expert fusion in early layers. While preserving performance, it enables reductions in deployed parameters, KV-cache storage, GPU memory, and inference latency.

the same connectivity budget achieve similar performance, despite preserving substantially different sets of expert interactions. These results suggest that successful downstream adaptation is not dependent on a particular fixed subset of indispensable connections, revealing substantial redundancy and flexibility in how pretrained experts communicate across network depth.

Building on this observation, we further ask whether the flexibility of expert connectivity can be translated into a structured pruning pattern that directly benefits deployment efficiency. We introduce an early-half fusion design that preserves cross-expert connectivity only in the first half of the network while removing it from later layers. This simple structure maintains comparable downstream task performance, while making the upper-layer computation substantially more separable across experts. As a result, unused upper conditioning layers can be removed to reduce deployed parameter count, and for autoregressive policies, restricting upper-layer attention to the current chunk can further eliminate historical KV-cache requirements. These results show that the redundancy identified in our empirical study can be converted into concrete efficiency gains without redesigning or pretraining a new robot foundation policy.

Our main takeaways are:

• **Expert connectivity is highly redundant.** Pretrained MoT robot policies can tolerate substantial sparsification during downstream adaptation with little performance loss.

• **No specific sparse topology is indispensable.** Different masks with similar sparsity budgets achieve comparable performance.

• **Fusion can be concentrated in early layers.** Restricting cross-expert communication to the first half of the network largely preserves downstream performance.

• **Structured pruning improves deployment efficiency.** Early-half fusion enables parameter truncation and KV-cache reduction, lowering memory usage and inference latency.

2 PRELIMINARIES

Mixture-of-Transformer based Robot Policy Models. The policy models considered in this work primarily follow a MoT design (Liang et al., 2024; Deng et al., 2025), in which function-

Table 1: Architectural variants of the studied robot policies. * denotes the publicly released shared-backbone implementation.

Model	Type	Denoising	Cross-Chunk	Backbone	Expert streams
$\pi_{0.5}$	VLA	Causal	Non-AR	Separate	Und, Action
InternVLA-A1	VLA	Causal	Non-AR	Separate	Und, Gen, Action
Motus	WAM	Joint	Non-AR	Separate	Und, Video, Action
LingBot-VA*	WAM	Causal	AR	Shared	Video, Action

ally distinct token groups, referred to as *experts*, exchange information through joint attention along each Transformer layer. Let $\mathcal{E} = \{e_1, \ldots, e_K\}$ denote the expert set and L the number of Transformer layers, indexed by $\ell \in \{1, \ldots, L\}$. The token representations entering layer ℓ are organized by expert identity as $\mathbf{X}^{(\ell)} = [\mathbf{X}^{(\ell)}_{e_1}; \ldots; \mathbf{X}^{(\ell)}_{e_K}]$. In general, the attention projection and feed-forward parameters are expert-specific, although the shared-backbone variants considered below reuse them across streams. Regardless of parameterization, the resulting queries, keys, and values are grouped by their associated experts. For two experts e_i and e_j, the attention-logit block at layer ℓ, through which queries $\mathbf{Q}^{(\ell)}_{e_i}$ read keys $\mathbf{K}^{(\ell)}_{e_j}$, is given by

$$\mathbf{S}^{(\ell)}_{e_i \to e_j} = \mathbf{Q}^{(\ell)}_{e_i} \mathbf{K}^{(\ell)\top}_{e_j} / \sqrt{d} \quad . \tag{1}$$

where d is the common query/key dimension of each attention head; the head index is omitted for clarity. The notation $e_i \to e_j$ indicates that queries from e_i read keys from e_j. Diagonal blocks $\mathbf{S}^{(\ell)}_{e_i \to e_i}$ correspond to within-expert self-attention, whereas off-diagonal blocks $\mathbf{S}^{(\ell)}_{e_i \to e_j}$, $i \neq j$, enable cross-expert information exchange.

Expert Connectivity Across MoT Layers. At each Transformer layer, a MoT policy defines which expert streams can exchange information through attention. Recall that $\mathbf{S}^{(\ell)}_{e_i \to e_j}$ denotes the attention-logit block through which queries from expert e_i attend to keys from expert e_j at layer ℓ. We define the native expert connectivity of a pretrained policy as

$$\Omega = \{(\ell, i, j) \mid \mathbf{S}^{(\ell)}_{e_i \to e_j} \text{ is permitted by the native attention structure.}\} \tag{2}$$

Each element $(\ell, i, j) \in \Omega$ therefore represents an expert-level connection at layer ℓ. Diagonal connections ($i = j$) correspond to within-expert attention, whereas off-diagonal connections ($i \neq j$) enable cross-expert information flow. Depending on the underlying architecture, these connections may be symmetric or directional. For example, a causal attention structure may allow one expert to read from another without permitting the reverse direction. This layer-wise connectivity provides a common abstraction across different MoT architectures. In our study, for each native connection, we associate a binary indicator $m^{(\ell)}_{e_i \to e_j} \in \{0, 1\}$, where 1 retains the connection and 0 removes it. The resulting attention block is

$$\widetilde{\mathbf{S}}^{(\ell)}_{e_i \to e_j} = \begin{cases} \mathbf{S}^{(\ell)}_{e_i \to e_j}, & m^{(\ell)}_{e_i \to e_j} = 1, \\ -\infty, & m^{(\ell)}_{e_i \to e_j} = 0. \end{cases} \tag{3}$$

The binary mask is applied on top of the native attention structure, so connections that are disallowed by the original architecture remain unavailable. A complete set of masks therefore specifies which within-expert and cross-expert interactions are preserved at each layer, providing the common representation of expert connectivity used throughout our study. Valid masks must leave each query expert with at least one accessible key expert at every layer after all masking. In particular, if a query expert can attend only to itself at a given layer, its within-expert block must be retained.

3 Empirical Study on Expert Connectivity Redundancy

We study whether pretrained MoT robot policies can adapt to sparser expert connectivity while retaining comparable performance. We evaluate this question across multiple representative architectures and manipulation benchmarks using controlled downstream sparsification.

Table 2: Fusion sparsification on RoboTwin. Entries are task-averaged success rates (%) on matched task sets within each model and setting, with 100 episodes per included task. Easy/Hard denote clean/domain-randomized evaluation. A-C are three random masks per model; Bold and underlining mark the best and second-best; parentheses give changes from the dense baseline.

			Random masks				Structured
Model	**Setting**	**Dense**	**A**	**B**	**C**	**Mean**	**Early-half fusion**
$\pi_{0.5}$	Easy	80.20	78.34	78.38	79.66	78.79	77.66
	Hard	80.50	81.22	81.28	82.38	81.63	79.54
	Avg.	**80.35**	79.78	79.83	81.02	<u>80.21</u> (−0.14)	78.60 (−1.75)
InternVLA-A1	Easy	88.52	87.80	87.34	85.94	87.03	87.74
	Hard	88.80	87.48	87.14	86.72	87.11	87.64
	Avg.	**88.66**	87.64	87.24	86.33	87.07 (−1.59)	<u>87.69</u> (−0.97)
Motus	Easy	87.72	88.16	87.04	87.32	87.51	86.16
	Hard	85.40	87.32	87.02	85.68	86.67	84.82
	Avg.	<u>86.56</u>	87.74	87.03	86.50	**87.09** (+0.53)	85.49 (−1.07)
LingBot-VA	Easy	91.52	86.89	87.10	86.94	86.98	92.76
	Hard	90.86	87.22	88.30	87.72	87.75	92.34
	Avg.	<u>91.19</u>	87.06	87.70	87.33	87.36 (−3.83)	**92.55** (+1.36)

3.1 Experimental Setup

Models. We study four representative pretrained MoT robot policies following the paradigm described in Sec. 2 but with different expert composition, denoising mechanism, temporal dependency, and backbone parameterization: $\pi_{0.5}$ (Intelligence et al., 2025), InternVLA-A1 (Cai et al., 2026), Motus (Bi et al., 2026), and LingBot-VA (Li et al., 2026b). Their details are summarized in Table 1 and Appendix A. We additionally report Fast-WAM (Yuan et al., 2026) as the supplementary experiments to examine a complementary setting in which the policy is trained directly on downstream task data rather than adapted from an embodied pretrained policy.

