MolmoAct2
Action Reasoning Models for Real-World Deployment

Haoquan Fang[♥1,2*] Jiafei Duan[♥1,2,3*]

Donovan Clay[♥1,2] Sam Wang[♥1,4] Shuo Liu[♥2] Weikai Huang[♥1,2] Xiang Fan[♥1,2] Wei-Chuan Tsai[♥2]
Shirui Chen[♥1,2] Yi Ru Wang[♥1,2] Shanli Xing[♥2]

Jaemin Cho[1,2,5] Jae Sung Park[1] Ainaz Eftekhar[1,2] Peter Sushko[1] Karen Farley[1] Angad Wadhwa[2]
Cole Harrison[6] Winson Han[1] Ying-Chun Lee[2] Eli VanderBilt[1] Rose Hendrix[1] Suveen Ellawela[7]
Lucas Ngoo[7]

Joyce Chai[8] Zhongzheng Ren[♥1,2,9] Ali Farhadi[♥1,2] Dieter Fox[♥1,2] Ranjay Krishna[♥1,2]

[1]Allen Institute for AI, [2]University of Washington, [3]National University of Singapore, [4]University of Pennsylvania, [5]Johns Hopkins University, [6]Amazon, [7]Cortex AI, [8]University of Michigan, [9]University of North Carolina at Chapel Hill

* denotes equal contribution in no particular order. ♥ marks core contributors. See full author contributions here.

🤗 **MolmoAct2:** `MolmoAct2-Finetune` `MolmoAct2` `MolmoAct2-Think` `MolmoAct2-Pretrain` `Molmo2-ER`

MolmoAct2 Data: `MolmoAct2-BimanualYAM` `MolmoAct2-SO100/101` `MolmoAct2-DROID` `Molmo2-ER`

Code: `allenai/molmoact2`

Blog: `allenai.org/blog/molmoact2`

Abstract

Vision-Language-Action (VLA) models aim to provide a single generalist controller for robots, but today's systems fall short for real-world deployment. Frontier models are closed; open-weight alternatives are tied to expensive hardware; reasoning-augmented policies pay prohibitive latency for their grounding; and fine-tuned success rates remain below the threshold for dependable use. We present MolmoAct2, a fully open action reasoning model built for practical deployment, advancing its predecessor, MolmoAct along five axes. (1) MolmoAct2 is built on top of our new Molmo2-ER, a VLM backbone specialized for spatial and embodied reasoning, trained on a 3.3M-sample corpus with a specialize-then-rehearse recipe. (2) We release three new robot datasets spanning low-to-medium cost platforms: MolmoAct2-BimanualYAM Dataset, 720 hours of teleoperated bimanual trajectories that constitute the largest open bimanual dataset to date; MolmoAct2-DROID Dataset, a quality-filtered Franka subset of DROID; and MolmoAct2-SO100/101 Dataset, a quality-filtered SO-100/101 subset. (3) We train and release MolmoAct2-FAST Tokenizer, an open-weight, open-data action tokenizer trained on millions of trajectories across five embodiments. (4) We design a new VLA architecture to graft the discrete-token VLM into the flow-matching continuous-action expert via per-layer key-value (KV) conditioning. (5) we propose MolmoAct2-Think, an adaptive-depth reasoning variant that re-predicts depth tokens only for scene regions that change between timesteps, retaining geometric grounding at a fraction of prior latency. In the most extensive empirical study of any open VLA to date, spanning 7 simulation and real-world benchmarks, MolmoAct2 outperforms strong baselines including $\pi_{0.5}$, while Molmo2-ER surpasses GPT-5 and Gemini Robotics ER-1.5 across 13 embodied-reasoning benchmarks. We release model weights, training code, and complete training data.

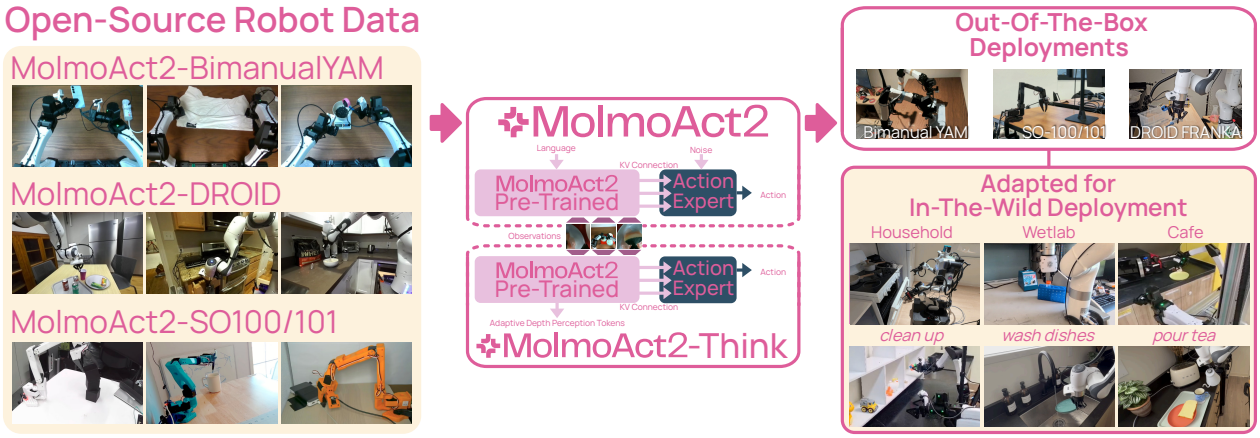

Figure 1 Overview of MolmoAct2. MolmoAct2 is a fully open action reasoning model for real-world deployment. From a suite of high-quality robot datasets that we collect, filter, and curate at scale across three platforms spanning the low-to-medium cost range (**left**), we train MolmoAct2 and its adaptive-depth reasoning variant MolmoAct2-Think, coupled to the VLM backbone through per-layer KV conditioning (**center**). The resulting models deploy out-of-the-box on bimanual YAM, SO-100/101, and DROID Franka, and adapt to in-the-wild tasks such as cleaning up, washing dishes, wetlab automation, and pouring tea (**right**).

1 Introduction

Physical intelligence is fundamentally organized around perception and action (Tversky, 2025). Rather than reasoning over abstract internal computation, human think by constructing spatial representations, simulating actions, and interacting with the world through their bodies (Tversky, 2019, 2013). Although we conflate today's robot foundation models as displaying such intelligence; from a cognitive science perspective, they remain incomplete models of intelligence. They often lack structured spatial representations Qu et al. (2025), rely on heavyweight internal reasoning processes that impede real-time interaction (Lee et al., 2025; Kim et al., 2026a; Zhu et al., 2025), and are difficult to adapt or extend due to limited openness (Black et al., 2024; Intelligence et al., 2025).

Recent work has shown promise that reasoning processes improve performance but the improvements come at the cost of high inference latency. Recent systems including MolmoAct (Lee et al., 2025) and others (Zhao et al., 2025; Sun et al., 2024; Zheng et al., 2024) have shown that grounded spatial reasoning, predicted goal images, point trajectories, or full world-model rollouts improve both action quality and interpretability. In current implementations, however, this reasoning dominates inference latency: hundreds of tokens or entire predicted frames must be generated before a single action is emitted; emerging world models (Kim et al., 2026a; Zhu et al., 2025) compound the problem with heavyweight per-step rollouts. The very mechanism intended to make policies more reliable thus renders them too slow for closed-loop control.

Reasoning, by itself, is only as good as the underlying foundation model that uses the reasoning process. Most frontier robot policies remain closed off; open-source alternatives are embodiment-specific and hard to adapt to new tasks or embodiments. Frontier vision-language-action (VLA) models (Team et al., 2024, 2025b; Team, 2026b,a) are effectively closed systems: their training data, recipes, and model weights are proprietary. The few exceptions release weights alone, withholding the data and training procedures needed to reproduce or extend them. This opacity both impedes scientific progress and prevents practitioners from adapting these models to their own robots or fine-tuning on in-house demonstrations. The few open-weights VLAs (Black et al., 2024; Intelligence et al., 2025) that can be run out-of-the-box are tied to expensive or specialized robot platforms beyond the reach of most academic labs and independent researchers. This constrains not only who can use these models, but also the diversity of settings in which they can be evaluated and improved. Zero-shot performance remains brittle, and even after task-specific fine-tuning, success rates on realistic tasks fall well below the threshold required for dependable deployment.

We present **MolmoAct2**, an action reasoning model built for real-world deployment: fully open, deployable out-of-the-box on multiple embodiments, performant, and capable of fast, interpretable reasoning (Figure 1).

MolmoAct2 improves over its predecessor MolmoAct (Lee et al., 2025) along five axes vital for a strong action reasoning model: (1) a stronger open embodied-reasoning VLM backbone, (2) new open-source training datasets, (3) MolmoAct2-FAST Tokenizer, an open-source multi-embodiment action tokenizer, (4) a new VLA architecture design, and (5) a new adaptive reasoning paradigm for efficient inference.

First, we train **Molmo2-ER** (Sec. 2), a VLM specialized for spatial and embodied reasoning. General-purpose VLMs rarely train on or test the embodied skills a robot policy needs: they need to understand metric distances, free space, cross-view object tracking, and scene geometry. We address this by training Molmo2-ER on a 3.3M-sample spatial–embodied corpus using a specialize-then-rehearse recipe.

Second, existing open robot datasets remain fragmented across embodiments, uneven in quality, and too small or noisy to support reliable multi-embodiment action learning. To support training action models on top of our VLM backbone, we release three action datasets (Sec. 3) targeting three platforms across a low-to-medium cost range: **MolmoAct2-BimanualYAM Dataset**, 720 hours of teleoperated YAM trajectories spanning tabletop and household tasks—the largest open bimanual dataset to date; **MolmoAct2-SO100/101 Dataset**, a filtered subset of internet SO-100/101 data with mislabeled and low-quality trajectories removed; and **MolmoAct2-DROID Dataset**, a quality-filtered Franka subset of DROID. For the two filtered datasets, we also re-annotate the language instructions for improved diversity and accuracy.

Third, existing action tokenizers are either closed, tied to specific action spaces, or insufficiently documented, making it difficult to train reproducible discrete-action VLAs across embodiments. To make the trajectories in our data usable under a discrete autoregressive objective, we introduce MolmoAct2-FAST Tokenizer, an open-weight, open-data implementation following FAST (Pertsch et al., 2025). We train and release its weights along with the millions of trajectories across five embodiments used to train it. MolmoAct2-FAST Tokenizer compresses one second of 32-D continuous actions into a compact discrete sequence.

Fourth, discrete-token VLMs provide strong reasoning grounding, but their native output space is poorly matched to the continuous, high-frequency trajectories required by robot control. A new architecture is therefore needed to connect the discrete reasoning capacity with smooth continuous actions. We design a new architecture that conditions each layer of the continuous action expert on the keys and values from the corresponding VLM layer. This design is in contrast to existing VLAs with action experts in two ways. First, we use a DiT-style transformer trained with flow matching objective, giving the continuous controller a modern denoising-transformer architecture that has proven more effective than many alternatives for diffusion and flow-based generative modeling. Second, instead of conditioning the expert on VLM hidden states, we condition each expert layer on the corresponding VLM keys and values. This preserves a similar compute profile while exposing the attention state used by the VLM itself, which has been shown to be more effective in our ablation studies.

Fifth, prior reasoning-based VLAs improve action quality by generating dense intermediate representations at every step Lee et al. (2025), but this repeats nearly identical computation across largely static scenes and makes inference too slow for real-time control. We introduce **MolmoAct2-Think**, the reasoning variant of MolmoAct2, which performs *adaptive* depth reasoning by autoregressively predicting only the tokens for scene regions that change between timesteps. This exploits trajectory-level temporal redundancy to reduce latency proportional to the static scene fraction while retaining the geometric grounding that significantly improves model performance.

To understand the capabilities of MolmoAct2 and assess its readiness for real-world deployment, we conduct the most extensive empirical study of any open VLA to date, spanning 7 environment benchmarks across both simulation and the real world. Across all of them, MolmoAct2, with MolmoAct2-Think, outperforms every strong baseline. We show that MolmoAct2's fine-tuned checkpoints, MolmoAct2-DROID and MolmoAct2-SO100/101, can be deployed out of the box on their respective embodiments without any additional fine-tuning, significantly surpassing $\pi_{0.5}$ (Intelligence et al., 2025) (Sec.6.2). We further demonstrate that MolmoAct2 is built on one of the strongest embodied-reasoning VLM backbones available: Molmo2-ER surpasses models such as GPT-5 (Singh et al., 2025) and Gemini Robotics Embodied Reasoning (ER)-1.5 (Team et al., 2025a) on 13 standard embodied-reasoning benchmarks (Sec.6.1).

For rapid real-world deployment via efficient fine-tuning to new embodiments, MolmoAct2 opens large performance gaps over strong general-purpose VLAs such as $\pi_{0.5}$ (Intelligence et al., 2025) not only on the

Table 1 MolmoAct2 multimodal web data training corpus (combining data from Molmo2, Molmo2-ER, and Tulu-3). Sizes denote the number of samples used in our mixture (we subsample several datasets to balance the mixture). Sampling weights are the marginal proportion of each group within the non-robotics portion of the pretraining mixture.

	#Samples	Weight		#Samples	Weight
Molmo2 Dataset			**Molmo2-ER Dataset**		
Image QA	2.4M	0.115	Image Embodied QA	1.3M	0.11
Video QA	2.4M	0.092	Image Pointing	780K	0.11
Image Pointing	1.1M	0.046	Image Detection	100K	0.01
Video Pointing	370K	0.069	Video Embodied QA	703k	0.10
Video Tracking	800K	0.069	Multi-image/ Ego-Exo	700K	0.09
Captions/Long QA	1.2M	0.069	Abstract Reasoning	150K	0.04
Subtotal	**8.3M**	**0.46**	**Subtotal**	**3.3M**	**0.46**
Tulu-3 SFT Dataset	**980k**	**0.08**	**Total**	**12.5M**	**1.0**

two mainstream simulation benchmarks LIBERO (Liu et al., 2023) and RoboEval (Wang et al., 2025b), but also on a comprehensive evaluation suite of 8 real-world tasks on the bimanual YAM setup (Sec.6.3). Our ablations demonstrate that MolmoAct2-Think models yield further gains over MolmoAct2 while providing additional interpretability that aids both diagnosis and performance (Sec.6.4).

MolmoAct2 is fully open in every respect, and beyond that, is capable of supporting real-world deployment for practical tasks: we release the model weights, training code, and the complete training dataset. We aim for MolmoAct2 to be more than an academic robotics foundation model; we want it to be a model that can be deployed in real-world workflows and deliver meaningful social impact.

2 Molmo2-ER

Existing VLM backbones are optimized for semantic image understanding rather than the metric, geometric, and temporally grounded reasoning required for robot control. We develop Molmo2-ER as a strong VLM backbone for embodied reasoning (ER). We finetune Molmo2 (Clark et al., 2026) for specialized embodied perception skills that downstream action reasoning depends on, including scene understanding, pixel-accurate pointing, multi-image and egocentric reasoning, exocentric correspondence, and video temporal reasoning. Molmo2-ER outperforms every open-weight baseline as well as the strongest closed-source models, including Gemini Robot-ER 1.5 Thinking and GPT-5, on 9 of 13 established embodied reasoning benchmarks (Table 3), reaching an overall average of 63.8% and improving over its Molmo2 starting point by 17 points. The remainder of this section describes the new training data we introduce for Molmo2-ER (summarized in Table 1) and the two-stage training recipe used to inject these skills.

2.1 Training data

On top of Molmo2's original multimodal pre- and mid-training data (Clark et al., 2026), we curate a new embodied reasoning corpus of approximately 3.3M samples spanning six complementary capability pillars: single-image embodied QA, image pointing, image detection, video embodied QA, multi-image and ego–exo reasoning, and abstract embodied reasoning. Each pillar is covered by two or three datasets with diverse supervision sources (simulator ground truth, 3D-annotated real scans, template-generated QA, and a small amount of LLM-generated chain-of-thought), so that the model is exposed to a wide distribution of spatial reasoning phenomena rather than over-fitting to a single template style. The composition of this corpus is summarized in Table 1; we briefly describe each constituent below.

Image embodied QA. We assemble a mixture of image QA sources chosen to cover complementary axes of spatial competence rather than to maximize any single one. SAT (Ray et al., 2025) supplies simulator-grounded supervision for *dynamic* reasoning (egocentric motion, perspective taking, action consequences) that is hard to harvest from static web data. RoboPoint-QA (Yuan et al., 2024) contributes general VQA breadth to

guard against forgetting base perception during spatial fine-tuning. RefSpatial (Zhou et al., 2025) bridges web images, real indoor scans, and procedural simulation, and is our main source of chain-of-thought referring—the bridge from spatial language to the pointing actions used downstream; we draw 250K multiple choice questions, 250K Chian-of-thought, and 80K pointing examples. VST-P (Yang et al., 2025c) normalizes inputs onto a uniform virtual camera, giving metric-consistent depth, distance, direction, and size supervision (200K single-image + 200K cross-view). VSI-590K (Yang et al., 2025d) extends coverage to in-the-wild robotics and tour footage via 3D-grounded label propagation (200K images and 300K videos). Together these span static/dynamic, synthetic/real, and single-/multi-view regimes.

Video embodied QA. To extend reasoning across time we pair two complementary sources. SIMS-VSI (Brown et al., 2025) gives clean simulator labels for distance, direction, count, and temporal-order questions over agent trajectories (203K). RoboVQA (Sermanet et al., 2023) covers the other end of the distribution: human-annotated long-horizon embodied video targeting planning, affordance, and future prediction—i.e. the question types that map most directly onto policy behavior; we use a 200K subset.

Pointing and object detection. Pointing is our primary action interface, so we deliberately oversample pixel-accurate localization. Beyond RefSpatial's pointing split, we use the full RoboPoint procedural pointing corpus (700K normalized (x, y) targets for object reference and free-space selection) together with its 100K LVIS-sourced detection split (Yuan et al., 2024), which ground spatial language directly in the coordinate format the policy consumes.

Multi-image and ego–exo correspondence. Embodied policies routinely reconcile multiple camera views and switch between first- and third-person frames, an ability under-served by single-image corpora. We therefore include SenseNova-SI (Cai et al., 2026) (500K subset), whose distinguishing emphasis is multi-image and egocentric–exocentric correspondence (Grauman et al., 2024), together with the 200K cross-view split of VST-P.

Abstract embodied reasoning. Finally, we add two synthetic diagnostics to harden compositional reasoning in a low-bias setting. CLEVR (Johnson et al., 2016) (50K) targets compositional attribute–relation reasoning. GRiD-3D (Lee et al., 2022) (100K) specifically isolates *object-intrinsic* relative direction (front/left of a referent's own frame, not the camera's)—a frame-of-reference distinction that matters for instruction following but is rarely labeled in natural data.

2.2 Specialize-then-rehearse training recipe

To avoid the redundant compute of re-running MOLMO2's full multimodal training with our new corpus integrated, we build on the released MOLMO2 checkpoint with a two-stage *specialize-then-rehearse* recipe.

Stage 1: Embodied specialization. Starting from the MOLMO2–4B mid-training checkpoint (Deitke et al., 2024; Yang et al., 2025a), we fine-tune for 20K steps on the MOLMO2-ER corpus augmented with 8% Tulu-3 (Lambert et al., 2025) text-only data to preserve language competence, using sequence length 4,200 and a global batch size of 64 (device batch size 4 across 2 nodes × 8 H100 GPUs). This stage rapidly moves the model onto the embodied data manifold: pointing accuracy, video embodied QA, and multi-image reasoning all improve sharply.

Stage 2: Joint refinement. We continue training the Stage 1 checkpoint for 1.5K additional steps on a mixture that interleaves our embodied corpus with MOLMO2's original multimodal mid-training data (general VQA, captioning, academic benchmarks, tracking, and MOLMO2 pointing). Holding the NLP rate at 8%, the remaining 92% budget is split as $p \cdot 0.92$ embodied and $(1-p) \cdot 0.92$ general, with each side's internal proportions preserved. Sweeping $p \in \{0.30, 0.50, 0.70, 0.90\}$, we find $p = 0.5$ yields the best Pareto trade-off between the embodied-reasoning benchmarks in Table 3 and MOLMO2's general benchmarks. To accommodate the long multi-image and long-video examples in the general mixture, Stage 2 uses a longer sequence length (16,384 vs. 4,200 in Stage 1) with per-device batch size reduced to 1; all other hyperparameters follow MOLMO2.

3 Data

Training a generalist vision-language-action model requires data that is simultaneously large in scale, diverse across embodiments, tasks, and scenes, and high in quality. The robotics community has released several substantial public corpora toward this goal—most notably the Open X-Embodiment (OXE) mixture (O'Neill et al., 2024), DROID (Khazatsky et al., 2024), and a rapidly growing collection of LeRobot community datasets contributed by SO-10x users (Shukor et al., 2025). While each is valuable, none of these resources individually, nor their union, is sufficient for training a model intended for real-world deployment. OXE offers breadth across embodiments but its constituent datasets vary widely in quality, control conventions, and language-annotation fidelity; DROID provides scale on a single Franka platform, but a substantial fraction of its episodes contain idle segments, failed attempts, or repetitive task instructions; and crowd-sourced LeRobot data, although rich in embodiment and scene diversity, frequently contains placeholder annotations such as "lerobot_test" alongside genuine demonstrations (Shukor et al., 2025). Critically, no public corpus contains high-quality bimanual manipulation data on the platform we target for deployment, and the tasks that are covered are often collected with limited variation in scene configuration, object instances, or spatial variation that mimic the realistic of real-world tasks for deployment.

To close these gaps, we assemble MolmoAct2's training mixture from three complementary sources, summarized in Fig. 2. First, we collect MolmoAct2-BimanualYAM Dataset, a new bimanual manipulation dataset emphasizing task repeatability and task, object, and scene diversity across a wide range of useful behaviors. Second, we curate and filter two large public corpora, MolmoAct2-SO100/101 Dataset (drawn from community LeRobot data) and MolmoAct2-DROID Dataset (drawn from DROID), using structural, licensing, and quality-based filtering pipelines, and re-annotate their language instructions to improve both accuracy and diversity. Third, we co-train with a targeted subset of academic robotics data for additional embodiment breadth, together with multimodal and embodied-reasoning data to preserve the broad visual and linguistic competence of the underlying VLM. The remainder of this section describes each component of the mixture in turn, together with the language re-annotation pipeline shared across our robot datasets.

MolmoAct2 is trained on a diverse set of datasets spanning robot data, multimodal reasoning, and embodied reasoning, summarized in Fig. 2. Additionally, we collected MolmoAct2-BimanualYAM Dataset, curated MolmoAct2-SO100/101 Dataset, and filtered MolmoAct2-DROID Dataset for use in pre-training and post-training. Below, we describe each dataset and detail its respective collection, curation, or filtering process.

3.1 MolmoAct2-BimanualYAM Dataset

To support real-world deployment, we introduce MolmoAct2-BimanualYAM Dataset, an open-source robot manipulation dataset that emphasizes task repeatability, task diversity, and object diversity across a broad range of useful behaviors spanning household, factory, and coffee-shop settings. All data is collected on our custom bimanual YAM (Yet Another Manipulator) setup, shown in Figure 3. MolmoAct2-BimanualYAM Dataset is designed to deliver both scale and quality. It contains over 28 unique real-world tasks from folding clothes and untangling cables to bussing tables, scanning groceries, and packing medication, each captured with substantial variation in scene configuration, object instances, and object placement. In total, the dataset comprises 34.5k robot demonstrations totaling over 720 hours of robot data, collected over a two-month period. Data collection was supported by Cortex AI, with strict protocols governing the number of permitted failure retries and the maximum duration of no-op segments to ensure consistently high data quality. Further details on MolmoAct2-BimanualYAM Dataset are provided in the appendix.

3.2 MolmoAct2-SO100/101 Dataset

SO-100/101 is a low-cost robotic platform from Hugging Face that is accessible to a broad community of users. As a result, we utilize the diverse open-source SO-10x data collected by the community to improve our model's capability for deployment in the wild (Chen, 2025). Specifically, we curate MolmoAct2-SO100/101 Dataset from 1,222 public LeRobot datasets contributed by 377 users. This corpus contains 38,059 robot demonstration episodes, 19.8M frames, and approximately 184 hours of interaction data.

