Wall-OSS-0.5 Technical Report

Pretrain Once, Act Anywhere

X Square Robot Team

Large-scale Vision-Language-Action (VLA) pretraining is increasingly adopted as the foundation for robot policies, yet the evidence for pretrained VLAs is almost invariably reported after task-specific fine-tuning. This leaves a foundational question unanswered: does VLA pretraining itself yield executable robot behavior, or does it merely furnish a better initialization for downstream policy learning? We present **Wall-OSS-0.5**, an open-source 4B VLA built upon a 3B VLM backbone augmented with action-generation components, designed so that pretrained robotic capability is directly measurable on physical hardware. The model is pretrained across more than 20 embodiments, processing over one million robot trajectories per epoch alongside a grounded multimodal corpus. We adopt a *gradient-bridged co-training* recipe in which three objectives play distinct and complementary roles: discrete action prediction routes strong VLM-native gradients into the backbone, multimodal prediction preserves grounded vision-language understanding, and continuous flow matching serves as the deployment-time action interface. *Before task-specific fine-tuning*, the pretrained checkpoint achieves non-trivial zero-shot real-robot behavior, completing several tasks—including a held-out deformable manipulation task—at high task progress on a 17-task suite. *After fine-tuning*, the same checkpoint serves as a markedly stronger adaptation prior, reaching 60.5% average task progress on 15 real-robot tasks and outperforming $\pi_{0.5}$ (1) by **17.5%**. Multimodal evaluations further confirm that action training does not erode grounded vision-language competence: the model preserves broad vision-language ability while strengthening embodied grounding. Together, these results reposition VLA pretraining from an initialization strategy to a directly testable—and already useful—source of robot capability.

Code: https://github.com/X-Square-Robot/wall-x

1 Introduction

Foundation models become credible when pretraining yields capabilities that are observable prior to task-specific adaptation. In language and vision, this standard is by now routine: a pretrained model is expected to answer questions, follow instructions, and transfer to novel tasks. In robotics, however, the evidence remains less direct. Vision-Language-Action (VLA) models (2, 3, 4, 1, 5, 6, 7, 8, 9) increasingly inherit perception and reasoning from pretrained VLMs, yet their strongest results are typically reported only after downstream fine-tuning. This raises a concrete operational question: *does VLA pretraining alone produce an executable policy under real-robot evaluation?*

We introduce **Wall-OSS-0.5**, an open-source 4B VLA built upon a 3B VLM backbone augmented with action-generation components, and organized around precisely this deployment-oriented criterion. The pretrained checkpoint is evaluated directly as a real-robot policy, rather than merely as an initialization. This imposes three requirements on the model: it must execute useful manipulation skills out of the box, retain sufficient VLM-derived vision-language competence to remain instruction-grounded, and furnish priors that render downstream adaptation more sample-efficient. We refer to this objective as *deployment-oriented VLA pretraining*.

The central empirical finding is that pretraining alone already produces measurable robot behavior, as shown in Figure 1. On a 17-task real-robot zero-shot suite spanning semantic, rigid-object, deformable-object, fine-grained, and long-horizon manipulation, the pretrained checkpoint completes several tasks with high task progress in the absence of any task-specific fine-tuning: Block Sorting (100%), Fruit Sorting (96%), Ring Stacking (86%), and the held-out deformable task Rope Tightening (82%). Held-out tasks track the

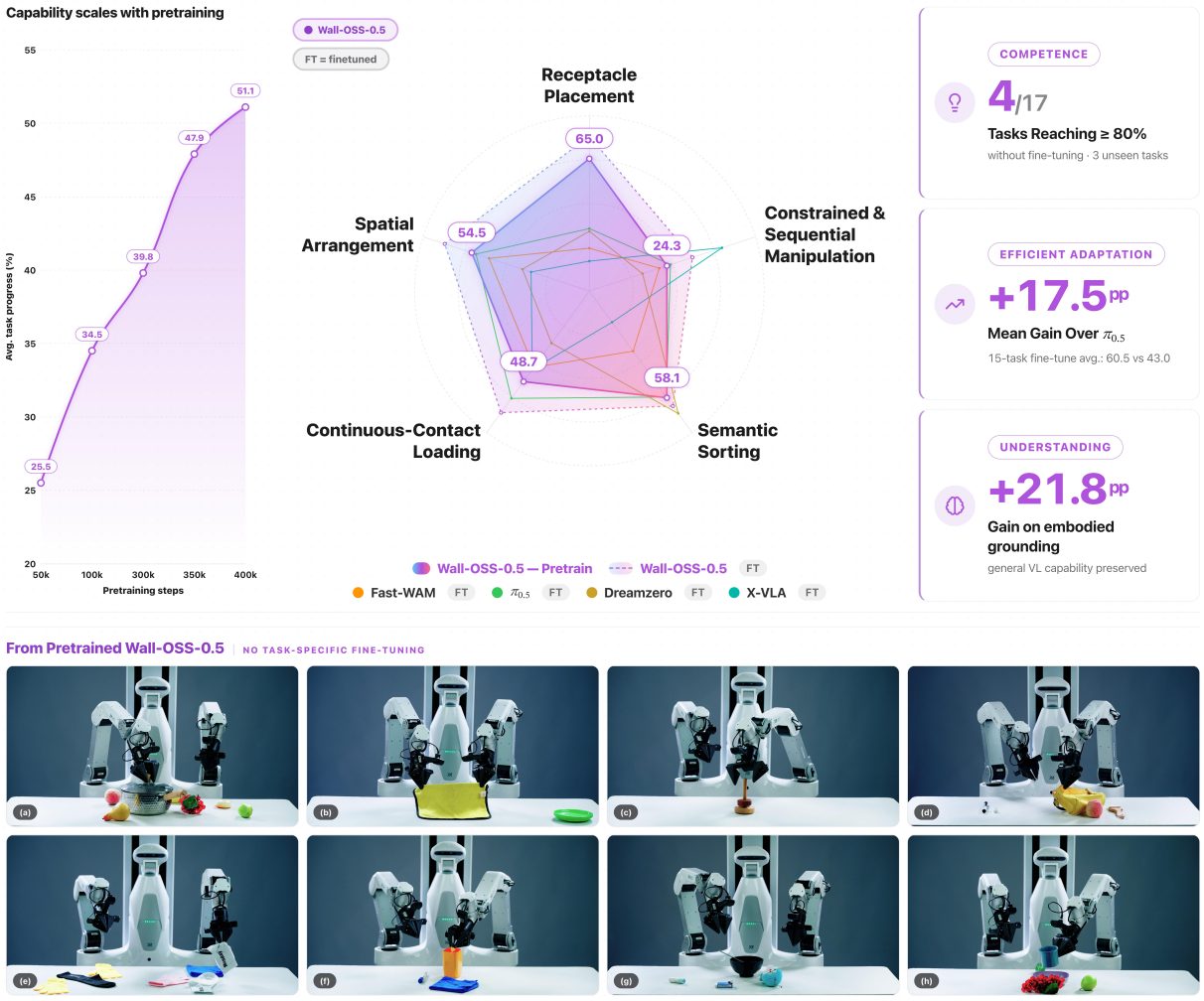


Figure 1 Wall-OSS-0.5 capability overview. The figure summarizes Wall-OSS-0.5 along three axes—pretrained model's real-robot behavior, downstream adaptation, and embodied multimodal understanding. Panels (a)–(h) show examples of the evaluated real-robot tasks and their instructions: **(a)** open the pot lid; **(b)** cover the blocks; **(c)** stack the ring; **(d)** put the pen into the bag; **(e)** pair the socks; **(f)** insert the screwdriver; **(g)** put the spoon into the bowl; **(h)** put the cup onto the plate. Notably, when deployed directly without any task-specific fine-tuning, the pretrained model already executes many of these tasks at high task progress. *Throughout, pp denotes percentage points.*

learning curves of seen tasks throughout pretraining, indicating that the model acquires reusable manipulation capability rather than merely memorizing task templates. We also find that the model continues to serve as a stronger prior after fine-tuning: on a 15-task real-robot suite, Wall-OSS-0.5 attains an average task progress of 60.5%, surpassing $\pi_{0.5}$ (1) by 17.5% and widening the margin to **26%** on the 10-task manipulation subset. Multimodal evaluation further reveals stable overall performance alongside a **21.8%** gain on embodied grounding, indicating that action training strengthens embodied perception without eroding general vision-language competence.

The core methodological contribution underlying these results is **gradient-bridged co-training**, which proves decisive for real-robot performance. The starting point is the tension inherent to VLA training. Continuous flow matching constitutes the natural execution interface, as it models unquantized robot actions directly; yet on its own, it only weakly updates the pretrained VLM backbone. Discrete action-token prediction exhibits the complementary property: next-token cross-entropy is native to the VLM training interface and strongly shapes

the backbone, but decoded discrete actions are too coarse for precise control. Freezing or truncating gradients preserves the VLM prior, but at the cost of preventing precise action targets from shaping the large pretrained backbone. The design problem is therefore not one of "continuous versus discrete," but rather how to leverage the discrete path during training while retaining continuous actions at deployment.

Wall-OSS-0.5 resolves this tension through gradient-bridged co-training. During pretraining, action-token cross-entropy supplies the gradient bridge: a strong, VLM-native action signal that updates the backbone in a direction aligned with continuous control. In strength, it drives backbone updates far more effectively than flow matching, by virtue of sharing the autoregressive interface of VLM pretraining; in alignment, its gradient direction remains positively correlated with that of flow matching, shaping features that continuous control can subsequently exploit. Multimodal cross-entropy on grounded vision-language data provides the anchor, tying the backbone to instruction following, visual grounding, and embodied scene understanding; its update direction is largely orthogonal to action optimization, and therefore complements the bridge rather than competing with it. Flow matching, in turn, trains the continuous action generator used at inference. Gradient analysis confirms why all three components are necessary: beyond the early training stage, flow matching contributes only a small though persistent share of the backbone update, while the dominant updates originate from the two cross-entropy losses. Ablating the gradient bridge, isolating gradients, or delaying co-training substantially degrades downstream behavior (Section 5.1).

This recipe is instantiated through three design choices. First, a **Mixture-of-Transformers (MoT) backbone** separates a *VL Expert* and an *Action Expert* at every layer. Vision, language, and discrete action tokens are routed through the VL Expert, whereas continuous action signals are routed through the Action Expert. Crucially, gradients still flow end-to-end across both experts, so the split amounts to a routing decomposition rather than gradient isolation. Second, a **Vision-Aligned RVQ Action Tokenizer** replaces rule-based FAST tokenization (10), rendering the discrete tokens a more semantic training interface for the VLM backbone. Third, **Action-Space Supervision** strengthens flow matching by imposing the loss directly in the raw action space, rather than on the velocity field as in standard flow matching, thereby accelerating convergence and stabilizing continuous action generation.

Wall-OSS-0.5 is trained in a single stage on a three-source mixture: high-quality self-collected manipulation data, a curated set of open-source multi-embodiment trajectories, and a 90M-sample multimodal corpus that includes embodied bridge samples synthesized from action trajectories. This mixture mirrors the training design: robot trajectories teach execution, multimodal samples preserve grounded understanding, and embodied bridge samples connect the two.

Beyond the modeling recipe, we engineer a deployment-grade inference stack—combining CUDA Graph capture and custom fused kernels—that achieves a 4× end-to-end speedup over a PyTorch eager-mode baseline and sustains real-time control (15 Hz) at high input resolution.

We summarize our contributions as follows:

- **A pretrained VLA that can be evaluated directly on robots.** We release an open-source 4B VLA, built upon a 3B VLM backbone, whose pretrained checkpoint is evaluated on a 17-task real-robot zero-shot suite prior to any task-specific fine-tuning, with 15 Hz inference speed at high input resolution.
- **A practical co-training recipe for action-aware VLM backbones.** We identify action-token cross-entropy as the gradient bridge, multimodal cross-entropy as the grounding anchor, and flow matching as the continuous execution objective, and show that the three are jointly necessary.
- **An integrated architecture and training design.** We combine an MoT routing backbone, a vision-aligned RVQ action tokenizer, and Action-Space Supervision to instantiate this recipe at scale, while leaving the deployment-time action interface unchanged.
- **A multi-axis evaluation of pretrained and adapted capability.** We evaluate out-of-the-box real-robot behavior, post-finetuning real-robot performance and multimodal capability, demonstrating that strong pretraining yields both direct real-robot capability and improved downstream adaptation.