Benchmarks. We evaluate on RoboTwin (Chen et al., 2025) and LIBERO (Liu et al., 2023). RoboTwin evaluates bimanual manipulation under both Easy and Hard settings, while LIBERO covers language-conditioned manipulation across the Spatial, Object, Goal, and Long suites. In addition, we test $\pi_{0.5}$ on RoboCasa365 (Nasiriany et al., 2026) and MolmoSpaces (Deshpande et al., 2026). We also include five real-world manipulation tasks on an AgiBot G1 robot. Benchmark-specific task definitions and evaluation details are provided with the corresponding experiments.

Sparse Adaptation. For each model and downstream setting in the study, we start from a dense checkpoint obtained either from an official release or through our own baseline training. Unless otherwise stated, Dense denotes direct evaluation of this checkpoint. The sparse variants are initialized from the corresponding dense checkpoints and continue fine-tuning under their respective masks, which remain fixed at inference. The training data, objective, and fine-tuning configuration are virtually the same as those of their dense baselines. This setup evaluates whether sparse adaptation can attain performance comparable to the dense reference. Model-specific mask constructions and training configurations are detailed with the corresponding experiments and in Appendices B and D.

Evaluation. We use task success rate as the primary metric. Within each model and evaluation setting, dense and sparse variants are evaluated on matched task sets under identical conditions, with benchmark-specific episode counts and aggregation rules reported alongside the results.

3.2 Random Sparsification Without Performance-Based Search

We first test whether substantial expert connectivity can be removed without searching for a favorable sparse topology. Starting from each dense checkpoint, we randomly sparsify its native expert

Table 3: Fusion sparsification on LIBERO. Entries are task-averaged success rates (%) over matched task sets, with 50 episodes per included task. The random mask is reused across all four suites.

Model	Variant	Spatial	Object	Goal	Long	Average
$\pi_{0.5}$	Dense	98.6	99.4	97.6	91.4	96.75
	Random	99.2	98.8	98.4	92.6	**97.25** (+0.50)
	Early-half fusion	98.6	97.4	97.4	92.8	96.55 (−0.20)
InternVLA-A1	Dense	94.0	99.6	94.2	90.0	**94.45**
	Random	91.4	99.6	91.8	92.6	93.85 (−0.60)
	Early-half fusion	88.0	97.6	94.0	84.8	91.10 (−3.35)
Motus	Dense	98.8	99.2	94.6	94.8	**96.85**
	Random	99.4	98.6	95.8	90.6	96.10 (−0.75)
	Early-half fusion	99.0	98.4	92.8	94.4	96.15 (−0.70)
LingBot-VA	Dense	98.4	98.2	96.4	96.0	**97.25**
	Random	95.8	96.8	94.0	90.6	94.30 (−2.95)
	Early-half fusion	97.6	99.2	97.6	93.6	97.00 (−0.25)

Table 4: Fusion sparsification of $\pi_{0.5}$ on RoboCasa365 Atomic-Seen. Success rates (%) are reported for all 18 tasks, with 50 episodes per task. Source-to-destination labels denote pick-and-place tasks.

Variant	Close blender lid	Close fridge	Close toaster oven door	Coffee mug setup	Navigate kitchen	Open cabinet	Open drawer	Open stand mixer head	Counter to cabinet	Counter to stove	Drawer to counter	Sink to counter	Toaster to counter	Slide dishwasher rack	Turn off stove	Turn on kettle	Turn on microwave	Turn on faucet	**Avg. (18)**
Dense	4	44	14	30	2	40	52	58	66	70	46	84	34	70	18	20	10	52	**39.67**
Random	0	64	4	16	0	40	74	24	56	62	48	78	38	72	8	42	10	32	37.11
Early-half fusion	10	54	8	24	0	50	30	56	66	64	40	78	54	58	22	26	6	40	38.11

connections and fine-tune the resulting policy under the fixed sparse structure.

$$\rho_i = \frac{1}{|\Omega_i|} \sum_{(\ell,j)\in\Omega_i} m^{(\ell)}_{e_i \to e_j}. \tag{4}$$

This quantity measures the fraction of native within- and cross-expert connections retained for query expert e_i. The dense policy has $\rho_i = 1$ for all experts. We construct random masks independently for each query stream and retain half of its feasible native connections, i.e., $\rho_i = 0.5$. This avoids concentrating sparsification on a particular expert and provides a common retention budget across architectures. All masks satisfy the native attention constraints defined in Sec. 2. For autoregressive LingBot-VA, removing an expert-level connection also removes access to both current-chunk and cached historical keys from that expert. Model-specific constructions are detailed in Appendix B. We fine-tune and evaluate all four models on RoboTwin and LIBERO, with additional experiments on RoboCasa365 and MolmoSpaces for $\pi_{0.5}$. On RoboTwin, we evaluate three independently sampled masks per model at the same retention ratio to assess sensitivity to the particular sparse topology. For the remaining benchmarks, we use one fixed random mask per model-setting pair. All masks are sampled before fine-tuning and are not selected based on downstream performance.

3.3 Result Analysis

Substantial Expert Connectivity Can Be Removed with Limited Performance Loss. On RoboTwin (Table 2), success rates averaged over three masks and both evaluation settings differ from the dense references by -0.14, -1.59, $+0.53$, and -3.83 percentage points for $\pi_{0.5}$, InternVLA-A1, Motus, and LingBot-VA, respectively. On LIBERO (Table 3), the corresponding changes are $+0.50$, -0.60, -0.75, and -2.95 percentage points. Additional evaluations of $\pi_{0.5}$ show the same trend:

Table 5: Fusion sparsification of $\pi_{0.5}$-based policies on five MolmoSpaces evaluation configurations. Entries are success rates (%), with 100 episodes per configuration. Bold marks the highest average.

Variant	Pick-MSProc	Pick-Classic	Pick&Place	P&P-NextTo	P&P-Color	Avg.
Dense	77	35	40	29	46	**45.40**
Random	82	40	36	30	33	44.20
Early-half fusion	80	34	42	23	47	45.20

random sparsification achieves 37.11% on RoboCasa365 (Table 4) and 44.20% on MolmoSpaces (Table 5), close to dense references. These results show that removing half of the expert connectivity can largely preserve the performance across architectures and task settings after adaptation.

No Particular Sparse Connectivity Pattern Is Indispensable. The RoboTwin experiments further show that different random masks with the same retention budget achieve similar performance, with at most 1.31% difference in setting-averaged success rates. Despite retaining different subsets of expert interactions, these masks converge to comparable downstream performance after adaptation. This suggests that effective sparse adaptation does not depend on a particular set of expert connections, revealing substantial distributed redundancy in the dense connectivity structure.

4 Structured Pruning Expert Connectivity

The previous experiments show that effective sparse adaptation does not depend on a particular expert-connectivity pattern. We therefore ask whether this flexibility can be converted into a structured pruning scheme that preserves downstream performance while simplifying deployment. To this end, we introduce a depth-wise pruning pattern that concentrates cross-expert fusion in the lower half of the network and removes it from upper layers.