To prioritize quality while preserving diversity, we apply a four-stage filtering pipeline: (i) structural validity

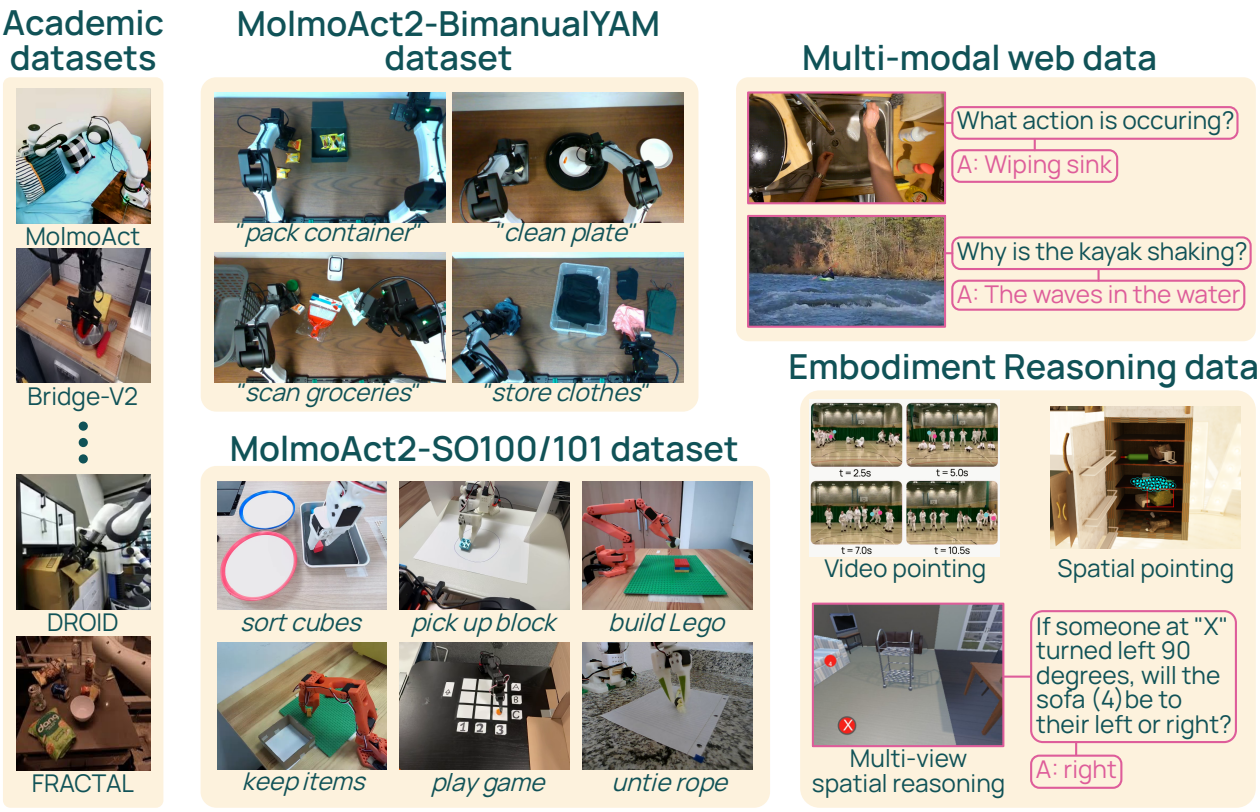

Figure 2 Overview of the MolmoAct2 training data. Our data mixture combines public academic robot datasets, MolmoAct2-BimanualYAM Dataset, MolmoAct2-DROID Dataset, MolmoAct2-SO100/101 Dataset, multimodal web data, and embodied-reasoning data.

checks (required schema fields, valid action/state tensors, no NaN/corrupt samples), (ii) removal of eval-style datasets, (iii) license/codebase eligibility checks, and (iv) a final TOPReward quality gate (Chen et al., 2026). In stage (iv), we keep datasets whose mean TOPReward over the last 3 sampled episodes is above a threshold obtained by averaging the TOPReward over a collection of human-audited high-quality datasets.

The filtered data spans both SO-100 and SO-101 embodiments, multiple camera configurations, varied object manipulation tasks, and diverse real-world collection environments. Compared with centrally collected robot datasets, this community-sourced corpus provides broader coverage of user setups, backgrounds, objects, and task annotations, making it a useful source of embodiment and environment diversity for improving real-world robustness.

3.3 MolmoAct2-DROID Dataset

DROID (Distributed Robot Interaction Dataset) (Khazatsky et al., 2024) is a large-scale in-the-wild robot manipulation dataset, collected across a wide range of real-world deployment scenarios with a unified Franka robot setup. To ensure the quality of our training data, we leverage the supplementary annotations released in the accompanying HuggingFace repository (Pertsch et al., 2024) to filter the original DROID release. Specifically, we use (i) the extended language annotations, which provide three natural-language instructions for 95% of the 75k successful episodes, and (ii) the provided idle-frame filter, which retains only contiguous non-idle action segments of at least one second. Furthermore, we did language re-annotation for the filtered DROID dataset before using for training. The resulting subset, which we refer to as MolmoAct2-DROID Dataset, contains 74,604 valid episodes comprising a total of 17,758,044 frames. Each episode in this subset is marked as successful, contains at least one valid language instruction, and is free of significant pauses.

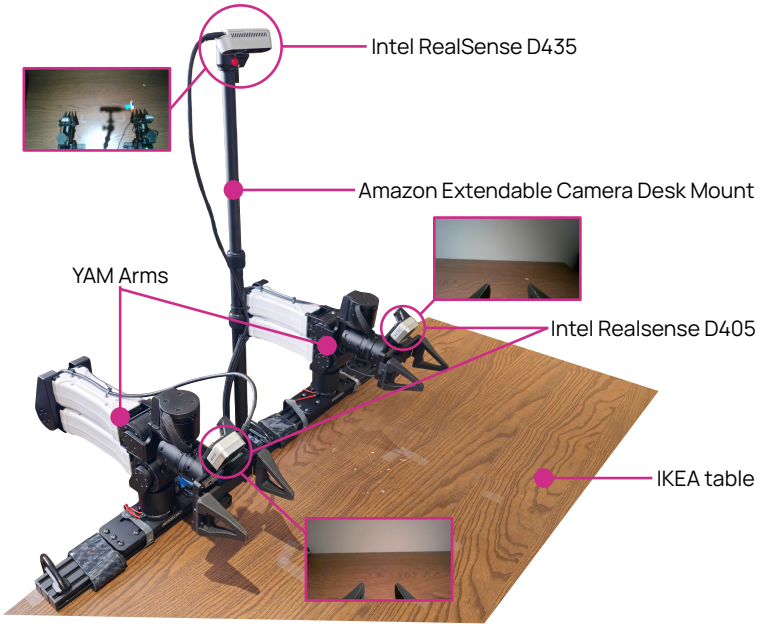

Figure 3 MolmoAct2-BimanualYAM Dataset collection setup. Our standardized MolmoAct2-BimanualYAM Dataset collection setup. Every component is readily available for purchase, and the total cost of the entire setup is under $6,000 USD.

3.4 Language annotation pipeline

Following standard practice in imitation learning, each demonstration in our datasets is paired with a language instruction that describes the task completed by the teleoperator. However, we find that these annotations often are imperfect along two key axes. First, datasets where a specific set of tasks is collected repeatedly at scale often have repetitive language instructions. For example, the dataset collected in Jang et al. (2022) contains 104 unique instructions for 39350 episodes, representing 0.26% unique instructions in the dataset. Secondly, as found in prior work (Shukor et al., 2025), crowd-sourced datasets like the MolmoAct2-SO100/101 Dataset often have inaccurate or meaningless task annotations, like "lerobot_test" and "Test run".

To improve both the diversity and accuracy of the language instructions in our robotics datasets, we use an open-source VLM (Qwen3.5-27B) to re-annotate the datasets. We prompt the VLM with a sample of the frames and the original instruction. The VLM is asked to generate a instruction that describes the demonstration. To increase the diversity of prompts, we randomly sample a number and ask the VLM to make this instruction roughly that many words. Relabeling the robotics datasets through this pipeline doubles the unique labels in the overall from 71121 (22%) to 146485 (46%) in dataset. More details and the prompt can be found in Appendix D.2.

3.5 Academic robotics datasets

To broaden the range of embodiments, camera layouts, task families, and control conventions seen during pre-training, we include additional public academic robotics data. This pool consists of a targeted subset of the Open X-Embodiment mixture (O'Neill et al., 2024), including BC-Z (Jang et al., 2022), BridgeData V2 (Walke et al., 2023), and RT-1 (Brohan et al., 2022), together with MolmoAct Dataset from MolmoAct (Lee et al., 2025). These datasets are used as complementary sources rather than as the primary deployment embodiments, since YAM, SO-100/101, and DROID supply the majority of robot training examples.

3.6 Multimodal datasets

Past research on robotics foundation models has found benefits in incorporating multimodal data into the training mix along with robotics datasets (Lee et al., 2025; Intelligence et al., 2025). We co-train the VLA

with a multimodal data mixture where 46% consists of the MOLMO2-ER dataset mixture described in Sec. 2.1, 46% consists of the MOLMO2 dataset mixture (Clark et al., 2026), and 8% consists of the text-only data from Tulu-3 (Lambert et al., 2025). The high level composition of this data can be seen in Table 1.

4 MolmoAct2

MOLMOACT2 follows a three-stage training pipeline, with the post-training architecture summarized in Fig. 4. Pre-training (Sec. 4.1) adapts the Molmo2-ER vision-language backbone into a discrete autoregressive robot policy, using the MOLMOACT2-FAST TOKENIZER (Sec. 4.1.1) to map continuous trajectories into a compact action vocabulary under a unified next-token recipe (Sec. 4.1.2). Post-training (Sec. 4.2) attaches a flow-matching action expert with per-layer KV conditioning on the autoregressive backbone (Sec. 4.2.1), and co-trains discrete and continuous action supervision (Sec. 4.2.2) to produce continuous control. Deployment (Sec. 4.3) then covers embodiment-specific fine-tuning (Sec. 4.3.1) and inference optimization (Sec. 4.3.2). Shared implementation details for training infrastructure, packing, prompt formatting, and stage-level hyperparameters are provided in Appendix B.

MOLMOACT2 is designed around a practical tension: we want to preserve the scaling properties and general visual-language competence of a pretrained VLM, while producing precise continuous robot actions across heterogeneous embodiments. Directly training a VLM and a continuous action expert together from the beginning makes optimization unnecessarily difficult: the model must simultaneously learn a robot-aware token interface, align new state/action embeddings, and fit a flow-matching controller. We therefore use a three-stage pipeline. Pre-training first turns MOLMO2-ER into an action-aware VLA using only autoregressive supervision, which gives the backbone aligned robot, state, and action-token embeddings under the same next-token objective used by the base VLM. Post-training then attaches the continuous action expert to this already action-aware backbone, allowing the flow-matching controller and its VLM conditioning interface to align quickly on a broad robot-data mixture. We keep post-training separate from embodiment-specific fine-tuning for efficiency: the post-training mixture is intentionally diverse across datasets, robots, control rates, camera layouts, and task families, so extending it until the model is directly deployment-ready for every target embodiment would require a long and expensive training stage. Instead, post-training produces a general continuous-control base model, and fine-tuning adapts that model quickly to a specific embodiment, environment, and use case while keeping the same architecture and output interfaces.

4.1 Pre-training

MOLMOACT2-PRETRAIN adapts the MOLMO2-ER vision-language backbone into a discrete autoregressive robot policy while keeping the Molmo2 token interface intact. Images and video frames are still encoded by a ViT, pooled and projected by the vision-language connector, and passed to the LLM together with text. Robot examples extend this sequence with two additional token streams: state tokens that describe the current robot configuration, and action tokens that describe the future one-second motion.

This subsection focuses on the two ingredients needed to make that formulation practical. First, we describe MOLMOACT2-FAST TOKENIZER, which converts continuous, embodiment-specific robot trajectories into compact discrete action tokens. Second, we describe the pre-training recipe that lets these robot sequences be mixed with standard multimodal data: single-crop visual inputs, a small number of sampled video frames, setup/control and action-output markers, state tokenization, and packed training sequences. Together, these choices preserve a single next-token prediction objective across text, vision-language, state, and action targets, without introducing a separate continuous action head at this stage.

4.1.1 MolmoAct2-FAST Tokenizer

Robot actions are continuous, embodiment-specific, and often produced at different control rates, so they cannot be inserted into a language-model pre-training stream directly. We therefore train MOLMOACT2-FAST TOKENIZER, an open-weight and open-data action tokenizer following FAST (Pertsch et al., 2025). The goal is not to introduce a different tokenization principle, but to make this component fully inspectable and reproducible: existing FAST weights are useful, but their training data mixture is not fully specified. MOLMOACT2-FAST TOKENIZER fills this transparency gap by releasing both the tokenizer weights and the

Table 2 Embodiment mixture for MolmoAct2-FAST Tokenizer training. The tokenizer is trained on one million subsampled action sequences spanning the main robot embodiments used by MolmoAct2, plus smaller sources that broaden control-mode coverage.

Dataset	Mix	Robot	Action representation
MolmoAct2-BimanualYAM Dataset	30%	YAM	Absolute joint
MolmoAct2-SO100/101 Dataset	30%	SO-100/101	Absolute joint
MolmoAct2-DROID Dataset	30%	Franka	Absolute joint
Fractal (Brohan et al., 2022)	3.33%	Google Robot	Delta end-effector
BC-Z (Jang et al., 2022)	3.33%	Google Robot	Delta end-effector
Bridge (Walke et al., 2023)	3.33%	WidowX	Delta end-effector

action mixture used to train it. It maps a one-second action trajectory into a compact sequence of discrete tokens by representing the trajectory with a frequency-domain transform, quantizing the resulting coefficients, and applying byte-pair encoding to produce tokens from a 2048-token action vocabulary.

Unlike the prior FAST tokenizer, whose released weights are not paired with a fully specified training distribution, MolmoAct2-FAST Tokenizer is trained from a transparent action mixture. We subsample one million action sequences across five embodiments, balancing the main MolmoAct2 deployment platforms with smaller sources that broaden the control vocabulary (Table 2). This gives MolmoAct2-FAST Tokenizer coverage over bimanual YAM, SO-100/101, DROID Franka, BC-Z (Jang et al., 2022), BridgeData V2 (Walke et al., 2023), RT-1 (Brohan et al., 2022), and MolmoAct Dataset trajectories, including both absolute joint control and delta end-effector control.

Before fitting the tokenizer, we put all raw robot telemetry into a common action format. Each training sequence corresponds to one second of robot motion, so the number of actions in a chunk is set by the control frequency of the source dataset. Each action in the chunk is padded to 32 dimensions, so embodiments with different action dimensionalities share the same tokenizer input space. Continuous dimensions are normalized with 1–99 percentile statistics, which limits the effect of outliers while preserving the useful dynamic range of each control dimension. Gripper commands are handled separately from this continuous normalization, since they are typically binary or narrow-range open/close signals. This standardized representation allows the same tokenizer to cover both joint-space and end-effector control across multiple robot platforms.

4.1.2 Pre-training recipe

Model. We initialize from the Molmo2-ER checkpoint and keep the Molmo2 vision-language architecture (Clark et al., 2026). Visual observations are encoded by the SigLIP2 ViT (Tschannen et al., 2025), converted into language-model tokens by the connector, and passed to the LLM together with the language instruction and robot-specific text. The connector uses the same Molmo2 design: features from the third-to-last and ninth-from-last ViT layers are pooled into compact image or video-frame tokens and projected into the LLM embedding space. Full backbone and added-token details are given in subsection A.1, with architecture hyperparameters in Table 15.

Data. The training mixture combines multimodal examples with robot trajectories so that the model retains vision-language capability while learning to predict actions; we describe the data sources and curation in Sec. 3. We allocate 10% of sampling to multimodal data and 90% to robot trajectories. Within the robot portion, YAM, SO-100/101, and DROID each receive 30% of the robot sampling weight; the remaining 10% is split across smaller BC-Z, BridgeData V2, RT-1, and MolmoAct Dataset sources. For all visual inputs in this stage, we use a single resized crop rather than high-resolution tiled crops. For videos, we sample at most 8 frames at up to 2 FPS, and we skip examples that exceed the sequence budget, which primarily removes unusually long text examples and videos with long captions. We also apply the image augmentation described in Appendix B during pre-training, combining light geometric perturbations with color jitter and occasional blur. For multi-camera robot episodes, we randomize the input camera order at the episode level to prevent the model from relying on a fixed camera slot. Each robot example also includes setup and

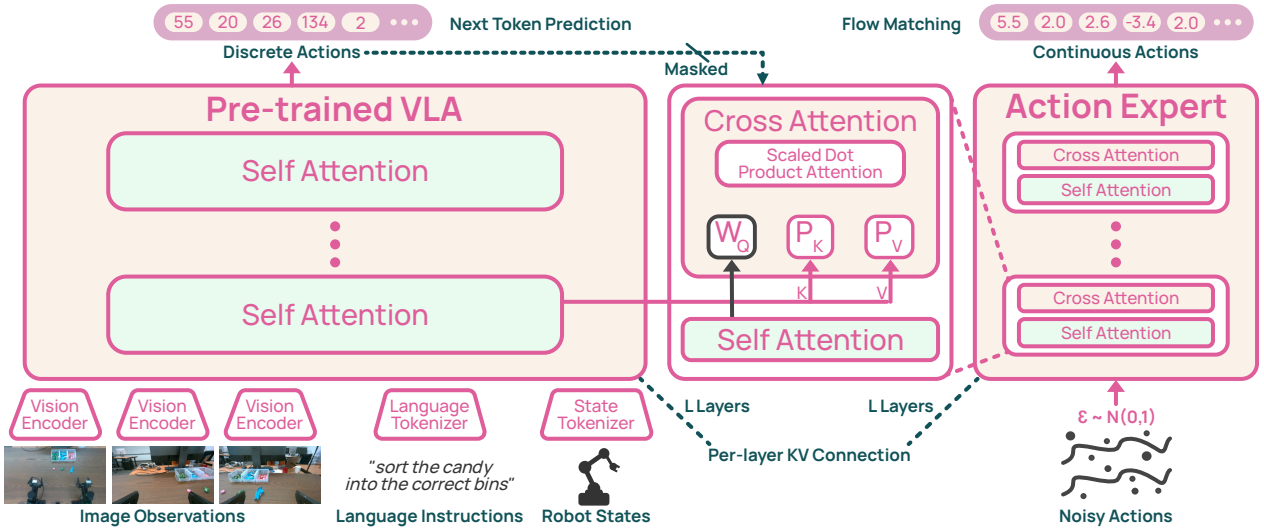

Figure 4 Overview of MolmoAct2. Image observations, language instructions, and robot states are tokenized and processed by a pre-trained VLA backbone with self-attention layers. Post-training attaches a DiT-style action expert with the same number of layers as the backbone. At each layer, the backbone key and value tensors are projected and reused as the key and value inputs to the action expert's cross-attention, enabling per-layer KV conditioning from visual-language context to continuous control. The expert is trained with flow matching to denoise a noisy action trajectory into a continuous robot trajectory. During training, the backbone is also supervised with next-token prediction over discrete action tokens, while the target action-token span is masked from the expert so continuous action prediction cannot condition on the ground-truth discrete actions.

control strings using the same special-token wrappers used during training, e.g., `<setup_start>bimanual yam robotic arms in molmoact2<setup_end>`, and `<control_start>absolute joint pose<control_end>` or `<control_start>delta end-effector pose<control_end>`; the corresponding chat template and output trigger formatting are detailed in Appendix B.

Action and state representation. This stage is deliberately limited to discrete autoregressive supervision: the model predicts action tokens, but does not yet train the continuous action expert used in post-training. Each robot target is represented as a one-second action chunk, so the number of raw actions in the chunk follows the control frequency of the source dataset. Before tokenization, continuous action and state dimensions are normalized with 1–99 percentile statistics; gripper commands are treated separately from this percentile scaling when they are represented as binary or narrow-range open/close signals. Action vectors are padded to a 32-dimensional space and then encoded with the 2048-token vocabulary of MolmoAct2-FAST Tokenizer. Proprioceptive state is represented separately: after normalization, each state value is uniformly discretized into one of 256 state tokens and appended to the prompt before the action target.

Training. We train for 200K steps with a maximum sequence length of 4200 tokens. Examples vary substantially in length: a text-only example may use only a few hundred tokens, while multi-camera robot examples must carry images, states, and action targets. We therefore use on-the-fly packing to combine multiple short examples into one 4200-token sequence. Packed examples share the same forward pass, but their text, visual tokens, state tokens, and action targets remain separated by the training attention mask, so the model does not condition one example on another. Appendix B gives the implementation details for this packing procedure and the shared training stack. We train the vision encoder, connector, language model, and newly added tokens' embeddings with a global batch size of 128 across 64 H100 GPUs for around 5,760 GPU hours. The vision encoder and connector use a learning rate of 5×10^{-6}, and the language model uses 1×10^{-5}. This produces a discrete VLA checkpoint that can later be adapted to continuous control.

4.2 Post-training

MolmoAct2-Pretrain learns a robot policy through the same autoregressive interface used by the VLM: given language, visual observations, setup/control descriptors, and state tokens, it predicts discrete action tokens produced by MolmoAct2-FAST Tokenizer. This makes large-scale robot pre-training simple and stable, but the deployed policy should output continuous action trajectories directly. Post-training therefore adds a flow-matching action expert to the pre-trained VLM, producing the final MolmoAct2 model.

This section describes the two parts of that adaptation. First, we introduce the action expert and its per-layer KV conditioning on the VLM, which lets the continuous controller condition on the full visual-language attention state without replacing the backbone. Second, we describe the post-training recipe: how we keep the pre-training data construction, co-train discrete and continuous action supervision, mask padded or target-leaking tokens, and adjust packing and sequence lengths for the added action-expert compute. Additional model hyperparameters and implementation details are given in subsection A.2, with shared training-system details in Appendix B.

4.2.1 Action expert and per-layer KV conditioning

Post-training adds a continuous action expert on top of the pre-trained autoregressive VLM. We use a DiT-style transformer expert because flow matching and diffusion models have shown that denoising transformers scale well for continuous generation, and because action trajectories are naturally represented as short continuous sequences rather than as text-like token streams. This choice separates responsibilities: the VLM provides visual-language grounding and robot-state context, while the expert models the continuous velocity field needed to transform noisy action chunks into executable trajectories.

The key architectural question is how this expert should receive VLM context. A shallow projection of the final hidden state compresses the backbone into a single residual-stream representation. Instead, MolmoAct2 conditions the expert on the VLM at every layer. Each action-expert block cross-attends to the corresponding VLM layer's keys and values, after lightweight learned projections map the VLM attention state into the expert's cross-attention width. This gives the continuous controller access to the same attention state used by the VLM itself, while preserving a modular interface between the backbone computation and the newly trained action expert.

Given a normalized target action chunk a, Gaussian noise ϵ, and a sampled time $t \in [0, 1]$, we interpolate between noise and data,

$$x_t = (1 - t)\epsilon + ta, \qquad u^\star = a - \epsilon. \tag{1}$$

The expert f_θ predicts the target velocity $u^\star$ from the noisy action chunk, the time embedding, and the VLM context c, which contains the task, visual observations, setup/control descriptors, and discrete state tokens:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{a,\epsilon,t}\left[\left\| m \odot \left(f_\theta(x_t, t, c) - u^\star\right)\right\|_2^2\right], \tag{2}$$

where m masks padded action steps and padded action dimensions. At inference, we start from Gaussian noise and integrate the predicted velocity field to produce a continuous action trajectory.

Architecturally, the expert has the same depth as the VLM, where both of them use $L = 36$ layers. The expert first embeds the current noisy action sequence. Each block then applies action self-attention, cross-attention to the VLM, and an MLP, with the time embedding producing DiT-style shift, scale, and gate parameters for all three residual branches. Schematically, block ℓ computes

$$h'_\ell = h_\ell + g_\ell^{\text{sa}}\,\text{SA}\big(\text{AdaRMS}_\ell^{\text{sa}}(h_\ell, t)\big), \tag{3}$$

$$\bar{h}_\ell = h'_\ell + g_\ell^{\text{ca}}\,\text{CA}\Big(\text{AdaRMS}_\ell^{\text{ca}}(h'_\ell, t), \tilde{K}_\ell, \tilde{V}_\ell\Big), \tag{4}$$

$$h_{\ell+1} = \bar{h}_\ell + g_\ell^{\text{ff}}\,\text{MLP}\Big(\text{AdaRMS}_\ell^{\text{ff}}(\bar{h}_\ell, t)\Big). \tag{5}$$

The action expert uses per-layer KV conditioning rather than hidden-state conditioning. For each VLM layer ℓ, we collect the keys and values $(K_\ell^{\text{vlm}}, V_\ell^{\text{vlm}})$ produced by that layer's self-attention. We then project them into the action-expert width with learned adapter projections P_K and P_V:

$$\tilde{K}_\ell = \text{reshape}\Big(P_K K_\ell^{\text{vlm}}\Big), \qquad \tilde{V}_\ell = \text{reshape}\Big(P_V V_\ell^{\text{vlm}}\Big). \tag{6}$$

Here, P_K and P_V are linear VLM-to-expert adapter layers that align the VLM KV dimensionality with the expert's cross-attention width; they are separate from the VLM self-attention projections that produced the keys and values. After projection, the conditioning context is organized into the expert's attention heads. The cross-attention in expert block ℓ attends to the projected keys and values from the corresponding VLM layer:

$$\mathrm{CA}(Q_\ell, \tilde{K}_\ell, \tilde{V}_\ell) = \mathrm{softmax}\left(\frac{Q_\ell \tilde{K}_\ell^\top}{\sqrt{d_h}}\right)\tilde{V}_\ell, \tag{7}$$

where d_h is the expert head dimension. This per-layer KV conditioning gives every action-expert block direct access to the visual-language attention state at the same depth in the backbone. During post-training we detach this conditioning path from the VLM, so the flow loss trains the expert and its adapter projections without sending gradients back through the VLM keys and values.