The remainder of this report is organized as follows: Section 2 describes the model and training recipe, Section 3

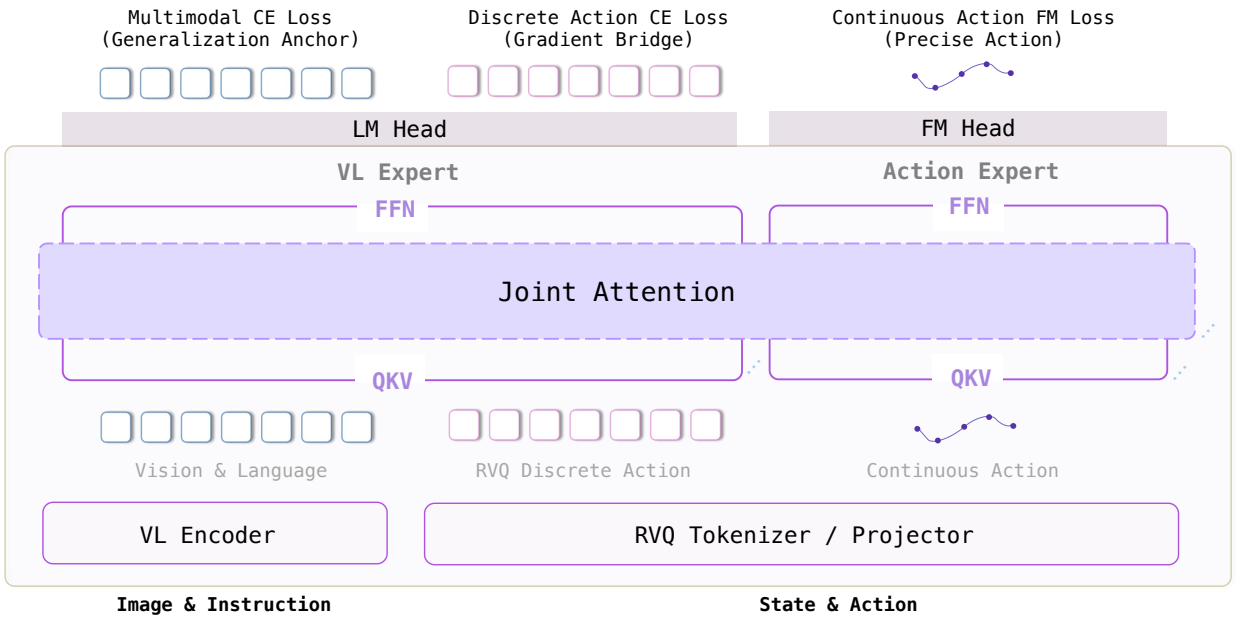


Figure 2 Gradient-bridged co-training and MoT routing in Wall-OSS-0.5. Three complementary objectives shape the pretrained policy: multimodal CE preserves grounded vision-language knowledge, action-token CE provides the gradient bridge that adapts the VLM backbone toward control, and flow matching supervises the continuous actions used for deployment. The Mixture-of-Transformers architecture routes vision-language tokens through the VL Expert and continuous-action computation through the Action Expert, with joint attention enabling end-to-end gradient flow between the two experts.

details the data pipeline, Section 4 presents the main experiment results, Section 5 validates the design choices, Section 6 discusses related work, and Section 7 concludes with limitations and future directions.

2 Methods

In Wall-OSS-0.5, pretraining is accordingly organized around three types of data modalities: discrete action tokens shape the VLM backbone during training, continuous flow matching supplies the action output at inference and grounded multimodal data preserves the vision-language prior. This section describes the architecture, tokenizer, training losses, and action interface used to realize this design in a single training stage.

2.1 Architecture

2.1.1 Backbone Routing

Wall-OSS-0.5 is initialized from Qwen2.5-VL-3B-Instruct (11) and extended with a Mixture-of-Transformers (MoT) backbone, yielding a model with over 4B parameters. The original 3B VLM is retained as the *VL Expert*, while the added *Action Expert*, together with action projections in continuous-action heads, provides the additional action-generation capacity. Four token streams—vision, language, proprioception, and discrete actions—are routed through the VL Expert, while noisy continuous action tokens are routed through the Action Expert, which is trained for flow-matching action generation.

This split constitutes a routing decomposition rather than a gradient-stopping mechanism. The two experts share the same sequence-level attention context, allowing the Action Expert to attend to visual and linguistic information when generating continuous actions. The attention mask renders discrete and continuous action tokens mutually invisible in the forward pass, which permits the two action pathways to be trained and

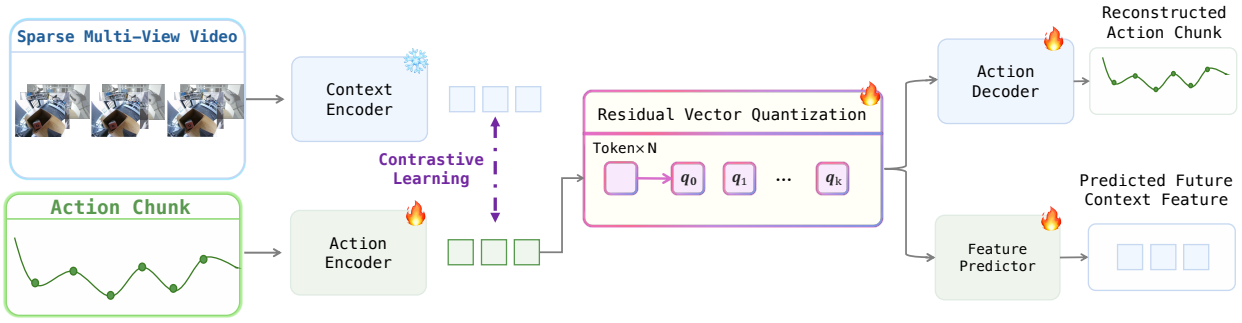


Figure 3 Overview of Vision-Aligned RVQ Action Tokenizer. Our framework compresses observation-conditioned delta action sequences into multi-level discrete tokens via residual vector quantization. By incorporating auxiliary visual-action and future-observation objectives, the tokenizer serves as a semantic training interface for the VLM backbone rather than a mere action compressor.

evaluated independently. In the meantime, gradients are not blocked from flow matching to VL Expert through shared attention.

2.1.2 Vision-Aligned RVQ Action Tokenizer

We adopt discrete action tokens because next-token cross-entropy is the training interface most directly compatible with the VLM backbone. The tokenizer must therefore expose structured action semantics for backbone training, not merely achieve low-distortion reconstruction. Accordingly, we replace the FAST tokenizer (10) with a learned Vision-Aligned Residual Vector Quantization (RVQ) Action Tokenizer, trained on heterogeneous robot data spanning diverse embodiments.

The tokenizer operates in the delta-action space and follows an Encoder–RVQ–Decoder structure, as shown in Figure 3. The encoder compresses observation-conditioned action chunks via temporal cross-attention; the RVQ codebooks capture coarse motion structure at early levels and fine residual corrections at later levels; and the decoder reconstructs action sequences conditioned on observation states. Beyond reconstruction, three objectives jointly shape the token space: visual-action alignment pulls action latents toward VLM visual features, next-frame prediction encourages tokens to encode action consequences, and DCT-domain reconstruction suppresses high-frequency jitter. The resulting discrete action representation is simultaneously reconstructable, visually aligned, and physically smooth.

2.2 Training Recipe

2.2.1 Gradient-Bridged Co-Training

During pretraining, we jointly optimize three objectives in a single stage. First, action-token cross-entropy predicts the RVQ action tokens autoregressively and serves as the gradient bridge between *VL Encoder* and *Action Expert*: a strong, VLM-native signal that renders the backbone action-aware. Second, multimodal cross-entropy loss trained on *VL Encoder* on grounded vision-language samples serves as the anchor that preserves instruction following, visual grounding, and embodied scene understanding. Third, flow matching trains the *Action Expert* to generate the continuous action chunks used at inference. We illustrate the model architecture of the gradient bridge in Figure 2. The discrete pathway is therefore primarily a training-time pathway, whereas the continuous pathway is the deployment-time pathway.

This design is motivated by the gradient dynamics analyzed in our ablations. Flow matching defines the execution objective, but beyond the early training stage its contribution to the VLM-backbone update stabilizes at a small but persistent share of roughly 5%, the dominant backbone updates instead originate from the two cross-entropy losses. Action-token cross-entropy is aligned with action generation and biases the backbone

toward control-relevant features, while multimodal cross-entropy anchors the update in the original vision-language prior. Accordingly, Wall-OSS-0.5 preserves end-to-end gradient flow rather than adopting the stop-gradient design of $\pi_{0.5}$ (1): the small flow-matching residual still matters for action quality, while action-token cross-entropy carries most of the backbone shaping.

In summary, Wall-OSS-0.5 is trained with the following composite objective:

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda_{\text{act}} \cdot \mathcal{L}_{\text{act-CE}} + \lambda_{\text{mm}} \cdot \mathcal{L}_{\text{mm-CE}}, \tag{1}$$

where $\mathcal{L}_{\text{act-CE}}$ denotes the autoregressive RVQ action-token loss, $\mathcal{L}_{\text{mm-CE}}$ denotes the multimodal next-token loss, and $\lambda_{\text{act}} = \lambda_{\text{mm}} = 0.01$. Empirically, $\mathcal{L}_{\text{flow}}$ is roughly two orders of magnitude smaller than the cross-entropy terms under action-space Supervision; the shared weight of 0.01 laces the cross-entropy losses on a scale comparable to the flow-matching loss, preventing language-style prediction from dominating action learning. The relative contribution of action-token and multimodal cross-entropy is in turn controlled by batch composition, with action and multimodal data mixed at a 9:1 ratio. At inference, decoding defaults to the continuous flow-matching pathway, which is decoupled from the discrete pathway by attention masking (Section 2.1.1); the discrete pathway's role is confined to carrying the gradient bridge during training.

2.2.2 Action-Space Supervision

Flow matching starts from a noisy action chunk and learns a velocity field that transports noise to the clean action. We use the linear Gaussian probability path We use the linear Gaussian probability path

$$\mathbf{A}_t^\tau = \tau \mathbf{A}_t + (1-\tau)\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2}$$

where $\tau = 0$ corresponds to pure noise and $\tau = 1$ to the clean action. Following π_0 (4), we bias timestep sampling toward the high-noise regime:

$$u \sim \text{Beta}(1.5, 1), \qquad \tau = s(1-u), \quad s = 0.999. \tag{3}$$

Since $\tau = 0$ corresponds to pure noise, this transformation concentrates probability mass on high-noise timesteps.

Robot action sequences are low-dimensional and smooth: their task-relevant structure resides primarily in the low-frequency trajectory shape rather than in high-frequency detail. We accordingly hypothesize that the high-noise regime is particularly critical for recovering the global shape of robot trajectories, whereas low-noise steps serve mainly to refine residual detail. Unlike natural images, in which both high- and low-frequency components carry rich semantics, robot actions tend to concentrate useful task structure in smooth trajectory trends; supervision quality in the high-noise regime therefore largely determines the generation ceiling. Guided by this intuition, we retain velocity prediction as the network output but define the loss on the recovered action:

$$\hat{A} = A^\tau + (1-\tau) \cdot f_\theta(A^\tau, \tau), \tag{4}$$

$$\mathcal{L}_A = \mathbb{E}_{\tau,\epsilon}\left[\left\|\hat{A} - A\right\|^2\right]. \tag{5}$$

This formulation is equivalent to a $(1-\tau)^2$-weighted loss in velocity space:

$$\mathcal{L}_A = \mathbb{E}_{\tau,\epsilon}\left[(1-\tau)^2 \left\|f_\theta(A^\tau, \tau) - (A - \epsilon)\right\|^2\right]. \tag{6}$$

The induced weighting emphasizes high-noise steps, at which the global action trajectory is formed, consistent with the trajectory-smoothness hypothesis above. This data-space formulation is related to x-prediction in diffusion models (12); here, however, the motivation derives from the spectral structure of robot actions rather than from variance considerations. The controlled ablation in Section 5.2 supports this design choice, demonstrating that action-space Supervision improves convergence speed, peak performance, and training stability.