4.1 Pruning MoT with Early-Half Fusion for Efficiency

Early-Half Fusion. We consider pruning with early-half fusion, which preserves all native expert connectivity in the first half of the Transformer layers and restricts the remaining layers to within-expert attention. Let $h = \lfloor L/2 \rfloor$ denote the last fusion layer. For each native attention block $(\ell, i, j) \in \Omega$, the expert-level mask is

$$m^{(\ell)}_{e_i \to e_j} = \begin{cases} 1, & \ell \le h \text{ or } i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

This structured mask concentrates cross-expert communication in the lower half of the network, while the upper half retains only within-expert computation. In contrast to random sparsification, which removes connections at scattered layers, early-half fusion eliminates cross-expert dependencies from a contiguous upper-layer region. This contiguous structure is particularly useful because once an expert is no longer consumed by any retained upper-layer computation, its corresponding computation or cached states can be removed from deployment. For each model, we initialize from the same dense checkpoint used in the random sparsification experiments and continue fine-tuning with the early-half mask fixed throughout training and inference. Although early-half fusion does not exactly match the query-wise retention ratios of random sparsification, both remove a comparable fraction of cross-expert connectivity in our settings.

Deployment Simplification. Early-half fusion enables two forms of deployment simplification, depending on the underlying MoT architecture. For policies with separate conditioning experts, action-only inference no longer requires the upper layers of a conditioning expert once its representations cease to participate in cross-expert fusion. When these layers do not share parameters with retained components, their weights can therefore be excluded from deployment. This allows us to truncate the Und stack in $\pi_{0.5}$, both the Und and Gen stacks in InternVLA-A1, and the Und expert in Motus, while retaining the full depth of diffusion experts needed for iterative denoising. For autoregressive LingBot-VA, the shared Video-Action backbone must retain all layers for current-chunk

Table 6: Deployment efficiency of early-half fusion. Params denotes parameters loaded for inference. Prefill measures conditioning computation and is included in total latency; Total reports the median time per action chunk. Speedup is computed from total latency relative to Dense. Peak memory denotes peak allocated GPU memory. Historical KV denotes persistent cross-chunk cache storage at a matched history length. Dash (–) indicates not applicable.

Model	Variant	Params	Latency (ms)		Speedup	Peak memory	Historical KV
		(B)	Prefill	Total		(GiB)	(GiB)
$\pi_{0.5}$	Dense	3.62	163.7	338.4	1.00×	13.76	–
	Early-half fusion	2.63	96.0	260.1	1.30×	10.05	–
InternVLA-A1	Dense	3.22	90.9	351.0	1.00×	12.25	–
	Early-half fusion	2.30	63.3	310.5	1.13×	8.78	–
LingBot-VA	Dense	5.79	627	7344	1.00×	20.63	6.72
	Early-half fusion	5.79	557	5187	1.42×	17.26	3.36

computation, so expert-layer truncation is not applicable. Instead, we remove cross-chunk KV access in the upper half of the backbone: upper-layer queries attend only to same-expert tokens from the current chunk, while the lower half retains the native expert connectivity and temporal access. Historical KV states are therefore required only for the lower half of the network, reducing persistent historical KV storage by half under equal per-layer cache sizes. Exact layer configurations, retained components, and model-specific implementations are provided in Appendices B and C.

4.2 Result Analysis

Early-Half Fusion Preserves Comparable Downstream Performance. Across all evaluated architectures, early-half fusion remains close to the original dense policies. On RoboTwin (Table 2), the setting-averaged success rates differ from the dense references by -1.75, -0.97, -1.07, and $+1.36$ percentage points for $\pi_{0.5}$, InternVLA-A1, Motus, and LingBot-VA, respectively. On LIBERO (Table 3), the corresponding changes are -0.20, -3.35, -0.70, and -0.25 percentage points. Additional evaluations of $\pi_{0.5}$ show the same trend: early-half fusion achieves 38.11% on RoboCasa365 Atomic-Seen (Table 4) and 45.20% on MolmoSpaces (Table 5), compared with dense references of 39.67% and 45.40%, respectively. These results show that cross-expert connectivity need not be maintained throughout the full network depth; concentrating fusion in the first half is sufficient to preserve downstream performance across the evaluated models and settings.

Structured Pruning Reduces Deployment Cost and Improves Inference Efficiency. We next measure the deployment benefits enabled by the structured early-half design. Using matched full-depth and pruned implementations on a single NVIDIA L40S (48 GiB), early-half fusion reduces GPU-resident parameters from 3.62B to 2.63B for $\pi_{0.5}$ and from 3.22B to 2.30B for InternVLA-A1, while achieving per-chunk speedups of $1.30\times$ and $1.13\times$, respectively. For LingBot-VA, pruning upper-layer historical dependencies reduces persistent KV storage from 6.72 to 3.36 GiB and peak allocated GPU memory from 20.63 to 17.26 GiB, while reducing median per-chunk latency from 7344 to 5187 ms ($1.42\times$). Measurements use matched model inputs and generation settings with eager PyTorch, and detailed implementation and measurement protocols are provided in Appendix C.1.

5 Further Analysis

The preceding sections show that connectivity sparsification can be accommodated through downstream adaptation and can be organized into a deployment-friendly structure. We further test it on more aggressive sparsification, non-pretrained methods, and real-world manipulation.

5.1 Fusion Is Redundant but Not Dispensable

The strong performance of early-half fusion raises a natural question: how far can expert fusion be reduced before downstream performance begins to degrade? We study this trade-off on InternVLA-

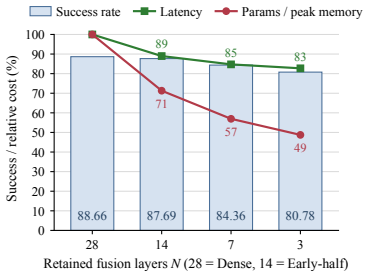

Figure 2: Performance-efficiency trade-off with reduced fusion depth in InternVLA-A1.

Table 7: Fusion sparsification results of Fast-WAM on LIBERO. Entries are task-averaged success rates (%) within each task suite, with 50 episodes per included task.

Variant	Spatial	Object	Goal	Long	Avg.
Dense	97.8	100.0	96.4	93.0	96.8
Random	96.2	97.6	94.6	89.2	94.4
Early-half fusion	97.4	99.8	96.8	94.4	97.1

Table 8: Real-world task successes out of 20 trials of InternVLA-A1 across five real robot tasks.

Variant	Pour Coffee	Bake Cake	Pick Bill	Place Bill	Change Table Cloth
Dense	20/20	19/20	14/20	14/20	11/20
Random	17/20	15/20	13/20	11/20	10/20
Early-half fusion	20/20	19/20	12/20	13/20	9/20

A1 by progressively restricting cross-expert communication to the first $N \in \{14, 7, 3\}$ of its 28 Transformer layers, using the same structured pruning as early-half fusion. As shown in Fig. 2, retaining fusion in 14 layers reduces the RoboTwin average success rate only from 88.66% to 87.69%. More aggressive sparsification, however, progressively lowers performance to 84.36% with 7 fusion layers and 80.78% with only 3. Meanwhile, deployed parameters decrease to approximately 71%, 57%, and 49% of Dense, while inference latency decreases more gradually to approximately 89%, 85%, and 83%, respectively. These results reveal a clear performance-efficiency boundary: substantial expert fusion is redundant, but increasingly aggressive sparsification eventually incurs meaningful performance loss while substantial savings are achieved in deployed parameter count and GPU memory, together with lower inference latency. Details are in Appendix E.

5.2 Sparse Fusion Remains Effective Without Embodied Policy Pretraining

To examine whether sparse expert connectivity relies on embodied-policy pretraining, we additionally evaluate Fast-WAM (Yuan et al., 2026) on LIBERO. Unlike the pretrained robot policies studied above, Fast-WAM is trained directly on downstream robot data while leveraging pretrained video-model components. We apply the same sparsification framework: random sparsification retains half of the Action-query attention blocks across depth ($\rho_{\text{Action}} = 0.5$), with both diagonal and cross-expert blocks participating, while early-half fusion preserves Video-Action fusion in the first 15 of 30 layers and restricts the remaining layers to within-expert attention. Dense, Random, and Early-half variants achieve four-suite average success rates of 96.8%, 94.4%, and 97.1%, respectively (Table 7). The comparable performance of the sparse variants suggests that the flexibility of expert connectivity is not limited to pretrained robot policies.