4.2.2 Post-training recipe

Overview. Post-training starts from the 200K-step MolmoAct2-Pretrain checkpoint and turns it into the final MolmoAct2 model. We keep the VLM architecture and pre-training data construction from Sec. 4.1.2, including the image augmentation described in Appendix B. The main change is that robot examples now supervise both output interfaces: the LLM continues to predict discrete action tokens, while the action expert described in Sec. 4.2.1 learns to generate continuous action chunks.

Multiple flow samples. For each robot action chunk, we evaluate the flow objective at multiple noise levels. Let a be a normalized continuous action chunk and c be the VLM context for the corresponding robot prompt. For each training example we draw K independent pairs $\{(\epsilon_i, t_i)\}_{i=1}^K$, form $x_{t_i} = (1 - t_i)\epsilon_i + t_i a$, and train the expert to predict $a - \epsilon_i$:

$$\mathcal{L}_{\mathrm{flow}}(a, c) = \frac{1}{K}\sum_{i=1}^{K} \left\| m \odot \left(f_\theta(x_{t_i}, t_i, c) - (a - \epsilon_i)\right)\right\|_2^2. \tag{8}$$

We use $K = 4$, so each robot sample contributes four points on the same flow trajectory while reusing the same visual-language context. We didn't use $K = 8$ as in the fine-tuning stage (Sec. 4.3.1) mainly due to GPU memory constraints.

Padding and packing. Post-training retains a common action tensor shape across embodiments. Continuous actions are right-padded to a maximum horizon of 30 steps and a maximum width of 32 dimensions. The horizon mask removes padded time steps from the flow loss, and the dimension mask zeroes padded action dimensions in the noisy inputs, targets, and predictions before averaging the loss over valid dimensions. This lets datasets with different control rates and action widths share the same expert without training on artificial padded coordinates.

Packing is also extended to the continuous expert. A packed language sequence can contain multiple robot examples; each action chunk is paired with the sub-example that produced it, and the expert attends only to that example's VLM context. To keep memory use stable when packed batches contain different numbers of robot chunks, we pad the packed action-chunk axis up to a small fixed cap of five chunks per packed sequence when possible. Padded chunk rows are marked invalid and excluded from the flow loss; sequences with more chunks are still used, but processed without this fixed-cap padding.

Training objectives. The post-training objective combines the autoregressive loss from pre-training with the flow-matching loss:

$$\mathcal{L}_{\mathrm{post}} = \mathcal{L}_{\mathrm{LM}} + \mathcal{L}_{\mathrm{flow}}. \tag{9}$$

The language-model loss is applied to the same next-token targets used in pre-training, including discrete action tokens for robot examples and text tokens for VLM examples. The flow loss is applied only to continuous robot action chunks. Since robot examples contain both the discrete action target and the continuous action target, we mask the discrete action-token span out of the expert's VLM conditioning path. The expert

therefore conditions on the task, observations, setup/control descriptors, and state tokens, but not on the target action tokens it is supposed to predict.

We also apply knowledge insulation (Driess et al., 2025), where the VLM is isolated from the continuous-action loss. The expert conditions on the VLM keys and values, but these tensors are detached before they enter the expert, so $\mathcal{L}_{\text{flow}}$ updates the action expert and its adapter projections without back-propagating through the VLM. The VLM is still updated by $\mathcal{L}_{\text{LM}}$.

Training. Robot batches use a sequence length of 2100 tokens, while the non-robot VLM batches keep the 4200-token context length from pre-training. We use this split because robot batches additionally run the action expert and four flow samples per action chunk, whereas VLM batches only train the autoregressive model. We train for 100K updates with a global batch size of 128 and a device batch size of 2 across 64 H100 GPUs for around 2,304 GPU hours. The VLM learning rates match pre-training: 5×10^{-6} for the vision encoder and connector, and 1×10^{-5} for the language model. The action expert is trained with a larger learning rate of 5×10^{-5}.

4.3 Deployment

4.3.1 Embodiment-specific fine-tuning

Shared recipe. Embodiment-specific fine-tuning starts from the post-trained MolmoAct2-Post checkpoint and keeps the same VLM–action-expert architecture described in Sec. 4.2. We continue to co-train the discrete autoregressive action output and the continuous flow-matching action expert, use the same 32-dimensional padded action width, and keep the same learning rates as post-training: 5×10^{-6} for the vision encoder and connector, 1×10^{-5} for the language model, and 5×10^{-5} for the action expert. We also keep single resized crops, packed sequences, setup/control descriptors, discrete state tokens, and the same action-token vocabulary, using the shared prompt, image-augmentation, and packing implementation described in Appendix B.

The fine-tuning stage differs from post-training in four ways. First, each run is robot-only: we do not include the multimodal VLM mixture or a separate VLM dataloader. Second, we increase the number of sampled flow times from 4 to 8 per action chunk, which gives the action expert denser supervision along the same flow trajectory. Third, we do not use knowledge insulation during fine-tuning: gradients from the flow loss are allowed to update the VLM through the action-expert conditioning path, since we did not observe a consistent performance gain from detaching this path at this stage. Finally, for the main embodiment checkpoints below, we do not tune the added-token input embeddings. As in standard VLM fine-tuning, we assume that the token embeddings learned during pre-training and post-training are already well placed, and tune the language-model output head and final normalization instead.

Bimanual YAM. The MolmoAct2-BimanualYAM checkpoint is fine-tuned on the bimanual YAM mixture. The camera order is fixed as top, left, and right, matching the data collection setup. We use annotated language instructions, absolute joint-pose actions, and a 30-step action chunk, corresponding to the 30 Hz control rate of the dataset. We train with a sequence length of 2100, global batch size 128, and 100K updates on 64 H100 GPUs, for roughly 2,304 GPU hours.

DROID. The MolmoAct2-DROID checkpoint is fine-tuned on the filtered DROID mixture. DROID provides two exterior cameras and one wrist camera. For DROID-style two-view evaluation, we use a fixed exterior-then-wrist ordering; when training variants expose both exterior choices as alternatives, the loader samples one exterior camera and pairs it with the wrist camera during training. We use absolute joint-pose actions with a 15-step action chunk, matching DROID's 15 Hz control rate. We do not use the additional language annotations in this fine-tune, to keep the comparison with prior DROID-trained models fair. We train with a sequence length of 2100, global batch size 64, and 100K updates on 32 H100 GPUs, for roughly 1,152 GPU hours.

SO-100/101. The MolmoAct2-SO100/101 checkpoint is fine-tuned on the SO-100/101 mixture. Because this data is aggregated from internet demonstrations with diverse and inconsistent camera layouts, we randomize the input camera order at the episode level rather than imposing a fixed view naming convention.

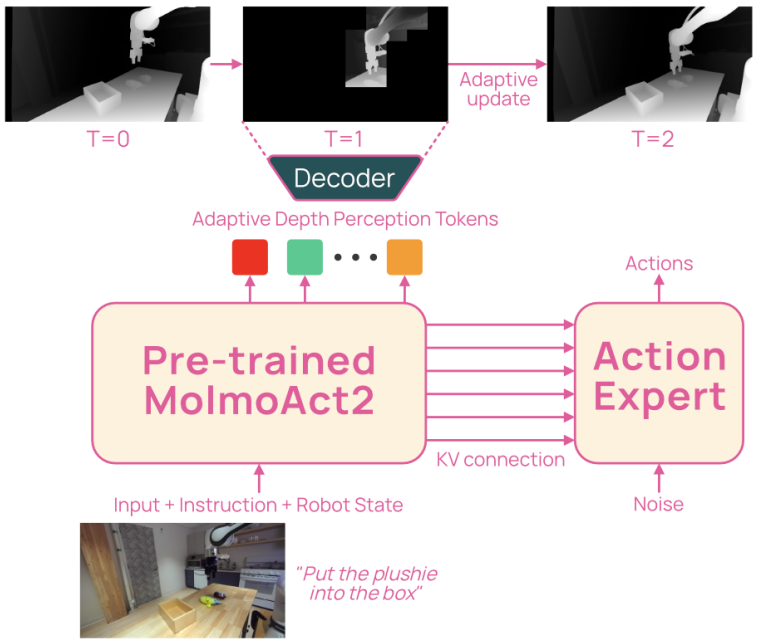

Figure 5 Overview of MolmoAct2-Think. MolmoAct2-Think augments the MolmoAct2 action-generation pipeline with adaptive depth-token reasoning, reusing cached depth codes for static regions and regenerating depth codes for changed regions before conditioning the action expert.

We use annotated language instructions, absolute joint-pose actions, and a 30-step action chunk for the 30 Hz control rate. We train with a sequence length of 2100, global batch size 64, and 100K updates on 32 H100 GPUs, for roughly 1,152 GPU hours.

LIBERO. The MolmoAct2-LIBERO checkpoint is fine-tuned on the full LIBERO training mixture, combining the Spatial, Object, Goal, and Long suites rather than training a separate model for each suite. The camera order is fixed as front view followed by wrist view. We do not use annotated language instructions. LIBERO uses relative end-effector control at 10 Hz, so we train with a 10-step action chunk. We use a sequence length of 2100, global batch size 64, and train for 50K updates on 32 H100 GPUs, for roughly 1,152 GPU hours; for evaluation, we select the best-performing checkpoint, which occurs at 40K updates.

Other evaluation fine-tunes. For smaller task- or benchmark-specific fine-tunes used in downstream evaluation, we follow the same recipe unless otherwise noted. These runs use fixed metadata camera order rather than camera-order randomization, no language annotations, 2100-token sequences, packing, 8 sampled flow times, and robot-only data. The action horizon is set to the dataset control frequency, and the control representation follows the dataset, either joint pose or end-effector pose. Real-world evaluation runs use 8 H100 GPUs, global batch size 16, and 50K updates, and we choose the best-performing checkpoint for evaluation.

4.3.2 Inference optimization

For the MolmoAct2 model, inference is dominated by the continuous action expert, which integrates a fixed-step flow-matching trajectory conditioned on the VLM context. Within one action chunk, this context is invariant across flow steps, while only the noisy action state and flow time evolve. We therefore cache reusable action-expert intermediates, including context-dependent cross-attention states and fixed position-dependent terms, and reuse them throughout the flow loop. We further capture the fixed-shape flow loop with CUDA Graphs, reducing Python and kernel-launch overhead during repeated action generation.

5 MolmoAct2-Think

Robotic manipulation depends on spatial information that is only indirectly supervised by action imitation: object distance, free space, occlusion, and surface layout all affect the action, but standard behavior-cloning objectives do not ask the model to make this structure explicit before acting. MolmoAct (Lee et al., 2025) addressed this problem by adding depth-token prediction as an intermediate reasoning step. MolmoAct2-Think extends the same idea to MolmoAct2: before producing an action, the model predicts a compact discrete depth representation that conditions the action expert through per-layer KV conditioning.

The depth representation is intentionally lightweight. Each observation depth map is quantized into a 10×10 grid, giving 100 spatial code positions, and each position takes one of 128 learned depth-code values. These codes are represented as ordinary autoregressive tokens, which makes depth reasoning compatible with the same next-token interface used throughout MolmoAct2 while exposing an interpretable intermediate prediction.

The main distinction from MolmoAct is that MolmoAct2-Think makes depth prediction adaptive across time. Robot trajectories contain substantial temporal redundancy: many cells in a scene-level depth grid remain unchanged from one control step to the next. Instead of re-predicting every depth code at every step, MolmoAct2-Think reuses cached codes for static regions and autoregressively predicts only the cells whose RGB evidence changes. The result is a depth-aware policy whose geometric reasoning cost scales with scene change rather than with the full 100-token grid.

5.1 Adaptive depth perception data

For every robotics dataset used by MolmoAct2-Think in the mixtures described in section 3, we attach depth annotations to the policy observation stream. We use the same camera stream that is presented to the policy for depth reasoning: single-view datasets use their canonical observation camera, and multi-view datasets use the first policy view with available depth annotations. For heterogeneous LeRobot datasets, when a camera is not specified explicitly, the video feature is selected from the dataset metadata.

For each RGB frame, we estimate a dense monocular depth map with Depth Anything V2 (Yang et al., 2024). We then quantize that depth map with the trained depth VQ-VAE used by MolmoAct (Lee et al., 2025), following the tokenization scheme of Ning et al. (2023). The VQ-VAE operates on a 320×320 depth image with a downsampling factor of 32, producing a 10×10 grid of codebook indices in $\{0, \ldots, 127\}$.

The same preprocessing pass also constructs the adaptive-depth side channels used during training and inference. Let $d_t \in \{0, \ldots, 127\}^{100}$ be the full VQ depth codes for frame t, flattened in raster order. We maintain a buffer b_t and update mask $m_t \in \{0, 1\}^{100}$. For the first frame of an episode, all positions are updated and $b_1 = d_1$. For later frames, the RGB image is resized to 320×320, divided into the same 10×10 grid of 32×32 patches, and each patch is compared to the corresponding patch in the previous frame by cosine similarity. A cell is marked updated when the similarity is below 0.996:

$$m_{t,i} = \mathbf{1}\left[\cos(x_{t,i}, x_{t-1,i}) < 0.996\right], \qquad b_{t,i} = \begin{cases} d_{t,i}, & m_{t,i} = 1, \\ b_{t-1,i}, & m_{t,i} = 0. \end{cases} \tag{10}$$

Thus each frame stores the full depth codes d_t, the carried-forward depth buffer b_t, and the binary update mask m_t. The model is supervised on the buffer codes, matching the representation it will maintain during adaptive inference.

5.2 Training

Post-training. MolmoAct2-Think starts from the same 200K-step MolmoAct2-Pretrain checkpoint as the standard MolmoAct2 post-training stage. We keep the post-training recipe in subsection 4.2: the same data construction, action expert, per-layer KV conditioning, sequence lengths, optimizer settings, 100K training updates, global batch size of 128, and 64 H100 GPUs. On top of MolmoAct2, we added depth-related special tokens for training MolmoAct2-Think, where the full depth-token interface is in subsection A.3.

Within robot data, post-training uniformly samples three tasks: action prediction, depth prediction, and depth-and-action prediction. The action task trains the usual discrete and continuous action objectives. The depth task trains autoregressive prediction of the 100 depth-buffer tokens. In the depth-and-action task, the model first predicts depth tokens, then predicts discrete action tokens autoregressively while the action expert conditions on the input context and predicted depth state to generate continuous actions. The prompt templates and assistant-side triggers for these output styles are given in Appendix B. As in subsection 4.2, the target action-token span is masked from the expert conditioning path, so the expert can use the task, observations, state, and depth tokens, but not the discrete action target it is trained to predict.

Fine-tuning. The MolmoAct2-Think-LIBERO checkpoint is fine-tuned from the depth-aware post-trained checkpoint using the same full-LIBERO recipe and compute budget as subsubsection 4.3.1: robot-only training on the combined Spatial, Object, Goal, and Long suites, with front-view then wrist-view camera order, 2100-token sequences, global batch size 64, 32 H100 GPUs, and 50K updates. We select the best checkpoint at 30K updates for evaluation.

Fine-tuning differs from depth post-training in three targeted ways. First, we sample only action and depth-and-action examples, again uniformly, and remove the pure depth-prediction style. Second, because inference conditions on depth tokens predicted by the model rather than oracle depth tokens, we inject noise into the teacher-forced depth prefix during training: 10% of depth-code input tokens are replaced by uniformly sampled depth codes, while the prediction targets remain unchanged. Third, we add a learned per-layer gate on the depth portion of the action expert's per-layer KV conditioning. For action-expert layer ℓ, let $M_t = 1$ denote positions belonging to the depth-output trigger, depth delimiters, or depth-code tokens, and let A_t denote valid context positions. The gate is computed from the non-depth context of the corresponding VLM layer,

$$c_\ell = \frac{\sum_t A_t(1 - M_t)V_{\ell,t}^{\mathrm{vlm}}}{\sum_t A_t(1 - M_t)}, \qquad g_\ell = \sigma\big(w_\ell^\top c_\ell + b_\ell\big), \tag{11}$$

and then applied only to depth-token keys and values:

$$\bar{K}_{\ell,t}^{\mathrm{vlm}} = \big(1 - M_t + M_t g_\ell\big)\, K_{\ell,t}^{\mathrm{vlm}}, \qquad \bar{V}_{\ell,t}^{\mathrm{vlm}} = \big(1 - M_t + M_t g_\ell\big)\, V_{\ell,t}^{\mathrm{vlm}}. \tag{12}$$

The gated $(\bar{K}_\ell^{\mathrm{vlm}}, \bar{V}_\ell^{\mathrm{vlm}})$ are then projected into the action expert as in subsection 4.2. The gate is initialized with bias -4, so fine-tuning begins close to the standard action-conditioning path and learns how strongly each expert layer should use the depth prefix. Additional model-level details for the depth extension are summarized in subsection A.3.

5.3 Adaptive depth inference

Inference pipeline. At inference, MolmoAct2-Think uses the depth-and-action output style. Given the task, current observation, and proprioceptive state, the prompt requests `<depth_output><action_output>`. The model first performs a prefill over the prompt and images. If no depth cache is available, as at the beginning of a rollout or after a reset, it autoregressively predicts the full depth sequence: `<depth_start>`, 100 depth-code tokens, and `<depth_end>`.

When a depth cache is available, the model compares the current first observation image to the cached previous image using the same 10×10 RGB-patch cosine threshold described in subsection 5.1. Updated cells are generated by argmax decoding from the depth-token logits. Unchanged cells are replayed from the previous predicted depth buffer and consumed by the model as known depth-token inputs. Consecutive unchanged spans are replayed together, whereas changed spans are decoded token by token. After all 100 cells are filled, the model emits `<depth_end>` and stores the current image and the newly filled 100-code depth buffer as the cache for the next control step.

The action is then generated from the depth-conditioned state. For continuous control, the action expert receives the VLM keys and values corresponding to the prompt plus the filled depth prefix and integrates the flow-matching velocity field to produce the action chunk. Adaptive depth changes only how the intermediate depth prefix is produced. The action interface remains the same as the depth-and-action training objective.

Table 3 Embodied reasoning results. We evaluate on benchmarks across spatial reasoning, pointing, and embodied QA tasks. The best-performing open-weight model on each benchmark is in **bold**, and the second best is underlined.

Models/Benchmarks	Point-Bench (Cheng et al., 2025)	RefSpatial (Zhou et al., 2025)	RoboSpatial-Poi (Song et al., 2025)	Where2Place (Yuan et al., 2024)	BLINK (Fu et al., 2024)	CV-Bench (Tong et al., 2024)	ERQA (Team et al., 2025b)	EmbSpatial (Du et al., 2024)	MindCube (Luo et al., 2026)	RoboSpatial-VQ (Song et al., 2025)	SAT (Ray et al., 2025)	OpenEQA (Majumdar et al., 2024)	VSI-Bench (Yang et al., 2025b)	Overall Avg.
API call only														
GR-ER 1.5 Thinking (Team et al., 2025b)	71.6	48.5	31.1	59.0	57.8	84.3	54.8	78.4	54.7	79.3	76.7	55.0	45.8	61.3
GR-ER 1.5 (Team et al., 2025b)	73.3	41.8	25.3	48.0	65.2	83.6	47.0	73.4	47.7	57.0	62.0	50.5	39.9	55.0
Gemini 2.5 Pro (Comanici et al., 2025)	62.7	33.6	8.3	37.0	69.2	85.9	56.0	78.0	59.2	71.3	74.7	55.0	51.1	57.1
GPT-5 (Singh et al., 2025)	43.6	23.5	19.0	37.0	71.3	86.1	59.0	81.5	58.0	69.3	86.7	64.4	52.9	57.9
GPT-5-mini (Singh et al., 2025)	39.5	23.0	12.5	33.5	56.4	85.9	57.3	78.8	55.6	70.7	81.3	59.2	46.2	53.8
Open weights only														
Qwen3-VL-4B (Yang et al., 2025a)	63.8	49.5	**62.3**	**63.0**	65.2	85.8	39.5	78.1	27.4	62.7	60.7	47.6	61.4	59.0
Qwen3-VL-8B (Yang et al., 2025a)	64.2	47.5	56.6	**63.0**	66.6	85.6	41.5	78.2	33.4	**73.7**	70.0	**49.3**	63.3	61.0
LLaVA-OV-7B (Li et al., 2024a)	9.4	3.0	0.0	19.0	44.2	70.3	39.8	69.7	45.6	68.9	58.0	36.2	31.2	38.1
InternVL3.5-4B (Wang et al., 2025a)	43.0	21.5	23.0	53.0	58.7	80.5	37.0	72.0	42.6	61.0	49.3	44.6	56.5	49.4
InternVL3.5-8B (Wang et al., 2025a)	45.5	35.0	27.0	57.0	59.1	80.9	41.0	74.7	39.7	62.7	58.7	43.5	56.4	52.4
Molmo family: Open weights, Open data, Open code														
Molmo2 (Clark et al., 2026)	76.9	**52.5**	13.9	18.0	50.8	64.4	46.3	67.0	37.6	57.9	50.0	46.9	26.1	46.8
MOLMO2-ER	**77.3**	**52.5**	32.0	54.0	**72.5**	**87.8**	**46.8**	**78.8**	**57.0**	73.4	**78.0**	44.7	**74.5**	**63.8**

Inference optimization. Adaptive depth inference introduces a different systems challenge from the continuous action expert: each frame can contain a different mixture of regenerated depth cell and replayed cached cells, so the overall decode schedule is data-dependent. Capturing the full adaptive loop would require a separate graph for each update pattern. We therefore keep the adaptive scheduler eager, including span-level replay for unchanged cells, while using a preallocated static KV cache to make the decode state stable across steps. For regenerated depth tokens, we capture the fixed-shape transformer work from post-attention through the next layer's pre-attention as CUDA Graph stages, and leave attention itself eager because its effective KV length changes throughout decoding. This preserves adaptive depth reuse while reducing the launch bubbles in the repeated one-token decode path.

6 Experiments

Our experimental evaluation comprises one of the most extensive suites of studies of MOLMOACT2, benchmarked against a diverse set of strong generalist baseline models. Specifically, we assess MOLMOACT2 across three main categories of evaluation: (i) we examine how MOLMO2-ER performs relative to generalist VLM models used as training backbones for VLAs, and quantify the performance gains attributable to a stronger backbone; (ii) we evaluate the out-of-the-box deployment capabilities of MOLMOACT2 across a variety of embodiments; and (iii) we investigate the ease and efficiency with which MOLMOACT2 can be adapted to novel tasks and unseen embodiments via fine-tuning.

Hence, we evaluate MOLMOACT2 across a comprehensive range of scenarios in both simulation and the real world, addressing the following research questions:

1. **How well does Molmo2-ER perform on established industry benchmarks for embodied reasoning in VLMs?** We address this question by benchmarking MOLMO2-ER against strong open-source and proprietary VLMs across 13 established embodied reasoning benchmarks.
2. **How well does MolmoAct2 perform out-of-the-box?** We investigate this by evaluating MOLMOACT2 on three benchmarks: two simulation benchmarks, MolmoBot (Deshpande et al., 2026) and Molmo-

Table 4 Results across MolmoSpace tasks. We evaluate all models on four manipulation skill categories—Pick, Pick & Place, Open, and Close—and report mean success rate (%) with standard error across three independent evaluation runs. **Bold** denotes the best and underline the second-best per task. MolmoAct2-DROID achieves the highest overall average (37.7), outperforming the strongest prior baseline $\pi_{0.5}$-DROID (34.5) by +3.2 points, with particularly large gains on Pick (+7.3) and Pick & Place (+13.1), where contact-rich, multi-stage reasoning is required. On Close, MolmoAct2-DROID also sets a new best at 70.8, while on Open it trails the leading methods, suggesting that articulated-object interaction remains a direction for further improvement. Standard errors remain comparable across methods (≤ 3.2), indicating that the observed gains are stable across runs rather than artifacts of evaluation variance.