2.2.3 Action Interface

The model follows a VLM-style conversation sequence. For action prediction, the input takes the form:

`[System]` Embodiment prompt `[User]` Observation: {camera}: ⟨image tokens⟩ ... Instruction: {task} Proprioception: ⟨proprio tokens⟩ `[Assistant]` ⟨action_ar_token⟩ ⟨action_flow_token⟩ $\times N$

Here ⟨action_ar_token⟩ denotes the discrete RVQ action tokens consumed by the autoregressive cross-entropy pathway, while ⟨action_flow_token⟩ denotes the continuous-action query tokens consumed by the flow-matching pathway; N is the number of frames in the predicted action horizon. For multimodal understanding samples, the output is text-only and contains no action tokens. The model supports an arbitrary number of camera views, sampled according to the data source. Each task is annotated with both a goal-level instruction (e.g., "tidy up the desk") and step-level sub-instructions (e.g., "throw it into the trash can"); during training, one granularity is sampled per step, allowing the model to follow instructions at multiple levels of abstraction. Paraphrases are additionally used to reduce overfitting to surface wording. Proprioception is discretized into text-numerical tokens and randomly dropped or perturbed during training, reducing dependence on noise-free robot state. At inference, discrete tokens are not decoded into executable actions; the Action Expert generates continuous action chunks through multi-step flow-matching denoising, with the two pathways kept decoupled in the forward pass by attention masking.

We adopt relative action representations and 6D rotations rather than Euler angles or quaternions, in order to avoid the discontinuities and ambiguities of SO(3). The resulting model action space is 26-dimensional: for each arm, a relative 3D position, a relative 6D rotation (both relative to the current end-effector pose), and a 1D gripper state (20D in total), plus a 3D mobile base velocity, a 1D lift height, and 2D head actuation. Both discrete and continuous pathways predict a one-second action horizon, with the number of frames adjusted to the control frequency of each data source.

2.2.4 Optimization and Configuration

We empirically find that Muon is effective in our co-training setup, yielding both faster convergence and improved training stability. Muon is applied to the 2D parameters of each expert, while AdamW handles the visual embeddings and LM head. Muon orthogonalizes the momentum prior to applying it as the update, producing spectrally normalized updates that are invariant to gradient magnitude. This invariance is critical in our setting, where the Action Expert and the VL Expert exhibit markedly different gradient scales.

To enable efficient Muon optimization at scale, we implement a distributed implementation (**DMuon**) that partitions the matrix-level Newton–Schulz computation across sharded parameters and reduces end-to-end overhead by up to 100× relative to a naive implementation.

Our pretraining uses an effective global batch size of 8192, bf16 mixed-precision training, gradient clipping at 1.0, a cosine learning-rate schedule with linear warmup, and a peak learning rate of 1×10^{-4}.

Images are resized with aspect ratio preserved and the longer side fixed at 448 pixels; stationary frames in all pretrained datasets are filtered out to prevent overfitting to idle actions. Fine-tuning uses a learning rate of 5×10^{-5} with all modules trainable, and retains the same joint discrete-plus-continuous objective; as shown in Section 5.1, this co-training setup yields a large relative gain over flow-only fine-tuning.

2.3 Inference Optimization

Real-time control is a defining requirement of VLA models deployed on physical robots. Unlike offline vision-language tasks, manipulation policies operate in closed loop with the environment, where every additional millisecond of inference latency translates into delayed actuation, degraded tracking of dynamic targets, and ultimately lower task success rates. A control frequency on the order of tens of Hertz is therefore a functional prerequisite rather than a performance luxury. This requirement is at odds with the architectural trend of modern VLAs, which couple a large vision encoder, a billion-scale language backbone, and an iterative action decoder into a single forward pass. The cost is further amplified when high-resolution visual input is required for fine-grained manipulation: ViT compute grows quadratically with the number of image tokens, and the resulting longer visual context inflates the key-value (KV) cache of the language backbone, lengthening

every attention operation downstream. Inference optimization for high-resolution VLA models is therefore substantially harder than for their low-resolution counterparts, and warrants dedicated treatment rather than off-the-shelf acceleration.

We optimize the inference pipeline of Wall-OSS-0.5 along two complementary axes, each targeting a dominant source of overhead identified through profiling. The denoising step is a memory-bound workload: GPU kernel execution time is shorter than the CPU launch latency, making CPU dispatch the bottleneck. The GPU consequently stalls between kernels, and the resulting bubbles—rather than compute—dominate end-to-end latency. Since the per-step computation graph is static, we capture the entire denoising step as a single CUDA Graph, which removes CPU dispatch from the critical path and eliminates these bubbles. Beyond GEMM and attention, the remaining operators (RoPE, RMSNorm, and similar elementwise and reduction patterns) are individually inexpensive but collectively incur substantial HBM traffic when executed as separate PyTorch ops, each of which materializes intermediate tensors. We fuse these operators into monolithic custom CUDA kernels that perform end-to-end computation in registers, thereby eliminating intermediate materialization and yielding 2–10× speedups over the native implementation.

Result. We evaluate the optimized inference stack on a single RTX 5090 with three-view image input at 224×224 and 448×448 resolutions. Our optimized implementation reaches approximately 21 Hz and 15 Hz at the two resolutions respectively with the standard denoising step $T = 10$ setting, corresponding to a 4× end-to-end speedup compared to a PyTorch eager-mode baseline. The relative speedup remains substantial at 448×448, where the baseline is most constrained by the quadratic cost of ViT and the inflated KV cache, confirming that the proposed optimizations are particularly effective in the high-resolution regime required by real-world manipulation. These numbers represent only a partial exploration of the optimization space; further gains are left to future work.

3 Data Recipe and Management

The pretraining dataset for Wall-OSS-0.5 is centered on diverse, high-quality, self-collected manipulation data, supplemented by open-source multi-embodiment data and augmented with targeted multimodal corpora. The composition is deliberately balanced across task variety, embodiment diversity, and source distribution, providing a solid foundation for the model to acquire robust manipulation capabilities through pretraining alone.

3.1 Action Data

Across self-collected and open-source action sources, we aim to unify the robot interface as much as platform diversity allows: action representations are mapped to a common semantic key set (bimanual end-effector poses, joint positions, gripper state, mobile base, lift/waist actuation, and head actuation), and camera streams are time-synchronized within each platform, though the camera number and viewpoint remain platform-specific. The cross-source preprocessing pipeline that implements this unification is detailed in Section 3.1.3.

3.1.1 Self-Collected Robot Manipulation Data

Our self-collected robot data forms the core of the pretraining manipulation corpus, covering thousands of independent tasks.

Embodiment composition. Our self-collected robot data spans two major platform categories—tabletop bimanual manipulation and mobile manipulation—respectively covering tabletop operations and mobile manipulation scenarios. We further complement these robot-platform data sources with an embodiment-free collection device.

Beyond teleoperated whole-robot data sources, we introduce XRZero-G0 (13), a proprietary embodiment-free device that enables low-cost data collection across diverse scenes without a fixed robot embodiment, enriching environmental and task diversity.

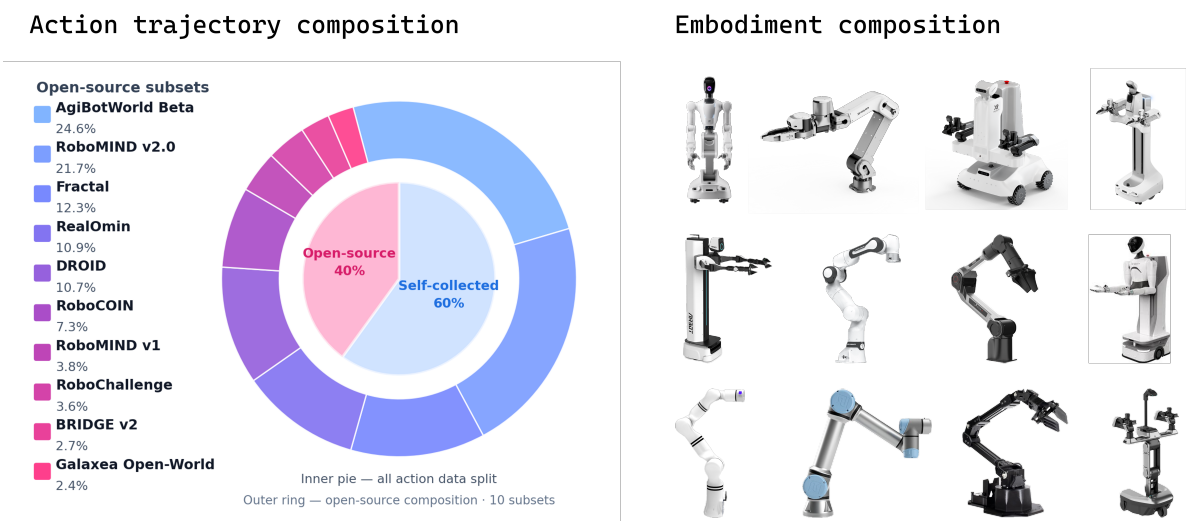


Figure 4 Action data landscape used for pretraining. Left: retained dataset subsets and trajectory composition, including self-collected action data, RoboMIND v1 (14), RoboMIND v2.0 (15), AgiBotWorld Beta (16), RoboCOIN (17), RoboChallenge (18), Galaxea Open-World (19), RealOmin (20), DROID (21), BRIDGE v2 (22), and Fractal/Google Robot (23). Right: embodiment composition summarizing the morphology-level diversity contributed by self-collected data and open-source subsets.

Scene and task distribution. The self-collected corpus is deliberately diverse in both scenes and tasks. Scenes span unstructured real-world environments (household, industrial, office), which target deployment realism with natural clutter and lighting variation, and controlled standardized collection rooms, which provide reproducibility and cross-batch alignment. Tasks are labeled along complementary axes—*manipulation complexity*, *trajectory duration*, and *special attributes* such as spatial reasoning or deformable-object interaction—and cover both industry-relevant operations and everyday behaviors. These task-level labels are also the unit in which we balance the training mixture (Section 3.1.3), so that a wide diversity is preserved rather than a few high-volume tasks dominating the data.

Task segmentation and language annotation. At the task level, each episode is annotated with both brief (goal-only) and detailed (step-by-step) instructions, then expanded into variants by large language models to diversify surface phrasing and reduce overfitting to any single wording. We additionally segment long teleoperated trajectories so that each segment maps to a single coherent sub-goal, producing instructions at a finer-than-episode granularity that supports cross-modal alignment and sub-goal modeling. Together, these two axes yield a large and varied instruction set—thousands of base instructions, tens of thousands after expansion—spanning diverse phrasings and granularities.

3.1.2 Open-Source Multi-Embodiment Data

Open-source manipulation data extends embodiment and scene coverage without requiring additional self-collected data collection. We perform unified curation on a selection of high-quality open-source datasets (format alignment, metadata verification, cross-source field mapping) and filter samples by timestamp continuity, action-observation plausibility, and frame anomalies. Figure 4 visualizes the 10 retained subsets and the resulting embodiment coverage, highlighting the broad scale and morphological diversity of the curated open-source corpus.

3.1.3 Data Preprocessing and Sampling Strategy

Multi-source robot data exhibits significant heterogeneity in quality, format, and action-space representation: different datasets use varying storage formats and field naming conventions, and even the same embodiment may have inconsistent action definitions across datasets (e.g., coordinate-frame orientation, rotation repre-

sentation conventions, gripper-state polarity). If left unaddressed, these differences directly introduce noise and degrade the effectiveness of joint training across sources. We therefore apply a systematic preprocessing pipeline to all datasets with action annotations.

Action space unification. We standardize each source into a unified action schema covering bimanual end-effector poses, joint positions, gripper states, mobile base motion, lift/waist actuation, and head motion. For datasets that provide only joint states, we recover end-effector poses through forward kinematics using the corresponding platform URDF. We further normalize the physical semantics of actions across embodiments: x points forward, y left, and z upward; the zero-rotation pose corresponds to a forward-facing gripper with a horizontal opening; and larger gripper values denote wider opening. When source annotations use Euler angles, we retain them only at the normalized source level and convert rotations to the 6D model representation before training to avoid discontinuities and gimbal lock.

Video–action temporal alignment. Different sensors often have inconsistent sampling frequencies and timestamp precision. We perform unified temporal alignment of video frames and action/state frames across all action data by timestamp. For cases where video and action frames cannot be strictly matched one-to-one, we use a nearest-timestamp strategy: for each video frame, the state frame with the closest timestamp is retrieved as the corresponding observation–action pair.

Erroneous data repair. Quality inspection further flags typical recording errors—such as swapped camera mappings or anomalous end-effector poses—which we correct where possible (e.g., remapping cameras or recovering poses via forward kinematics) and discard otherwise.

Stationary frame filtering. Near-stationary frames—identical observations paired with both near-zero and non-zero actions—introduce supervision noise and induce idle pauses at inference. We filter them by normalizing actions with global statistics and removing frames whose per-dimension difference from the last retained frame falls below a threshold. Beyond cleaner gradient signal, we observe visibly more compact task execution cadence post-training (reduced redundant pauses).