5.3 Real-World Experiments

To test whether sparse expert fusion transfers beyond simulation, we deploy InternVLA-A1 on an AgiBot G1 dual-arm robot across five living-room manipulation tasks: Pour Coffee, Bake Cake, Pick Bill, Place Bill, and Change Table Cloth. Policy inference runs on a single NVIDIA RTX 5090, and each variant is evaluated for 20 trials per task without human intervention; a trial is counted as successful if the task goal is achieved at the end of the episode (training and evaluation details in Appendix F). As shown in Table 8, Early-half fusion matches the dense policy on Pour Coffee and Bake Cake and loses only one to two trials on each of the remaining three tasks, while Random remains within a few trials of Dense on every task. It indicates that the structured early-half pattern preserves most of the dense performance on real hardware.

Figure 3: **Case study of Place Bill task.** From left to right, the robot first pulls the drawer open, places the bill inside, and then pushes the drawer closed. Red circles indicate the bill location.



6 Related Work

Mixture-of-Transformers Architecture. MoT couples modality- or function-specific Transformer streams through shared attention (Liang et al., 2024), a design closely related to MMDiT (Esser et al., 2024). LMFusion and BAGEL combine specialized streams for multimodal understanding and generation (Shi et al., 2026; Deng et al., 2025). Applications of this broader architectural family span text-image-audio (Li et al., 2025a), video-sound (Shan et al., 2025), video generation (Bian et al., 2026), and omnimodal world modeling (Agarwal et al., 2026). Further applications include 3D generation (Chen et al., 2026a; Lin et al., 2026; Yang et al., 2026c), human-motion modeling (Li et al., 2026e), and time-series forecasting (Wang et al., 2026b). HBridge further pretrains cross-expert model through intermediate-layer bridges (Wang et al., 2026a).

Robot Foundation Policies. Robot foundation policies connect pretrained representations to actions through token prediction (Brohan et al., 2023; Kim et al., 2024) or dedicated diffusion and flow networks (Liu et al., 2025; Li et al., 2024; Bjorck et al., 2025; Black et al., 2024; Intelligence et al., 2025). InternVLA-A1 combines understanding, generation, and action experts (Cai et al., 2026), while world-action policies incorporate future visual prediction or video priors (Cen et al., 2025; Bi et al., 2026; Li et al., 2026b; Ye et al., 2026; Kim et al., 2026). Recent work also explores intermediate features or learned feature-to-action interfaces (Reuss et al., 2025; Ma et al., 2026b; Wang et al., 2026c). Selective fusion also appears in MotuBrain's middle-layer interactions (Team et al., 2026) and SparseMoT's strided video reads with cross-layer KV aggregation (Zhao et al., 2026).

Efficient Inference and Structural Sparsification of Robot Policies. Robot-policy efficiency has been improved through compact backbones (Wen et al., 2025; Shukor et al., 2025), quantization (Wang et al., 2025b; Xu et al., 2026b), token reduction (Jiang et al., 2025; Li et al., 2026d; Wang et al., 2025a), temporal reuse (Xu et al., 2026a; Tan et al., 2025; Li et al., 2025b), and faster action decoding (Pertsch et al., 2025; Kim et al., 2025; Song et al., 2025). Structural approaches use dynamic layer execution (Yue et al., 2024; Zhang et al., 2026; Yang et al., 2026b), pruning or distillation (Jeon et al., 2026; Jabbour et al., 2025; Yang et al., 2026a), and post-pruning fine-tuning (Chen et al., 2026b; Nguyen et al., 2026; Sun et al., 2026). WAM-specific methods additionally reduce future-video generation costs or action-module depth (Yuan et al., 2026; Ma et al., 2026a; Li et al., 2026a), or retain KV projections from removed language blocks (Sun et al., 2026).

7 Conclusion

We studied whether dense expert connectivity must be preserved throughout network depth when adapting pretrained Mixture-of-Transformers robot policy models to downstream tasks. Across multiple representative architectures and robot manipulation benchmarks, our experiments showed that substantial portions of expert connectivity can be removed during fine-tuning while largely preserving task performance. Moreover, different sparse masks with matched budgets achieved similar results, suggesting that successful adaptation does not depend on a particular fixed subset of expert connections. Building on this observation, we introduced early-half fusion, which concentrates cross-expert connectivity in the first half of the network and applies structured pruning to the remaining layers. This simple pattern maintained comparable downstream performance while enabling practical deployment simplifications, including truncation of unused conditioning layers and reduction of historical KV-cache storage in autoregressive policies. Deployment measurements further demonstrated reductions in loaded parameters, GPU memory, and inference latency.

AI USE STATEMENT

We used generative AI tools, including Claude, ChatGPT, and Codex, for language editing, manuscript organization, LaTeX formatting, literature navigation, and code generation and debugging. All AI-assisted code was manually reviewed and tested by the authors before use in experiments. All AI-assisted text, technical claims, citations, and references in the manuscript were also reviewed and, where applicable, verified against the original sources and experimental results. Generative AI was not used as a substitute for experimental validation or scientific judgment. The authors take full responsibility for the research methodology, experimental results, analysis, conclusions, and final manuscript.

ETHICS STATEMENT

This work studies the efficiency and expert-connectivity structure of existing robot foundation policies. Our experiments do not involve human subjects or the collection of personal or sensitive data. Evaluations are conducted primarily on established simulation benchmarks, with additional real-world robot manipulation experiments performed in a controlled research environment. Real-world experiments are limited to standard manipulation tasks and are conducted with appropriate supervision and hardware safety precautions.

REPRODUCIBILITY STATEMENT

We provide the model architectures, sparsification formulation, training and evaluation settings, and benchmark protocols needed to reproduce the main experiments in the paper and appendix. Model-specific random and structured mask constructions are described in Appendix B, and the training and evaluation configuration of every experiment (checkpoint sources, initialization, fine-tuning budgets, optimizer settings, and reported checkpoints) in Appendix D. Additional ablations supporting the analysis of fusion depth are also reported. We use publicly available robot-policy checkpoints and established benchmarks whenever applicable, and report the corresponding evaluation settings and episode counts with each experiment. We plan to release our implementation, sparsification configurations, and supporting experimental data to facilitate reproduction of the reported results.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Yuxuan Bian, Xin Chen, Zenan Li, Tiancheng Zhi, Shen Sang, Linjie Luo, and Qiang Xu. Video-as-prompt: Unified semantic control for video generation. In *International Conference on Learning Representations*, volume 2026, pp. 114558–114583, 2026.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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

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.

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

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.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7220–7232, 2026a.

Yuxuan Chen, Yixin Han, Yize Huang, and Xiao Li. Rlrc: Reinforcement learning-based recovery for compressed vision-language-action models. *IEEE Robotics and Automation Letters*, 2026b.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Abhay Deshpande, Maya Guru, Rose Hendrix, Snehal Jauhri, Ainaz Eftekhar, Rohun Tripathi, Max Argus, Jordi Salvador, Haoquan Fang, Matthew Wallingford, et al. Molmob0t: Large-scale simulation enables zero-shot manipulation. *arXiv preprint arXiv:2603.16861*, 2026.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

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

Jason Jabbour, Dong-Ki Kim, Max Smith, Jay Patrikar, Radhika Ghosal, Youhui Wang, Ali Agha, Vijay Janapa Reddi, and Shayegan Omidshafiei. Don't run with scissors: Pruning breaks vla models but they can be recovered. *arXiv preprint arXiv:2510.08464*, 2025.