Models/Tasks	Pick	Pick & Place	Open	Close	Average
StereoVLA (Deng et al., 2025)	7.0±2.3	–	–	–	7.0
LAP-VLA (Zha et al., 2026)	24.9±2.7	6.6±1.6	11.4±1.5	45.9±2.9	22.2
π_0-DROID (Black et al., 2024)	16.2±2.6	12.5±2.0	11.0±2.0	53.1±3.2	23.2
$\pi_{0.5}$-DROID (Intelligence et al., 2025)	36.4±2.9	13.6±2.2	**22.7**±2.6	65.1±3.1	34.5
MolmoAct2-DROID	**43.7**±3.1	**26.7**±2.8	9.5±2.0	**70.8**±3.0	**37.7**

Table 5 Evaluation on simulation held-out environments. Simulation success rates are evaluated over 1000 episodes per task. For pick-and-place tasks, we report both oracle success (first number, which is the success conditions being fulfilled at any timestep) and success at end (second number, the success conditions being fulfilled at the final timestep). The delta between captures both unstable/unsuitable placement and the inability of policies to determine when a specified task is already completed, e.g. by repeatedly picking up an object which has already been placed correctly. We additionally report the half-width of the 95% confidence interval bounds for each result. **Bold** denotes the best and underline the second-best per task.

Models/Tasks	Pick MSProc	Pick Classic	Pick	Pick Rand.-Cam.	Pick&Place	PnP Next-To	PnP Color	Avg.
StereoVLA (Deng et al., 2025)	6.6±2.6	4.3±1.5	1.1±1.0	-	-	-	-	-
X-VLA (Zheng et al., 2025)	3.3±1.0	0.5±0.5	0.7±0.5	0.8±0.5	0.1±0.2/0.1±0.2	1.9±0.9/1.0±0.7	0.9±0.6/0.5±0.5	1.2
LAP-VLA (Zha et al., 2026)	19.4±2.4	2.4±1.0	3.1±1.1	2.7±1.0	3.81±1.5/1.59±1.0	6.48±2.8/3.41±2.1	3.1±1.1/1.5±0.8	4.8
$\pi_{0.5}$-DROID (Intelligence et al., 2025)	18.1±2.4	6.4±1.5	7.0±1.6	8.0±1.9	11.7±2.1/7.6±1.7	8.2±2.2/6.2±1.9	10.4±1.9/6.7±1.6	10.0
MolmoAct2-DROID	**35.6**±3.0	**18.9**±2.6	**20.5**±2.6	**15.4**±2.4	**15.8**±2.4/**7.8**±1.8	**20.9**±2.6/**14.4**±2.3	**17.2**±2.5/**8.8**±2.0	**20.6**

Spaces (Kim et al., 2026b), and a series of real-world zero-shot evaluations across two embodiments (DROID and SO-100/101 setups) against strong baselines pretrained or fine-tuned on the DROID (Khazatsky et al., 2024) and large-scale SO-100/101 datasets.

3. **How effectively can MolmoAct2 be fine-tuned for new tasks and embodiments?** We study this by fine-tuning MolmoAct2 on custom datasets collected across three benchmarks: two simulation benchmarks, RoboEval (Wang et al., 2025b) and LIBERO (Liu et al., 2023), and one large-scale real-world evaluation suite consisting of in-the-wild bimanual YAM tasks.

4. **Do adaptive depth-perception tokens improve the performance of MolmoAct2-Think?** We investigate this question by comparing MolmoAct2 against MolmoAct2-Think on LIBERO (Liu et al., 2023), MolmoSpace (Kim et al., 2026b), and the MolmoBot Benchmark (Deshpande et al., 2026).

5. **How robust is MolmoAct2 under out-of-distribution perturbations?** To investigate this, we fine-tune MolmoAct2-Post checkpoint on 4 tasks from the real-world Bimanual YAM benchmark and evaluate each task under 4 distinct out-of-distribution variants.

6. **What is the quality of MolmoAct2's trajectory rollouts beyond raw success rate?** For real-world deployment, trajectory quality matters beyond task success (e.g., stability, smoothness). We therefore evaluate MolmoAct2 against strong generalist baselines on RoboEval (Wang et al., 2025b) to assess the quality of the trajectories it generates while solving the tasks.

7. **Which components contributed the largest performance gains to MolmoAct2?** We investigate this question through systematic component-level ablations on the MolmoAct2 architecture and training configurations to understand how each of our novel improvements contributes to the model's performance, and we verify our findings on LIBERO (Liu et al., 2023) for reproducibility.

Table 6 Success rates (%) across manipulation tasks. Each cell reports the percentage of successful trajectories out of 15 trials. **Bold** denotes the best and underline the second-best per task.

Models/Tasks	Apple on plate	Pipette in tray	Red cube in tape roll	Knife in box	Objects in bowl	Average
$\pi_{0.5}$-DROID (Intelligence et al., 2025)	66.7	33.3	53.3	26.7	46.2	45.2
MolmoBot (Deshpande et al., 2026)	86.7	53.3	33.3	40.0	28.6	48.4
MolmoAct2-DROID	**100.0**	**86.7**	**93.3**	**93.3**	**62.0**	**87.1**

Table 7 Success rates (%) across manipulation tasks on the SO-100 platform. Each cell reports the mean score over 15 trials, where partial credit is awarded for sub-task completion (0.25 for reaching, 0.5 for pickup, 1.0 for successful placement). **Bold** denotes the best and underline the second-best per task.

Models/Tasks	Fork on plate	Stack blocks	Tissues in basket	Pen on notebook	Block in box	Average
SmolVLA (Shukor et al., 2025)	3.3	5.0	0.0	3.3	0.0	2.3
π_0-SO100/101 (Chen, 2025)	30.0	6.7	20.0	80.0	**90.0**	45.3
MolmoAct2-SO100/101	**70.0**	**20.0**	**73.3**	**86.7**	33.3	**56.7**

8. **How fast is MolmoAct2 at inference?** We measure end-to-end inference latency and control rate for MolmoAct2 and MolmoAct2-Think on LIBERO (Liu et al., 2023), and evaluate the impact of our caching and CUDA Graph optimizations.

6.1 Molmo2-ER evaluation

Evaluation setup. We evaluate Molmo2-ER on 13 established vision-language benchmarks that are widely used to measure the embodied reasoning capabilities of VLMs: Point-Bench (Cheng et al., 2025), RefSpatial (Zhou et al., 2025), RoboSpatial-Point (Song et al., 2025), Where2Place (Yuan et al., 2024), BLINK (Fu et al., 2024), CV-Bench (Tong et al., 2024), ERQA (Team et al., 2025b), EmbSpatial (Du et al., 2024), MindCube (Luo et al., 2026), SAT (Ray et al., 2025), OpenEQA (Majumdar et al., 2024), and VSI-Bench (Yang et al., 2025b). Together, these benchmarks span visual question answering, 2D pointing, multi-image reasoning, ego–exo understanding, and video-based spatial reasoning.

Baselines. We compare Molmo2-ER against a diverse set of proprietary VLMs, reasoning-oriented VLMs, and open-weight VLMs, including the Gemini family (Team et al., 2024; Comanici et al., 2025), the GPT-5 family (Singh et al., 2025), LLaVA-OneVision (Li et al., 2024a), the Qwen3-VL family (Bai et al., 2025), and InternVLA (Wang et al., 2025a). We additionally compare against Molmo2 (Clark et al., 2026), the base model from which Molmo2-ER is trained.

Results. Molmo2-ER outperforms all baseline VLMs on 9 of the 13 benchmarks and achieves the highest overall average score of 63.8%, exceeding the runner-up, Gemini-ER 1.5 Thinking, by 2.5 points. Notably, Molmo2-ER improves over its base model Molmo2 by 17%, demonstrating the substantial gains in embodied reasoning capability conferred by our approach.

6.2 Out-of-the-box deployment

Evaluation setup and baselines. To verify the out-of-the-box deployment capability of MolmoAct2, we evaluate it in both simulation and real-world settings. In simulation, we evaluate the MolmoAct2-DROID checkpoint on two benchmarks: MolmoSpaces (Kim et al., 2026b) and MolmoBot (Deshpande et al., 2026). Both target single-cycle pick-and-place tasks across diverse objects and environments, with MolmoBot featuring more challenging objects and scenarios. Each benchmark replicates the original DROID (Khazatsky et al., 2024) setup, enabling zero-shot evaluation of any policy pretrained or fine-tuned on the DROID dataset. On MolmoSpaces, we compare MolmoAct2-DROID against LAP-VLA (Zha et al., 2026), StereoVLA (Deng et al., 2025), π_0-DROID (Black et al., 2024), and $\pi_{0.5}$-DROID (Intelligence et al., 2025); on MolmoBot, we compare against LAP-VLA, $\pi_{0.5}$-DROID (Intelligence et al., 2025), and X-VLA (Zheng et al., 2025). All simulation evaluations strictly follow the protocols, trial counts, and procedures of the respective original papers. Further details are provided in the Appendix.

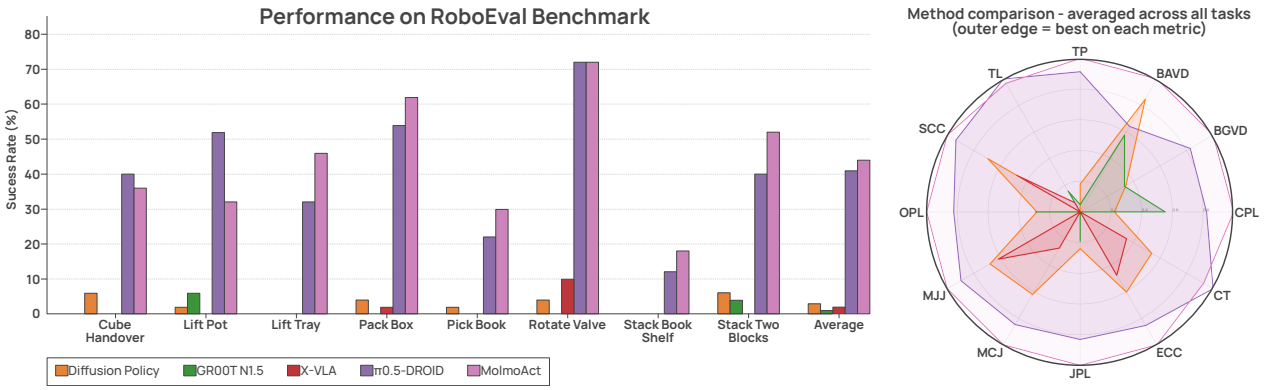

Figure 6 Performance comparison on the RoboEval benchmark. (A) Task-wise success rates (%) across eight manipulation tasks for Diffusion Policy, GR00T N1.5, X-VLA, $\pi_{0.5}$-DROID, and MolmoAct. MolmoAct achieves the highest or near-highest success rates on most tasks, with particularly strong performance on long-horizon tasks such as *Pack Box* and *Rotate Valve*. $\pi_{0.5}$-DROID is the strongest baseline overall, while other methods exhibit lower and more variable performance across tasks. (B) Radar plot of normalized performance across behavioral and outcome metrics, averaged over all tasks (outer edge indicates best performance per metric). MolmoAct consistently dominates across nearly all metrics, indicating improvements not only in task success but also in efficiency, stability, and trajectory quality. *Abbreviations:* CT = completion time, TL = trajectory length, JPL = joint path length, CPL = Cartesian path length, CJ = Cartesian jerk, JJ = joint jerk, SC = self-collisions, SL = slip count.

In the real world, we test MolmoAct2-DROID and MolmoAct2-SO100/101 on their respective pretrained embodiment setups under challenging out-of-distribution conditions: camera poses are randomly initialized without conforming to any dataset visual distribution, using two cameras (wrist and a single allocentric view); all objects are unseen during training; and the environments themselves are out-of-distribution relative to the training data. For the DROID embodiment, we compare MolmoAct2-DROID against $\pi_{0.5}$-DROID (Intelligence et al., 2025), fine-tuned on the DROID dataset, and MolmoBot (Deshpande et al., 2026), fine-tuned on the large-scale MolmoSpaces dataset. We evaluate on five tasks ranging from single- to multi-object pick-and-place: `apple_on_plate`, `pipette_in_tray`, `red_cube_in_tape_roll`, `knife_in_box`, and `objects_in_bowl`. For the SO-100/101 setup, we use the SO-100 robot, retaining the wrist camera and adding a third-person external camera with a randomly initialized position. We compare MolmoAct2-SO100/101 against SmolVLA (Shukor et al., 2025) and π_0-SO100/101, a variant of π_0 that we fine-tuned on MolmoAct2-SO100/101 Dataset. Both baselines are pretrained or fine-tuned at scale on the SO-100/101 dataset. We evaluate five pick-and-place tasks with novel objects and randomly initialized camera poses. Every real-world policy is assessed over 15 trials, with partial credit awarded for near-successful executions. Further details are provided in the Appendix.

Results. MolmoAct2-DROID achieves strong zero-shot performance on all DROID-style setups, in both simulation and the real world. In simulation, MolmoAct2-DROID is the state-of-the-art VLA model across MolmoSpaces and MolmoBot, outperforming all baselines on both benchmarks. $\pi_{0.5}$-DROID is the runner-up on each, with MolmoAct2 achieving an absolute gain of +10.6% on MolmoBot and +3.2% on MolmoSpaces, averaged across tasks; trial counts in every case are sufficient for statistical significance. In real-world evaluation on the DROID setup, which is substantially more out-of-distribution given the random camera initialization and entirely novel scenes and objects relative to the DROID dataset (Khazatsky et al., 2024), MolmoAct2 again attains the highest performance, reaching 87.1% and surpassing the runner-up MolmoBot by 38.7%. On the second zero-shot embodiment, SO-100/101, MolmoAct2-SO100/101 reaches 56.7%, an 11.4% gain over our open implementation of π_0 on SO-100/101, demonstrating affordable deployment on low-cost robots such as the SO-100/101.

6.3 Effective fine-tuning

Evaluation setups and baselines. For rapid real-world deployment on novel tasks and embodiments, VLAs must adapt from only a handful of demonstrations. To assess MolmoAct's capacity for such efficient adaptation, we benchmark it against several strong VLA baselines on downstream tasks, domains, and embodiments

Table 8 LIBERO benchmark success rates across four task categories (Spatial, Object, Goal, and Long-horizon) along with the average performance. MolmoAct2 achieves the highest overall average success rate of 97.2%, outperforming all strong baselines, with strong performance across all categories, particularly scoring 100% on the LIBERO-Object tasks. Furthermore improvement could be seem going from MolmoAct2 to MolmoAct2-Think. **Bold** denotes the best and underline the second-best per task.

Baseline	Spatial	Object	Goal	Long	Average
TraceVLA (Zheng et al., 2024)	84.6%	85.2%	75.1%	54.1%	74.8%
OpenVLA (Kim et al., 2024)	84.7%	88.4%	79.2%	53.7%	76.5%
SpatialVLA (Qu et al., 2025)	88.2%	89.9%	78.6%	55.5%	78.1%
CoT-VLA (Zhao et al., 2025)	87.5%	91.6%	87.6%	69.0%	83.9%
π_0 (Black et al., 2024)	96.8%	98.8%	95.8%	85.2%	94.2%
ThinkAct (Huang et al., 2025)	88.3%	91.4%	87.1%	70.9%	84.4%
MolmoAct-7B-D (Lee et al., 2025)	87.0%	95.4%	87.6%	77.2%	86.6%
GR00T N1.7 (NVIDIA et al., 2025)	97.7%	97.5%	98.5%	94.4%	97.0%
$\pi_{0.5}$ (Intelligence et al., 2025)	98.8%	98.2%	98.0%	92.4%	96.9%
NORA-1.5 (Hung et al., 2025)	97.3%	96.4%	94.5%	89.6%	94.5%
MolmoAct2	97.8%	**100.0%**	97.8%	93.2%	97.2%
MolmoAct2-Think	**98.8%**	99.8%	**98.5%**	**95.4%**	**98.1%**

unseen during training. In all fine-tuning experiments, we initialize from the MolmoAct2-Post checkpoint and fine-tune on the benchmark-specific training data.

In simulation, we evaluate on two benchmarks. RoboEval (Wang et al., 2025b) comprises a bimanual Franka Emika Panda setup with human-expert demonstrations across 8 tasks requiring bimanual coordination. LIBERO (Liu et al., 2023), following prior work (Kim et al., 2024), consists of four task suites—LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long—each containing 500 demonstrations across 10 tasks. We fully fine-tune MolmoAct2 and compare it against state-of-the-art VLAs trained under similar protocols, including TraceVLA (Zheng et al., 2024), OpenVLA (Kim et al., 2024), SpatialVLA (Qu et al., 2025), CoT-VLA (Zhao et al., 2025), π_0 (Black et al., 2024), ThinkAct (Huang et al., 2025), MolmoAct (Lee et al., 2025), GR00T-N1 (NVIDIA et al., 2025), $\pi_{0.5}$ (Intelligence et al., 2025), and NORA-1.5 (Hung et al., 2025).

For real-world evaluation, we conduct a large-scale, systematic study on both a bimanual YAM setup and a mobile bimanual YAM setup, covering 8 tasks that span static laboratory settings, in-the-wild environments (kitchen, study room, pantry, and wet labs), and mobile manipulation. The tasks—`stack_cups`, `store_test_tubes`, `store_candy`, `hang_tools`, `store_toys`, `shelf_cups`, `prepare_pipette`, and `make_popcorn`—are each evaluated over 50 trials, with three spatial variants per task used for data collection. We fine-tune from the MolmoAct2-Post checkpoint for every task and compare against four strong VLA and world-model baselines: Cosmos Policy (Kim et al., 2026a), X-VLA (Zheng et al., 2025), OpenVLA-OFT (Kim et al., 2025), and $\pi_{0.5}$-DROID (Intelligence et al., 2025).

Results. On LIBERO, MolmoAct2 achieves an average success rate of 97.2%, the highest among all compared methods—reaching 100% on LIBERO-Object and improving over our previous MolmoAct-7B-D by 10.6%. This trend is reinforced on RoboEval, where MolmoAct2 attains a 44.3% success rate, surpassing the second-best model, $\pi_{0.5}$ (Intelligence et al., 2025), by 3.8%. In the real-world evaluation, MolmoAct2 outperforms all baselines on 7 of 8 deployment tasks, achieving an average success rate of 50.1%—15% above the runner-up, OpenVLA-OFT. Hence, MolmoAct2 demonstrate its effectiveness for rapid adaptation to new tasks, domains, and embodiments.

6.4 Performance of MolmoAct2-Think

To isolate the contribution of MolmoAct2-Think, we fine-tuned it with the adaptive-depth training pipeline and compared it against MolmoAct2, fine-tuned from MolmoAct2-Post under the standard recipe on

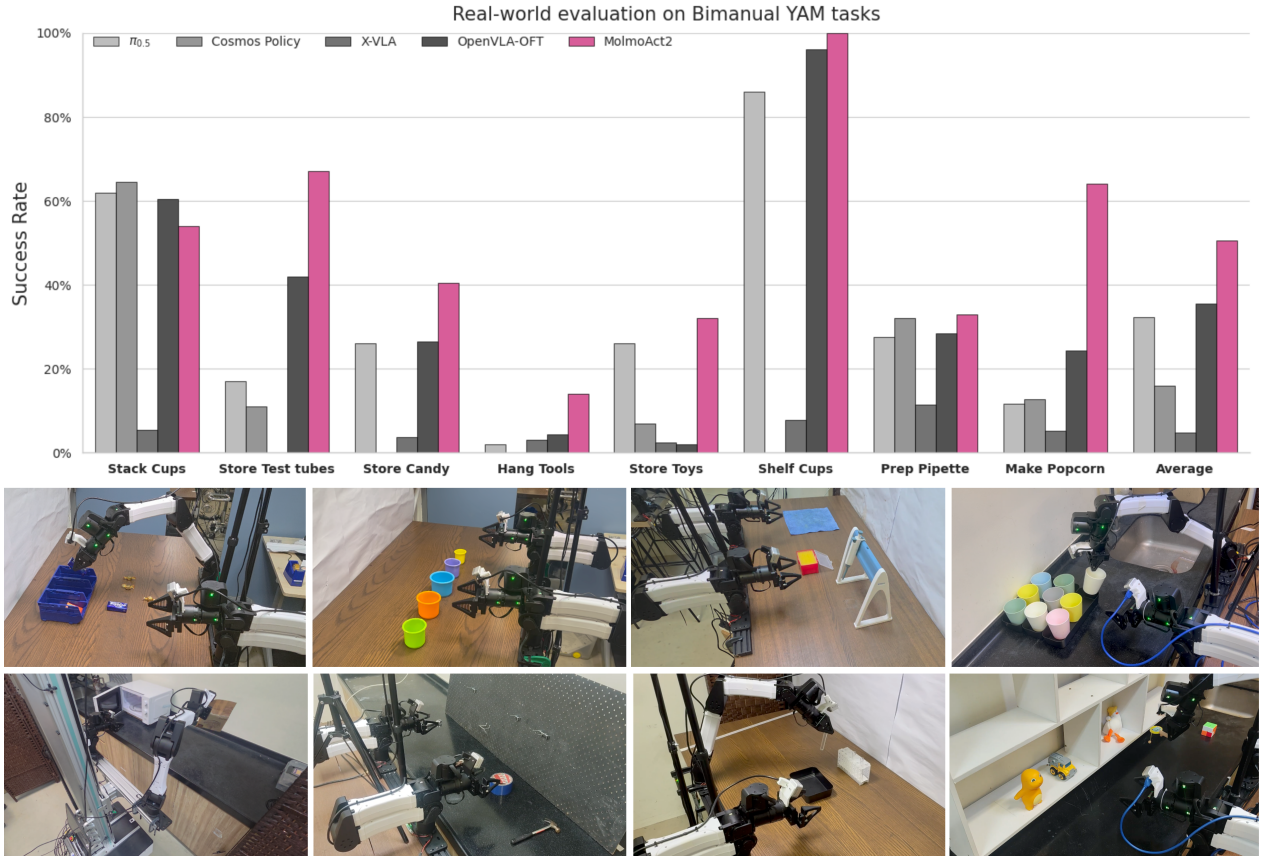

Figure 7 Overview of efficient fine-tuning of MolmoAct2. We conduct a comprehensive evaluation across 8 real-world tasks outside of a laboratory setting, ranging from preparing a pipette for a chemist to placing toys back onto a shelf. MolmoAct2, fine-tuned on the collected evaluation data, outperforms 4 strong baselines by a large margin of 15% over the runner-up model.

LIBERO. MolmoAct2-Think improves over MolmoAct2 on three of the four task suites and matches it on the fourth, where the baseline is already saturated at 100%. Crucially, the largest gain (+2.2%) appears on the most challenging suite, where the baseline leaves the most headroom (93.2%), while gains on the easier suites are correspondingly smaller as performance approaches ceiling. Averaged across all suites and 2,000 rollouts, MolmoAct2-Think achieves 98.1% versus 97.2%, a +0.9% improvement that is consistent in sign and concentrated where the task is hardest which indicates that the adaptive-depth pipeline yields a real, non-incidental gain rather than noise around saturation.

6.5 Robustness of MolmoAct2 to out-of-distribution shifts

Evaluation setup and baselines. To investigate how robustly MolmoAct2 transfers after parameter-efficient fine-tuning on a new dataset, we select 4 of the 8 real-world bimanual tasks (`stack_cups`, `store_test_tubes`, `store_candy`, and `prepare_pipette`) and fine-tune a separate checkpoint for each on top of MolmoAct2-Post. At test time, we evaluate each checkpoint under four out-of-distribution variants: *spatial*, in which we place the target objects at locations outside the training distribution; *lighting*, in which we alter the illumination of the scene; *language variants*, in which we rephrase the language instruction in three different ways; and *distractors*, in which we add unseen distractor objects while keeping the original in-distribution spatial layout. We compare MolmoAct2 to strong baselines such as $\pi_{0.5}$ (Intelligence et al., 2025), Cosmos Policy (Kim et al., 2026a), X-VLA (Wen et al., 2025), and OpenVLA-OFT (Kim et al., 2025) .For each task each set, we will evaluate 20 trials, 5 trials for each perturbation.