Data sampling strategy. Multi-source data is imbalanced at two levels: inter-source (trajectory counts differ by orders of magnitude between self-collected and open-source subsets) and intra-source (per-task long-tail within each dataset). Proportional sampling would let high-frequency tasks dominate each epoch's gradient and leave rare tasks underrepresented. We therefore define sampling groups using source and task labels, then apply power sampling within the resulting groups: for the i-th group with n_i trajectories, sampling weight $w_i = n_i^p$ and normalized count $s_i = N \cdot w_i / \sum_j w_j$, where N is the total trajectory budget per epoch. We set $p = 0.5$ (square-root sampling), boosting small-group frequency while preserving the statistical weight of large groups. A per-group cap with iterative reallocation prevents very large groups from monopolizing the budget. After sampling, one training epoch comprises over one million trajectories (~60% self-collected, ~40% open-source).

3.2 Multimodal Data

Multimodal data provides the generality anchor in gradient-bridged co-training: it carries the multimodal CE loss that keeps the VLM backbone tied to grounded vision-language understanding while action-token CE shapes it for control. Beyond generic vision-language alignment, the corpus supervises object recognition, scene composition, spatial relations, affordance, and task-relevant interaction cues, all of which help the model transfer to unseen scenes, object instances, and instructions. The multimodal corpus totals approximately 90 million samples, consisting of 78 million open-source samples and 12 million embodied bridge samples constructed directly from action trajectories. During training, action and multimodal samples are mixed at a 9:1 batch-sampling ratio: multimodal samples are trained with autoregressive next-token CE, while action trajectories jointly provide action-token CE and flow-matching supervision.

Open-source multimodal data. The open-source portion serves two complementary goals—maintaining general VLM capabilities and injecting embodiment-relevant understanding—and is instantiated as three dataset categories. *General vision-language data* provides broad supervision for captioning, general question answering, pointing, and open-ended reasoning; *embodied perception data* targets object grounding, affordance understanding, and spatial reasoning; and *embodied cognition data* covers task VQA, interaction understanding, and long-horizon reasoning. Table 1 lists representative datasets for each category. We apply strict quality control to open-source data, combining model-based verification with human annotation to filter inaccurate, low-information, and excessively repetitive samples.

Table 1 Open-source multimodal data organized by category.

Category	Task Types	Datasets
General vision-language	Captioning, VQA, pointing, reasoning	CAPSFUSION (24), Cambrian (25), PixMo-Cap (26), COCO (27), VQAv2 (28), PixMo-Point (26), OneThinker (29)
Embodied perception	Grounding, pointing, spatial, affordance	RoboPoint (30), SpaceThinker (31), OpenSpaces (32), SpaceOm (33), RefSpatial (34), CrossPoint (35), SenseNova-SI (36)
Embodied cognition	Task VQA, interaction, long-horizon reasoning	Robo2VLM (37), EO-Data (38), RoboVQA (39), Cosmos-Reason1 (40)

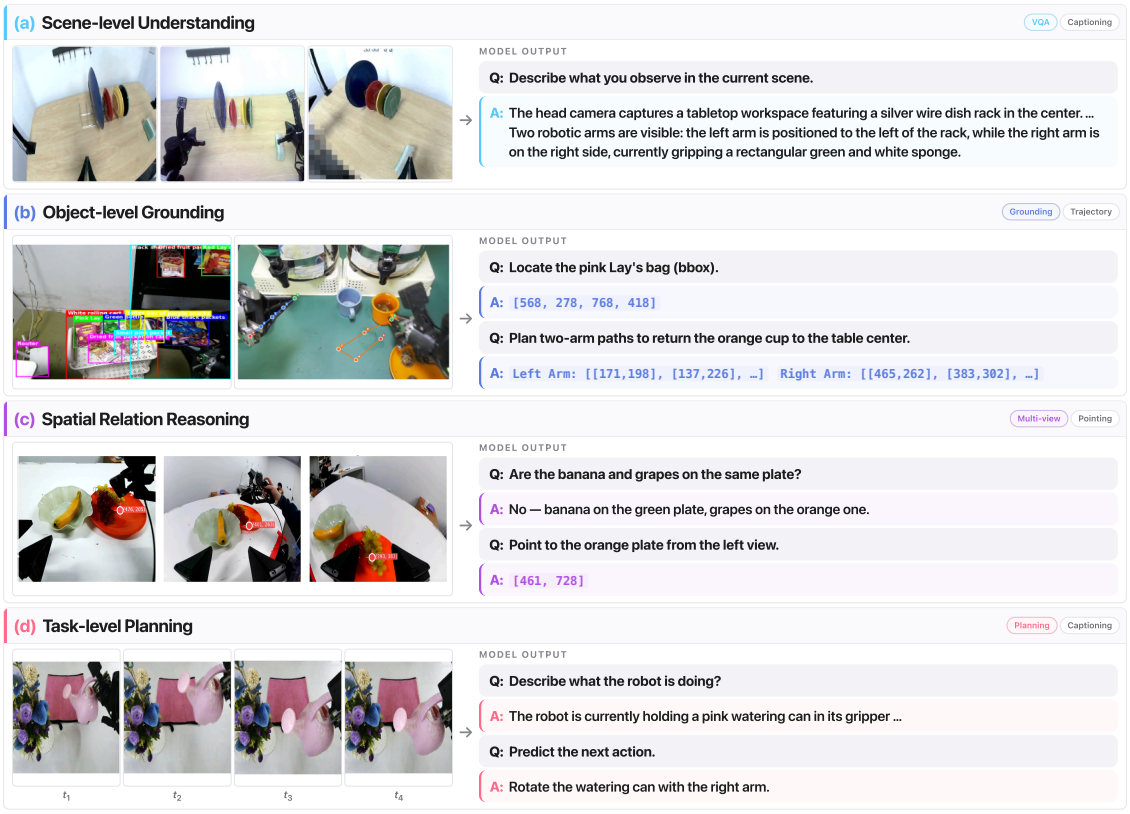


Figure 5 Embodied bridge data construction and task taxonomy. Bridge samples are generated from action trajectories and organized into object, spatial, scene, and task understanding objectives, aligning multimodal supervision with executable robot behavior.

Table 2 Average task progress across pretraining checkpoints for seen and unseen zero-shot evaluation tasks.

	50k	100k	200k	300k	350k	400k
Seen avg. (12 tasks)	26.1	31.7	40.1	40.4	48.1	**50.0**
Unseen avg. (5 tasks)	24.2	41.0	38.8	34.8	47.6	**53.6**
Overall avg. (17 tasks)	25.5	34.5	39.8	38.7	47.9	**51.1**

Embodied bridge data. The second component, which we term *embodied bridge data*, is constructed through an automated data pipeline directly from the action pretraining corpus. We use the term "bridge" because these samples provide an explicit connection from multimodal understanding to action prediction—they originate from the same trajectories, observations, and task contexts as action learning, and are thus natively aligned with executable robot behavior.

As shown in Figure 5, bridge data is organized along four levels of understanding. *Object understanding* supervises object grounding, pointing, 2D trajectory prediction, and attribute VQA; *spatial understanding* covers inter-object relations and multi-view pointing; *scene understanding* addresses scene captioning and scene-level VQA; and *task understanding* targets task-progress prediction, affordance estimation, and next-step action planning.

For pointing and grounding tasks, dedicated spatial tokens are introduced, unifying all coordinate annotations into text format: `<box>[x1,y1,x2,y2]</box>` and `<point>[x,y]</point>`.

4 Experiments and Results

4.1 Pretrained Model Zero-Shot Evaluation

We evaluate the pretrained model without fine-tuning using a real-robot zero-shot suite that spans diverse manipulation dimensions. Evaluation tasks are divided into two groups: seen tasks (12) drawn from within the pretraining data distribution, and unseen tasks (5) comprising held-out task configurations not collected as identical tasks on the current embodiment, serving to test generalization and transfer capabilities. Task types span five dimensions: semantic understanding, rigid-object manipulation, deformable-object manipulation, fine-grained manipulation, and long-horizon multi-step manipulation. Each task is scored according to a predefined scoring rubric (task progress, maximum 100, evaluated over 10 trajectories). This metric captures fine-grained progress at the "partial completion" level, making it more suitable than binary success rate for evaluating VLA models' foundational manipulation capabilities. Multiple milestone checkpoints are sampled during training to track capability evolution trends.

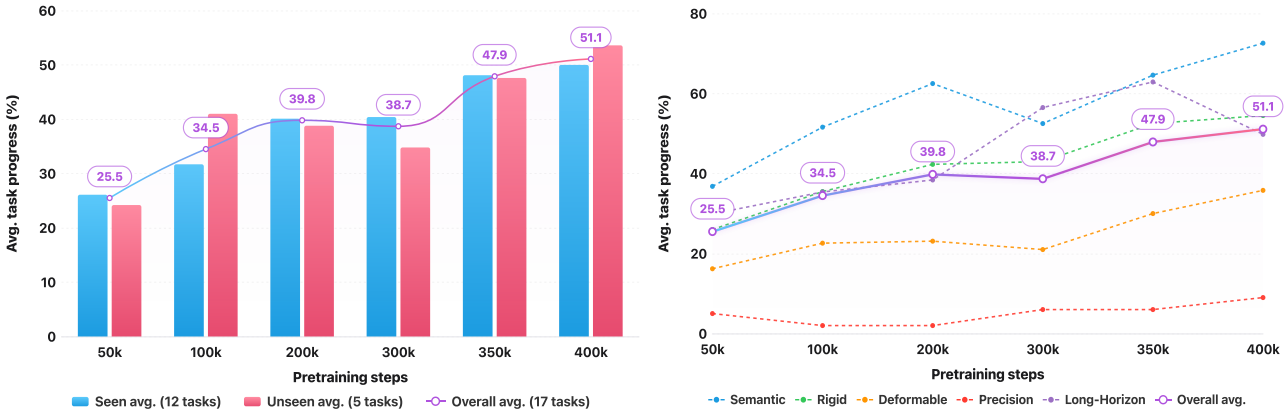
4.1.1 Results

Table 2 summarizes the average task progress for seen and unseen tasks across milestone checkpoints (per-task detailed results in Section A.1).

Before unpacking the aggregate trends, we highlight the strongest pretrained capabilities: at the 400k checkpoint, six tasks already reach or exceed a task-progress threshold of 60% without any task-specific fine-tuning (Table 3), including four tasks above 80% and two held-out (unseen) tasks. The unseen deformable task *Rope Tightening* reaching 82% is particularly notable, making pure task-template memorization less likely and suggesting transferable manipulation capability. Figure 6a shows the average task progress trends for seen and unseen tasks as training progresses; Figure 6b further groups tasks by manipulation type, showing the evolution trajectories across five capability dimensions.

4.1.2 Analysis

From the above experimental results, we distill the following core findings:



(a) Seen vs. Unseen training trends.

(b) Task progress evolution by task category.

Figure 6 Zero-shot real-robot evaluation trends across pretraining checkpoints. (a) Average task progress for seen and unseen tasks shows an overall upward trend despite checkpoint-level fluctuations, with held-out tasks reaching 53.6 at the 400k checkpoint. (b) Category-level curves show how capability emerges unevenly across semantic understanding, rigid-object manipulation, deformable-object manipulation, fine-grained manipulation, and long-horizon manipulation, with semantic tasks becoming the strongest zero-shot dimension while precision-demanding categories remain more difficult.

Semantic grounding transfers to action. From Figure 6b, semantic understanding tasks are the model's strongest dimension (400k average task progress of 72.6%), exceeding other categories. Several high-performing seen tasks—Block Sorting (100%), Ring Stacking (86%), Fruit Sorting (96%)—require the model to ground visual-semantic concepts such as "which color corresponds to which position" or "the ring should be placed on the pole" before executing the manipulation. Among unseen tasks, Toy Basket Placement reaches 58% at 400k and peaks at 72% during pretraining, suggesting that VLM-derived semantic knowledge is being transferred to action execution while some held-out tasks remain checkpoint-sensitive. This pattern is consistent with the training design: action-token CE exposes the action pathway to VLM semantic priors, so tasks dominated by semantic-grounding decisions can benefit from the backbone's strongest capabilities, while tasks requiring precise low-level control (fine-grained insertion, deformable folding) remain bottlenecked by downstream execution.

