Actions as Language: Fine-Tuning VLMs into VLAs Without Catastrophic Forgetting

Asher J. Hancock[1†], **Xindi Wu**[1*], **Lihan Zha**[1†], **Olga Russakovsky**[1*], **Anirudha Majumdar**[1†]

[1]Princeton University

[†]Department of Mechanical and Aerospace Engineering, [*]Department of Computer Science

Fine-tuning vision-language models (VLMs) on robot teleoperation data to create vision-language-action (VLA) models is a promising paradigm for training generalist policies, but it suffers from a fundamental tradeoff: learning to produce actions often diminishes the VLM's foundational reasoning and multimodal understanding, hindering generalization to novel scenarios, instruction following, and semantic understanding. We argue that this catastrophic forgetting is due to a distribution mismatch between the VLM's internet-scale pretraining corpus and the robotics fine-tuning data. Inspired by this observation, we introduce VLM2VLA: a VLA training paradigm that first resolves this mismatch at the data level by *representing low-level actions with natural language*. This alignment makes it possible to train VLAs *solely with Low-Rank Adaptation (LoRA)*, thereby minimally modifying the VLM backbone and averting catastrophic forgetting. As a result, the VLM can be fine-tuned on robot teleoperation data without fundamentally altering the underlying architecture and without expensive co-training on internet-scale VLM datasets. Through extensive Visual Question Answering (VQA) studies and over 800 real-world robotics experiments, we demonstrate that VLM2VLA preserves the VLM's core capabilities, enabling zero-shot generalization to novel tasks that require open-world semantic reasoning and multilingual instruction following. Website with additional information, videos, and code: https://vlm2vla.github.io/.

Keywords: VLAs, Embodied Reasoning, Action Representation

IRoM Lab

1 Introduction

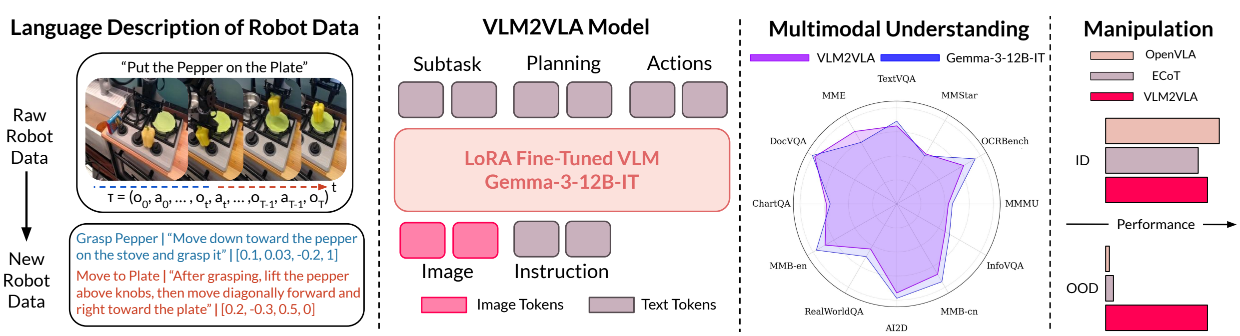

Figure 1 We present VLM2VLA, a data pipeline and training methodology for fine-tuning VLMs into VLAs while preserving their foundational perceptual and reasoning capabilities. Our policy retains its pretraining knowledge, enabling strong VQA performance and superior generalization in real robotic manipulation tasks.

The pursuit of generalist robot policies capable of understanding and executing human commands has been significantly advanced by the integration of vision-language models (VLMs) [1–3] throughout the autonomy stack. Trained on internet-scale datasets of image-text pairs, these models have acquired sophisticated capabilities in perception, semantic understanding, and commonsense reasoning. To endow robots with similar capabilities, the prevailing paradigm involves fine-tuning pretrained VLMs on robot demonstration data, transforming them into vision-language-action models (VLAs) that map from natural language commands and visual observations to robot actions. This approach has yielded impressive results across a wide range of robotic manipulation tasks [4–11].