Boseong Jeon, Yunho Choi, and Taehan Kim. Shallow-{\pi}: Knowledge distillation for flow-based vlas. *arXiv preprint arXiv:2601.20262*, 2026.

Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

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

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

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Jiajun Li, Tiecheng Guo, Yifan Ye, Rongyu Zhang, Xiaowei Chi, Qianpu Sun, Ying Li, Yunfan Lou, Yan Huang, Zhihe Lu, et al. Efficient-wam: A 1b-parameter world-action model with low-cost future imagination. *arXiv preprint arXiv:2606.10040*, 2026a.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026b.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Shalfun Li, Victor Yao, Charles Yang, Truth Qu, Regis Cheng, Ryan Yu, Howard Lu, Newton Von, Vincent Chen, Yohann Tang, et al. Wall-wm: Carving world action modeling at the event joints. *arXiv preprint arXiv:2606.01955*, 2026c.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Zichun Liao, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. Omniflow: Any-to-any generation with multi-modal rectified flows. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13178–13188. IEEE, 2025a.

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

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. In *The Fourteenth International Conference on Learning Representations*, 2025b.

Zekun Li, Sizhe An, Chengcheng Tang, Chuan Guo, Ivan Shugurov, Linguang Zhang, Amy Zhao, Srinath Sridhar, Lingling Tao, and Abhay Mittal. Llamo: Scaling pretrained language models for unified motion understanding and generation with continuous autoregressive tokens. *arXiv preprint arXiv:2602.12370*, 2026e.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Siyou Lin, Zhou Xue, Hongwen Zhang, Liang An, Dongping Li, Shaohui Jiao, and Yebin Liu. Mix3r: Mixing feed-forward reconstruction and generative 3d priors for joint multi-view aligned 3d reconstruction and pose estimation. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–12, 2026.

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

Songming Liu, 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*, volume 2025, pp. 29982–30009, 2025.

Hao Luo, Ye Wang, Wanpeng Zhang, Sipeng Zheng, Ziheng Xi, Chaoyi Xu, Haiweng Xu, Haoqi Yuan, Chi Zhang, Yiqing Wang, et al. Being-h0.5: Scaling human-centric robot learning for cross-embodiment generalization. *arXiv preprint arXiv:2601.12993*, 2026.

Liheng Ma, Rui Heng Yang, Zhanguang Zhang, Mateo Clemente, Ziwen Hu, Tongtong Cao, and Yingxue Zhang. Faster-wam: Do world action models need deep action modules? *arXiv preprint arXiv:2608.02365*, 2026a.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026b.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

Gia-Binh Nguyen, Trong-Bao Ho, Thien-Loc Ha, Khoa Vo, Philip Lund Møller, Quang T Nguyen, Long Dinh, Tung M Luu, Tuan Dam, Vu Duong, et al. Finetuning vision-language-action models requires fewer layers than you think. *arXiv preprint arXiv:2606.20246*, 2026.

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

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies. *arXiv preprint arXiv:2509.04996*, 2025.

Sizhe Shan, Qiulin Li, Yutao Cui, Miles Yang, Yuehai Wang, Qun Yang, Jin Zhou, and Zhao Zhong. Hunyuanvideo-foley: Multimodal diffusion with representation alignment for high-fidelity foley audio generation. *arXiv preprint arXiv:2508.16930*, 2025.

Weijia Shi, Xiaochuang Han, Chunting Zhou, Weixin Liang, Xi Lin, Luke Zettlemoyer, and Lili Yu. Lmfusion: Adapting pretrained language models for multimodal generation. *Advances in Neural Information Processing Systems*, 38:9163–9182, 2026.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Zhijun Li, Donglin Wang, Lujia Wang, et al. Pd-vla: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13162–13169. IEEE, 2025.

Guoheng Sun, Kaixi Feng, Shwai He, Xiaochuan Gong, Yexiao He, Ziyao Wang, Zheyu Shen, Wanghao Ye, Ramana Rao Kompella, Gaowen Liu, et al. Drop-then-recovery: How redundant are vision-language-action models? *arXiv preprint arXiv:2606.27755*, 2026.

Xudong Tan, Yaoxin Yang, Peng Ye, Jialin Zheng, Bizhe Bai, Xinyi Wang, Jia Hao, and Tao Chen. Think twice, act once: Token-aware compression and action reuse for efficient inference in vision-language-action models. *arXiv preprint arXiv:2505.21200*, 2025.

MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, Hongzhe Bi, James Li, Jiabao Liu, Jingrui Pang, Kiro Jing, et al. Motubrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025a.

Hongyu Wang, Chuyan Xiong, Ruiping Wang, and Xilin Chen. Bitvla: 1-bit vision-language-action models for robotics manipulation. *arXiv preprint arXiv:2506.07530*, 2025b.

Xiang Wang, Zhifei Zhang, He Zhang, Zhe Lin, Yuqian Zhou, Qing Liu, Shiwei Zhang, Yijun Li, Shaoteng Liu, Haitian Zheng, et al. Hbridge: H-shape bridging of heterogeneous experts for unified multimodal understanding and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14767–14778, 2026a.

Xinyue Wang, Jiacheng Pang, Kun Zhou, Kexin Zhang, Defu Cao, Fan Feng, Faisal, Songyao Jin, Yan Liu, and Biwei Huang. Timebraid: Unifying time series and language for understanding and forecasting, 2026b. URL https://arxiv.org/abs/2609.29792.

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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 40, pp. 18638–18646, 2026c.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *Advances in Neural Information Processing Systems*, 38:164448–164473, 2026a.

Yuhao Xu, Yantai Yang, Zhenyang Fan, Yufan Liu, Yuming Li, Bing Li, and Zhipeng Zhang. Qvla: Not all channels are equal in vision-language-action model's quantization. *arXiv preprint arXiv:2602.03782*, 2026b.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40891–40914, 2026a.

Zebin Yang, Yijiahao Qi, Tong Xie, Bo Yu, Shaoshan Liu, and Meng Li. Dysl-vla: Efficient vision-language-action model inference via dynamic-static layer-skipping for robot manipulation. *arXiv preprint arXiv:2602.22896*, 2026b.

Zongyuan Yang, Mingjing Yi, Wanli Ma, Chenzhuo Fan, Bocheng Li, Baolin Liu, Yuke Lou, Yingde Song, Yongping Xiong, Zhengdong Guo, et al. Eva01: Unified native 3d understanding and generation via mixture-of-transformers. *arXiv preprint arXiv:2605.16745*, 2026c.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

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, 2026.

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-wam: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026.