Held-out tasks improve with seen tasks. From Figure 6a, the average task progress of unseen tasks (24.2% → 53.6%) and seen tasks (26.1% → 50.0%) both improve overall despite checkpoint-level fluctuations, with unseen tasks (53.6%) slightly exceeding seen tasks (50.0%) at 400k. Because the seen and unseen groups are not difficulty-matched, the trend is more informative than the absolute ordering. The parallel improvement makes it less likely that the model relies only on memorized tasks within the training distribution and instead suggests manipulation capabilities that transfer across scenes and tasks. In the context of large-scale pretraining, however, "unseen" is a relative concept—while these tasks have never been collected as identical task data on the current embodiment and use entirely new prop combinations, open-source data in the pretraining corpus may contain semantically related manipulation experience. The generalization here thus primarily reflects cross-scene, cross-prop skill transfer rather than learning entirely novel skills from scratch. Notably, Rope Tightening (deformable manipulation) reaches task progress 82 at 400k, providing a strong example of transfer in a held-out deformable scenario.

Zero-shot capability has clear limits. Based on 400k performance, tasks can be categorized into three tiers:

- *Zero-shot proficient* (task progress ≥ 60%): Block Sorting (100%), Fruit Sorting (96%), Ring Stacking (86%), Rope Tightening (82%), etc.—characterized by clear semantics and moderate precision requirements, where VLM semantic understanding provides a strong scaffold for effective actions.
- *Partially proficient* (task progress 40%–60%): Switch Pressing (55%), Number Ordering (54%), Flower

Arranging (51%), Package Sorting (48.5%), etc.—the model has mastered basic operations (approach, grasp, move) but still falls short on final precise execution. This gap precisely defines the capability delta that fine-tuning needs to bridge.

- *Currently beyond zero-shot reach* (task progress < 20%): Towel Folding (10%), Table Setting (9%), Charger Plugging (9%)—these tasks primarily involve deformable objects and fine-grained manipulation. We attribute this to two factors: first, these tasks are inherently far more difficult than rigid pick-and-place, with precision and state perception requirements exceeding what can be achieved without fine-tuning; second, the action patterns required (e.g., pinching fabric edges, folding along trajectories, fine-adjusting alignment for insertion) are relatively isolated in the overall dataset, with low overlap with common task action distributions, making effective cross-task transfer difficult.

Capabilities emerge during pretraining. Multiple tasks show breakthrough progress at specific training stages (see Section A.1 for details): Block Sorting jumps from approximately 50% to 100% in the mid-to-late period, Ring Stacking crosses into the fully successful regime by reaching 100% at 350k before remaining strong at 86% at 400k, and Fruit Sorting rises from 61% to 96%. This "staircase" emergence pattern resembles the emergent abilities observed in large language models: capabilities may first appear as sharp checkpoint-level breakthroughs before stabilizing into the broader pretrained policy. Meanwhile, the overall average task progress at 400k continues to rise (seen 50.0%, unseen 53.6%), suggesting that pretraining may not yet be saturated and that further scaling could yield additional gains.

Table 3 Highlighted zero-shot tasks at the 400k checkpoint (task progress ≥ 60). *Unseen* tasks are bolded; deformable tasks are particularly notable for generalization. Full per-task results in Section A.1.

Task	Category	Seen / Unseen	Task progress
Block Sorting	Semantic understanding	Seen	**100%**
Fruit Sorting	Semantic understanding	Seen	96
Ring Stacking	Rigid manipulation	Seen	86
Rope Tightening	Deformable manipulation	**Unseen**	82
Cup Grasping	Rigid manipulation	Seen	64
Bean Pouring	Deformable manipulation	**Unseen**	60

4.2 Real-Robot Fine-Tuning Results

4.2.1 Baseline Comparison

We benchmark against two pretrained robot foundation models from the two dominant paradigms: $\pi_{0.5}$, a vision-language-action (VLA) model, and DreamZero, a world-action model (WAM). All three models—ours, $\pi_{0.5}$, and DreamZero—are initialized from their respective official pretrained weights, and are fine-tuned and evaluated on 15 real-robot tasks (10 manipulation + 5 reasoning) under identical fine-tuning data (~500 demonstration trajectories per task) and evaluation protocols.

Table 4 Real-robot fine-tuning baseline comparison. Values are average task progress per category (max 100). Per-task details in Section A.3.

Model	Manipulation (10)	Reasoning (5)	Overall (15)
Wall-OSS-0.5	**61.1**	**59.3**	**60.5**
$\pi_{0.5}$	35.0	58.9	43.0
DreamZero	33.7	32.7	33.4

The results indicate:

Overall result. Wall-OSS-0.5 achieves 60.5% average task progress, outperforming $\pi_{0.5}$ (43.0%) by 17.5% and DreamZero (33.4%) by 27.1%, with the highest score on 10 of 15 tasks. Wall-OSS-0.5 now leads on both subsets simultaneously—manipulation (61.1% vs. 35.0%) and reasoning (59.3% vs. 58.9%).

Manipulation advantage. On tasks such as Color Block Sorting (96% vs. 42%), Ring Stacking (91% vs. 60%), Drawer Organization (52% vs. 7%), and Spoon-in-Bowl (80% vs. 43%), Wall-OSS-0.5 leads $\pi_{0.5}$ by 30% or more. These tasks span semantic understanding, medium-precision positioning, and multi-step manipulation. Glasses Rack Placement (66% vs. 87%) is the only manipulation task where $\pi_{0.5}$ leads.

Reasoning tasks. Wall-OSS-0.5 leads $\pi_{0.5}$ on three of five reasoning tasks—Shape Sorting (68% vs. 63%), Earphone Sorting (82% vs. 73%), and Sequential Button Pressing (16% vs. 13%)—driving the reasoning-subset average to 59.3% vs. $\pi_{0.5}$'s 58.9%. Fruit Basket Placement (86% vs. 94%) and Object Matching (44.5% vs. 51.5%) remain the two task-level gaps to $\pi_{0.5}$ on the reasoning subset.

Hard remaining case. Pencil Case Packing (18.5%) remains the main low-score manipulation case for Wall-OSS-0.5, reflecting genuine difficulty in fine-grained bimanual manipulation involving deformable and articulated interactions under the current fine-tuning data budget.

Fine-tuning builds on pretraining. The strongest post-fine-tuning tasks (Color Block Sorting 96%, Ring Stacking 91%) already exhibited high task progress in the Section 4.1 zero-shot evaluation (100% and 86%, respectively), while the tasks with the largest fine-tuning gains (e.g., Drawer Organization jumping from a low zero-shot baseline to 52%) demonstrate fine-tuning's amplification effect on pretrained base capabilities. This suggests that the capability foundation established during pretraining significantly influences the ceiling of fine-tuning performance.

Why manipulation improves. We analyze the manipulation gains through three training-framework choices introduced in Section 1: end-to-end co-training, the vision-aligned RVQ Action Tokenizer, and Action-Space Supervision. Section 5 quantifies the individual contribution of each.

4.2.2 Multi-Task Fine-Tuning Scaling

Setup. To evaluate the pretrained model's fine-tuning capability in multi-task settings, we progressively expand the fine-tuning task set from 5 to 10 to 19 tasks, all fine-tuned from the same pretrained checkpoint, analyzing performance trends as task scale grows. The three configurations are:

- **5 tasks**: A basic manipulation task subset selected from Section 4.1.
- **10 tasks**: The 5-task set augmented with 5 higher-difficulty or multi-step reasoning tasks.
- **19 tasks**: The 10-task set further augmented with 9 additional manipulation tasks from the broader real-robot evaluation suite. These 9 added tasks differ notably from the original 10 in background environments and embodiment distributions, so this configuration expands not only task count but also scene and object coverage.

All experiments use identical training configurations with a uniform 6 epochs, ensuring each task receives the same number of training passes. Overall results are shown in Figure 7; per-task details, including Towel Folding (which lies outside the scope of this scaling study—see Section A.4), are in the appendix.

Task expansion helps shared tasks. As the fine-tuning task set expands, performance on shared subsets improves monotonically. On the 5 simple shared tasks, scaling $5 \rightarrow 10 \rightarrow 19$ tasks raises the average from 73.96% to 74.75% to **83.75%** (a +9.8% gain end-to-end). On the 10 shared tasks (5 simple + 5 complex & reasoning), scaling $10 \rightarrow 19$ raises the average from 59.98% to **64.78%** (+4.8%). Notably, the gains on shared subsets persist after introducing 9 additional tasks with distinctly different backgrounds and embodiment distributions, where the model also reaches 65.59% average on these out-of-distribution tasks. This shows that under the current model capacity and training setup, task expansion benefits not only adaptation to new tasks but also sustained improvement on shared tasks.

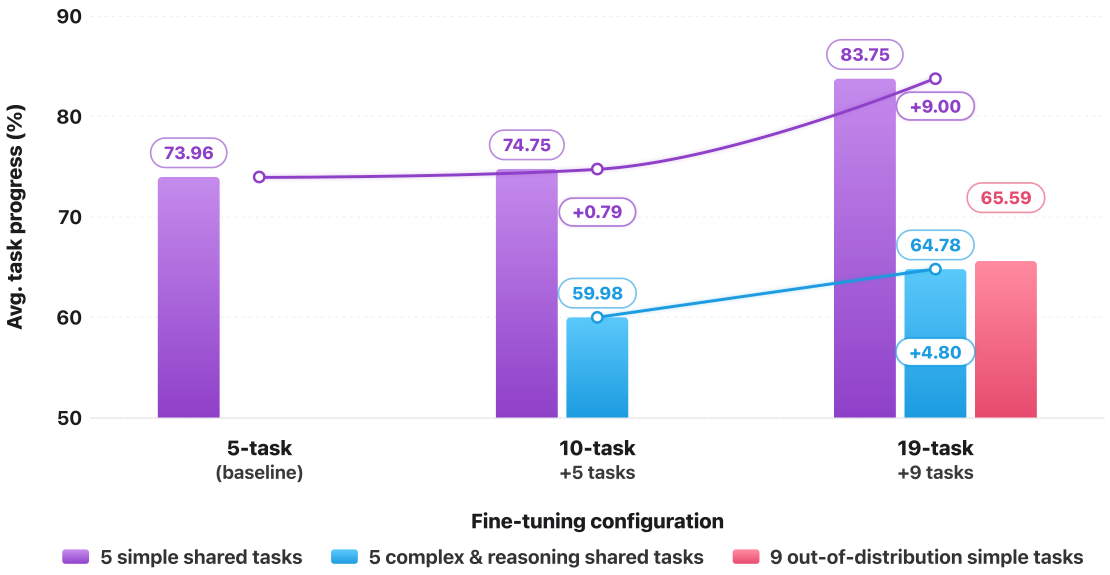


Figure 7 Multi-task fine-tuning scaling results on real-robot tasks. The figure compares models fine-tuned on progressively larger task sets (5, 10, and 19 tasks) and reports average task progress on the shared 5 simple-task subset, the shared 10-task subset, and the 9 newly added out-of-distribution simple tasks. Expanding the fine-tuning set improves performance on shared tasks rather than diluting them: the 5-task subset rises from 73.96% to 83.75%, the 10-task subset improves from 59.98% to 64.78%, and the added out-of-distribution tasks reach 65.59% under the 19-task configuration.

Gains come from broader capability coverage. We hypothesize that this improvement more likely stems from supplementation at the fine-grained capability level rather than direct whole-task transfer. Although the 9 added tasks differ substantially from the original 10 in overall form, they may still supplement the original training data's coverage gaps in atomic action patterns, language instruction expressions, and state change distributions. For example, basic approach, align, grasp, adjust, and place action primitives may be more thoroughly observed in the new tasks; similarly, different object description styles, object reference forms, and environmental perturbation conditions may expand the model's coverage of the action-language-state combinatorial space. The new tasks' contribution thus more likely manifests as completing reusable intermediate capabilities, improving generalization and robustness on original tasks rather than simply learning a set of independent new task policies.

4.3 Embodied Multimodal Understanding

To evaluate the impact of co-training on multimodal understanding capabilities, we select 5 representative benchmarks for evaluation (detailed scores in Section A.2), divided into two groups: general visual question answering (RealWorld VQA (41), ERQA (5)) and embodied understanding directly relevant to robot execution (EO-Bench (38) scene understanding, Embodied Grounding target localization, Where2Place (30) placement reasoning), with the VLM backbone Qwen2.5-VL-3B (11) as baseline. Embodied Grounding is an internally constructed benchmark whose samples are drawn and annotated from robot action trajectories.

Figure 8 shows the score changes of Wall-OSS-0.5 relative to the backbone on each benchmark. The results suggest a *specialization* effect from co-training: performance shifts away from open-domain VQA and toward embodied perception signals more relevant to robot execution.