However, the standard methodology of fundamentally modifying the VLM's architecture, tokenization vocabulary, or a combination thereof, coupled with full parameter fine-tuning on robot imitation learning data, introduces a crucial yet often overlooked trade-off. In adapting the VLM for robotic control, we risk overfitting to the robot fine-tuning data, thereby overwriting the general-purpose world knowledge acquired during pretraining (see Fig. 2). The consequences of this trade-off are far-reaching: current VLAs often exhibit a diminished ability to generalize to novel objects, handle linguistic variations, be robust to distractions, or reason about concepts outside the narrow scope of their robotic training data [4, 9, 12, 13].

Preserving the VLM's foundational world knowledge during VLA fine-tuning is essential for creating truly generalist robot policies; consequently, numerous techniques have been developed to address this challenge. The most common approach is to co-train with non-robotic data. This training regimen regularizes the VLM against overfitting to robot datasets, thereby mitigating the loss of its foundational capabilities [6, 9, 14]. While these methods can mitigate knowledge loss, co-training with VLM-scale datasets is inherently expensive and requires a carefully constructed dataset mixture for optimal performance.

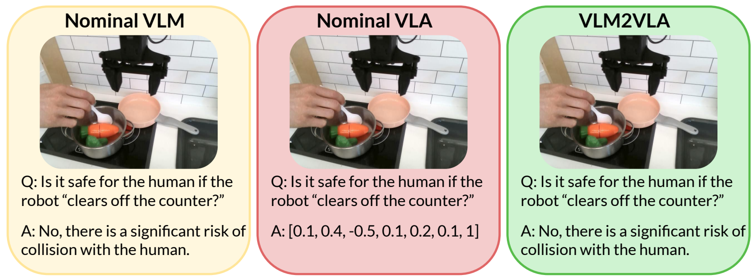

Figure 2 Traditional VLA training procedures often overfit to the robot training data, sacrificing their original reasoning capabilities for low-level action prediction (center). In contrast, VLM2VLA (right) preserves the world understanding of the nominal VLM (left), allowing the model to reason about potential safety risks instead of just motor commands.

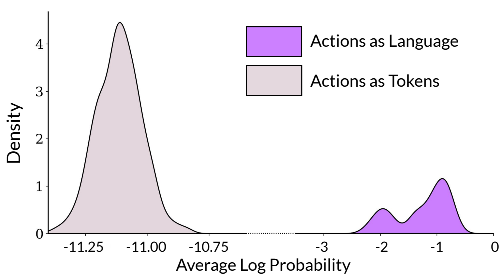

Figure 3 Distribution of action probabilities under Gemma-3-12B-IT *before* fine-tuning on robot teleoperation data. The model assigns significantly higher log-probabilities to actions represented as language compared to those defined by explicit tokenization modifications, e.g., least likely token assignment.

This paper aims to preserve the world knowledge of the VLM while adapting it for robotic control *without* co-training. We address this issue by resolving the *distribution mismatch between the low-level action spaces needed for robotic control and the image-text distributions of the VLM's pretraining corpus.* This mismatch often compels researchers to use full parameter fine-tuning when training VLAs, which contributes to catastrophic forgetting by overfitting to the robot teleoperation data.

Our key insight is that while parameter-efficient methods like Low-Rank Adaptation (LoRA) [15] can avert catastrophic forgetting, their effectiveness relies on the fine-tuning data being sufficiently close to the model's pretrained representations. We therefore propose resolving this representational mismatch *at the data level.* Our data-centric approach *re-represents robot actions as natural language descriptions, thereby aligning the VLA fine-tuning data directly with the VLM's pretrained representation space.* This alignment enables LoRA to effectively adapt the VLM for robotic control without significantly perturbing its pretrained weights. Fig. 3 illustrates this idea, showing our language-based actions are assigned significantly higher probabilities by the VLM backbone than actions mapped to arbitrary tokens, a common strategy in state-of-the-art VLAs [4, 8–11]. Our method is model agnostic and simple to implement, obviating the need for sophisticated architectures, complex co-training schemes, or multi-stage training procedures to achieve robust knowledge retention and superior generalization capabilities.