Results. Based on the evaluation, we find that MolmoAct2 is substantially more robust to perturbations than

Table 9 Robustness of MolmoAct2 under environmental perturbations. We evaluate MolmoAct2 against four baselines on out-of-distribution scenarios spanning spatial variation, lighting, language rephrasing, and visual distractors. MolmoAct2 is the most robust, achieving an average success rate of **50.69%** across all perturbation types—a **10.80%** absolute improvement over the next-best model. Values report average success rates (%); **bold** = best, underline = second-best.

Model	Spatial Var.	Lighting	Language	Distractor	Overall
$\pi_{0.5}$ (Intelligence et al., 2025)	<u>15.00</u>	33.70	26.15	33.20	27.01
Cosmos Policy (Kim et al., 2026a)	8.75	17.50	5.00	13.75	11.25
X-VLA (Wen et al., 2025)	3.75	8.30	9.95	3.75	6.44
OpenVLA-OFT (Kim et al., 2025)	13.75	<u>46.25</u>	<u>51.25</u>	<u>48.30</u>	<u>39.89</u>
MolmoAct2-Think	**26.25**	**62.05**	**60.35**	**54.10**	**50.69**

all other evaluated models, achieving a 50.7% average success rate across all perturbation types, with a margin of ~10.8% over the second-best model, OpenVLA-OFT. While MolmoAct2 leads on every perturbation category, its advantage is narrowest on Distractor (a 5.8% margin over OpenVLA-OFT), and it attains its lowest absolute score on Spatial Variance (26.25%), indicating room for improvement on fine-grained spatial generalization.

6.6 Quality of MolmoAct2 trajectories for real-world deployment

While task success is a necessary measure of performance, it is insufficient for assessing readiness for real-world deployment. In practical settings, trajectory quality directly impacts efficiency, stability, and safety. We therefore evaluate MolmoAct2 fine-tuned on RoboEval for beyond success rates using a suite of behavioral and outcome metrics from RoboEval (Wang et al., 2025b), summarized in Fig. 6B. Across tasks, MolmoAct2 demonstrates consistent improvements in efficiency-related metrics. For example, on *Stack Two Blocks*, MolmoAct2 reduces completion time from 5.87s ($\pi_{0.5}$) and 7.27s (Diffusion) to 4.70s, while also reducing joint path length from 2.16 to 1.04 (approximately 2× shorter). Similar trends hold on longer-horizon tasks such as *Rotate Valve*, where MolmoAct2 achieves the lowest completion time (8.51s vs. 9.69s for $\pi_{0.5}$), along with reduced trajectory length.

In addition to efficiency, MolmoAct2 exhibits strong performance on stability-related metrics. As shown in Fig. 6B, MolmoAct2 consistently operates near the best normalized value across both Cartesian and joint-space measures, indicating smoother and more stable trajectories compared to baselines. In contrast, other methods exhibit greater variability across these metrics, suggesting less consistent control behavior. Together, these results show that MolmoAct2 not only improves task success, but also produces shorter, more stable, and more efficient trajectories, which are critical properties for real-world deployment.

6.7 Systematic analysis

We ablate the main design choices in MolmoAct2 on LIBERO to isolate where the performance gains come from. Unless otherwise noted, all variants use the same data, image augmentation, action normalization, action horizon, and evaluation protocol as the corresponding LIBERO fine-tuning run. For the full-suite ablations, each row is evaluated on Spatial, Object, Goal, and Long, and the average is computed across the four suites.

Embodied-reasoning backbone. We first test whether the Molmo2-ER backbone improves action learning even before the continuous action expert is introduced. We directly fine-tune Molmo2 and Molmo2-ER with the same discrete-action architecture on the single LIBERO Long task suite, using action tokens produced by MolmoAct2-FAST Tokenizer and no continuous action expert. Both models are trained for 60,000 steps with the same configuration. As shown in Table 10, replacing Molmo2 with Molmo2-ER improves success from 77.6% to 83.6%, a 6.0-point gain. This confirms that the embodied-reasoning specialization is not only useful for VLM benchmarks, but also transfers directly into action-token prediction.

Table 10 Backbone ablation on LIBERO Long. Both variants are trained directly from the VLM checkpoint with only discrete action prediction using MolmoAct2-FAST Tokenizer. The final row is marked in pink, which is what we initialize with for MolmoAct2-Pretrain.

Backbone	Action objective	Long
Molmo2 (Clark et al., 2026)	Discrete	77.6%
Molmo2-ER	Discrete	**83.6%**

VLM-to-expert conditioning. We next ablate how the continuous action expert receives context from the VLM. All variants start from MolmoAct2-Pretrain, attach an action expert directly, and use the same LIBERO training configuration. We compare hidden-state conditioning, standard per-layer KV conditioning, and a per-head per-layer KV conditioning variant. In the standard design, the VLM key and value heads at each token are flattened before learned projections map them into the action expert cross-attention space. In the per-head variant, the VLM keys and values remain separated by head, and each head is projected into the corresponding action-expert head. This preserves the head structure, but requires the number of VLM KV heads to match the number of action-expert KV heads; we use 8 for both in all settings. The standard per-layer KV conditioning is strongest on average, reaching 95.9%, compared with 94.8% for the per-head variant and 94.0% for hidden-state conditioning (Table 11). Hidden-state conditioning is competitive on Spatial, but it drops more on Object, Goal, and Long. This supports the architectural choice in Sec. 4.2.1, where the action expert consumes the same attention state used by the VLM rather than a single residual-stream representation.

Table 11 Ablation of the VLM-to-action-expert conditioning source on LIBERO. All variants are trained from MolmoAct2-Pretrain with the same configuration. **Bold** denotes the best result per column. The final row is marked in pink, which is what we use in all MolmoAct2 and MolmoAct2-Think post-training and fine-tuning runs.

Conditioning source	Spatial	Object	Goal	Long	Average
Hidden-state conditioning	**97.0%**	95.4%	96.6%	87.0%	94.0%
Per-head per-layer KV conditioning	95.8%	97.2%	96.6%	89.6%	94.8%
Per-layer KV conditioning	96.2%	**99.0%**	**98.6%**	**89.8%**	**95.9%**

Multiple flow samples. We then vary the number of flow samples K per action chunk while holding per-layer KV conditioning and all other training settings fixed. These runs also start from MolmoAct2-Pretrain without a prior post-training stage. Table 12 shows that increasing K generally improves average performance. $K = 1$ is the weakest overall at 94.15%, while $K = 8$ gives the best average at 95.90%. The effect is not monotonic on every suite, because Long benefits strongly from two flow samples, but the aggregate trend favors denser supervision along the flow trajectory.

Table 12 Ablation of the number of flow samples on LIBERO. All rows use per-layer KV conditioning and are trained from MolmoAct2-Pretrain with the same configuration. **Bold** denotes the best result per column. The final row is marked in pink, which is what we stick to in all of MolmoAct2 and MolmoAct2-Think fine-tuning runs.

K	Spatial	Object	Goal	Long	Average
1	95.0%	95.8%	98.2%	87.6%	94.15%
2	**96.6%**	96.0%	95.4%	**92.2%**	95.05%
4	96.2%	98.4%	97.0%	89.0%	95.15%
8	96.2%	**99.0%**	**98.6%**	89.8%	**95.90%**

Fine-tuning design. Table 13 compares the final MolmoAct2 fine-tuning recipe against targeted alternatives along three axes: whether we co-train the discrete autoregressive action loss, whether we apply knowledge insulation to the flow-matching loss, and which parameters are updated. Removing discrete action co-training

yields a similar average but shifts performance across suites, improving Long while reducing Spatial and Object. Adding knowledge insulation is also close but slightly worse than the final recipe. LoRA remains strong, especially on Spatial, but loses 2.8 points on Long relative to full fine-tuning. Tuning only the action expert is the clearest failure mode, dropping the average to 93.05%. Overall, full-model adaptation with discrete and continuous action co-training provides the best average outcome.

Table 13 Ablation of MolmoAct2 fine-tuning design choices on LIBERO. Component columns indicate whether each design choice is enabled: ✓ denotes enabled and ✗ denotes disabled. **Bold** denotes the best result per column. The final row is the MolmoAct2-LIBERO recipe, where its training design is marked in pink, which is what we stick to in all of MolmoAct2 fine-tuning runs.

Discrete co-training	Knowledge insulation	Training type	Spatial	Object	Goal	Long	Average
✗	✗	Full fine-tuning	95.8%	98.6%	**98.4%**	**95.0%**	96.95%
✓	✓	Full fine-tuning	96.8%	99.4%	97.2%	94.8%	97.05%
✓	✗	LoRA	**98.0%**	99.4%	97.2%	90.4%	96.25%
✗	✗	Action expert only	91.6%	97.6%	93.8%	89.2%	93.05%
✓	✗	Full fine-tuning	97.8%	**100.0%**	97.8%	93.2%	**97.20%**

Depth-aware fine-tuning. Finally, we ablate the MolmoAct2-Think depth fine-tuning recipe. The baseline uses both 10% depth-token noise and the learned per-layer depth gate described in Sec. 5.2, while uniformly mixing action-only and depth-and-action examples. Removing the depth-token noise and per-layer depth gate reduces the average from 98.10% to 97.65%, mostly through a 1.8-point drop on Goal. Removing mixed training as well, so that training uses only depth-and-action examples, reduces the average further to 97.50%. These results show that the depth pathway is most useful when it is regularized for imperfect inference-time depth predictions and when the policy retains a strong action-only path.

Table 14 Ablation of MolmoAct2-Think depth fine-tuning choices on LIBERO. Mixed training denotes uniform sampling of action-only and depth-and-action examples. ✓ denotes enabled and ✗ denotes disabled. **Bold** denotes the best result per column. The final all-enabled row is the MolmoAct2-Think-LIBERO recipe, where its training design is marked in pink, which is what we stick to in all of MolmoAct2-Think fine-tuning runs.

Mixed training	Noise injection	Depth gate	Spatial	Object	Goal	Long	Average
✗	✗	✗	97.6%	99.6%	96.6%	**96.2%**	97.50%
✓	✗	✗	**98.8%**	**99.8%**	96.6%	95.4%	97.65%
✓	✓	✓	**98.8%**	**99.8%**	**98.4%**	95.4%	**98.10%**

6.8 Inference speed

We measure end-to-end action-generation latency on LIBERO using a single H100 GPU and an action horizon of 10, and report the amortized control rate as action horizon divided by latency. We compare three inference paths: the original implementation, an optimized eager path using reusable-cache optimizations, and the same optimized path with CUDA Graph replay enabled.

As shown in Figure 8, caching alone improves MolmoAct2 from 23.02 Hz to 27.39 Hz and MolmoAct2-Think from 8.04 Hz to 9.72 Hz. Enabling CUDA Graphs gives the larger gain: MolmoAct2 reaches 55.79 Hz, a 2.42× speedup over the original path, while MolmoAct2-Think reaches 12.71 Hz, a 1.58× speedup. MolmoAct2 benefits substantially from CUDA Graph replay: its fixed-shape flow-matching steps form a regular, repeated computation pattern whose runtime is dominated by kernel launch overhead, which graph replay largely eliminates. MolmoAct2-Think sees a smaller gain, since its adaptive-depth stage performs autoregressive decoding, whose sequential dependencies and variable-length execution are less amenable to graph capture.

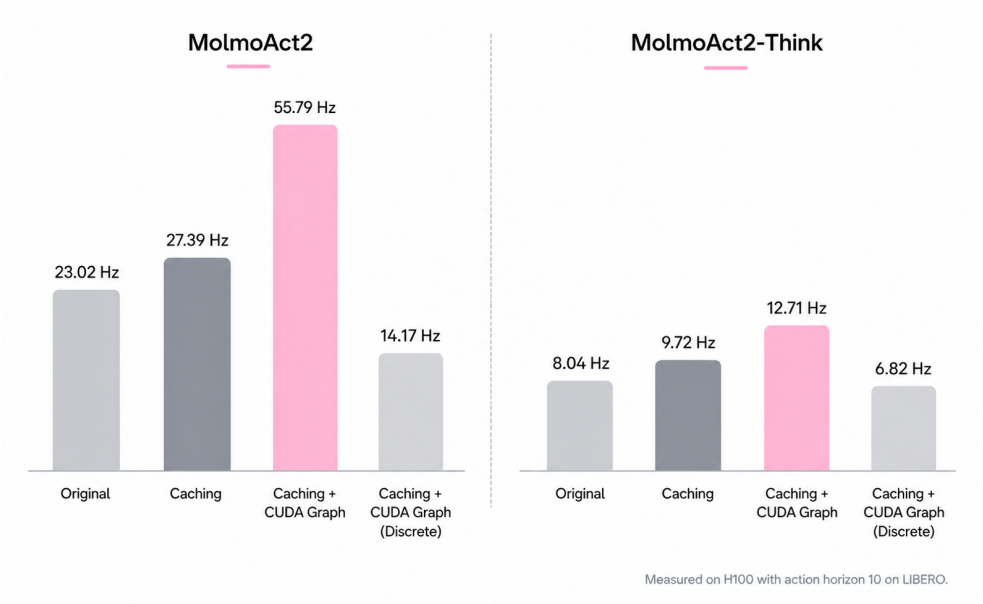

Figure 8 Inference control rate after caching and CUDA Graph optimizations. Control rate is action horizon divided by end-to-end action-generation latency; measurements use horizon 10 on LIBERO with a single H100.

MolmoAct2 also exposes a discrete action path, in which the VLM autoregressively decodes action tokens for the entire action chunk. We measured this path under the same optimized graph as above. MolmoAct2 runs at 14.17 Hz and MolmoAct2-Think at 6.82 Hz i.e., 3.94× and 1.86× slower than the corresponding continuous path. This gap comes from the large VLM decoding overhead, while the action expert emits the entire chunk in a small fixed number of flow matching steps through a smaller cross-attention head, with the VLM cache reused across steps. We therefore use the continuous path as the default deployment option.

7 Related work

Generalist robot manipulation policies. In recent years, robotic manipulation research has shifted focus from task-specific designs toward developing generalist policies capable of operating across diverse scenarios and embodiments (Berscheid et al., 2019; Brohan et al., 2022; Dasari et al., 2019; Ebert et al., 2021; Fang et al., 2023; Jang et al., 2022; Khazatsky et al., 2024; Mandlekar et al., 2018; Walke et al., 2023; Shafiullah et al., 2023). A representative approach is the development of vision-language-action (VLA) models. In this paradigm, vision-language models (VLMs)—pretrained on vast, web-scale multimodal data (e.g., image, video, text)—are further trained on collection of diverse robotics datasets (Black et al., 2024; Brohan et al., 2022; Zheng et al., 2024; Zhao et al., 2025; Team et al., 2025b; Li et al., 2024b; Qu et al., 2025; Kim et al., 2024). This knowledge transfer significantly reduces the reliance on domain-specific robotics data, which is typically expensive to collect via teleoperation. However, with the notable exception of our previous work, MolmoAct (Lee et al., 2025), reproducing frontier VLA models (e.g., the π series (Black et al., 2024; Intelligence et al., 2025)) remains highly challenging due to closed-source training data and code. MolmoAct represents our latest effort to democratize VLA research by releasing full training code, data, and model checkpoints that achieve state-of-the-art performance across a wide range of robotic manipulation benchmarks.

Embodied reasoning for robotic manipulation. While textual chain-of-thought (CoT) prompting (Wei et al., 2022) has advanced reasoning in domains like program synthesis and mathematics, its benefits for robotic manipulation remain limited, likely because tacit physical knowledge is rarely articulated in the text corpora used to train backbone LLMs. This has motivated a growing body of work on non-textual 'embodied CoT' that leverages visual or spatial representations, including object bounding boxes and points (Zawalski et al., 2024; Yuan et al., 2024; Li et al., 2025), future image predictions (Zhao et al., 2025), latent reasoning (Tur

et al., 2026), and 2D/3D point trajectories (Sun et al., 2024; Huang et al., 2025; Lee et al., 2025). Closest to our adaptive depth approach is PEEK (Zhang et al., 2025), which instead masks the RGB observation and conditions on visual tracing to learn a low-level policy. A common drawback of these methods is heavy token consumption before action generation, which severely degrades inference latency. MolmoAct2-Think addresses this with adaptive depth reasoning: new depth tokens are computed only for novel visual signals, substantially improving latency, especially in third-person viewpoint setups.

Bridging VLM and action expert. While the architecture of VLA models has evolved significantly, the interface between VLM and action generators remains an open research problem. Recent approaches either fully discretize actions via vector quantization (Brohan et al., 2022; Kim et al., 2024; Pertsch et al., 2025) or append a continuous generative action expert with diffusion (Chi et al., 2025) or flow-matching objective (Lipman et al., 2022; Black et al., 2024). Standard continuous action experts typically condition only on the VLM's final hidden states. MolmoAct2 departs from this by introducing a deep, layer-wise conditioning interface. After pre-training the VLM on discrete action tokens produced by MolmoAct2-FAST Tokenizer for seamless multimodal data mixing, we post-train a continuous flow-matching expert using per-layer KV conditioning. Unlike final-layer conditioning (NVIDIA et al., 2025; Wang et al., 2026), this per-layer KV conditioning gives the action expert dense access to the backbone's hierarchical visual-semantic features. Finally, to explicitly ground this reasoning in 3D space without bottlenecking latency, MolmoAct2-Think precedes this continuous action generation with an adaptive generation of discrete depth tokens only for dynamic scene regions.

8 Conclusion

We introduced MolmoAct2, a family of fully open action reasoning models built for real-world deployment across heterogeneous robot platforms. Building upon a spatially specialized Molmo2-ER backbone, MolmoAct2 produces performant and geometrically grounded behaviors across diverse manipulation tasks. We additionally release MolmoAct2-Think, a thinking variant equipped with adaptive depth reasoning that delivers interpretable, depth-aware control with efficient inference. Our evaluations across simulation and real-world settings demonstrate that MolmoAct2 consistently outperforms strong VLA baselines out-of-the-box, fine-tunes efficiently from a handful of demonstrations, and transfers across three structurally different embodiments spanning the low-to-medium cost range. We release all model weights, training code, and data, including MolmoAct2-BimanualYAM Dataset, the largest open bimanual manipulation dataset to date, alongside MolmoAct2-DROID Dataset and MolmoAct2-SO100/101 Dataset, to enable reproducibility and foster community-driven research toward open foundation models that researchers and practitioners can both build on and deploy in the real world.

Author contributions

This project was made possible through the equal contributions of both co-first authors, listed in no particular order:

- **Haoquan Fang**: Led the design and implementation of the MolmoAct2 model, training, and inference pipelines and infrastructure; contributed to data curation, evaluations, and writing.
- **Jiafei Duan**: Led the project and core method design; proposed and curated MolmoAct2-BimanualYAM Dataset; led paper writing and the design of all simulation and real-world evaluations.

All other contributors are also deeply appreciated for their effort, which is critical to the success of the MolmoAct2 project. As not all of these can be captured, we indicate their primary contributing role in MolmoAct2:

- **Donovan Clay**: Led the training and curation of MolmoAct2-FAST Tokenizer
- **Sam Wang**: Led the curation and filtering of the pre-training data mixture for MolmoAct2
- **Weikai Huang**: Led the curation and training of Molmo2-ER
- **Xiang Fan**: Led the development of MolmoAct2-Think
- **Shirui Chen**: Led the curation of MolmoAct2-SO100/101 Dataset
- **Shanli Xing**: Led the inference optimization of MolmoAct2 and MolmoAct2-Think
- For real-world robot infrastructure: Shuo Liu, Wei-Chuan Tsai, Ying-Chun Lee, Shanli Xing, Angad Wadhwa and Rose Hendrix
- For real-world data collection, curation, and evaluation: Jiafei Duan, Donovan Clay, Angad Wadhwa, Suveen Ellawela, Lucas Ngoo, Cole Harrison and Sam Wang
- For simulation training and evaluation: Yi Ru Wang, Haoquan Fang, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, and Peter Sushko
- For paper writing and figures: Jiafei Duan, Haoquan Fang, Zhongzheng Ren, Shirui Chen, Jaemin Cho, Cole Harrison, Donovan Clay, Sam Wang, Shuo Liu, Xiang Fan, Winson Han, and Eli VanderBilt
- For project management: Karen Farley.
- For research advisory: Ranjay Krishna, Dieter Fox, Joyce Chai, Zhongzheng Ren, and Ali Farhadi.
- Project PI: Ranjay Krishna

Acknowledgment

This work would not be possible without the support of our colleagues at Ai2:

- We thank Christopher Clark, Abhay Deshpande, Yejin Kim, Max Argus, Ishneet Sukhvinder Singh and Wilbert Pumacay for helpful research discussions and sharing of relevant findings across related projects.
- We thank for David Albright, Crystal Nam, Kristin Cha, Cailin Brashear, Jae Pak, Sophie Lebrecht, Kyle Wiggers, Kelsey MacMillan, Katie Morigi, Peter Clark and Megan Bartot for project management, support to robot room and publicity of MolmoAct2
- We thank Yoganand Chandrasekhar, Johann Dahm, Michael Wilson, Fangzhou Hu, and Caroline Wu for their work on the Ai2 cluster.

MolmoAct2 would not have been possible without the support of many other institutions. In particular, we thank Cirrascale for their on-going support of Ai2's cluster. Jiafei Duan is supported by the Agency for Science, Technology and Research (A*STAR) National Science Fellowship.

References

S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

L. Berscheid, P. Meißner, and T. Kröger. Robot learning of shifting objects for grasping in cluttered environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

E. Brown, A. Ray, R. Krishna, R. Girshick, R. Fergus, and S. Xie. Sims-v: Simulated instruction-tuning for spatial video understanding, 2025. URL https://arxiv.org/abs/2511.04668.

Z. Cai, R. Wang, C. Gu, F. Pu, J. Xu, Y. Wang, W. Yin, Z. Yang, C. Wei, Q. Sun, T. Zhou, J. Li, H. E. Pang, O. Qian, Y. Wei, Z. Lin, X. Shi, K. Deng, X. Han, Z. Chen, X. Fan, H. Deng, L. Lu, L. Pan, B. Li, Z. Liu, Q. Wang, D. Lin, and L. Yang. Scaling spatial intelligence with multimodal foundation models, 2026. URL https://arxiv.org/abs/2511.13719.

Q. Chen, J. Yu, M. Schwager, P. Abbeel, Y. Shentu, and P. Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. *arXiv preprint arXiv:2509.25358*, 2025.

S. Chen. Depi: Pi0 training with large decentralized-collected so100 dataset. https://github.com/chinsengi/depi, 2025.

S. Chen, C. Harrison, Y.-C. Lee, A. J. Yang, Z. Ren, L. J. Ratliff, J. Duan, D. Fox, and R. Krishna. Topreward: Token probabilities as hidden zero-shot rewards for robotics. https://topreward.github.io/webpage/, 2026. Project page.

L. Cheng, J. Duan, Y. R. Wang, H. Fang, B. Li, Y. Huang, E. Wang, A. Eftekhar, J. Lee, W. Yuan, et al. Pointarena: Probing multimodal grounding through language-guided pointing. *arXiv preprint arXiv:2505.09990*, 2025.

C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2025.

C. Clark, J. Zhang, Z. Ma, J. S. Park, M. Salehi, R. Tripathi, S. Lee, Z. Ren, C. D. Kim, Y. Yang, V. Shao, Y. Yang, W. Huang, Z. Gao, T. Anderson, J. Zhang, J. Jain, G. Stoica, W. Han, A. Farhadi, and R. Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding, 2026. URL https://arxiv.org/abs/2601.10611.

G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine. The ingredients for robotic diffusion transformers. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15617–15625. IEEE, 2025.

M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv e-prints*, pages arXiv–2409, 2024.

S. Deng, M. Yan, Y. Zheng, J. Su, W. Zhang, X. Zhao, H. Cui, Z. Zhang, and H. Wang. Stereovla: Enhancing vision-language-action models with stereo vision. *arXiv preprint arXiv:2512.21970*, 2025.

A. Deshpande, M. Guru, R. Hendrix, S. Jauhri, A. Eftekhar, R. Tripathi, M. Argus, J. Salvador, H. Fang, M. Wallingford, et al. Molmob0t: Large-scale simulation enables zero-shot manipulation. *arXiv preprint arXiv:2603.16861*, 2026.

D. Driess, J. T. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Z. Ren, H. Walke, Q. Vuong, L. X. Shi, and S. Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025. URL https://arxiv.org/abs/2505.23705.

M. Du, B. Wu, Z. Li, X.-J. Huang, and Z. Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2024.

F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. *arXiv preprint arXiv:2307.00595*, 2023.