Significant embodied understanding gains. The most pronounced improvement is in robot manipulation-oriented target localization (Embodied Grounding, +21.8), which requires the model to localize manipulation

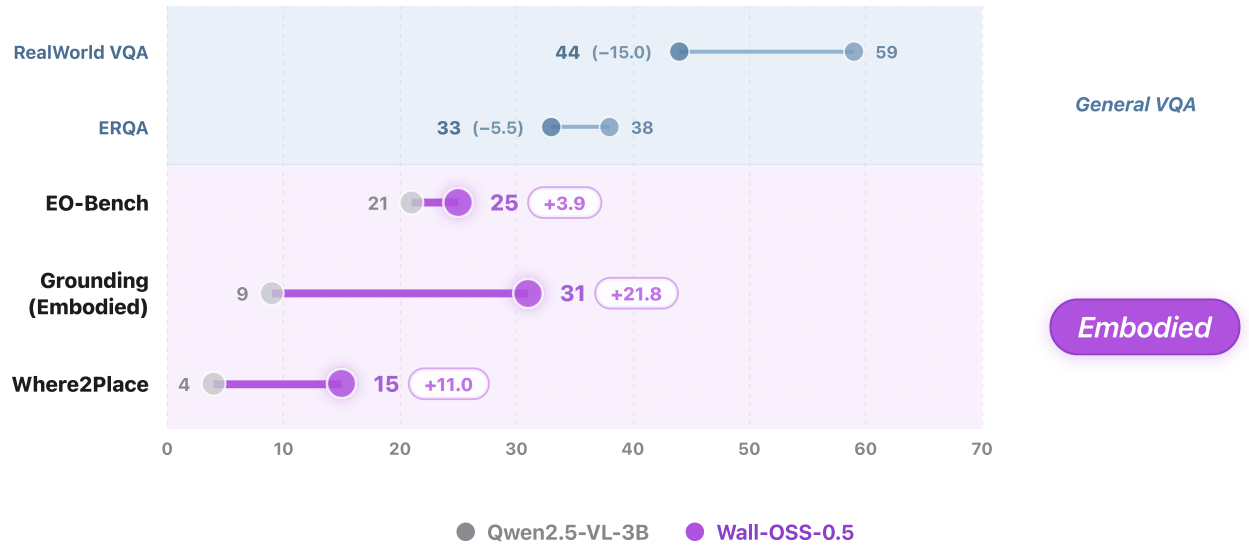


Figure 8 Multimodal capability changes from co-training, measured relative to the backbone Qwen2.5-VL-3B.

targets in the robot's ego-centric view (e.g., "find the red cup on the table"). This differs from open-domain grounding settings: the backbone has a low baseline (9.0) on our embodied manipulation grounding benchmark, which lies outside its original pretraining focus. Placement reasoning Where2Place (+11.0) and embodied scene understanding EO-Bench (+3.9) also show clear gains. These improvements correspond closely to the core perception needs in the robot execution pipeline—"where to look, where to point, where to place". In pilot pretraining runs, we observed that the strong action-token prediction objective can substantially reduce multimodal scores on common robot-observation distributions when embodied bridge data is insufficient. The 12 million bridge samples described in Section 3.2 were therefore introduced not only as extra VQA data, but as robot-view grounding and spatial-decision supervision that counterbalances the specialization pressure from action-token CE.

General VQA regression under specialization. General visual question answering regresses (RealWorld VQA −15.0, ERQA −5.5), consistent with the specialization pressure introduced by VLA co-training. For a VLA model whose primary objective is action execution, this trade-off is reasonable: the model gives up part of its open-domain VQA performance while improving embodied spatial decision signals, rather than competing with dedicated VLMs on general image understanding.

Visualization examples. Figure 9 shows grounding and spatial reasoning comparisons between Wall-OSS-0.5 and the backbone Qwen2.5-VL-3B in embodied scenes. In the robot's ego-centric view, Wall-OSS-0.5 more accurately localizes manipulation targets and selects placement positions that align with actionable regions, while the backbone is more likely to drift toward visually salient but less task-relevant areas. These examples visually illustrate the embodied perception enhancement from co-training.

5 Ablation Studies

5.1 Effect on Co-training Strategies

We compare four training strategies trained from scratch (with only the VLM backbone initialized from pretraining): co-training, stop-gradient, stop-gradient to co-training, and flow-only. All experiments run for 70k training steps on 5 real-robot ablation tasks with a 25% multimodal-CE mixing ratio, under identical configurations. These 5 tasks are a subset selected for tractable from-scratch training; their absolute numbers are not directly comparable to the 17-task pretrained evaluation in Section 4.1—this experiment only supports

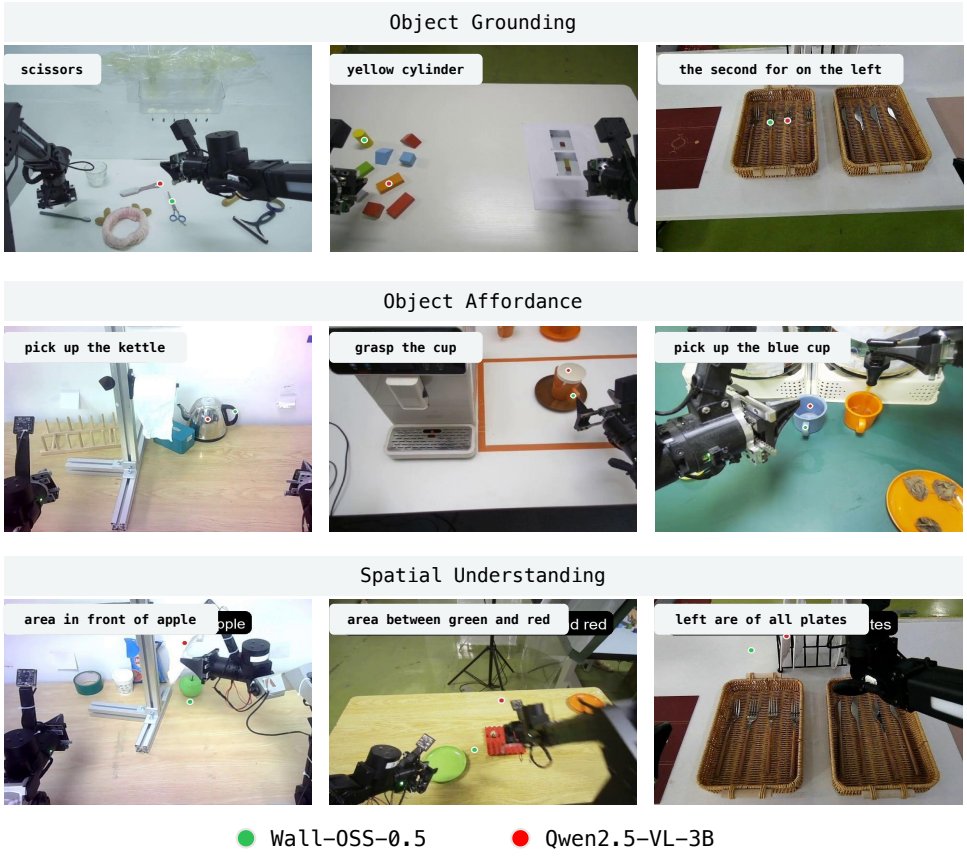


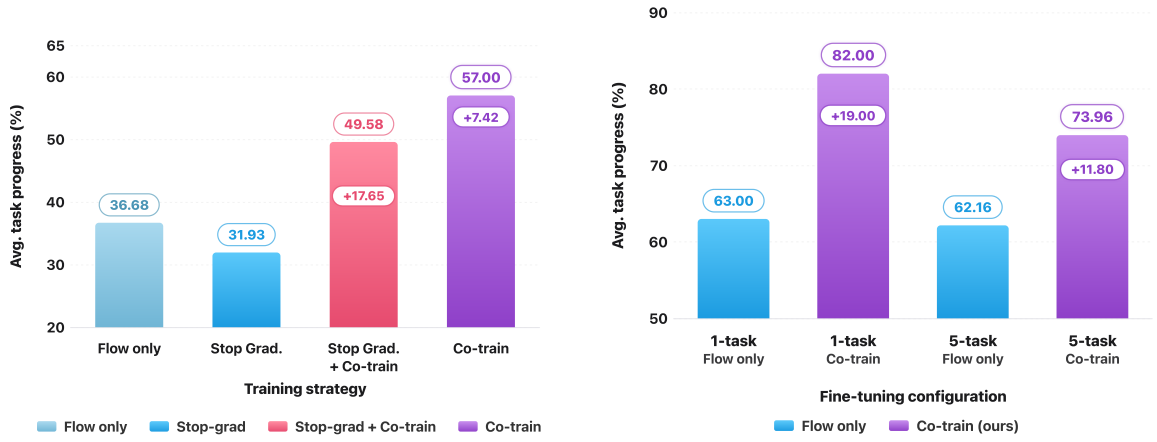
Figure 9 Multimodal understanding visualization comparison in embodied scenes (Wall-OSS-0.5 vs. Qwen2.5-VL-3B).

relative comparison between the four strategies. The strategies differ in whether flow-matching gradients reach the VLM backbone and whether action-token cross-entropy is present:

- *Co-training*: cross-entropy on RVQ action tokens, flow matching, and multimodal cross-entropy are jointly optimized, with all gradients backpropagating to the VLM backbone (the full gradient-bridged co-training setup).
- *Stop-gradient*: Same as co-training, but flow gradients are cut off from the backbone (the Action Expert is still optimized by flow); the backbone is updated only by action-token and multimodal cross-entropy. This matches the knowledge-isolation strategy proposed by (42).
- *Stop-gradient to co-training*: 60k steps of stop-gradient training, followed by 10k steps after the gradient block is removed.
- *Flow-only*: flow matching and multimodal cross-entropy only, with no action-token cross-entropy pathway.

As shown in Figure 10a, co-training reaches an average task progress of 57.0%, outperforming flow-only (36.6%), stop-gradient (31.9%), and stop-gradient to co-training (49.6%) on 5 ablation tasks. This supports the gradient-bridged co-training claim in Section 1: removing any of the three signals (flow alone, no bridge to backbone, or two-stage substitution) degrades real-robot performance by 7.4 – 25.1 percentage points. Across all four strategies, the VQA scores stay tightly clustered, with stop-gradient marginally ahead of the other three.

Stop-gradient—the strategy marginally ahead on VQA—achieves the lowest score among all action tasks, and this gap is consistent across all tasks. During training, stop-gradient exhibits slower flow-loss convergence and a higher final loss value, indicating Action Expert underfitting. Co-training's flow loss, in contrast, converges



(a) From-scratch results on 5 real-robot ablation tasks.

(b) Fine-tuning stage results.

Figure 10 Training strategy comparison. (a) Co-training achieves the best from-scratch performance (70k steps). (b) Co-training advantages persist during fine-tuning.

faster than both flow-only and stop-gradient, with a final value lower than stop-gradient and comparable to flow-only. Based on these observations, co-training is adopted as the training strategy for the pretraining stage.

This finding extends beyond pretraining: during fine-tuning, discrete action tokens likewise provide more efficient adaptation signals to the VLM backbone, accelerating learning on new tasks, as shown in Figure 10b. Co-training with discrete action tokens and multimodal dataset is therefore also adopted for the fine-tuning stage.

5.2 Effect on Action-Space Loss

To compare the action-space loss (Section 2.2.2) with the traditional flow-matching loss in velocity fields, we carry out controlled experiments in the LIBERO simulation environment. Both conditions use the same model architecture as the pretraining backbone, with weights initialized from Qwen2.5-VL and the Action Expert trained from scratch. Training uses a global batch size of 128, with all remaining hyperparameters held identical across the two conditions.

Performance. As shown in Figure 11, the action-space loss variant achieves 96.5% peak average success rate at 25k steps, exceeding the velocity-space loss peak by 6.2%. For the convergence speed, the action-space loss reaches 95.8% average success rate at only 20k training steps, while the velocity-space loss fails to surpass 90.3% after the full 35k steps.