Statement of Contributions. We present VLM2VLA, a data pipeline and training methodology for fine-tuning VLMs into VLAs while preserving their foundational perceptual and reasoning capabilities. Our core contributions are as follows: **1) Representing actions as language**: We propose translating low-level robotic imitation data into text, thereby aligning the VLA fine-tuning data with the VLM's pretraining distribution to mitigate catastrophic forgetting. **2) A data re-labeling and training pipeline for knowledge retention**: Building on our action representation, we present a scalable methodology for re-labeling robot teleoperation datasets for fine-tuning a VLM into a VLA through LoRA. **3) Empirical validation of action and reasoning capabilities**: We provide extensive empirical validation showing our VLA preserves a suite of crucial capabilities that are often lost in other state-of-the-art models. Specifically, our method's efficacy is demonstrated through extensive real-world evaluation (over 800 robotic experiments), showing generalization to novel tasks with objects and language instructions unseen during training. Moreover, our policy averts catastrophic forgetting, retaining over 85% of the base model's performance across challenging VQA benchmarks.

2 Related Work

2.1 Vision-Language Action (VLA) Models

VLA models, which fine-tune pretrained VLM backbones for robotic control, have become the dominant paradigm in training generalist robotic policies [4, 5, 8–11, 14, 16–23]. The central promise of this approach is knowledge transfer: one can translate the rich semantic knowledge of the VLM, learned from internet-scale pretraining, directly to action prediction. One challenge in adapting transformer-based VLMs for action prediction is bridging the gap between their discrete, token-based nature and the continuous, vector-valued representations utilized in robotic control. Consequently, many state-of-the-art VLAs diverge architecturally in how they address the problem of action generation.

2.1.1 Action Representation in VLAs

The first dominant strategy is discretization, where continuous action vectors are explicitly mapped to a finite set of tokens [11, 24], typically the VLM's least likely tokens [4, 8–10]. This approach casts robot control as a standard next-token prediction problem, allowing the VLA to generate outputs autoregressively.

The second paradigmatic strategy augments the VLM with a separate, lightweight action head (e.g., using diffusion or flow-matching) to generate continuous actions directly [5, 6, 13, 21, 25, 26]. While this design choice enables faster action prediction [24], it introduces new, randomly initialized parameters to the VLM during VLA fine-tuning which can corrupt the original pretrained representations [13, 26, 27].

Our work introduces a third approach: *representing actions as natural language descriptions all within the VLM's existing vocabulary*. For instance, the command 'move forward by 4.2 centimeters' is treated as a standard text string, allowing our method to bootstrap the VLM's intrinsic understanding of numerical magnitude for grounding in physical space. We provide extensive experimental evidence in Section 4.2 that this 'actions as language' representation is more effective for policy learning via LoRA fine-tuning than strategies based on least likely token assignment. Unlike the concurrent work of [28], which use full parameter fine-tuning and co-training, we use only LoRA and our action representation to preserve the VLM's foundational reasoning capabilities.

2.1.2 Reasoning Models

To improve performance on complex, long-horizon tasks, a growing body of work has focused on enabling VLAs to perform explicit reasoning. These methods typically use a chain-of-thought approach [29], where the model first generates a high-level plan or reasoning trace that then conditions the prediction of low-level actions. The form of this reasoning varies across the literature. One set of approaches provides task-level context, leveraging (vision) language models to generate natural language sub-goals or task plans that guide the policy [6, 30–38]. A second set of works condition the policy on more structured, mid-level representations like trajectory sketches, code, or spatial affordances [39–45].