APPENDIX CONTENTS

A Architectural Variants **15**

B Sparsification Details **15**

B.1 $\pi_{0.5}$ 16

B.2 InternVLA-A1 16

B.3 Motus 16

B.4 LingBot-VA 16

C Deployment Details **17**

C.1 Deployment Measurement Protocol 17

D Training and Evaluation Details **18**

E Progressive Reduction of Fusion Depth **20**

F Real-World Experiment Details **21**

A ARCHITECTURAL VARIANTS

Recent robot foundation policies instantiate MoT design by combining an action diffusion expert with other pretrained experts for action generation, while usually differing along several architectural axes. First, *vision-language-action (VLA) models* (Black et al., 2024; Intelligence et al., 2025; Cai et al., 2026; Luo et al., 2026) emphasize pretrained vision-language representations for semantic understanding, whereas *world-action models (WAM)* (Bi et al., 2026; Li et al., 2026b; Ye et al., 2026; Li et al., 2026c; Team et al., 2026) explicitly incorporate world models for future-state video prediction. Second, *joint denoising* (Bi et al., 2026; Ye et al., 2026; Li et al., 2026c; Team et al., 2026) simultaneously denoises multiple diffusion experts, whereas *causal denoising* (Intelligence et al., 2025; Cai et al., 2026; Li et al., 2026b; Luo et al., 2026) conditions action denoising on upstream representations computed beforehand, with directed information flow enforced by the native attention mask. Third, *autoregressive (AR) denoising* (Li et al., 2026b; Ye et al., 2026; Team et al., 2026) conditions the current chunk on cached keys and values from preceding chunks, whereas *non-autoregressive (Non-AR) denoising* (Black et al., 2024; Intelligence et al., 2025; Cai et al., 2026; Bi et al., 2026; Luo et al., 2026) operates without such cross-chunk caching. We additionally consider shared-backbone variants, in which a single Transformer backbone processes multiple expert token groups (Li et al., 2026b; Ye et al., 2026). These architectural choices are partially coupled. We primarily study four representative policies, $\pi_{0.5}$ (Intelligence et al., 2025), InternVLA-A1 (Cai et al., 2026), Motus (Bi et al., 2026), and LingBot-VA (Li et al., 2026b), which span distinct combinations of these choices, as summarized in Table 1.

B SPARSIFICATION DETAILS

This section describes how the random and structured masks in Sec. 3.2 and Sec. 4 are instantiated for each model. Sparsification only removes native attention paths and never introduces connections prohibited by the original architecture. All masks remain fixed throughout fine-tuning and inference and must leave every query token with at least one accessible key after all masking.

B.1 $\pi_{0.5}$

$\pi_{0.5}$ combines a vision-language understanding stream (Und) with a flow-matching Action expert. Its native attention mask permits Und queries to attend only to Und keys, while Action queries can attend to both Und and Action keys.

Random Sparsification. We leave Und unchanged ($\rho_{\mathrm{Und}} = 1$), since removing its only accessible key-expert block would violate the validity constraint. At each layer, the Action query stream randomly retains exactly one of its two key-expert blocks, Und or Action, yielding $\rho_{\mathrm{Action}} = 0.5$. Both the cross-expert Und-key block and the within-expert Action-key block participate in sampling. Native attention constraints within the retained blocks remain unchanged.

Early-half Fusion. We retain the native connectivity in the first half of layers. In the second half, Action queries attend only to Action keys, while Und queries continue to attend only to Und keys. Thus, Action-to-Und attention is restricted to the first half, and within-expert attention remains available throughout.

B.2 InternVLA-A1

InternVLA-A1 contains understanding (Und), generation (Gen), and Action streams. Und and Gen use non-diffusion vision-language components, whereas Action is a diffusion expert. Its native connectivity is causal: Und queries attend only to Und keys, Gen queries can attend to Und and Gen keys, and Action queries can attend to Und, Gen, and Action keys.

Random Sparsification. We leave Und unchanged ($\rho_{\mathrm{Und}} = 1$), since removing its only accessible key-expert block would violate the validity constraint. We construct separate masks for the Gen and Action query streams over their layer-query-key attention blocks, retaining exactly half of each stream's native blocks across depth ($\rho_{\mathrm{Gen}} = \rho_{\mathrm{Action}} = 0.5$). Both diagonal within-expert and off-diagonal cross-expert blocks participate in sampling, subject to the validity constraint. The retention budget is enforced across depth rather than separately within each layer, and native causal masking remains unchanged.

Early-half Fusion. We preserve all native connections in the first half of layers. In the second half, Gen queries attend only to Gen keys and Action queries attend only to Action keys; Und remains unchanged throughout. Consequently, Gen-to-Und and Action-to-Und/Gen attention occurs only in the first half, while all within-expert blocks are retained.

B.3 Motus

Motus contains an Und expert operating on upstream VLM features and two diffusion experts, Video and Action. Video and Action are jointly denoised, while all three streams participate in joint attention: each query expert can attend to Und, Video, and Action keys.

Random Sparsification. For each query expert, we construct a separate mask over its layer-indexed Und-, Video-, and Action-key blocks. Both diagonal within-expert and off-diagonal cross-expert blocks participate in sampling. Each query stream retains exactly half of its native attention blocks across depth ($\rho_i = 0.5$ for all three experts), subject to the validity constraint. As in InternVLA-A1, the budget is enforced across depth rather than separately within each layer.

Early-half Fusion. We retain full three-expert connectivity in the first half of layers. In the second half, all off-diagonal blocks are removed, leaving only Und-to-Und, Video-to-Video, and Action-to-Action attention. These upper layers therefore introduce no further cross-expert attention.

B.4 LingBot-VA

LingBot-VA contains Video and Action streams that share a diffusion backbone. Video is generated before Action, and both streams may attend to KV states from preceding chunks according to the native temporal mask. Current Action queries can additionally attend to the current predicted video.

Random Sparsification. For each layer and query expert, we randomly retain exactly one of the two key-expert categories, Video or Action. The two query streams use separate masks, yielding $\rho_{\text{Video}} = \rho_{\text{Action}} = 0.5$. Dropping a key-expert category removes all otherwise accessible keys from that expert for the corresponding query stream at that layer, including historical and current-chunk keys when applicable.

The masking decision depends only on expert identity, not on temporal distance, frame identity, or cache window. Native temporal masking within each retained category remains unchanged. A category excluded for one query stream may still be accessed by the other; its historical KV cache must remain available whenever a retained computation requires it.

Early-half Fusion. We preserve native Video–Action connectivity and cross-chunk KV access in the first half of the shared backbone. In the second half, each query attends only to same-expert tokens within the current chunk: Video queries retain current Video keys, and Action queries retain current Action keys.

All upper-half historical KV access is disabled, including access to same-expert history. This additional temporal restriction eliminates the need to maintain a persistent historical cache for those layers. Native attention constraints within the retained current-chunk blocks remain unchanged.

C Deployment Details

Table 9 summarizes the layer and cache retention configurations enabled by early-half fusion. Conditioning-layer truncation applies to action-only deployment, where the omitted representations and final outputs are no longer consumed by the retained inference procedure. Layers required for other outputs or subsequent generation steps are not assumed removable.

Conditioning-layer Truncation. For $\pi_{0.5}$, the Und stack can be truncated after the last fusion layer, while the upstream visual encoder and input-processing components remain unchanged. For InternVLA-A1, both Und and Gen can be truncated under the sparse action-only execution path. Their layer parameters are separate, so the unused upper-layer weights of both stacks can be excluded from the deployed model.

For Motus, the truncated Und expert operates on features extracted by an upstream VLM; the VLM itself remains unchanged. The Video and Action diffusion experts retain their full depth because their upper-layer outputs produce the denoising updates used in subsequent iterations. Thus, the simplification removes unused conditioning computation without shortening the diffusion experts.

Historical KV-cache Retention. LingBot-VA's full shared backbone remains active for current-chunk computation. Its lower half retains native historical KV access, whereas its upper half attends only to same-expert tokens within the current chunk. Persistent historical KV is therefore maintained only for the lower half, while current-chunk keys and values remain available throughout the backbone. At a fixed history length, batch size, and numerical precision, equal-sized per-layer caches imply a 50% reduction in persistent historical KV storage.

C.1 Deployment Measurement Protocol

Hardware and Execution. Measurements are performed on an eight-GPU NVIDIA L40S host using a single 48 GiB GPU per run, with NVIDIA driver version 580.178. The policy batch size is one; LingBot-VA's classifier-free guidance internally uses a video-denoising batch size of two. We use FP32 for $\pi_{0.5}$ and InternVLA-A1 and BF16 for Motus and LingBot-VA. Within each model, both variants use matched input shapes, action horizons, and denoising schedules.

Inputs and Latency. Timing inputs consist of fixed-seed synthetic images and zero-valued proprioceptive states, providing reproducible workloads with matched tensor shapes. Equivalence between the masked and structurally pruned implementations is checked separately using real inputs.