X. Fu, Y. Hu, B. Li, Y. Feng, H. Wang, X. Lin, D. Roth, N. A. Smith, W.-C. Ma, and R. Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, 2024.

K. Grauman, A. Westbury, L. Torresani, K. Kitani, J. Malik, T. Afouras, K. Ashutosh, V. Baiyya, S. Bansal, B. Boote, E. Byrne, Z. Chavis, J. Chen, F. Cheng, F.-J. Chu, S. Crane, A. Dasgupta, J. Dong, M. Escobar, C. Forigua, A. Gebreselasie, S. Haresh, J. Huang, M. M. Islam, S. Jain, R. Khirodkar, D. Kukreja, K. J. Liang, J.-W. Liu, S. Majumder, Y. Mao, M. Martin, E. Mavroudi, T. Nagarajan, F. Ragusa, S. K. Ramakrishnan, L. Seminara, A. Somayazulu, Y. Song, S. Su, Z. Xue, E. Zhang, J. Zhang, A. Castillo, C. Chen, X. Fu, R. Furuta, C. Gonzalez, P. Gupta, J. Hu, Y. Huang, Y. Huang, W. Khoo, A. Kumar, R. Kuo, S. Lakhavani, M. Liu, M. Luo, Z. Luo, B. Meredith, A. Miller, O. Oguntola, X. Pan, P. Peng, S. Pramanick, M. Ramazanova, F. Ryan, W. Shan, K. Somasundaram, C. Song, A. Southerland, M. Tateno, H. Wang, Y. Wang, T. Yagi, M. Yan, X. Yang, Z. Yu, S. C. Zha, C. Zhao, Z. Zhao, Z. Zhu, J. Zhuo, P. Arbelaez, G. Bertasius, D. Crandall, D. Damen, J. Engel, G. M. Farinella, A. Furnari, B. Ghanem, J. Hoffman, C. V. Jawahar, R. Newcombe, H. S. Park, J. M. Rehg, Y. Sato, M. Savva, J. Shi, M. Z. Shou, and M. Wray. Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives, 2024. URL https://arxiv.org/abs/2311.18259.

M. Guerquin. Introducing AI2's beaker. *AI2 Blog*, 2022. URL https://web.archive.org/web/20241231204439/https://medium.com/ai2-blog/beaker-ed617d5f4593. Accessed: 2024-12-31. Original: https://medium.com/ai2-blog/beaker-ed617d5f4593.

C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *arXiv preprint arXiv:2507.16815*, 2025.

C.-Y. Hung, Q. Sun, P. Hong, A. Zadeh, C. Li, U. Tan, N. Majumder, S. Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2022.

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, 2016. URL https://arxiv.org/abs/1612.06890.

A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026a.

Y. Kim, W. Pumacay, O. Rayyan, M. Argus, W. Han, E. VanderBilt, J. Salvador, A. Deshpande, R. Hendrix, S. Jauhri, et al. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *arXiv preprint arXiv:2602.11337*, 2026b.

N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, S. Lyu, Y. Gu, S. Malik, V. Graf, J. D. Hwang, J. Yang, R. L. Bras, O. Tafjord, C. Wilhelm, L. Soldaini, N. A. Smith,

Y. Wang, P. Dasigi, and H. Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, W. Han, W. Pumacay, A. Wu, R. Hendrix, K. Farley, E. VanderBilt, A. Farhadi, D. Fox, and R. Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

J. H. Lee, M. Kerzel, K. Ahrens, C. Weber, and S. Wermter. What is right for me is not yet right for you: A dataset for grounding relative directions via multi-task learning. In *International Joint Conference on Artificial Intelligence*, Jul 2022. URL https://www.ijcai.org/proceedings/2022/0145.pdf.

B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024b.

Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, R. Yu, C. R. Garrett, F. Ramos, D. Fox, A. Li, et al. Hamster: Hierarchical action models for open-world robot manipulation. *arXiv preprint arXiv:2502.05485*, 2025.

Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

Z. Luo, C. Zhang, S. Yong, C. Dai, Q. Wang, H. Ran, G. Shi, K. P. Sycara, and Y. Xie. pyspatial: Generating 3d visual programs for zero-shot spatial reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

T. Motoda, M. Murooka, R. Nakajo1, M. Muttaqien, K. Makihara, H. Oh, K. Shirai, F. Erich, R. Hanai, and Y. Domae. Aist bimanual manipulation dataset: A dataset for learning bimanual robot control, collected via leader-follower teleoperation. https://aistairc.github.io/aist_bimanip_site/, 2025.

J. Ning, C. Li, Z. Zhang, Z. Geng, Q. Dai, K. He, and H. Hu. All in tokens: Unifying output space of visual tasks via soft token, 2023. URL https://arxiv.org/abs/2301.02229.

NVIDIA, J. Bjorck, N. C. Fernando Castañeda, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. 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)*, 2024.

K. Pertsch, A. Khazatsky, and DROID Team. DROID annotations: Extended language labels and improved camera calibrations. https://huggingface.co/KarlP/droid, 2024.

K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

A. Ray, J. Duan, E. Brown, R. Tan, D. Bashkirova, R. Hendrix, K. Ehsani, A. Kembhavi, B. A. Plummer, R. Krishna, K.-H. Zeng, and K. Saenko. Sat: Dynamic spatial aptitude training for multimodal language models, 2025. URL https://arxiv.org/abs/2412.07755.

P. Sermanet, T. Ding, J. Zhao, F. Xia, D. Dwibedi, K. Gopalakrishnan, C. Chan, G. Dulac-Arnold, S. Maddineni, N. J. Joshi, P. Florence, W. Han, R. Baruch, Y. Lu, S. Mirchandani, P. Xu, P. Sanketi, K. Hausman, I. Shafran, B. Ichter, and Y. Cao. Robovqa: Multimodal long-horizon reasoning for robotics, 2023. URL https://arxiv.org/abs/2311.00899.

N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, S. Alibert, M. Cord, T. Wolf, and R. Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

A. Singh, A. Fry, A. Perelman, A. Tart, A. Ganesh, A. El-Kishky, A. McLaughlin, A. Low, A. Ostrow, A. Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

C. H. Song, V. Blukis, J. Tremblay, S. Tyree, Y. Su, and S. Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025.

Q. Sun, P. Hong, T. D. Pala, V. Toh, U. Tan, D. Ghosal, S. Poria, et al. Emma-x: An embodied multimodal action model with grounded chain of thought and look-ahead spatial reasoning. *arXiv preprint arXiv:2412.11974*, 2024.

G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

G. A. Team. Gen-1: Scaling embodied foundation models to mastery. *Generalist AI Blog*, 2026a. https://generalistai.com/blog/apr-02-2026-GEN-1.

G. R. Team, A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025a.

G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025b.

R. A. Team. Causal video models are data-efficient robot policy learners. *Rhoda AI Blog*, 2026b.

S. Tong, E. Brown, P. Wu, S. Woo, M. Middepogu, S. C. Akula, J. Yang, S. Yang, A. Iyer, X. Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. In *NeurIPS*, 2024.

M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Y. Tur, J. Naghiyev, H. Fang, W.-C. Tsai, J. Duan, D. Fox, and R. Krishna. Recurrent-depth vla: Implicit test-time compute scaling of vision-language-action models via latent iterative reasoning. *arXiv preprint arXiv:2602.07845*, 2026.

B. Tversky. Visualizing thought. In *Handbook of human centric visualization*. Springer, 2013.

B. Tversky. *Mind in motion: How action shapes thought*. Basic Books, 2019.

B. Tversky. Your body thinks as much as your mind. *IAI News*, Aug. 2025. URL https://iai.tv/articles/your-body-thinks-as-much-as-your-mind-auid-3282. Institute of Art and Ideas.

H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, 2023.

W. Wang, M. Ghobadi, K. Shakeri, Y. Zhang, and N. Hasani. Rail-only: A low-cost high-performance network for training llms with trillion parameters. *2024 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 1–10, 2023. URL https://api.semanticscholar.org/CorpusID:260125277.

W. Wang, Z. Gao, L. Gu, H. Pu, L. Cui, X. Wei, Z. Liu, L. Jing, S. Ye, J. Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025a.

Y. Wang, P. Ding, L. Li, C. Cui, Z. Ge, X. Tong, W. Song, H. Zhao, W. Zhao, P. Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *AAAI*, 2026.

Y. R. Wang, C. Ung, G. Tannert, J. Duan, J. Li, A. Le, R. Oswal, M. Grotz, W. Pumacay, Y. Deng, et al. Roboeval: Where robotic manipulation meets structured and scalable evaluation. *arXiv preprint arXiv:2507.00435*, 2025b.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

J. Yang, S. Yang, A. Gupta, R. Han, L. Fei-Fei, and S. Xie. Thinking in Space: How Multimodal Large Language Models See, Remember and Recall Spaces. In *CVPR*, 2025b.

L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao. Depth anything v2. *arXiv:2406.09414*, 2024.

R. Yang, Z. Zhu, Y. Li, J. Huang, S. Yan, S. Zhou, Z. Liu, X. Li, S. Li, W. Wang, Y. Lin, and H. Zhao. Visual spatial tuning, 2025c. URL https://arxiv.org/abs/2511.05491.

S. Yang, J. Yang, P. Huang, E. Brown, Z. Yang, Y. Yu, S. Tong, Z. Zheng, Y. Xu, M. Wang, D. Lu, R. Fergus, Y. LeCun, L. Fei-Fei, and S. Xie. Cambrian-s: Towards spatial supersensing in video, 2025d. URL https://arxiv.org/abs/2511.04670.

W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

L. Zha, A. J. Hancock, M. Zhang, T. Yin, Y. Huang, D. Shah, A. Z. Ren, and A. Majumdar. Lap: Language-action pre-training enables zero-shot cross-embodiment transfer. *arXiv preprint arXiv:2602.10556*, 2026.

J. Zhang, M. Memmel, K. Kim, D. Fox, J. Thomason, F. Ramos, E. Bıyık, A. Gupta, and A. Li. Peek: Guiding and minimal image representations for zero-shot generalization of robot manipulation policies. *arXiv preprint arXiv:2509.18282*, 2025.

Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *CVPR*, 2025.

T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

E. Zhou, J. An, C. Chi, Y. Han, S. Rong, C. Zhang, P. Wang, Z. Wang, T. Huang, L. Sheng, et al. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. *arXiv preprint arXiv:2506.04308*, 2025.

C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

Appendix

The appendix includes the following sections:

A Model Details

This appendix expands the model description in Sec. 4.1 and Sec. 4.2. MolmoAct2 is built in two architectural stages. First, MolmoAct2-Pretrain adapts the Molmo2-ER vision-language backbone into a discrete autoregressive robot policy. It keeps the Molmo2 visual and text token interface, but adds robot-specific state, setup/control, and action-output tokens so that robot trajectories can be trained with the same next-token objective as vision-language data. Second, post-training attaches a continuous DiT-style action expert to this already robot-aware backbone. The expert uses per-layer KV conditioning on the VLM and learns a flow-matching velocity field over continuous robot action chunks. MolmoAct2-Think keeps the same architecture and adds an autoregressive depth-token prefix before action prediction.

A.1 MolmoAct2 Backbone

Backbone initialization and visual inputs. MolmoAct2 initializes from Molmo2-ER, a Molmo2-based VLM specialized for embodied and spatial reasoning. The backbone therefore follows the Molmo2 architecture (Clark et al., 2026): visual inputs are encoded by a SigLIP2 vision transformer, pooled and projected by a vision-language connector, and consumed by an autoregressive language model together with text. The important MolmoAct2-specific choice is how this interface is used for robot control. Robot examples can include multiple camera views, proprioceptive state tokens, setup/control descriptors, and action targets, so sequence length is a central constraint. For the robot-policy stages in Sec. 4.1 and Sec. 4.2, each camera observation is represented as a single resized crop rather than a tiled high-resolution image. Video examples are sampled at up to 2 FPS and capped to at most 8 frames. Multi-camera robot episodes are serialized as multiple image inputs, and the camera order is randomized at the episode level during pre-training.

Vision encoder and connector. Each crop is encoded by a SigLIP2 vision transformer (Tschannen et al., 2025). The connector follows the Molmo2 design. It reads patch features from the third-to-last and ninth-from-last ViT layers, then pools local windows with a multi-head attention layer. For images, 2×2 patch windows are pooled into one visual token using the mean patch feature as the query. For videos, a 3×3 pooling window is used to further reduce the number of frame tokens. The pooled features are projected into the language-model embedding space with a shared MLP. This produces a sequence of visual tokens that can be interleaved with text tokens.

Language model input. The language model receives visual tokens together with the robot instruction and robot-specific text. Multi-camera observations are identified by image-index text, and video frames use the same frame markers as Molmo2. Since MolmoAct2 robot observations use single resized crops, the model does not rely on the multi-crop column-token machinery for robot control. Visual tokens are allowed to forward-attend to one another, including across different frames or camera views, so the backbone can reason over the full visual context before producing state-conditioned robot outputs.

Added tokens. The robot interface keeps the model in the same autoregressive format as the base VLM by adding robot-specific tokens to the backbone vocabulary. A robot example contains visual observations, a language instruction, optional setup and control descriptors, discrete state tokens, and an output marker that tells the model which response type to produce. The main markers are `<setup_start>`, `<setup_end>`,

Hyperparameter	Image Encoder	V/L Connector	LLM	Action Expert
Params	380M	57M	4.0B	621M
Dim	1152	1152	2560	768
MLP Dim	4304	9728	9728	3072
Act.	GELU	SwiGLU	SwiGLU	SwiGLU
Heads	16	16	32	8
KV Heads	16	–	8	–
Layers	27	–	36	36
Dropout	0.0	0.0	0.1	0.0
Image Size	384×384	–	–	–
Patch Size	14	–	–	–
Image Pool Size	–	2×2	–	–
Video Pool Size	–	3×3	–	–
Pool Dim	–	1152	–	–
Pool Heads	–	16	–	–
Embed	–	–	151936	–
Theta	–	–	1M	–
Max Horizon	–	–	–	30
Max Action Dim	–	–	–	32
Time Embed Dim	–	–	–	256
Conditioning	–	–	–	Per-layer KV

Table 15 MolmoAct2 architecture hyperparameters. The image encoder, connector, and LLM columns follow the 4B Molmo2-ER backbone used by MolmoAct2. The action-expert column describes the MolmoAct2 continuous flow-matching module used for continuous action prediction.

`<control_start>`, `<control_end>`, `<state_start>`, `<state_end>`, and `<action_output>`. We also add indexed token families for discrete robot quantities: action tokens `<action_0>`–`<action_2047>` and state tokens `<state_0>`–`<state_255>`. The depth tokens are added only for MolmoAct2-Think post-training: the model uses `<depth_output>`, `<depth_start>`, and `<depth_end>`, together with depth-code tokens `<depth_0>`–`<depth_127>`.

Setup and control descriptors make embodiment and action semantics explicit in the prompt. For example, a bimanual YAM prompt can indicate that the robot is a pair of YAM arms and that the control target is an absolute joint pose, while a DROID prompt can indicate a Franka arm with delta end-effector control. This lets the same tokenizer and backbone support different robots without forcing all datasets into a single physical coordinate convention.

Continuous robot states are normalized using dataset statistics and discretized into one of 256 state tokens per scalar dimension. The resulting state-token sequence is appended to the prompt before the action output marker. Continuous actions are normalized, padded to a maximum width of 32 dimensions, and encoded by MolmoAct2-FAST Tokenizer into discrete action tokens with a 2048-token action vocabulary. During pre-training, these action tokens are the only robot target. During post-training and fine-tuning, the same example also provides a continuous action chunk for the action expert. The discrete action-token span is masked from the expert conditioning path, so the continuous policy cannot condition on the ground-truth action tokens it is trained to predict.

A.2 Continuous Action Expert

Input and output. The action expert is a noncausal transformer over a short action chunk. It maps a noisy normalized trajectory $x_t \in \mathbb{R}^{H \times D_{\max}}$ to a velocity prediction of the same shape, where H is the action horizon and $D_{\max} = 32$ is the shared maximum action width. The released MolmoAct2 checkpoint uses $H = 30$

for post-training, while the LIBERO fine-tuned checkpoints use $H = 10$. The corresponding padding masks remove padded action steps and padded action dimensions from both the noisy input and the loss.

Flow-matching objective. Let a be a normalized target action chunk and let $\epsilon \sim \mathcal{N}(0, I)$. For a sampled time $t \in [0, 1]$, we construct

$$x_t = (1 - t)\epsilon + ta, \qquad u^\star = a - \epsilon. \tag{13}$$

The action expert f_θ predicts $u^\star$ from the noisy trajectory, the time embedding, and the VLM context c:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{a,\epsilon,t}\Big[\big\| m \odot \big(f_\theta(x_t, t, c) - u^\star\big)\big\|_2^2\Big], \tag{14}$$

where m masks padded action steps and padded action dimensions. In post-training, we evaluate multiple sampled flow-matching times for each robot action chunk while reusing the same VLM context. At inference, we initialize the trajectory from Gaussian noise and integrate the learned velocity field for a fixed number of Euler steps:

$$z_{i+1} = z_i + \Delta t\, f_\theta(z_i, t_i, c), \qquad t_i = \frac{i}{N}, \qquad \Delta t = \frac{1}{N}. \tag{15}$$

The released checkpoints use $N = 10$ inference steps. The final normalized trajectory is sliced to the embodiment-specific action width and unnormalized with the corresponding dataset statistics.

Expert block. The action expert has one transformer block for each VLM layer, giving $L = 36$ expert blocks. A noisy action chunk is first projected from 32 continuous dimensions to the expert hidden width. Each block contains bidirectional self-attention over the action chunk, cross-attention to the VLM context, and an MLP. A sinusoidal time embedding is passed through a small MLP, then used to produce DiT-style shift, scale, and gate parameters for all three residual branches. Schematically, block ℓ computes

$$h'_\ell = h_\ell + g^{\text{sa}}_\ell\, \text{SA}\big(\text{AdaRMS}^{\text{sa}}_\ell(h_\ell, t)\big), \tag{16}$$

$$\bar{h}_\ell = h'_\ell + g^{\text{ca}}_\ell\, \text{CA}\Big(\text{AdaRMS}^{\text{ca}}_\ell(h'_\ell, t), \tilde{K}_\ell, \tilde{V}_\ell\Big), \tag{17}$$

$$h_{\ell+1} = \bar{h}_\ell + g^{\text{ff}}_\ell\, \text{MLP}\Big(\text{AdaRMS}^{\text{ff}}_\ell(\bar{h}_\ell, t)\Big). \tag{18}$$

Here, AdaRMS denotes RMS normalization followed by the time-dependent affine modulation. The self-attention and cross-attention layers use query-key normalization. The expert uses rotary position embeddings for the action sequence, which gives the denoising transformer an explicit ordering over the predicted trajectory steps. After the final expert block, a modulated RMS normalization layer and a linear projection map the hidden states back to 32 action dimensions. The final projection is initialized to produce near-zero velocity predictions at the beginning of post-training, which makes the newly attached expert easier to optimize.

Per-layer KV conditioning. The action expert receives VLM context through per-layer KV conditioning. For VLM layer ℓ, let $(K^{\text{vlm}}_\ell, V^{\text{vlm}}_\ell)$ be the key and value tensors produced by that layer's self-attention over the prompt, images, state tokens, and non-target robot text. The expert uses learned adapter projections P_K and P_V to map these tensors into the expert cross-attention width:

$$\tilde{K}_\ell = \text{reshape}\Big(P_K K^{\text{vlm}}_\ell\Big), \qquad \tilde{V}_\ell = \text{reshape}\Big(P_V V^{\text{vlm}}_\ell\Big). \tag{19}$$

Expert block ℓ then cross-attends to $(\tilde{K}_\ell, \tilde{V}_\ell)$:

$$\text{CA}(Q_\ell, \tilde{K}_\ell, \tilde{V}_\ell) = \text{softmax}\left(\frac{Q_\ell \tilde{K}_\ell^\top}{\sqrt{d_h}}\right)\tilde{V}_\ell. \tag{20}$$

This gives each expert block direct access to the attention state at the same depth in the VLM. In the released architecture, the VLM has 8 KV heads with head dimension 128, so each token contributes a 1024-dimensional key and value state. The adapter projections map this state to the action expert width of 768, which is reshaped into 8 expert attention heads of width 96. In post-training, this conditioning path is detached from the VLM for the flow loss. The language-model loss still updates the VLM, while the flow loss trains the action expert and the VLM-to-expert adapter projections.

A.3 Adaptive-Depth Extension

MolmoAct2-Think keeps the same backbone and action expert, but adds an autoregressive depth-token interface before continuous action prediction. A depth response is requested with `<depth_output>` and serialized between `<depth_start>` and `<depth_end>`. The released depth checkpoints predict a 10×10 depth-token grid, giving 100 depth positions, and each position takes one of 128 learned depth-code values. For depth-only examples, the assistant response begins with `<depth_output>` and the model predicts depth tokens. For depth-and-action examples, the response begins with `<depth_output><action_output>`. The depth tokens are generated autoregressively, then included in the context used by the action expert.

The LIBERO depth fine-tuned model also includes a learned depth gate on the expert conditioning path. For action-expert layer ℓ, let $M_t = 1$ denote positions belonging to the depth-output trigger, depth delimiters, or depth-code tokens, and let A_t denote valid context positions. The gate is computed from the non-depth context of the corresponding VLM layer,

$$c_\ell = \frac{\sum_t A_t(1 - M_t)V_{\ell,t}^{\mathrm{vlm}}}{\sum_t A_t(1 - M_t)}, \qquad g_\ell = \sigma\big(w_\ell^\top c_\ell + b_\ell\big), \tag{21}$$

and then applied only to depth-token keys and values:

$$\bar{K}_{\ell,t}^{\mathrm{vlm}} = \big(1 - M_t + M_t g_\ell\big)\, K_{\ell,t}^{\mathrm{vlm}}, \qquad \bar{V}_{\ell,t}^{\mathrm{vlm}} = \big(1 - M_t + M_t g_\ell\big)\, V_{\ell,t}^{\mathrm{vlm}}. \tag{22}$$

The gated $\big(\bar{K}_\ell^{\mathrm{vlm}}, \bar{V}_\ell^{\mathrm{vlm}}\big)$ are then projected into the action expert as in subsection 4.2. The gate is initialized with bias -4, so fine-tuning begins close to the standard action-conditioning path and learns how strongly each expert layer should use the depth prefix.

B Training Details

B.1 Implementation

Training infrastructure. Our training implementation mainly follows Molmo2. We train in PyTorch with Fully Sharded Data Parallel v2 (FSDP2), and use PyTorch's scaled dot-product attention implementation rather than FlashAttention because our packed multimodal sequences and robot-conditioning masks require custom attention masks. We use `torch.compile` for throughput and keep the ViT and LLM shapes static through padding and fixed sequence budgets, which allows the compiled graph to be reused efficiently across training steps.

We use automatic mixed precision with `bfloat16` for most operations, while keeping numerically sensitive operations such as layer normalization and RoPE in full precision. Gradients are computed on local minibatches and then averaged across devices. For token-level losses, each device divides by the average number of loss tokens across all devices rather than by its local number of loss tokens. This avoids over-weighting examples with short target spans, which is important because our mixtures contain both short robot-action targets and longer language or video examples. During fine-tuning, dataset mixing is performed within each batch so that each update contains examples from multiple sources. Examples longer than the stage-specific maximum sequence length are truncated; this affects fewer than 0.1% of examples and usually occurs for long video examples with subtitles or dense annotations. We find training stable under this setup, without loss spikes or NaNs.

Packing. Our packing implementation also follows Molmo2. Each data-loader worker keeps a pool of $M = 48$ preprocessed and tokenized examples. When the pool is not full, new examples are drawn from the training mixture and added to the pool. Once the pool is full, a dynamic-programming solver selects the subset that maximizes

$$T + w_i I \quad \text{subject to} \quad T \le T_{\max},\ I \le I_{\max}, \tag{23}$$

where T is the total number of text tokens, I is the total number of image crops, and $w_i = 30$ balances token and image utilization. The selected examples are yielded as one packed sequence and removed from the pool. In the Molmo2 setting, $T_{\max} = 16384$ and $I_{\max} = 128$, while long-context training uses $T_{\max} = 36864$ and $I_{\max} = 384$. For MolmoAct2, the same solver is used with the stage-specific sequence budgets described in subsubsection 4.1.2 and subsubsection 4.2.2. In practice, we solve a quantized version of the packing problem by rounding token counts to the nearest multiple of 32.