Our method utilizes a similar staged inference procedure, first generating subtasks (e.g., 'move to object'), then predicting mid-level trajectory plans (e.g., 'move primarily downward and slightly right'), before finally generating action commands conditioned on both. Architecturally, our approach aligns more closely with the recent trend toward monolithic models, where a single VLM is responsible for the entire reasoning hierarchy

[45–47]. This stands in contrast to modular pipelines that use distinct models for each stage of reasoning and control [31, 33, 35, 41, 43].

Moreover, our work is distinct from prior VLAs and embodied reasoning models [5, 6, 13, 26, 30, 36, 39] in that (1) we not only represent high-level reasoning in language, but also *represent low-level actions (e.g., end-effector movements) using text*, (2) we do not utilize any explicit action decoders, and (3) *we train the VLA solely with LoRA*. We posit that modifications to the VLM, combined with full-parameter fine-tuning, create a distribution shift deleterious to the VLM's world knowledge and reasoning capabilities. We demonstrate evidence of this claim in Section 4.1.

2.2 Averting Catastrophic Forgetting

A critical and often overlooked phenomenon of many VLAs is catastrophic forgetting, where the rich, general-purpose knowledge of the base VLM is lost during fine-tuning on narrow robotics data [13]. By compromising the visual and semantic understanding of the base VLM, existing VLAs often fail to generalize when prompted with scenarios outside the scope of their robot fine-tuning data [12, 13].

2.2.1 Co-Training for Knowledge Retention

To combat this, the most common effective strategy has been large-scale co-training [6, 9, 11, 13, 18, 26, 48, 48], which involves mixing robot teleoperation data with large, non-robotic datasets (e.g., VQA, image captioning). Co-training effectively regularizes the VLA fine-tuning process, whereby the VLM is continuously trained on data drawn from its pretraining distribution to preserve general reasoning and visual understanding.

However, co-training does not fundamentally resolve the underlying distribution shift problem between the VLM's pretraining corpus and the VLA's robotic fine-tuning data. Instead, it attempts to mitigate the effects of this mismatch by introducing an additional hyperparameter, namely the mixture ratio of robotics and web data. This ratio is non-trivial to tune, requiring numerous training runs and real-world evaluations to find an effective setting for both VQA performance and dexterous robotic control. The difficulty of this search is compounded by the substantial computational cost of training even a single VLA, often billions of parameters [4, 5, 9, 10], making a thorough hyperparameter sweep impractical in many settings.

2.2.2 Advanced Training Schemes

Sophisticated VLA training procedures in Mixture-of-Experts (MoE)-style VLAs have recently been developed as an alternative means of preventing catastrophic forgetting. For instance, [27] attempt to shield the VLM from otherwise destructive gradient updates of the action module via stop-gradient operators. Moreover, [26] have explored a sequential training procedure whereby the VLM's weights are frozen during training of the action expert. Our work challenges the necessity of co-training and other sophisticated training schemes to prevent catastrophic forgetting; by representing robot actions as natural language, our VLA can be effectively fine-tuned with only low-rank adaptation, a simple procedure requiring no major architectural changes.

3 Methodology

In this section, we discuss our VLA training pipeline. Our approach is predicated on a simple principle: to successfully leverage LoRA fine-tuning and avert catastrophic forgetting, we must first align the robot data with the VLM's existing representational space. To this end, we begin by first translating robot teleoperation datasets into natural language descriptions. We posit that this data transformation minimizes distribution shift incurred during LoRA fine-tuning (as evidence by Fig. 3), thereby retaining the VLM's nominal capabilities while enabling effective robot control.