We measure model-side latency for generating a complete action chunk from device-resident inputs, including the visual encoding, conditioning, and denoising computations required by the

Table 9: Layer and cache retention enabled by early-half fusion during action-only deployment. Arrows indicate original-to-retained expert Transformer depth. For LingBot-VA, the backbone remains intact; only persistent historical KV storage is restricted to the first half of layers.

Model / configuration	**Native expert depth**	**Early-half simplification**	**Retained at full depth**
$\pi_{0.5}$ Standard OpenPI configuration	Und: 18 Action: 18	Und: $18 \rightarrow 9$	Action: 18 layers
InternVLA-A1-3B	Und: 28 Gen: 28 Action: 28	Und: $28 \rightarrow 14$ Gen: $28 \rightarrow 14$	Action: 28 layers
Motus Wan2.2-5B configuration	Und: 30 Video: 30 Action: 30	Und: $30 \rightarrow 15$	Video: 30 layers Action: 30 layers
LingBot-VA Released shared-backbone variant	Video and Action: shared 30-layer backbone	Historical KV cache: layers 1–15 only	Shared backbone: all 30 layers

model. Model loading, one-time compilation, data transfer, and environment interaction are excluded. CUDA synchronization is performed at the timing boundaries.

For $\pi_{0.5}$ and InternVLA-A1, we report the median of 30 timed calls after 5 warm-up calls. For Motus, we report the median of 8 timed calls after 2 warm-up calls. LingBot-VA is evaluated through repeated autoregressive rollouts under a saturated attention window.

Memory and Parameter Accounting. Parameter counts include all GPU-resident model parameters, including frozen components, with shared parameters counted once. The T5 encoders used by Motus and LingBot-VA reside on the CPU and are excluded from both GPU-resident parameter counts and GPU-memory measurements.

Peak allocated GPU memory is measured using `torch.cuda.max_memory_allocated`; it does not include the CUDA context. Persistent historical KV storage is reported separately from current-chunk conditioning caches and temporary activations. It is already included in the GPU allocation and must not be added to the peak-memory figure.

Compiled Execution. Table 6 reports eager execution. Since the released $\pi_{0.5}$ serving code compiles the policy with `torch.compile` whereas InternVLA-A1 runs eagerly by default, we also measure both models with the action-sampling function compiled (`mode="max-autotune"`, FP32 with `float32_matmul_precision="highest"`), keeping the hardware, inputs, checkpoints, and timing protocol unchanged. As Table 10 shows, compilation lowers the per-chunk latency of every configuration by 1.6–$2.4\times$, and the advantage of early-half fusion grows, from $1.30\times$ to $1.50\times$ for $\pi_{0.5}$ and from $1.13\times$ to $1.19\times$ for InternVLA-A1. Compilation removes most of the kernel-launch overhead of the denoising loop, so the conditioning stage, which early-half fusion halves, becomes a larger share of the total (70% for $\pi_{0.5}$ and 43% for InternVLA-A1). Compiled and eager actions agree to within 4.8×10^{-7} for $\pi_{0.5}$ and 1.9×10^{-6} for InternVLA-A1 (with the output of its image tokenizer held fixed), peak memory changes by less than 0.3 GiB, and a second run reproduces all values within 3%. Enabling TF32 matrix multiplication, the default of the upstream compile path, reduces latency further ($101.8/77.1$ ms for $\pi_{0.5}$ and $95.5/83.0$ ms for InternVLA-A1) at the cost of 10^{-3}-level deviations in the actions; we therefore report FP32.

D Training and Evaluation Details

Table 11 lists, for every experiment, the dense checkpoint, the initialization and fine-tuning budget of the sparse variants, and the checkpoint whose results are reported. Optimizer and numerical settings follow the official recipe of each model and are summarized in Table 12. Masks are applied

Table 10: Per-chunk latency (ms) under `torch.compile` on the L40S host of Table 6 (FP32, batch size 1, median of 30 calls after warm-up). Eager totals repeat Table 6. The conditioning/denoising split is measured with the two stages compiled separately and therefore does not sum exactly to Total. Speedup is relative to Dense.

		Eager	Compiled			
Model	Variant	Total	Conditioning	Denoising	Total	Speedup
$\pi_{0.5}$	Dense	338.4	149.7	55.2	210.0	1.00×
	Early-half fusion	260.1	81.6	50.8	139.9	1.50×
InternVLA-A1	Dense	351.0	68.7	85.0	155.5	1.00×
	Early-half fusion	310.5	51.2	78.6	131.2	1.19×

inside attention and are never folded into the weights; the same mask is used during fine-tuning and evaluation.

Protocol. For each model and benchmark we fix a dense checkpoint: a released checkpoint adapted to the benchmark when one exists, and otherwise one that we train with the official recipe. Dense entries evaluate this checkpoint as is. Each sparse variant is initialized from its weights with a freshly initialized optimizer and schedule, and is fine-tuned on the same data with the same objective and hyperparameters, with the mask fixed throughout; all fine-tuning is full-parameter. Sparse variants therefore receive additional fine-tuning relative to the dense checkpoint they start from. On RoboTwin, LIBERO, and the real robot we report a checkpoint fixed in advance, namely the final checkpoint of the sparse schedule or the same intermediate step for all sparse variants; on RoboCasa365 and MolmoSpaces we evaluate every 5k steps and report the best checkpoint of each variant.

Table 11: Training configuration of every experiment. "Dense checkpoint" is the checkpoint evaluated as Dense; "Sparse init." is the initialization of the sparse variants; "Steps (reported)" gives the sparse fine-tuning budget and, in parentheses, the reported step (R: Random, E: Early-half fusion). Batch sizes are global, in samples per optimizer update.

Model	Benchmark	Dense checkpoint	Sparse init.	Steps (reported)	Batch	LR schedule
$\pi_{0.5}$	RoboTwin	Ours: $\pi_{0.5}$-base → 200k steps	Dense (200k)	200k (200k)	128	5×10^{-5}
$\pi_{0.5}$	LIBERO	Released `pi05_libero`	Released ckpt	30k (30k)	256	5×10^{-5} constant, warm-up 10k
$\pi_{0.5}$	RoboCasa365	Released RoboCasa365 checkpoint	Released ckpt	75k (R 40k, E 65k)	64	$2.5\times10^{-5} \rightarrow 2.5\times10^{-6}$, cosine over 100k, warm-up 1k
$\pi_{0.5}$	MolmoSpaces	Released DeMiAn mix5	Released ckpt	R 60k, E 100k (R 55k, E 65k)	256	5×10^{-5} constant, warm-up 1k
InternVLA-A1	RoboTwin	Released `InternVLA-A1-3B-RoboTwin`	Released ckpt	200k (200k)	128	$5\times10^{-5} \rightarrow 5\times10^{-6}$, cosine, warm-up 2k
InternVLA-A1	LIBERO	Ours: `InternVLA-A1-3B` → 100k schedule, reported at 40k	Dense (40k)	60k of a 100k schedule (50k)	128	as RoboTwin, cosine over 100k
InternVLA-A1	Real robot	Ours: `InternVLA-A1-3B` → 60k schedule, deployed at 40k	Dense (40k)	60k (40k)	128	as RoboTwin, cosine over 60k
Motus	RoboTwin	Released `Motus_robotwin2`	Released ckpt	40k (40k)	256	5×10^{-5}, warm-up 200, ≈constant
Motus	LIBERO	Ours: Stage-2 Motus → 30k steps	Dense (15k)	15k (15k)	256	as RoboTwin
LingBot-VA	RoboTwin	Released `posttrain-robotwin`	Released ckpt	50k (50k)	64	1×10^{-5} constant, warm-up 10
LingBot-VA	LIBERO	Ours: `lingbot-va-base` → 5k steps per suite, reported at 4k	Long: released LIBERO-Long ckpt; other suites: Dense (4k)	5k (4k)	120	as RoboTwin
Fast-WAM	LIBERO	Released `libero_uncond_2cam224`	Released ckpt	10 epochs, 21.7k (final)	128	1×10^{-4}, cosine

$\pi_{0.5}$. We use the PyTorch trainer of OpenPI with bf16 autocast and fp32 master weights. On RoboTwin, where no released checkpoint exists, the dense policy is trained from $\pi_{0.5}$-base on the

Table 12: Optimizer and numerical settings, shared by all benchmarks of a model unless noted. "fp32 master" denotes bf16 compute with fp32 master weights; "bf16" denotes pure bf16 parameters, as in the corresponding release.