Increasing the pool beyond 48 examples gives diminishing returns in packing efficiency. The method is also generally robust to w_i, although values that are too small can leave the pool dominated by examples with many image crops, which are difficult to pack with other examples. Implementing packing inside PyTorch's `DataLoader` makes it easy to use, since each worker runs the algorithm independently. This adds some overhead when many workers are used, but in practice data loading is still dominated by video decoding and frame extraction.

Image augmentation. We use the same image augmentation scheme in all of our robot pre-training, post-training, and fine-tuning runs. The image augmentation is applied to all images and videos during training, not limited to robot data. Augmentation is disabled for evaluation and inference.

The image augmentation applies the stochastic transforms before image normalization and final resizing. In simplified form, the recipe is:

```
import torchvision.transforms as T

height, width = image.height, image.width

transform = T.Compose([
    T.RandomCrop(size=(int(height * 0.95), int(width * 0.95))),
    T.Resize((height, width)),
    T.RandomRotation(degrees=5),
    T.ColorJitter(
        brightness=0.2,
        contrast=(0.8, 1.2),
        saturation=(0.8, 1.2),
        hue=0.05,
    ),
    T.RandomApply(
        [T.GaussianBlur(kernel_size=5, sigma=(0.1, 1.0))],
        p=0.2,
    ),
])

image = transform(image)
```

Input prompts. Robot examples are formatted with one of three output styles. Standard MolmoAct2 uses the action style, which asks the model to predict the robot action for the given instruction, setup, state, and control mode. MolmoAct2-Think uses all three styles: action prediction, depth prediction, and depth-then-action prediction. The depth-only style asks for the depth map of the main image. The depth-then-action style asks the model to first produce the depth representation and then produce the action, so that the continuous action expert can condition on the input context together with the predicted depth tokens.

After constructing the natural-language prompt, we wrap it with the same user/assistant chat template used by the Molmo2 formatter. For robot runs, the user message is placed between `<|im_start|>user` and `<|im_end|>`, followed by the assistant prefix `<|im_start|>assistant`. We then append an output trigger on the assistant side: action examples use `<action_output>`, depth examples use `<depth_output>`, and depth-then-action examples use `<depth_output><action_output>`. During training, the target response follows this trigger. During inference, we provide the same formatted user message, assistant prefix, and trigger as the generation prefix, which selects the desired output style before autoregressive decoding or continuous action generation begins.

Before insertion into the prompt, raw task strings are normalized into concise natural-language task descriptions. Dataset-specific bookkeeping is removed, punctuation and whitespace are standardized, and embodiment or control information is placed into the separate setup and control fields rather than being duplicated in the task text. The setup and control fields can also be wrapped with dedicated special tokens, while the robot state is included as an optional state clause when discrete state tokens are available. Representative formatted prompt prefixes are:

```
Action:
<|im_start|>user
The task is to put the red mug on the tray. The setup is <setup_start>bimanual
    yam robotic arms in molmoact2<setup_end>. The current state of the robot is <
    state_start><state_14><state_87>...<state_203><state_end>. The expected
    control mode is <control_start>absolute joint pose<control_end>. Given these,
     what action should the robot take to complete the task?
<|im_end|>
<|im_start|>assistant
<action_output>

Depth:
<|im_start|>user
The task is to open the middle drawer. The setup is <setup_start>single franka
    robotic arm in libero<setup_end>. The expected control mode is <control_start
    >delta end-effector pose<control_end>. Given these, what is the depth map of
    the main image?
<|im_end|>
<|im_start|>assistant
<depth_output>

Depth then action:
<|im_start|>user
The task is to place the bowl next to the plate. The setup is <setup_start>single
     franka robotic arm in libero<setup_end>. The current state of the robot is <
    state_start><state_31><state_52>...<state_118><state_end>. The expected
    control mode is <control_start>delta end-effector pose<control_end>. Given
    these, first predict the depth map of the main image and then predict the
    action the robot should take to complete the task?
<|im_end|>
<|im_start|>assistant
<depth_output><action_output>
```

Training hyperparameters. We summarize the stage-level training hyperparameters in Table 16. The columns correspond to the three training stages used throughout the paper: pre-training, post-training, and embodiment-specific fine-tuning.

B.2 MolmoAct2-FAST Tokenizer

The MolmoAct2-FAST Tokenizer architecture is an open-data implementation based on the FAST framework released by Physical Intelligence (Pertsch et al., 2025). While we utilize the core logic of frequency-domain compression (DCT-BPE), MolmoAct2-FAST Tokenizer is distinguished by its use of a fully transparent training mix addressing the data opacity of prior open-weight releases.

Data Format We adopt a standardized 32-dimensional vector format to accommodate various robot morphologies. The vector is structured as follows: Single-Arm: $[A_1, \ldots, A_n, G_1, 0, \ldots, 0]$ where A represents n arm joints and G_1 is the gripper state. Bimanual: $[A_{L1}, \ldots, A_{Ln}, G_L, A_{R1}, \ldots, A_{Rn}, G_R, 0, \ldots, 0]$ representing Left and Right arms respectively.

Multimodal Representation : MolmoAct2-FAST Tokenizer does not require a unified coordinate frame (e.g., converting all data to Delta End-Effector). Instead, the tokenizer is trained on a heterogeneous mix of "dialects," including Absolute Joint positions and Delta End-Effector velocities. During training, the VLM backbone learns to associate these disparate action representations with the specific embodiment mentioned in the task prompt or visual context.

B.3 GPU Cluster

MolmoAct2 was trained on Jupiter, an Ai2 GPU cluster in Austin, Texas. MolmoAct2 workloads were scheduled using Beaker (Guerquin, 2022), a custom workload management system. Jupiter comprises 128

Hyperparameter	Pre-train	Post-train	Fine-tune
Warm-up ViT	200	200	200
Warm-up Conn.	200	200	200
Warm-up LLM	200	200	200
Warm-up AE	200	200	200
LR ViT	5×10^{-6}	5×10^{-6}	5×10^{-6}
LR Conn.	5×10^{-6}	5×10^{-6}	5×10^{-6}
LR LLM	1×10^{-5}	1×10^{-5}	1×10^{-5}
LR AE	5×10^{-5}	5×10^{-5}	5×10^{-5}
Cosine Decay	10%	10%	10%
Eps.	10^{-6}	10^{-6}	10^{-6}
Betas	0.9/0.95	0.9/0.95	0.9/0.95
Steps	200k	100k	100k
Global Batch Size	128	128	64 or 128 (BimanualYAM)
Sequence Length	4200	2100 (Robot) and 4200 (Multimodal)	2100
GPUs (H100s)	64	64	32 or 64 (BimanualYAM)
Time (Hours)	90	36	36
GPU Hours	5760	2304	1152 or 2304 (BimanualYAM)

Table 16 MolmoAct2 training hyperparameters. We summarize the stage-level hyperparameters used for pre-training, post-training, and embodiment-specific fine-tuning. We only show the fine-tuning hyperparameters for the training on major datasets.

GPU nodes and is operated by Cirrascale Cloud Services[1].

Compute Jupiter provides 1,024 NVIDIA H100 GPUs (80GB HBM3, 700W) across 128 servers. Each server has 2,×,Intel Xeon Platinum8468 CPUs, 2TB DDR5 system memory, and 18 TB local NVMe storage.

Storage The servers are connected over an 800Gbps local network to a WEKA high-performance storage cluster[2]. The storage system provides 1PB of NVMe SSD across 11 storage servers and 5PB of HDD across 12 hosts. Each Jupiter server has two bonded 25Gbps Mellanox Ethernet NICs (50Gbps per host). In benchmarks, we achieved 761Gbps aggregate read/write throughput using 64 client machines.

Interconnect Cross-node GPU communication uses RDMA over InfiniBand on a two-tier Rail-Optimized, balanced, full-bisection network (Wang et al., 2023). Each server is equipped with eight 400Gbps InfiniBand adapters (3.2Tbps peak per host), supporting concurrent distributed jobs without topological scheduling.

Cooling The servers are racked in *Dynamic Density Cabinets*[3]. Each cabinet houses five servers with dedicated cooling and power. Air circulates in a closed loop through an overhead plenum where it is cooled via heat transfer to water, enabling a datacenter PUE of 1.2. Under heavy utilization, H100 temperatures peak around 75°C, with typical averages between 60°C and 65°C.

[1] cirrascale.com
[2] weka.io
[3] cirrascale.com/products-and-services/cabinet-technologies

C Evaluation Details

C.1 RoboEval

Training. We fine-tune the MolmoAct2-Post checkpoint on the RoboEval dataset using 8 GPUs with a global batch size of 64 for 10,000 steps. All eight tasks are trained jointly in a multi-task setup, with demonstrations from every task and variation pooled into a single training distribution.

Evaluation. We evaluate MolmoAct2-Post on the eight bimanual manipulation tasks introduced in RoboEval—Cube Handover, Lift Pot, Lift Tray, Pack Box, Pick Single Book From Table, Rotate Valve, Stack Single Book Shelf, and Stack Two Blocks—each instantiated with 3–5 structured variations that systematically perturb the initial scene. Following the benchmark's protocol, every task is run under a Static configuration as well as Position (Pos), Orientation (Rot), and combined Position+Rotation (PR) perturbations of the manipulated objects, with additional task-specific variants where applicable (e.g., a Vertical configuration for Cube Handover, a Drag setup for Lift Tray that initializes the tray outside the bimanual workspace, and an obstacle-occluded valve in Rotate Valve). For each (task, variation) pair, we roll out the policy from initial states sampled from the task's distribution ρ_0 and score each rollout against the task's geometric success set $\mathcal{S}_{\text{success}}$, reporting the mean success rate together with its standard error across rollouts. Beyond binary success, we log the full set of RoboEval diagnostic metrics during inference: trajectory-based metrics (joint and Cartesian path length, and joint and Cartesian jerk) to characterize motion efficiency and smoothness; spatial metrics (self-collisions, environment collisions, and object slip counts) to capture contact stability and execution safety; coordination metrics (vertical end-effector height discrepancy Δz and inter-arm velocity divergence Δv) to assess bimanual alignment and synchronization; and stagewise task-progression flags that mark completion of each semantically grounded subphase (e.g., grasp, lift, transfer, place). This combination lets us report not only whether MolmoAct2-Post completes each task, but also where in the task pipeline it tends to fail and how cleanly it executes when it succeeds.

C.2 Real-world Zero-shot DROID

Implementation. We adopt the DROID setup (Khazatsky et al., 2024) in a kitchen environment, and design five tasks, each with three spatial variants, for evaluation. Camera positions are held fixed across all three models.

Figure 9 shows sample trajectories of MolmoAct2 from 5 different tasks. Table 18 presents the evaluation results for MolmoAct2, π_0, and MolmoBot in the DROID setup.

C.3 Real-world Bimanual YAM

Implementation. We trained MolmoAct2-Post as eight separate single-task policies; to ensure a fair comparison, we applied the identical training protocol to each of the four baseline policies. All policies were trained to convergence, and the resulting checkpoints were used for evaluation. Evaluation and setup were carried out by Cortex AI: each task was collected under three spatial variants and evaluated under the same three variants.

C.4 Real-world SO-100/101 zero-shot

Implementation. We conduct zero-shot evaluation on the SO-100 platform using a fixed, pre-initialized camera viewpoint, and compare against DePi and SmolVLA. Episodes are graded under a partial-credit scoring scheme, and no additional training of the policies is performed.

Table 17 Summary of performance metrics across all methods on the combined variation. Bold values indicate the best result for each metric within the task. Values are reported as mean ± 95% confidence interval. The final row reports human demonstration statistics (computed from successful teleoperated demos only), which provide a reference band for efficient execution.

Method	Success ↑	TP ↑	BAVD ↓	BGVD ↓	CPL ↓	CT ↓	ECC ↓	JPL ↓	MCJ ↓	MJJ ↓	OPL ↓	SCC ↓	SC ↓	TL ↓
Cube Handover														
Diffusion	0.06 ± 0.10	0.27 ± 0.07	0.32 ± 0.03	0.13 ± 0.01	5.07 ± 0.35	4.80 ± 0.28	2.94 ± 0.49	26.14 ± 2.02	24.73 ± 0.90	133.00 ± 4.88	19.61 ± 1.59	2.46 ± 0.57	0.22 ± 0.14	939.76 ± 52.04
GR00T	0.00 ± 0.07	0.01 ± 0.01	0.50 ± 0.01	0.10 ± 0.01	4.69 ± 0.26	9.52 ± 0.26	2.90 ± 0.49	33.15 ± 1.11	42.62 ± 1.49	235.44 ± 4.45	23.34 ± 0.97	25.88 ± 1.91	**0.00** ± 0.07	976.10 ± 23.48
xVLA	0.00 ± 0.07	0.04 ± 0.03	0.56 ± 0.04	0.15 ± 0.02	5.51 ± 0.68	8.50 ± 3.26	3.40 ± 0.59	31.99 ± 3.01	31.00 ± 3.20	146.84 ± 13.51	23.25 ± 2.47	13.68 ± 2.53	0.04 ± 0.09	962.24 ± 42.87
Pi_0.5	**0.40** ± 0.14	**0.65** ± 0.10	0.34 ± 0.03	0.05 ± 0.01	2.22 ± 0.44	**3.66** ± 0.68	**1.32** ± 0.47	11.22 ± 2.31	18.14 ± 1.78	88.53 ± 7.98	8.00 ± 1.68	2.30 ± 0.99	0.26 ± 0.13	**620.58** ± 120.85
MolmoAct	0.36 ± 0.14	0.58 ± 0.10	0.29 ± 0.03	**0.04** ± 0.00	2.04 ± 0.31	4.21 ± 0.66	1.40 ± 0.44	10.78 ± 1.73	18.43 ± 1.91	90.54 ± 7.73	7.99 ± 1.37	**1.10** ± 0.43	0.26 ± 0.17	695.86 ± 111.27
Human	1.00 ± 0.00	1.00 ± 0.00	0.33 ± 0.03	0.03 ± 0.00	0.53 ± 0.03	0.51 ± 0.07	0.10 ± 0.16	2.26 ± 0.18	28.94 ± 3.07	120.52 ± 11.54	1.38 ± 0.18	0.17 ± 0.17	0.03 ± 0.13	103.53 ± 14.28
Lift Pot														
Diffusion	0.02 ± 0.08	0.07 ± 0.03	0.36 ± 0.05	0.18 ± 0.03	4.58 ± 0.56	8.87 ± 0.40	0.20 ± 0.18	24.46 ± 2.63	27.14 ± 3.77	142.55 ± 18.69	16.55 ± 1.88	10.64 ± 2.02	**0.12** ± 0.12	978.06 ± 32.64
GR00T	0.06 ± 0.10	0.30 ± 0.05	0.51 ± 0.02	0.09 ± 0.02	2.13 ± 0.30	7.43 ± 1.22	4.54 ± 0.76	21.44 ± 2.66	28.17 ± 1.91	231.37 ± 6.96	13.14 ± 1.64	5.88 ± 2.26	1.34 ± 0.53	628.14 ± 92.83
xVLA	0.00 ± 0.07	0.21 ± 0.05	0.67 ± 0.06	0.22 ± 0.04	5.31 ± 0.84	8.29 ± 3.48	2.26 ± 0.69	30.12 ± 3.74	36.35 ± 3.80	173.17 ± 16.93	21.79 ± 2.89	4.96 ± 1.66	0.32 ± 0.20	747.02 ± 90.35
Pi_0.5	**0.52** ± 0.13	0.41 ± 0.08	0.34 ± 0.03	0.07 ± 0.02	2.73 ± 0.62	3.86 ± 0.78	0.06 ± 0.10	14.23 ± 3.13	27.71 ± 2.54	139.02 ± 12.27	9.53 ± 2.06	0.80 ± 0.67	0.28 ± 0.15	550.78 ± 117.31
MolmoAct	0.32 ± 0.14	0.47 ± 0.07	0.28 ± 0.04	**0.05** ± 0.01	2.21 ± 0.42	4.67 ± 0.83	0.44 ± 0.28	11.36 ± 1.87	26.56 ± 4.39	129.13 ± 20.69	7.86 ± 1.25	**0.48** ± 0.38	0.52 ± 0.28	646.72 ± 116.82
Human	1.00 ± 0.00	1.00 ± 0.00	0.49 ± 0.05	0.03 ± 0.01	0.61 ± 0.07	0.45 ± 0.06	0.00 ± 0.14	3.33 ± 0.34	62.59 ± 4.70	287.53 ± 20.08	2.40 ± 0.21	0.00 ± 0.14	0.00 ± 0.14	71.04 ± 9.68
Lift Tray														
Diffusion	0.00 ± 0.07	0.47 ± 0.06	0.47 ± 0.04	0.21 ± 0.04	5.01 ± 0.52	9.85 ± 0.28	5.28 ± 1.29	29.97 ± 2.22	27.50 ± 3.48	147.21 ± 13.54	21.38 ± 1.78	6.70 ± 2.20	0.48 ± 0.20	1000.00 ± 0.00
GR00T	0.00 ± 0.07	0.46 ± 0.05	0.51 ± 0.02	0.12 ± 0.02	3.93 ± 0.30	13.28 ± 0.25	6.62 ± 1.25	32.10 ± 1.15	34.08 ± 1.98	228.16 ± 6.87	20.66 ± 0.93	17.06 ± 2.25	0.50 ± 0.23	1000.00 ± 0.00
xVLA	0.00 ± 0.07	0.35 ± 0.04	0.63 ± 0.06	0.24 ± 0.04	7.01 ± 0.92	10.49 ± 0.43	4.88 ± 0.94	37.46 ± 3.72	32.82 ± 4.39	151.68 ± 16.81	28.46 ± 3.09	10.46 ± 2.68	**0.16** ± 0.12	1000.00 ± 0.00
Pi_0.5	0.32 ± 0.14	0.75 ± 0.07	0.52 ± 0.05	0.14 ± 0.03	4.83 ± 0.84	5.87 ± 0.94	2.62 ± 0.99	29.67 ± 5.06	31.27 ± 2.31	181.95 ± 11.42	21.82 ± 3.78	5.22 ± 2.07	0.92 ± 0.22	719.16 ± 115.13
MolmoAct	**0.46** ± 0.14	**0.91** ± 0.05	0.38 ± 0.03	0.07 ± 0.02	2.18 ± 0.46	**5.04** ± 1.06	**0.98** ± 0.57	14.06 ± 2.95	21.12 ± 2.41	127.85 ± 13.57	9.58 ± 2.04	**1.98** ± 1.17	0.90 ± 0.35	**589.02** ± 125.85
Human	1.00 ± 0.00	1.00 ± 0.00	0.39 ± 0.04	0.03 ± 0.00	0.52 ± 0.03	0.56 ± 0.03	0.00 ± 0.06	2.87 ± 0.13	26.77 ± 2.93	153.00 ± 16.82	2.04 ± 0.08	0.00 ± 0.06	0.00 ± 0.06	83.62 ± 3.96
Pack Box														
Diffusion	0.04 ± 0.09	0.27 ± 0.06	0.43 ± 0.04	0.11 ± 0.02	7.19 ± 0.41	5.42 ± 0.31	0.58 ± 0.34	36.50 ± 2.51	29.98 ± 2.07	154.68 ± 11.16	25.85 ± 1.81	9.82 ± 3.05	0.44 ± 0.16	962.98 ± 44.27
GR00T	0.00 ± 0.07	0.07 ± 0.03	0.50 ± 0.02	0.24 ± 0.04	4.77 ± 0.36	10.75 ± 0.20	1.04 ± 0.47	29.29 ± 1.37	40.49 ± 3.08	207.72 ± 9.51	19.50 ± 1.10	13.30 ± 2.92	0.30 ± 0.19	1000.00 ± 0.00
xVLA	0.02 ± 0.08	0.12 ± 0.05	0.79 ± 0.05	0.25 ± 0.03	9.76 ± 0.88	7.29 ± 0.35	1.30 ± 0.53	47.57 ± 3.10	46.60 ± 3.63	200.34 ± 12.90	34.53 ± 2.36	9.24 ± 3.08	**0.14** ± 0.11	988.12 ± 23.28
Pi_0.5	0.54 ± 0.14	0.55 ± 0.06	0.73 ± 0.07	0.10 ± 0.01	4.60 ± 0.89	3.70 ± 0.72	0.90 ± 0.41	22.61 ± 4.56	27.59 ± 1.39	130.35 ± 7.35	15.84 ± 3.25	1.24 ± 1.19	0.44 ± 0.20	566.54 ± 113.98
MolmoAct	**0.62** ± 0.14	**0.57** ± 0.05	0.58 ± 0.06	0.08 ± 0.02	3.22 ± 0.71	**3.14** ± 0.78	**0.44** ± 0.31	15.26 ± 3.72	24.28 ± 1.58	112.64 ± 7.83	10.10 ± 2.59	1.12 ± 0.72	0.48 ± 0.25	**470.42** ± 115.06
Human	1.00 ± 0.00	1.00 ± 0.00	0.51 ± 0.04	0.05 ± 0.00	1.30 ± 0.06	0.68 ± 0.04	0.84 ± 0.37	5.38 ± 0.34	28.24 ± 1.50	134.77 ± 6.73	3.35 ± 0.21	0.18 ± 0.16	0.03 ± 0.07	129.66 ± 8.00
Pick Single Book From Table														
Diffusion	0.02 ± 0.08	0.15 ± 0.09	0.59 ± 0.04	0.24 ± 0.03	4.42 ± 0.49	11.20 ± 0.83	5.52 ± 1.27	28.17 ± 2.59	21.24 ± 1.98	136.43 ± 10.11	17.38 ± 1.69	4.26 ± 1.50	0.04 ± 0.09	926.04 ± 61.77
GR00T	0.00 ± 0.07	0.05 ± 0.05	0.56 ± 0.03	0.26 ± 0.03	2.96 ± 0.36	15.47 ± 1.63	9.12 ± 1.30	28.12 ± 2.69	28.97 ± 1.90	237.89 ± 7.47	18.37 ± 1.90	10.90 ± 2.44	0.04 ± 0.09	821.42 ± 83.40
xVLA	0.00 ± 0.07	0.02 ± 0.08	0.72 ± 0.06	0.21 ± 0.03	5.10 ± 0.72	13.53 ± 0.81	5.58 ± 0.96	31.56 ± 2.94	22.35 ± 2.75	114.50 ± 11.31	22.67 ± 2.35	7.90 ± 1.75	**0.02** ± 0.08	949.08 ± 47.67
Pi_0.5	0.22 ± 0.13	0.44 ± 0.13	0.72 ± 0.07	0.20 ± 0.02	1.89 ± 0.47	6.03 ± 1.54	3.02 ± 0.88	11.37 ± 2.88	17.94 ± 1.51	91.87 ± 6.23	6.67 ± 1.76	0.28 ± 0.24	0.12 ± 0.11	**458.76** ± 120.93
MolmoAct	**0.30** ± 0.14	**0.48** ± 0.13	0.62 ± 0.08	**0.18** ± 0.01	1.96 ± 0.35	8.79 ± 1.68	2.70 ± 0.77	13.06 ± 2.38	15.55 ± 2.22	82.43 ± 8.31	7.10 ± 1.30	0.78 ± 0.59	**0.02** ± 0.08	641.82 ± 123.19
Human	1.00 ± 0.00	1.00 ± 0.00	0.87 ± 0.00	0.18 ± 0.00	0.76 ± 0.00	1.20 ± 0.00	0.00 ± 0.00	4.09 ± 0.00	30.32 ± 0.00	138.21 ± 0.00	1.90 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	118.00 ± 0.00
Rotate Valve														
Diffusion	0.04 ± 0.09	0.23 ± 0.07	0.44 ± 0.04	0.10 ± 0.02	2.10 ± 0.23	11.62 ± 0.52	10.18 ± 1.47	13.61 ± 1.18	5.88 ± 0.54	41.95 ± 3.08	9.76 ± 0.94	2.86 ± 1.06	**0.00** ± 0.07	977.16 ± 35.82
GR00T	0.00 ± 0.07	0.11 ± 0.04	0.63 ± 0.03	0.08 ± 0.01	2.69 ± 0.17	18.11 ± 0.63	23.70 ± 2.64	21.46 ± 0.77	14.40 ± 0.81	88.73 ± 2.50	15.09 ± 0.75	16.40 ± 2.50	**0.00** ± 0.07	1000.00 ± 0.00
xVLA	0.10 ± 0.11	0.15 ± 0.06	0.73 ± 0.05	0.13 ± 0.02	3.07 ± 0.29	11.74 ± 0.62	12.34 ± 2.19	19.40 ± 1.35	7.30 ± 0.55	38.85 ± 2.53	14.54 ± 1.06	2.40 ± 0.91	**0.00** ± 0.07	953.00 ± 44.60
Pi_0.5	**0.72** ± 0.14	0.88 ± 0.05	0.36 ± 0.02	0.03 ± 0.00	0.62 ± 0.14	8.65 ± 1.59	12.42 ± 2.36	4.67 ± 0.95	2.99 ± 0.22	19.81 ± 1.29	3.43 ± 0.69	**0.02** ± 0.08	**0.00** ± 0.07	527.64 ± 95.25
MolmoAct	**0.72** ± 0.14	**0.90** ± 0.05	**0.30** ± 0.02	**0.03** ± 0.00	**0.45** ± 0.10	**8.51** ± 1.96	**8.84** ± 1.28	**3.42** ± 0.73	**2.56** ± 0.20	**16.77** ± 1.05	**2.48** ± 0.51	0.14 ± 0.24	**0.00** ± 0.07	**481.94** ± 107.26
Human	1.00 ± 0.00	1.00 ± 0.00	0.37 ± 0.04	0.01 ± 0.00	0.20 ± 0.02	1.98 ± 0.22	6.38 ± 0.73	1.24 ± 0.11	3.60 ± 0.31	22.26 ± 2.06	1.04 ± 0.10	0.00 ± 0.19	0.00 ± 0.19	150.12 ± 16.61
Stack Single Book Shelf														
Diffusion	0.00 ± 0.07	0.01 ± 0.01	0.78 ± 0.06	0.24 ± 0.02	6.01 ± 0.63	11.95 ± 0.62	5.74 ± 0.83	35.11 ± 2.58	29.84 ± 3.97	166.93 ± 15.36	22.73 ± 1.84	3.54 ± 1.22	**0.00** ± 0.07	966.84 ± 45.49
GR00T	0.00 ± 0.07	0.04 ± 0.03	0.56 ± 0.02	0.29 ± 0.03	3.17 ± 0.41	16.17 ± 1.54	9.46 ± 1.34	28.67 ± 2.49	31.38 ± 2.57	236.75 ± 5.66	18.50 ± 1.71	11.38 ± 2.70	0.04 ± 0.09	842.26 ± 76.69
xVLA	0.00 ± 0.07	0.01 ± 0.01	0.78 ± 0.06	0.24 ± 0.04	5.53 ± 0.77	13.64 ± 0.82	7.28 ± 1.31	33.77 ± 3.27	29.08 ± 3.57	150.93 ± 15.57	24.37 ± 2.47	10.16 ± 2.45	0.02 ± 0.08	937.06 ± 49.54
Pi_0.5	0.12 ± 0.12	0.27 ± 0.10	0.79 ± 0.06	0.20 ± 0.01	2.04 ± 0.56	**5.96** ± 1.60	**2.16** ± 0.58	12.82 ± 3.59	19.90 ± 1.25	104.69 ± 5.90	7.39 ± 2.09	2.16 ± 1.45	0.16 ± 0.17	**414.04** ± 114.28
MolmoAct	**0.18** ± 0.13	**0.33** ± 0.11	**0.67** ± 0.07	**0.19** ± 0.02	**2.27** ± 0.40	8.76 ± 1.69	2.80 ± 0.86	**14.01** ± 2.59	**16.60** ± 1.33	**93.43** ± 6.99	**7.45** ± 1.41	**0.30** ± 0.31	0.10 ± 0.11	600.90 ± 117.81
Human	1.00 ± 0.00	1.00 ± 0.00	0.85 ± 0.00	0.20 ± 0.00	1.10 ± 0.00	2.12 ± 0.00	0.00 ± 0.00	6.29 ± 0.00	19.79 ± 0.00	100.20 ± 0.00	2.90 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	194.00 ± 0.00
Stack Two Blocks														
Diffusion	0.06 ± 0.10	0.13 ± 0.05	0.34 ± 0.03	0.08 ± 0.01	4.57 ± 0.23	7.27 ± 0.24	4.58 ± 0.47	24.32 ± 1.55	18.74 ± 0.83	106.54 ± 4.95	18.54 ± 1.23	2.04 ± 0.66	0.10 ± 0.11	978.98 ± 29.57
GR00T	0.04 ± 0.09	0.05 ± 0.03	0.50 ± 0.02	0.12 ± 0.02	4.70 ± 0.35	12.16 ± 0.63	2.56 ± 0.50	32.41 ± 1.85	41.87 ± 2.31	233.11 ± 6.53	22.17 ± 1.39	23.60 ± 2.49	**0.00** ± 0.07	965.66 ± 47.12
xVLA	0.00 ± 0.07	0.01 ± 0.02	0.54 ± 0.05	0.17 ± 0.03	4.34 ± 0.46	8.74 ± 0.36	3.22 ± 0.59	27.38 ± 2.32	22.73 ± 2.43	115.21 ± 11.41	19.44 ± 1.86	9.34 ± 2.24	**0.00** ± 0.07	974.08 ± 29.35
Pi_0.5	0.40 ± 0.14	0.44 ± 0.07	0.34 ± 0.02	0.05 ± 0.01	2.03 ± 0.31	5.87 ± 0.88	2.16 ± 0.59	10.44 ± 1.76	11.37 ± 0.75	57.57 ± 4.17	7.60 ± 1.29	1.94 ± 1.38	0.12 ± 0.12	710.78 ± 108.67
MolmoAct	**0.52** ± 0.13	**0.48** ± 0.07	**0.31** ± 0.04	**0.04** ± 0.00	**1.34** ± 0.29	**4.70** ± 1.05	**1.04** ± 0.36	**6.16** ± 1.35	**12.91** ± 1.46	**58.79** ± 5.30	**4.49** ± 1.02	**0.38** ± 0.63	0.04 ± 0.09	**545.88** ± 123.97
Human	1.00 ± 0.00	1.00 ± 0.00	0.40 ± 0.03	0.05 ± 0.00	0.42 ± 0.04	0.58 ± 0.06	0.24 ± 0.11	1.56 ± 0.17	23.24 ± 1.28	95.97 ± 5.04	1.11 ± 0.14	0.00 ± 0.05	0.00 ± 0.05	83.78 ± 9.01