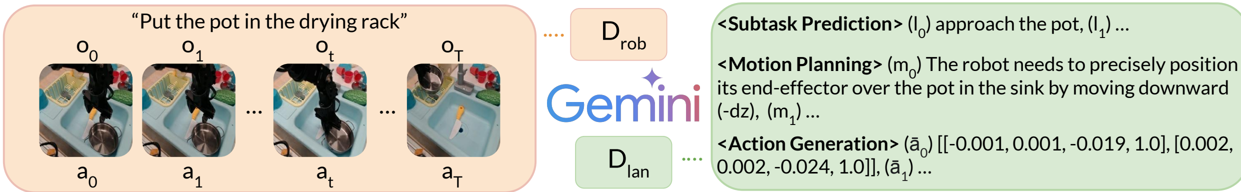

Figure 4 VLM2VLA's pipeline for annotating existing robot datasets $\mathcal{D}_{\text{rob}}$ into $\mathcal{D}_{\text{lan}}$ described via natural language. We use Gemini 2.5 [3] to decompose each trajectory into sub-trajectories, each with an associated subtask, motion plan, and action chunk.

3.1 Actions as Language: A Hierarchical Representation

VLM2VLA frames action prediction as a three-stage VQA hierarchical reasoning process, which has been shown to improve generalization to new tasks and environments [6, 45]; the fundamental difference between our action representation and other VLAs [4, 9, 10, 36, 39, 45] is that high-level and low-level actions are represented directly via language. Let the main task be broken into a series of N steps, indexed by i:

High-Level Subtask Prediction (l_i): given observation $\bar{o}_i$ and language instruction L, the model first describes what immediate subtask l_i is necessary to complete the main task.

Mid-Level Motion Planning (m_i): conditioned on the current subtask and observation, the model generates a spatially informative motion plan m_i, with respect to the robot's end-effector, detailing the movements necessary to complete the subtask. Our motion plans describe only directional movements, e.g., 'move left' and 'move down and slightly forward.' This coarse description was specifically chosen to benefit from the VLM's latent spatial reasoning, which excels at these types of tasks [1, 2].

Low-Level Action Generation ($\bar{a}_i$): conditioned on the current subtask and motion plan, the policy generates a variable-length action-chunk $\bar{a}_i$ to execute directly on the robot. In practice, the action-chunk is a list of lists, where each inner list contains the individual commands *represented as text* for each degree of freedom (DoF) of the robot. In this work, we only considered translational DoFs.

Formally, the distribution we seek to capture with our VLA is $p_\theta(\bar{a}_i, m_i, l_i|\bar{o}_i, L)$, which we decompose as:

$$p_\theta(\bar{a}_i, m_i, l_i|\bar{o}_i, L) = \underbrace{p_\theta(l_i|\bar{o}_i, L)}_{\substack{\text{1) Subtask} \\ \text{Prediction}}} \underbrace{p_\theta(m_i|l_i, \bar{o}_i)}_{\substack{\text{2) Motion} \\ \text{Planning}}} \underbrace{p_\theta(\bar{a}_i|m_i, l_i, \bar{o}_i)}_{\substack{\text{3) Action} \\ \text{Generation}}}, \tag{1}$$

where parameters θ are the weights of the robot policy. This parameterization corresponds to a vision-language model (transformer) taking as input a sequence of multimodal image-text input tokens and predicting a sequence of output text tokens. All observations reside in RGB image space.

3.1.1 Inference-Time Procedure

The decomposition given in Equation (1) describes how our VLA operates at test-time. Every action-chunk prediction is conditioned on the current observation, subtask description, and proposed motion-plan; in practice, we generate all N subtasks at once given the initial observation $\bar{o}_0$ and keep this set fixed for the duration of the rollout. To improve robustness of the policy, we operate in closed-loop fashion with a verifier. At the end of every action-generation cycle, the verifier determines if the model should re-try the current subtask or proceed onto the next one, a process which continues until all N subtasks are completed. In this work, we utilize Gemini 2.5 Pro [3] as the verifier (with details in Appendix C.5).