Model	Optimizer	(β_1, β_2)	Weight decay	Grad. clip	Precision	EMA
$\pi_{0.5}$	AdamW	(0.9, 0.95)	10^{-10}	1.0	fp32 master	0.999 (LIBERO only)
InternVLA-A1	AdamW	(0.9, 0.95)	0.01	1.0	bf16	–
Motus	AdamW	release default	0.01	–	ZeRO-1, fp32 master	–
LingBot-VA	AdamW	(0.9, 0.95)	0.1	2.0	FSDP, fp32 master	–
Fast-WAM	AdamW	release default	0.01	1.0	bf16 mixed precision	–

27,500-episode RoboTwin 2.0 set for 200k steps, and the sparse variants continue from its final weights under the identical schedule; actions are executed in chunks of 15 of the 50 predicted steps. On LIBERO, RoboCasa365, and MolmoSpaces the sparse variants continue from the released `pi05_libero`, RoboCasa365, and DeMiAn mix5 checkpoints with the corresponding released recipes (EMA 0.999 on LIBERO; on RoboCasa365 the released 100k-step cosine schedule, stopped at 75k).

InternVLA-A1. All runs use the released LeRobot-based trainer in bf16. On RoboTwin every sparse variant, including LayerCut-7 and LayerCut-3 of Appendix E, is fine-tuned from the released `InternVLA-A1-3B-RoboTwin` checkpoint for 200k steps. On LIBERO the dense policy is trained from `InternVLA-A1-3B` with a 100k-step schedule and reported at 40k steps; the sparse variants start from these weights with a fresh 100k-step schedule and are reported after 50k steps. The real-robot setting is described in Appendix F.

Motus. We use the released DeepSpeed ZeRO-1 trainer with the VLM frozen. On RoboTwin the sparse variants are fine-tuned from the released Stage-3 `Motus_robotwin2` checkpoint for 40k updates at an effective batch of 256. On LIBERO the dense policy is trained from the Stage-2 Motus release for 30k updates; the sparse variants start from its 15k weights and are fine-tuned for a further 15k updates, so that all variants receive 30k updates in total.

LingBot-VA. We use the released FSDP trainer. On RoboTwin the sparse variants are fine-tuned from the released `posttrain-robotwin` checkpoint for 50k steps. On LIBERO each suite is trained separately for 5k steps at an effective batch of 120 and reported at 4k steps, following the released LIBERO-Long recipe; the dense policies are trained from `lingbot-va-base`, and our LIBERO-Long reproduction matches the released checkpoint at 96.0%. The sparse variants start from the released LIBERO-Long checkpoint on Long and from our dense checkpoint on the other suites, for which no released checkpoint exists.

Fast-WAM. Both sparse variants are fine-tuned from the released `libero_uncond_2cam224` checkpoint with the released LIBERO recipe (10 epochs, cosine learning rate 10^{-4}), and the final checkpoint is reported.

E Progressive Reduction of Fusion Depth

This section provides the detailed ablations underlying Sec. 5.1. All layer-cut variants follow the same structured rule as early-half fusion: native expert connectivity is retained in the first N Transformer layers, while subsequent layers preserve only within-expert attention. We evaluate $N \in \{14, 7, 3\}$ in addition to the 28-layer dense model.

Performance remains close to Dense when fusion depth is reduced from 28 to 14 layers, but degrades increasingly under more aggressive cuts. In particular, as shown in Table 13, reducing fusion depth from 14 to 7 layers incurs an additional 3.33 percentage-point drop, and reducing it further from 7 to 3 layers incurs another 3.58 points. This monotonic trend supports the conclusion that expert fusion is substantially redundant, but not fully dispensable.

Table 13: Progressive fusion-depth ablations on InternVLA-A1. N denotes the number of Transformer layers that retain native cross-expert connectivity. Avg. equally weights the Easy and Hard RoboTwin settings; Δ denotes the change from Dense in percentage points.

Variant	N	Easy	Hard	Avg.	Δ
Dense	28	88.52	88.80	88.66	0.00
LayerCut-14	14	87.74	87.64	87.69	-0.97
LayerCut-7	7	84.24	84.48	84.36	-4.30
LayerCut-3	3	81.06	80.50	80.78	-7.88

F Real-World Experiment Details

Platform. The AgiBot G1 is a dual-arm mobile manipulator; in our experiments the base, torso, and head are fixed and the policy controls the two 7-DoF arms and their grippers. Actions are 16-D absolute joint targets (seven joint angles and one gripper command per arm), where the gripper command is a continuous value rather than a binary open/close signal. Observations consist of three RGB cameras-a head camera (800×1280) and one wrist camera per arm (480×848)-captured at 30 Hz, together with the current 16-D joint state. Images are resized to 224×224 with padding, and the policy receives two frames per camera: the current frame and the frame 15 steps (0.5 s) earlier.

Data. We collect teleoperated demonstrations of the five tasks in a living-room environment, 2,496 episodes (1.09M frames, about 10 hours) in total; Table 14 lists the instruction and the number of demonstrations for each task. Bake Cake demonstrations use two instruction variants that differ only in the description of the cake. Actions and states are normalized with per-dimension mean and standard deviation computed over the training data. Because the number of demonstrations differs by more than an order of magnitude across tasks, training samples each task with probability proportional to the square root of its frame count.

Table 14: Real-world tasks, language instructions, and number of teleoperated demonstrations.

Task	Instruction	Episodes
Pour Coffee	Pour the coffee beans from the bag into the coffee machine's bean hopper.	1,398
Bake Cake	Put the cake with white icing into the air-fryer basket and return the basket.	116
Pick Bill	Pick up the bill from the sofa.	223
Place Bill	Put the bill in the coffee table drawer.	168
Change Table Cloth	Spread the clean tablecloth on the coffee table.	591

Training. All variants share the same configuration: global batch size 128, AdamW (β_1=0.9, β_2=0.95, weight decay 0.01), a cosine learning-rate schedule with 2k warm-up steps decaying from 5×10^{-5} to 5×10^{-6} over 60k steps, bf16 precision, absolute joint-position actions, and an action chunk of 50 steps. The dense policy is initialized from the released InternVLA-A1-3B checkpoint and fine-tuned on the demonstrations above. The Random and Early-half fusion variants are initialized from the weights of the dense policy, with the optimizer and learning-rate schedule re-initialized, and are fine-tuned under their respective masks and otherwise identical settings. Every run follows the 60k-step schedule.

Deployment and evaluation. The policy runs as a server on a single NVIDIA RTX 5090, and the attention masks of the sparse variants are applied at inference exactly as during fine-tuning. The robot queries the policy at 3 Hz and executes the returned 50-step chunks in a receding-horizon manner, committing to at most 30 steps of a chunk and blending over 10 steps when switching to a newer chunk; all controller settings are identical across variants. Each variant is evaluated for 20 trials per task. Object placements are varied across trials and are not paired across variants. A trial is counted as a success if the task goal is achieved at the end of the episode, and no human intervention is applied during a trial.