Abbreviations: Success = Success Rate [%]; TP = Task Progression [%]; BAVD = Bimanual Arm Velocity Difference [m/s]; BGVD = Bimanual Gripper Vertical Difference [m]; CPL = Cartesian Path Length [m]; CT = Completion Time [s]; ECC = Environment Collision Count; JPL = Joint Path Length [rad]; MCJ = Mean Cartesian Jerk [m/s^3]; MJJ = Mean Joint Jerk [rad/s^3]; OPL = Orientation Path Length [rad]; SCC = Self-Collision Count; SC = Slip Count; TL = Trajectory Length [steps].

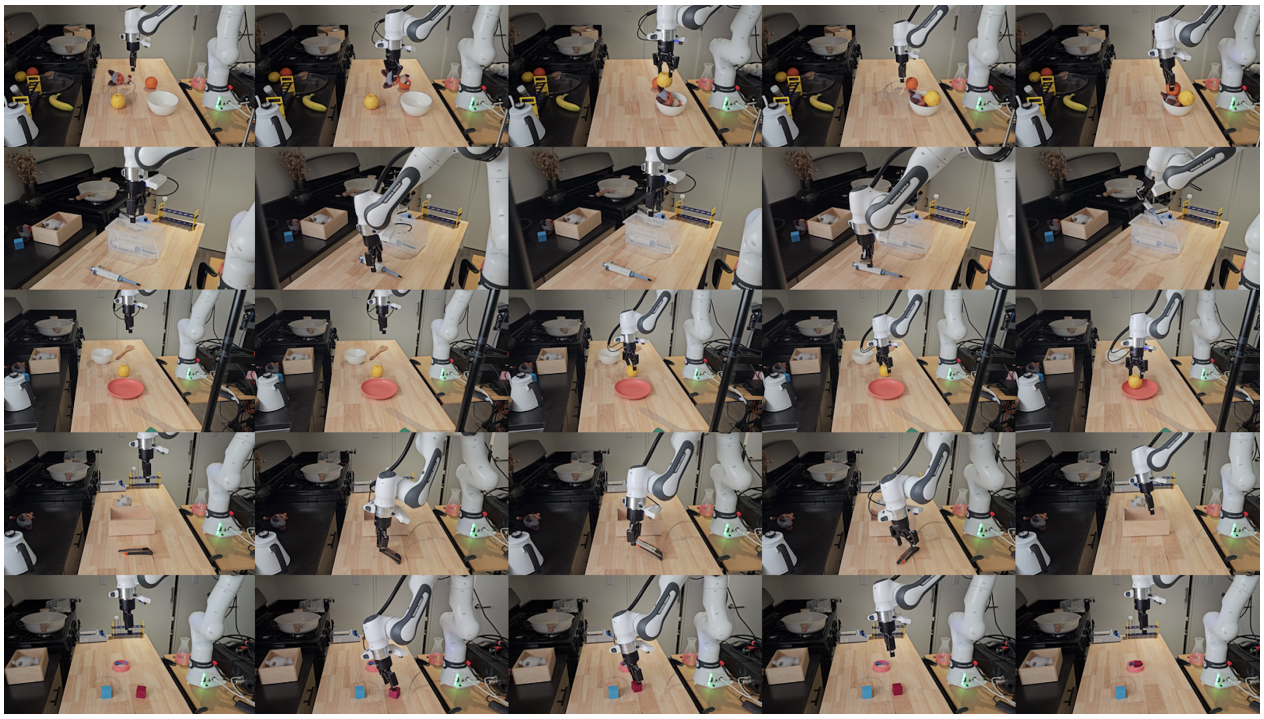

Figure 9 **Sample trajectories from the DROID evaluation of MolmoAct2.**

Table 18 Per-task and per-position success rates across policies. Each task uses 15 trajectories (5 per position).

Task	Policy	Pos. 1	Pos. 2	Pos. 3	Overall
Put the apple on the plate	MolmoAct2	5/5 (100%)	5/5 (100%)	5/5 (100%)	**15/15 (100.0%)**
	$\pi_0$5-DROID	5/5 (100%)	3/5 (60%)	2/5 (40%)	10/15 (66.7%)
	MolmoBot	5/5 (100%)	4/5 (80%)	4/5 (80%)	13/15 (86.7%)
Put the pipette in the tray	MolmoAct2	4/5 (80%)	4/5 (80%)	5/5 (100%)	**13/15 (86.7%)**
	$\pi_0$5-DROID	2/5 (40%)	3/5 (60%)	0/5 (0%)	5/15 (33.3%)
	MolmoBot	3/5 (60%)	2/5 (40%)	3/5 (60%)	8/15 (53.3%)
Put the red cube inside the tape roll	MolmoAct2	5/5 (100%)	4/5 (80%)	5/5 (100%)	**14/15 (93.3%)**
	$\pi_0$5-DROID	3/5 (60%)	5/5 (100%)	0/5 (0%)	8/15 (53.3%)
	MolmoBot	1/5 (20%)	0/5 (0%)	4/5 (80%)	5/15 (33.3%)
Put the knife in the box	MolmoAct2	5/5 (100%)	5/5 (100%)	4/5 (80%)	**14/15 (93.3%)**
	$\pi_0$5-DROID	3/5 (60%)	1/5 (20%)	0/5 (0%)	4/15 (26.7%)
	MolmoBot	1/5 (20%)	0/5 (0%)	5/5 (100%)	6/15 (40.0%)
Put the objects in the bowl[a]	MolmoAct2	73.2%	53.0%	59.8%	**62.0%**
	$\pi_0$5-DROID	39.6%	39.6%	59.4%	46.2%
	MolmoBot	33.0%	26.4%	26.4%	28.6%

[a] Partial-credit task scored on {0, 0.33, 0.66, 1.0} per trajectory; reported as mean success rate.

Figure 10 Sample trajectories from the Bimanual YAM evaluation of MolmoAct2.

Figure 11 Sample trajectories from the out-of-distribution Bimanual YAM evaluation of MolmoAct2.

Table 19 Overall success rate (%) across 8 manipulation tasks (50 episodes each). Best model per task is **bold**.

Task	$\pi_{0.5}$	Cosmos	X-VLA	OpenVLA	MolmoAct
Cup Stacking	62.0	**64.5**	5.5	60.5	54.0
Linearbot	11.6	9.2	5.2	24.4	**64.0**
Pegboard	2.0	0.0	3.0	4.5	**14.0**
Cup Storing	85.9	0.0	7.9	96.0	**100.0**
Candy Storing	26.0	0.0	3.8	26.5	**40.5**
Store Test Tube	17.0	11.0	0.0	42.0	**67.0**
Store Toys	26.0	7.0	2.5	2.0	**32.0**
Prepare Pipette	27.5	32.0	11.5	28.5	**33.0**
Average	32.2	15.5	4.9	35.5	**50.6**

Table 20 Zero-shot success rate (%) on SO-100 tasks with fixed initial camera position. Each task is graded with partial credit at three milestones; partial scores of 0.25, 0.5, and 1.0 are awarded for reaching the corresponding stage. 15 episodes per task. Best model per task is **bold**.

	Partial-Credit Milestones			Success Rate (%)		
Task	0.25	0.5	1.0	SmolVLA	MolmoAct v2	DePi$_0$
Place fork on plate	reach	pickup fork	place on plate	3.3	**70.0**	30.0
Stack blocks	reach	pickup block	stack blocks	5.0	**20.0**	6.7
Place tissues in basket	reach	pickup	place	0.0	**73.3**	20.0
Place pen on notebook	reach	pickup	place	3.3	**86.7**	80.0
Place block in box	reach	pickup	place	0.0	33.3	**90.0**
Average				2.3	**56.7**	45.3

Figure 12 Sample trajectories from the SO-100 evaluation of MolmoAct2.

D Data Details

D.1 MolmoAct2-SO100/101 Dataset Statistics

We use MolmoAct2-SO100/101 Dataset in the MolmoAct2 training mixture. We first collect 1,660 dataset entries; then we use TOPReward (Chen et al., 2026) to filter out 438 low-quality dataset entries, yielding 1,222 final datasets. All statistics below are computed only over this final filtered set. Table 21 presents the summary statistics for the SO-100/101 training partition. Table 22 presents the number of datasets by the type of SO-10x variants. Table 23 presents the distribution of different metadata such as FPS, camera configuration, and action/state dimensions. Figure 13 presents the distribution of action verbs before and after the language annotation pipeline described in Section 3.4. Figure 14 shows several sample trajectories from the dataset partition.

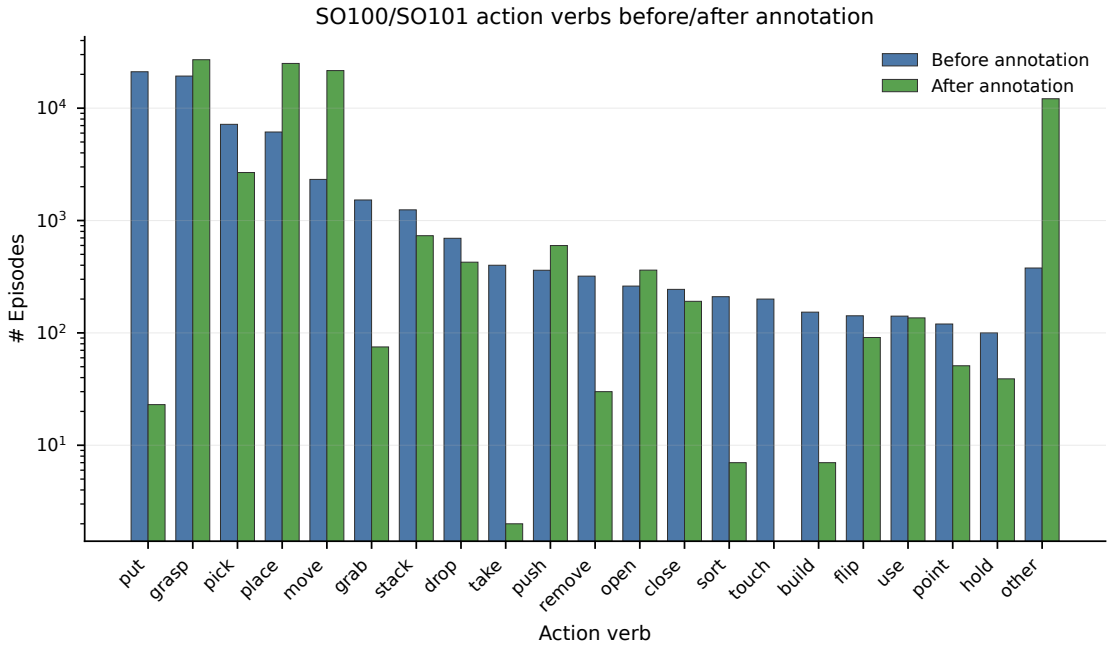

Figure 13 Action verb distribution before and after language annotation.

Table 21 Summary statistics for the SO-100/101 training subset. We report statistics for the final filtered MolmoAct2-SO100/101 Dataset partition.

Statistic	Value
# datasets	1,222
Contributors	377
Episodes	38,059
Frames	19.8M
Estimated duration	183.6 hours
Metadata task count	1,322
Instruction rows	38,059
Globally unique instructions	12,818
Unique-instruction ratio	33.7%
Datasets with annotated task metadata	1,222

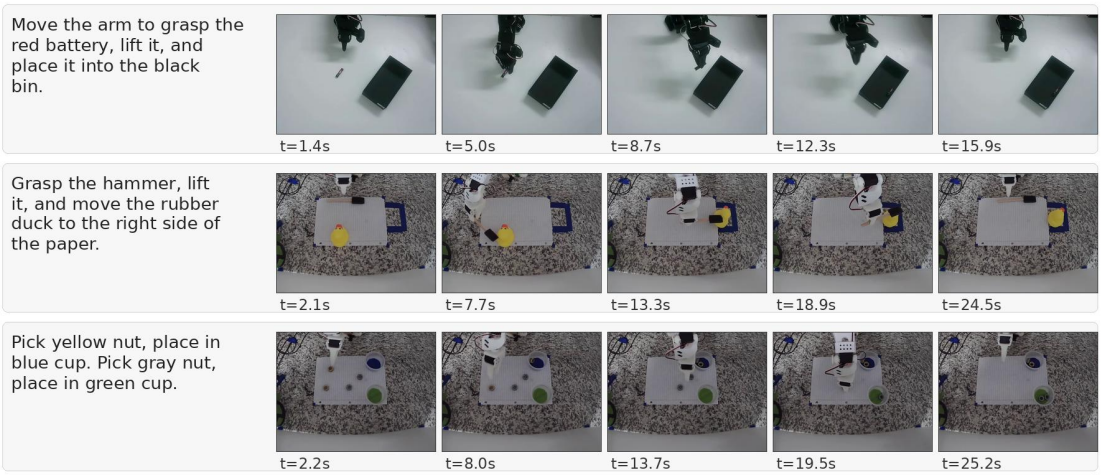

Figure 14 Sample trajectories from the SO10x dataset partition.

Table 22 SO-10x data partition breakdown by robot type.

Robot type	Datasets	Episodes	Frames	Hours
SO-100	921	31,101	15.18M	141.7
SO-101	299	6,898	4.57M	41.6
MOSS	2	60	15.6K	0.2

D.2 Language Annotation Pipeline Implementation Details

Running inference with temperature 0.1 and max_length set to 1024 tokens, we feed the following prompt into Qwen3.5-27B:

> You are an expert robot arm task instruction generator. You are provided with multiple demonstration videos, each showing the exact same task execution captured simultaneously from different camera angles. The task is to {original_instruction}. Based on these videos, generate a clear imperative instruction describing all of the actions performed by the robot arm. Carefully study the evolution of the state of the scene, to understand how the robot arm achieves the goal. Pay attention to the color and position of the objects the robot interacts with. Start directly with an action verb and make sure you do not miss any actions. Note that the robot arm is not always grasping an object and maybe pushing objects around instead. Verify any grasps by checking the gripper state, and only say the arm is grasping an object if this is shown in the wrist camera when the gripper is closed. Your prompts should be almost exactly {word_limit} words long.

word_limit is a number randomly sampled from a right tailed power distribution, with a minimum value of 5 and a maximum value of 25. In addition to the text prompt, for each of the cameras in the trajectory that are not frozen, we sample 12 frames that are each equally temporally spaced and pass the frames to the model.

Running this pipeline results in the changes to the number of unique instructions in the datasets shown in Table 24.

Table 23 Recording and feature metadata. Most demonstrations are recorded at 30 Hz with two camera streams; all final filtered datasets use six-dimensional action and state vectors.

Metadata field	Value	Datasets	Episodes	Frames
FPS	30	1,114	36,828	18.96M
FPS	20	78	527	261.4K
FPS	25	18	423	256.7K
FPS	60	8	208	231.6K
Camera count	1	230	4,159	2.26M
Camera count	2	835	24,151	13.05M
Camera count	3	151	9,235	4.11M
Camera count	4	6	514	342.1K
Action dimension	6	1,222	38,059	19.76M
State dimension	6	1,222	38,059	19.76M

Table 24 Dataset Language Annotation Diversity. Running the language annotation pipeline increases the number of unique instructions for all of the robotics datasets in MolmoAct2's training mix.

Dataset	Original Instructions	Relabeled Instructions
MolmoAct2-BimanualYAM Dataset	34 (0.1%)	11448 (35%)
MolmoAct2-DROID Dataset	49623 (67%)	55905 (75%)
MolmoAct2-SO100/101 Dataset	707 (1.5%)	16205 (34%)
MolmoAct Dataset	134 (1%)	2959 (28%)
BC-Z	104 (0.3%)	9981 (25%)
Fractal	598 (0.7%)	16729 (19%)
Bridge	19973 (37%)	35096 (66%)
Total	**71121 (22%)**	**146485 (46%)**

Table 25 Past Tabletop Bimanual Manipulation Datasets. MolmoAct2-BimanualYAM Dataset is the largest open-source tabletop bimanual robotics dataset. In this table, we compare against other open and closed-source tabletop bimanual datasets from the literature.

Dataset Name	Embodiment	Number of Trajectories	Open / Closed?
MolmoAct2-BimanualYAM Dataset (Ours)	Bimanual YAM	34,500	Open
AIST Bimanual Manipulation Dataset (Motoda et al., 2025)	ALOHA	10,000	Open
BiPlay Dataset (Dasari et al., 2025)	ALOHA	7,000	Open
RDT-1B Finetuning Dataset (Liu et al., 2024)	ALOHA	6,000	Open
ALOHA Unleashed Training Data (Zhao et al., 2024)	ALOHA	26,000	Closed
SARM RA-BC Dataset (Chen et al., 2025)	Bimanual YAM	200 Hours	Closed

E Limitations and Potential Solutions

While MolmoAct2 is all quite capable as a general-purpose action reasoning model, it is not without limitations. In the following sections, we discuss some of these limitations and potential solutions.

Action chunks and the absence of real-time re-chunking. MolmoAct2 predicts fixed-horizon action chunks (30 steps at 30 Hz for YAM/SO-100/101, 15 at 15 Hz for DROID, 10 at 10 Hz for LIBERO) and executes them open-loop before re-querying the policy. This has two consequences worth acknowledging. First, motion smoothness across chunk boundaries is not enforced anywhere in the pipeline: the flow-matching expert is trained to denoise each chunk independently, with no continuity loss tying the end of chunk t to the start of chunk $t + 1$. In practice this can produce visible velocity or acceleration discontinuities at the seams, particularly when the VLM context shifts between queries (new observation, slightly different attention state). Second, because the policy commits to a full chunk before observing its consequences, it cannot react within-chunk to perturbations, contact events, or its own tracking error — the 55.79 Hz number in Sec 6.8 is amortized chunk throughput, not closed-loop reactivity.

Embodiment-specific zero-shot deployment. The zero-shot deployment story for MolmoAct2 is currently tied to the three embodiments for which we have large-scale training data: MolmoAct2-BimanualYAM, MolmoAct2-SO100/101, and MolmoAct2-DROID can each be deployed out-of-the-box on their respective robot setups (bimanual YAM, SO-100/101, and DROID Franka) Sec. 6.2. This is a direct consequence of our data strategy — concentrating collection and curation effort on these three platforms to span the low-to-medium cost range — but it also bounds what zero-shot means here. MolmoAct2 is not a universal controller that transfers without adaptation to arbitrary new embodiments: deployment on robots outside this set (e.g., other bimanual platforms, dexterous hands, mobile manipulators with different kinematics, or humanoids) requires effective fine-tuning on target-embodiment demonstrations, as shown in our fine-tuning studies in Sec 6.3. Extending the zero-shot envelope to a broader range of embodiments is a matter of scaling data collection along the same axes we have begun here, and we view the released datasets and training recipe as a foundation for that effort rather than a finished product.