3.2 Data Curation: Translating Robot Trajectories into a Hierarchical Format

To teach a VLM the aforementioned spatially-grounded reasoning chain, we must first re-label existing robot datasets in natural language. We assume access to a dataset of human teleoperated robot trajectories, $\mathcal{D}_{\text{rob}}$, where each trajectory is a state-action sequence $\tau = \{(o_t, a_t)\}_{t=0}^{T}$ of length T controlling the relative position of the robot's end-effector [10, 49, 50]. Each trajectory comes equipped with a main-task language instruction L. We do not assume access to state information such as joint angles or absolute positioning in a pre-defined reference frame. Our goal is to automatically describe each trajectory $\tau \in \mathcal{D}_{\text{rob}}$ at the high-, mid-, and low-levels described in Section 3.1 to construct a new robot dataset annotated with natural language, $\mathcal{D}_{\text{lan}}$. Herein, we utilize the subset of the Bridgev2 dataset equipped with main-task instructions as $\mathcal{D}_{\text{rob}}$ [50].

Our pipeline for constructing $\mathcal{D}_{\text{lan}}$ is similar in spirit to the data curation strategies of [36, 45]. We use Gemini [3] to automatically annotate trajectories and construct our our natural language dataset (see Fig. 4 for an illustration). Specifically, Gemini decomposes each trajectory into a sequence of N steps. To provide strong spatial grounding for motion planning, we prompt Gemini with the robot's base frame coordinate system, where each axis corresponds to a DoF. We found this enabled Gemini to generate high-quality annotations for relative end-effector movements. Each resulting step i comes equipped with an initial observation $\bar{o}_i$, subtask l_i, motion plan m_i, and variable-length action chunk $\bar{a}_i$, allowing us to represent the trajectory described via natural language, $\bar{\tau}$, as the tuple $\bar{\tau} = \{(\bar{o}_i, l_i, m_i, \bar{a}_i)\}_{i=0}^{N-1} \in \mathcal{D}_{\text{lan}}$. See Appendix A.1 and A.2 for additional details on dataset curation.

In summary, we transform our original robot dataset of state-action pairs into a dataset of image-text pairs, thereby casting robotic control as a standard supervised fine-tuning task. We fine-tune the Gemma-3-12B-IT model [1] using LoRA applied to all its linear modules using the cross-entropy loss. This approach instills action prediction capabilities while preserving the VLM's world knowledge. Additional training details are outlined in Appendix D.

4 Experiments

In this section, we evaluate VLM2VLA on a suite of tasks to answer the following questions:

Q1–Multimodal Understanding: Does VLM2VLA effectively preserve the base VLM's multimodal reasoning ability in standard VQA tasks after fine-tuning on robot data $\mathcal{D}_{\text{lan}}$?

Q2–Robotic Manipulation: Can VLM2VLA achieve a level of in-distribution manipulation performance that is competitive with state-of-the-art VLAs?

Q3–Reasoning Generalization: Does the preserved knowledge from Q1 directly enable VLM2VLA to generalize to novel, out-of-distribution robotics tasks that require reasoning beyond the fine-tuning data $\mathcal{D}_{\text{lan}}$?

Baselines: We contrast our policy against two state-of-the-art tokenization-based VLAs: OpenVLA [4], an autoregressive policy constructed from fine-tuning a pretrained 7B Prismatic VLM on the large-scale Open-X-Embodiment dataset [10, 51], and Embodied Chain-of-Thought (ECoT), a variant of OpenVLA trained to produce reasoning traces before action prediction [45]. ECoT, similar to VLM2VLA, is only fine-tuned on the Bridgev2 dataset. The aforementioned models were chosen as baselines because they can run out-of-the-box on our hardware setup, requiring no additional fine-tuning.