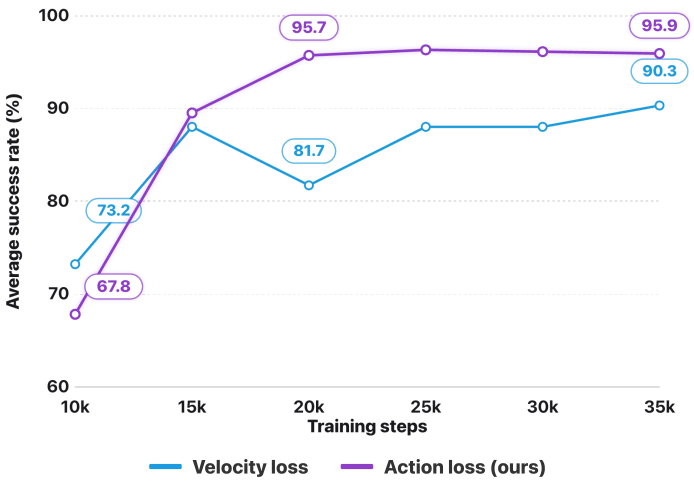


Figure 11 action-space loss vs. velocity-space loss on LIBERO.

Training stability. The velocity-space loss exhibits significant performance fluctuations at 20k, while the action-space loss consistently maintains the 92.5%–96.5% range after 20k steps. The only exception occurs in early training (10k steps), where the velocity-space loss slightly outperforms the action-space loss. This behavior is consistent with the design rationale of action-space supervision: since the action-space loss is equivalent to $(1-\tau)^2$-reweighting the velocity-space loss, gradient signals in the low-noise regime are relatively sparse during early training. Once the model has acquired the low-frequency trajectory structure through high-noise supervision, performance rapidly improves.

These experimental results support defining the loss function in the action space and align with the spectral characteristics of robot action signals: useful information concentrates in low-frequency components, and the high-noise regime constitutes the dominant quality bottleneck.

5.3 Effect on RVQ Action Tokenizer

To validate the effectiveness of the proposed Vision-Aligned RVQ Action Tokenizer relative to the FAST tokenizer (10), we conduct a controlled experiment that replaces only the tokenizer under identical co-training settings, training fully from the VLM backbone weights. Evaluation covers four real-robot tasks and one VQA task; real-robot evaluation uses the continuous actions generated by the flow-matching pathway under co-training.

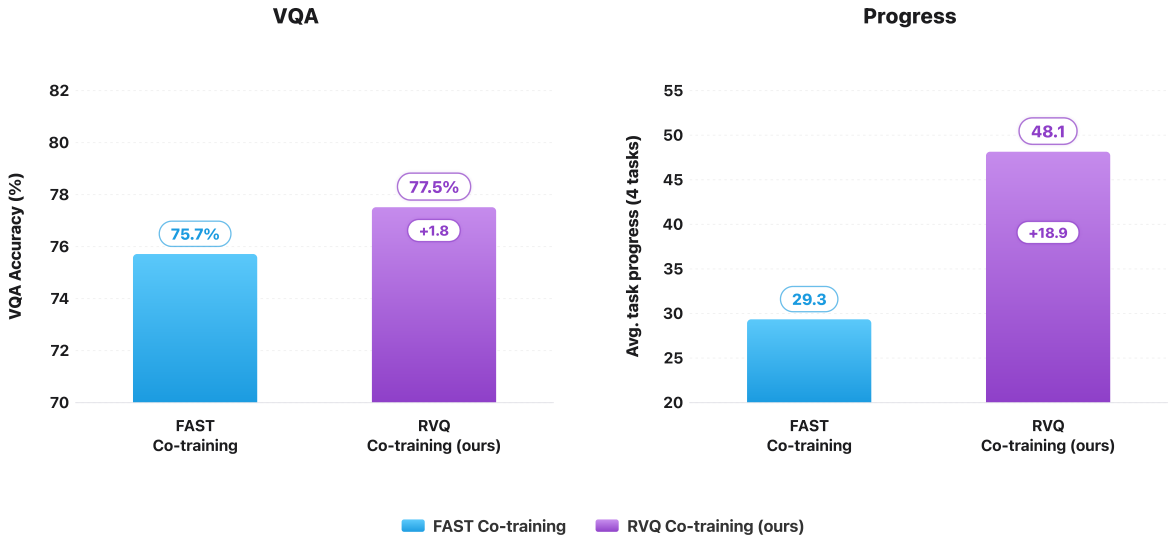


Figure 12 Vision-aligned RVQ Action Tokenizer vs. FAST tokenizer under identical co-training settings. Action task values are average task progress over 4 real-robot tasks (max 100), with RVQ leading on every task individually.

Multimodal understanding. As shown on the left of Figure 12, our proposed RVQ tokenizer not only preserves VQA performance but improves accuracy from 75.7% to 77.5%. This indicates that the auxiliary multimodal

objectives used to train the tokenizer—visual-action alignment and next-frame prediction (Section 2.1.2)—shape token representations that also contribute to the VLM's overall visual-semantic understanding.

Task Performance. As shown on the right of Figure 12, our proposed RVQ Tokenizer improves average task progress from 29.3% to 48.1%. Notably, real-robot evaluation uses continuous actions generated by flow matching rather than direct decoding of discrete tokens, so the tokenizer's gains are not confined to the discrete pathway: higher-quality discrete representations during co-training also enhance the quality of continuous action generation.

6 Related Work

Action representations. Adapting pretrained vision-language models (VLMs) into Vision-Language-Action (VLA) models is a central route to embodied control. RT-2 (2) and OpenVLA (3) discretize robot actions into text-like tokens, demonstrating that VLM semantic knowledge can transfer to robot control while exposing the precision limits of naive per-dimension discretization. FAST (10) improves token efficiency through DCT and BPE, but remains a rule-based action compressor that carries limited high-level semantics. Continuous policies, including Diffusion Policy (43) and π_0 (4), model high-precision action distributions more naturally. The closest comparison is $\pi_{0.5}$ (1), which co-trains a FAST-based (4) autoregressive pathway with a flow-matching pathway. Wall-OSS-0.5 differs in three respects: it employs a learned Vision-Aligned RVQ Action Tokenizer, analyzes the discrete pathway as a gradient bridge and introduces Action-Space Supervision for flow matching.

Architectures for action-ready VLMs. A central design challenge for VLAs is how to add action generation without overwriting the semantic knowledge stored in the VLM backbone. CogACT (44) separates cognition and action into independent modules, while HPT (45) uses embodiment-specific stems coupled to a shared trunk. Wall-OSS-0.5 adopts a Mixture-of-Transformers architecture with a VL Expert and an Action Expert at each Transformer layer; the two experts interact through shared attention while retaining separate parameters, shifting the role of expert partitioning from generic knowledge sharding (46, 47) to modality–function routing.

Robot data and embodied grounding. Open X-Embodiment (48), DROID (21), RoboMIND (14), and AgiBot World (16) have continued to expand the scale and diversity of available robot data. Our pretraining mixture is comparable in scale—over one million trajectories per epoch across more than 20 embodiments—but is organized around deployment rather than scale alone. We apply task-level power sampling to mitigate long-tail imbalance and supplement the mixture with large-scale embodied bridge samples synthesized from action corpora, linking visual understanding to executable action semantics rather than relying on generic VQA data alone (2, 1).

7 Discussion and Limitations

This report advances an operational claim: sufficiently large-scale VLA pretraining can already produce measurable real-world robotic behavior, while the resulting checkpoint simultaneously serves as a substantially stronger prior for downstream adaptation. We conclude by discussing the broader design implications, current limitations, and promising future directions.

Design implications. The gradient-bridge mechanism suggests that discrete action tokens remain valuable even when deployment ultimately relies on continuous actions. Their primary role emerges during pretraining: action-token cross-entropy provides a strong, VLM-native supervisory channel that directly shapes the backbone toward controllable representations. In contrast, continuous-only training exposes the backbone primarily to a relatively weak residual flow-matching signal, while stop-gradient formulations decouple representation learning from continuous control optimization, preventing the two objectives from fully reinforcing one another.

This perspective also unifies two seemingly independent design choices. MoT routing (Section 2.1.1) allocates dedicated parameter capacity to continuous-action computation without severing gradients flowing into the

VLM backbone. Meanwhile, embodied bridge data (Section 3.2) makes the multimodal anchor more action-aware, enabling the model to preserve grounded understanding in environments closer to those encountered during physical execution. Together, both choices pursue the same objective: transforming a pretrained multimodal model into an executable robotic policy without collapsing it into a narrow task-specific controller.

Limitations. Several limitations remain. First, the gradient-bridge dynamics have thus far only been validated with a 3B VLM backbone; scaling to larger VLM backbones may substantially alter the relative geometry and interaction strength among the three training signals. Second, the current model operates on single-frame image inputs, which likely constrains zero-shot performance on long-horizon tasks requiring temporal memory and persistent state tracking. Third, the Vision-Aligned RVQ Action Tokenizer and the associated training pipeline are currently tied to a fixed 26-dimensional action representation, limiting direct applicability to dexterous hands and other high-DoF embodiments that require richer action interfaces. Finally, task evaluation still depends on manually designed scoring rubrics (Section A.5), and the present real-robot benchmark does not yet cover multi-robot collaboration, long-duration deployment, or broader open-world interaction settings.

Future Work. Future work will scale the framework to larger VLM backbones, incorporate temporal observations and hierarchical planning for long-horizon tasks, and explore more general action representations capable of supporting diverse robot morphologies. As an open-source project, we will continue releasing model weights, training code, and evaluation tools to facilitate future research and reproducibility.

8 Contributors

Wall-OSS-0.5 is a collaborative effort of the X Square Robot team. The full contributor list is given below; $*$ denotes core contributors, † denotes the project lead, and ‡ denotes the corresponding author.

Ryan Yu[*], Pushi Zhang[*], Starrick Liu[*], Brae Liu[*], Miracle Kang[*], Shalfun Li[*], Lights Shi, Ellie Ma, Ping Yang, Chris Pan, Jerry Chen, Dongxiu Liu, Rain Sun, Miles Guo, Byron Zhang, Hugo Zhou, Zach Xu, Vincent Chen, Harrison Huang, Dance Kuzi, Andy Zhai, Hang Su, Roy Gan, Lucy Liang[†], Hao Wang[‡], Qian Wang.

References

[1] 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.

[2] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[3] 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.

[4] 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.

[5] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[6] Figure AI. Helix: A vision-language-action model for generalist humanoid control. https://www.figure.ai/helix, 2025.

[7] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

[8] GigaBrain Team, Boyuan Wang, Bohan Li, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigabrain-0.5 m*: a vla that learns from world model-based reinforcement learning. *arXiv preprint arXiv:2602.12099*, 2026.

[9] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic VLA foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

[10] 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.

[11] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

[12] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[13] Junming Wang, Teng Pu, Wingmun Fung, Jindong Wang, Shanchang Wang, Yuan Deng, Shuyuan Wang, Ziwei Liu, Kunhao Pan, Ping Yang, et al. Xrzero-g0: Pushing the frontier of dexterous robotic manipulation with interfaces, quality and ratios. *arXiv preprint arXiv:2604.13001*, 2026.

[14] Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

[15] Chengkai Hou, Kun Wu, Jiaming Liu, Zhengping Che, Di Wu, Fei Liao, Guangrun Li, Jingyang He, Qiuxuan Feng, Zhao Jin, et al. Robomind 2.0: A multimodal, bimanual mobile manipulation dataset for generalizable embodied intelligence. *arXiv preprint arXiv:2512.24653*, 2025.

[16] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[17] Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, et al. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

[18] Adina Yakefu, Bin Xie, Chongyang Xu, Enwen Zhang, Erjin Zhou, et al. RoboChallenge: Large-scale real-robot evaluation of embodied policies. *arXiv preprint arXiv:2510.17950*, 2025.

[19] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and G0 dual-system VLA model. *arXiv preprint arXiv:2509.00576*, 2025.