Ablations: We ablate our action representation by mapping the digits (0-9) in our robot action data to the decoded strings of Gemma-3's ten least likely tokens; therefore, during VLA fine-tuning, the model will learn to autoregressively predict actions via these reserved tokens (see Appendix A.3). We call this variation of our method, with action tokens, VLM2VLA-AT. To ensure an accurate comparison, both models are trained identically and trained on the same data, except for the action representation.

4.1 Evaluation of Multimodal Understanding

Many real-world manipulation scenarios deviate from the VLA's fine-tuning distribution, such as natural language instructions with variations in phrasing [52], the presence of novel objects and backgrounds [6, 12], or misaligned camera angles requiring fine-grained perceptual reasoning. In such settings, policies that retain strong visual-semantic priors from pretraining are better equipped to generalize more effectively [6]. In this section, we quantify the extent to which VLM2VLA, and its variants, preserve these priors by thorough evaluation on numerous VQA benchmarks.

Table 1 Multimodal understanding evaluation. Comparison of VLMs and VLAs across multimodal understanding benchmarks. We compare against Prismatic VLM [51], OpenVLA [4], ECoT [45], Gemma-3 [1], MolmoAct [11] and $\pi_{0.5}$ [6]. Our models preserve strong performance across diverse multimodal understanding tasks despite training on robot data. The best and second best results for each benchmark are shown in **bold** and <u>underlined</u>, respectively.

Method	#Params	MMMU	MMStar	MME	OCRBench	MMB-en	MMB-cn	TextVQA	DocVQA	InfoVQA	AI2D	ChartQA	RealWorldQA
Prismatic VLM Family													
Prismatic VLM	7b	35.0	38.8	**1456.6**	32.0	66.2	55.7	42.5	17.5	19.7	54.6	16.7	30.8
OpenVLA	7b	26.3	0	0	0	0	43.0	0	0	0	0	0	0
ECoT	7b	26.6	0	0	0.01	3.7	4.1	0	0	0	0	0	25.6
Gemma-3 Family (with VLM2VLA)													
Gemma-3-4B-IT	4b	39.3	37.1	1205.8	<u>70.2</u>	68.6	64.3	61.5	68.8	40.9	70.5	50.3	44.0
Gemma-3-12B-IT	12b	**46.0**	<u>46.3</u>	1182.3	**75.0**	**76.9**	**74.7**	**68.9**	**80.6**	**50.4**	**78.5**	55.1	**50.6**
VLM2VLA-AT	12b	<u>45.9</u>	45.2	1082.2	65.5	<u>70.9</u>	66.8	64.2	74.6	44.8	<u>74.1</u>	41.8	<u>44.5</u>
VLM2VLA (Ours)	12b	42.7	**48.0**	<u>1391.7</u>	63.9	68.5	<u>67.6</u>	<u>64.9</u>	<u>78.4</u>	<u>46.2</u>	74.0	**58.3**	43.3
Open-Source Co-Trained VLAs													
MolmoAct	7b	28.4	1.2	1224.5	52.7	55.1	46.3	57.5	58.7	41.9	2.0	<u>55.9</u>	8.6
$\pi_{0.5}$	3b	24.0	21.7	1061.9	6.8	6.8	0.3	10.0	4.6	7.7	27.0	5.1	2.7

We report the results for all policies and their original VLM backbones in Tab. 1. As demonstrated, both OpenVLA and ECoT suffer from catastrophic forgetting relative to the original Prismatic VLM [13, 51]. In contrast, VLM2VLA experiences only minor losses in performance across all VQA benchmarks, thereby conclusively answering Q1. While the VLM2VLA-AT ablation achieved comparable VQA scores, suggesting that our LoRA training scheme is primarily responsible for mitigating catastrophic forgetting, we argue this is a necessary but not sufficient condition for creating a generalist policy. As we show in Section 4.2, the choice of action representation is an important factor affecting generalization in downstream robotic tasks.