[20] GenRobot AI. RealOmin: 10kh RealOmin-open dataset. https://huggingface.co/datasets/genrobot2025/10Kh-RealOmin-OpenData, 2025. Open robot manipulation dataset; see also https://www.genrobot.ai/data/open-dataset.
[21] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.
[22] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.
[23] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
[24] Qiying Yu, Quan Sun, Xiaosong Zhang, Yufeng Cui, Fan Zhang, Yue Cao, Xinlong Wang, and Jingjing Liu. Capsfusion: Rethinking image-text data at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14022–14032, 2024.
[25] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.
[26] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 91–104, 2025.
[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
[28] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.
[29] Kaituo Feng, Manyuan Zhang, Hongyu Li, Kaixuan Fan, Shuang Chen, Yilei Jiang, Dian Zheng, Peiwen Sun, Yiyuan Zhang, Haoze Sun, et al. Onethinker: All-in-one reasoning model for image and video. *arXiv preprint arXiv:2512.03043*, 2025.
[30] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.
[31] Remyx AI. Spacethinker. https://huggingface.co/datasets/remyxai/SpaceThinker, 2025. Hugging Face dataset page.
[32] Remyx AI. Openspaces. https://huggingface.co/datasets/remyxai/OpenSpaces, 2025. Hugging Face dataset page.
[33] Remyx AI. Spaceom. https://huggingface.co/datasets/remyxai/SpaceOm, 2025. Hugging Face dataset page.
[34] Jingkun An. Refspatial. https://huggingface.co/datasets/JingkunAn/RefSpatial, 2025. Hugging Face dataset page.
[35] Yipu Wang, Yuheng Ji, Yuyang Liu, Enshen Zhou, Ziqiang Yang, Yuxuan Tian, Ziheng Qin, Yue Liu, Huajie Tan, Cheng Chi, et al. Towards cross-view point correspondence in vision-language models. *arXiv preprint arXiv:2512.04686*, 2025.
[36] Zhongang Cai, Ruisi Wang, Chenyang Gu, Fanyi Pu, Junxiang Xu, et al. Scaling spatial intelligence with multimodal foundation models. *arXiv preprint arXiv:2511.13719*, 2025. URL https://arxiv.org/abs/2511.13719.
[37] Kaiyuan Chen, Shuangyu Xie, Zehan Ma, Pannag R Sanketi, and Ken Goldberg. Robo2vlm: Visual question answering from large-scale in-the-wild robot manipulation datasets. *arXiv preprint arXiv:2505.15517*, 2025.
[38] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Dong Wang, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, Maoqing Yao, Haoran Yang, Jiacheng Bao, Bin Zhao, and Xuelong Li. Eo-1: An open unified embodied foundation model for general robot control, 2026. URL https://arxiv.org/abs/2508.21112.
[39] Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J Joshi, et al. Robovqa: Multimodal long-horizon reasoning for robotics. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 645–652. IEEE, 2024.
[40] Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, et al. Cosmos-reason1: From physical common sense to embodied reasoning. *arXiv preprint arXiv:2503.15558*, 2025.

[41] xAI. RealWorldQA: A benchmark for real-world spatial understanding of multimodal models. https://x.ai/blog/grok-1.5v, 2024. Dataset released with Grok-1.5V.

[42] Danny Driess, Jost Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026.

[43] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44 (10-11):1684–1704, 2025.

[44] 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.

[45] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Advances in neural information processing systems*, 37:124420–124450, 2024.

[46] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[47] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[48] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

A Appendix

A.1 Pretrained Model Zero-Shot Evaluation Per-Task Results

The following tables list per-task task progress for all seen tasks in Table 5 and unseen tasks in Table 6 across milestone checkpoints from section 4. Each task is evaluated over 10 trajectories, scored according to predefined scoring rubrics (maximum 100). Category abbreviations (**Cat.**): **Sem.** = semantic understanding, **Rigid** = rigid-object, **Def.** = deformable-object, **Fine** = fine-grained, **Long** = long-horizon multi-step manipulation.

Table 5 Seen tasks zero-shot per-task progress. Bold indicates the highest task progress across checkpoints for each task.

Task	Cat.	50k	100k	200k	300k	350k	400k
Block Sorting	Sem.	46	85	87.5	51.5	96	**100**
Fruit Sorting	Sem.	41	19	81	61	61	**96**
Number Ordering	Sem.	45	**65**	32	60	56	54
Switch Pressing	Sem.	7	18	46	30	38	**55**
Ring Stacking	Rigid	18	18	51	73	**100**	86
Cup Grasping	Rigid	48	**70**	63	58	56	64
Towel Folding	Def.	9	8	**11.5**	10	10	10
Table Setting	Def.	6	**15**	8	10	10	9
Paper Shredding	Def.	**28**	10	23	12	18	18
Charger Plugging	Fine	5	2	2	6	6	**9**
Flower Arranging	Long	30	34.5	37.5	54.5	**57**	51
Package Sorting	Long	29.75	36.25	39.25	58.5	**68.75**	48.5
Seen avg.		26.1	31.7	40.1	40.4	48.1	**50.0**

Table 6 Unseen tasks zero-shot per-task progress. Bold indicates the highest task progress across checkpoints for each task.

Task	Cat.	50k	100k	200k	300k	350k	400k
Toy Basket Plcmt.	Sem.	45	71	66	60	**72**	58
Rope Tightening	Def.	26	30	54	60	62	**82**
Bean Pouring	Def.	12	50	19	13	50	**60**
Table Wiping	Rigid	**38**	36	32	26	28	38
Pot Lid Covering	Rigid	0	18	23	15	26	**30**
Unseen avg.		24.2	41.0	38.8	34.8	47.6	**53.6**

A.2 Multimodal Understanding Evaluation Detailed Results

Table 7 lists detailed scores and task descriptions for each benchmark in section 4.

Table 7 Multimodal understanding benchmark detailed scores. *Embodied Grounding* is an internally constructed benchmark; evaluation samples are directly sampled and annotated from robot action dataset trajectories.

Dimension	Benchmark	Task Description	Qwen2.5-VL-3B	Wall-OSS-0.5	Change
General VQA	RealWorld VQA	Open-ended VQA on real-world scene images	59.2%	44.2%	−15.0
General VQA	ERQA	Entity recognition and relational QA for embodied scenes	38.3%	32.8%	−5.5
Embodied scene	EO-Bench	Comprehensive visual understanding in embodied manipulation scenes	20.8%	24.7%	+3.9
Embodied grounding	Emb. Grounding*	Localizing target objects via point coordinates in robot ego-centric images	9.0%	30.8%	+21.8
Placement reasoning	Where2Place	Predicting reasonable placement position coordinates	4.0%	15.0%	+11.0

A.3 Fine-Tuning Evaluation Per-Task Detailed Results

Table 8 Fine-tuning evaluation per-task results. Values are task progress (max 100); bold indicates row maximum. All three models use their respective official pretrained weights, fine-tuned and evaluated under identical data (~500 trajectories per task) and protocols.

Task	Cat.	Wall-OSS-0.5	$\pi_{0.5}$	Dream.
Color Block Sort.	Manip.	**96**	42	27
Ring Stacking	Manip.	**91**	60	27
Spoon-in-Bowl	Manip.	**80**	43	54
Obj.-to-Basket	Manip.	74.8	37	**97.8**
Glasses Rack	Manip.	66	**87**	37
Cup Triangle	Manip.	**58**	18	25
Drawer Org.	Manip.	**52**	7	7
Power Cord	Manip.	**50**	21	24
Water Pouring	Manip.	**25**	19	12
Pencil Case	Manip.	18.5	16	**26**
Fruit Basket	Reas.	86	**94**	45
Earphone Sort.	Reas.	**82**	73	66
Object Matching	Reas.	44.5	**51.5**	13.5
Shape Sorting	Reas.	**68**	63	36
Seq. Button	Reas.	**16**	13	3

A.4 Multi-Task Fine-Tuning Per-Task Detailed Results

Table 9 Multi-task fine-tuning per-task detailed results. Values are task progress (max 100). Empty cells indicate the task was not included in that configuration.

Task	Cat.	19-task	10-task	5-task	1-task	350k pretrain	400k pt/ft
Cup Grasping	Rigid	81	91	82	82	56	64
Package Sorting	Long	82.75	50.75	50.8		68.75	48.5
Block Sorting	Sem.	100	100	100		96	100
Switch Pressing	Sem.	76	53	70		38	55
Flower Arranging	Long	79	79	67		57	51
Table Setting	Def.	18	18			10	9
Charger Plugging	Fine	35	24			6	9
Fruit Sorting	Sem.	90	91			61	96
Number Ordering	Sem.	59	66			56	54
Paper Shredding	Def.	27	27			18	18
Cup Triangle Stack.	Long	60					58
Spoon-in-Bowl	Rigid	98					80
Glasses Rack	Rigid	65					66
Ring Stacking	Rigid	89					91
Color Block Sort.	Sem.	97					96
Water Pouring	Fine	27					25
Power Cord	Fine	36					50
Obj.-to-Basket	Sem.	100					74.8
Pencil Case	Fine	26.5					18.5

A.5 Evaluation Task Descriptions and Scoring Rubrics

Real-robot evaluation in this report covers 31 tasks. All tasks follow a unified step-wise scoring protocol: each task has a maximum score of 10, accumulated step by step according to key manipulation stages. We define task progress = actual score/maximum score $\times$ 100 as the fine-grained evaluation metric. Each task is evaluated over 10 trajectories; success rate is defined as the proportion of trajectories that fully complete all steps. Details can be found in table 10.

Table 10 Real-robot evaluation task descriptions and scoring rules.

Task	Description	Scoring Rule
Block Sorting	Sort 5 colored blocks by color into matching trays	2 pts per correct placement, 5 blocks total
Fruit Sorting	Grasp specified fruit per instruction and place into designated tray	Grasp (4), correct tray (5), retract (1)
Number Ordering	Arrange number cards in correct sequence based on a pattern	1 pt per correct card (7 total), 3 bonus for all correct
Switch Pressing	Move the arm to the light switch and press it	Move to position (3), press switch (4), retract (3)
Ring Stacking	Place a ring object onto a vertical pole	Grasp ring (3), move to pole (3), stack (3), retract (1)
Cup Grasping	Place a cup onto a plate; must first push plate to center	Push plate (3), upright cup (2), pick up (2), place (2), retract (1)
Towel Folding	Sequentially pick up, spread, fold, and place 2 towels	Per towel: pick up (1), spread (1.5), fold (1.5), place (1)
Table Setting	Set a Western-style table: fold napkin, place plate, cutlery, etc.	Fold napkin (2), 1 pt per step (7 steps)
Paper Shredding	Take paper from rack and feed into shredder	Per sheet: grasp (1) + feed (2); 3 sheets, retract (1)
Charger Plugging	Pick up charger plug and insert into outlet; sub-cm alignment	Pick up (2), grip (3), insert (4), retract (1)
Flower Arranging	Pick up 3 flowers and place into a vase	Per flower: grasp (1.5) + place (1.5); 3 flowers, retract (1)
Package Sorting	Extract packages, flip barcode up, place in area	Per pkg: pick up, place, barcode up; 6 packages
Toy Basket Plcmt.	Pick up 4 toys and place into basket without tipping	2 pts per toy (4), basket stable (1), retract (1)
Rope Tightening	Bimanual: grasp slack rope and pull taut	Each hand approaches (2), grasp rope (4), pull taut (2)
Bean Pouring	Pour beans from bag into basin; bimanual	Move basin (2), bag above (2), grasp bottom (3), pour (3)
Table Wiping	Pick up cloth and wipe grease stains	Approach (2), pick up (2), wipe all (5), return (1)
Pot Lid Covering	Pick up pot lid and place on pot	Approach (2), pick up lid (3), cover (4), return (1)
Cup Triangle Stack.	Invert 3 cups in triangle: 2 bottom, 1 top	Per cup: pick up (1) + position (1–3); top (3), retract (1)
Spoon-in-Bowl	Move bowl to center, place spoon in bowl	Bowl (2), move (2), spoon (2), place in bowl (3), retract (1)
Glasses Rack Plcmt.	Pick up glasses, adjust, place on rack	Pick up (2), center (2), adjust (2), rack (3), retract (1)
Drawer Organization	Place items into correct drawer levels by category	Per drawer: open (1) + place (1) + close (1); 3 drawers, retract (1)
Color Block Sort.	Place RGB blocks onto matching color patches	3 pts per correct, 3 blocks, retract (1)
Water Pouring	Pour water from kettle into cup	Move cup (2), kettle (2), pour (3), set down (2), retract (1)
Power Cord Plug.	Bimanual: left picks cable, passes to right, inserts	Left grasp (3), right receive (3), insert (3), retract (1)
Obj.-to-Basket	Pick up 4 objects and place into basket	Per object: pick up (1) + place (1.25); 4 objects, retract (1)
Pencil Case Pack.	Bimanual: open zipper, insert items, close	Open (3.5), 1 pt per item (3), zip (1.5), retract (1)
Earphone Sorting	Find earphones in clutter, place in basket	Identify (2), pick up (3), place (4), retract (1)
Shape Sorting	Sort 3 objects by shape to positions	3 pts per correct, 3 objects, retract (1)
Seq. Button Press	Press 3 buttons in instructed order	3 pts per correct press, 3 buttons, retract (1)

Continued on next page

Task	Description	Scoring Rule
Object Matching	Match object pairs, place at positions	Per pair: grasp (2) + place (2.5); 2 pairs, retract (1)
Fruit Basket Plcmt.	Place specified fruits into basket	3 pts per correct fruit, 3 fruits, retract (1)