To compare VLM2VLA against state-of-the-art co-trained VLAs, we also evaluate two recently released open-source models: MolmoAct [11] and $\pi_{0.5}$ [6]. While MolmoAct is strong in VQA ability, VLM2VLA outperforms it across all reported benchmarks. We acknowledge that model size is a confounding variable in the comparison, yet, the results in Tab. 1 demonstrate that our methodology is competitive with co-trained VLAs. On the other hand, the significantly lower performance of $\pi_{0.5}$ across most VQA benchmarks underscores that co-training is not a guaranteed solution to catastrophic forgetting.

4.2 Evaluating Robotic Manipulation

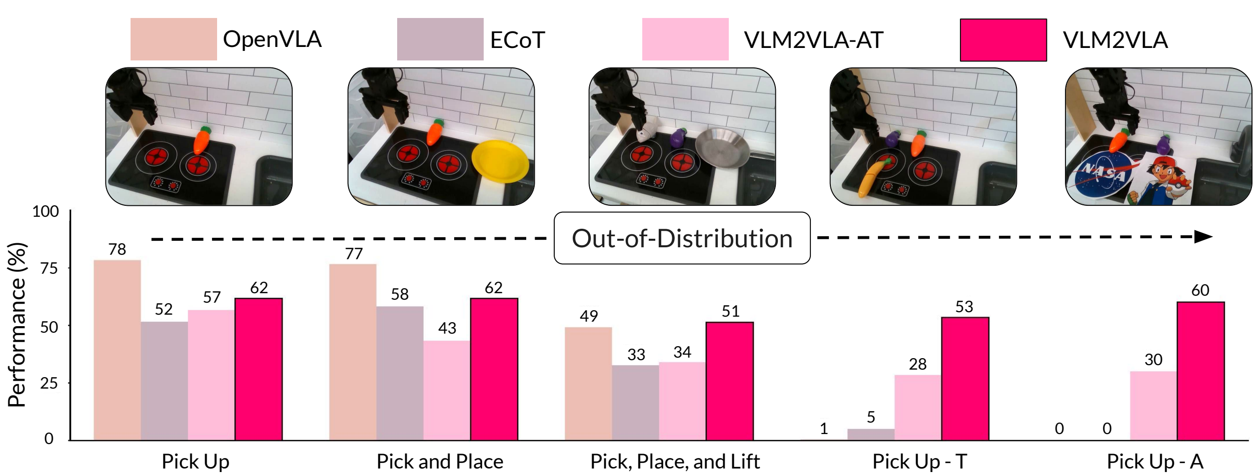

Figure 5 Comparative evaluation of VLA performance on in-distribution (ID) and out-of-distribution (OOD) robotic manipulation tasks. VLM2VLA maintains high success rates on OOD tasks, highlighting its superior generalization capabilities. Each bars corresponds to an average over thirty trials, except for the 'Pick Up -T' task, where each bar corresponds to an average over ninety trials.

Having established that VLM2VLA and its variants retain their foundational pretraining knowledge, we now evaluate if this training procedure translates to robotic control. Specifically, we assess VLM2VLA's ability to perform robotic manipulation tasks of varying complexity in in-distribution (ID) and out-of-distribution (OOD) scenarios. All real-world policy evaluations receive a total of 30 trials and are performed on a 6-DoF WidowX 250S robotic arm in a toy kitchen environment, as prescribed in [50].

We compare VLM2VLA to the baselines and ablations on two ID tasks, one borderline ID task, and two OOD tasks. The in-distribution experiments probe each policy's action prediction capabilities over short and long horizons. The out-of-distribution experiments investigate if the policies can leverage the latent world knowledge of their VLM backbone to generalize beyond the VLA fine-tuning data. All skills tested are 'pick up' or 'pick and place' tasks, which are common in policy evaluation [4, 7, 12, 45, 53], and we describe each in detail below.

Pick Up the Carrot (Pick Up): A baseline task testing the model's fundamental localization and grasping capabilities. This task is considered in-distribution because the task object (carrot) and instruction are within the VLA fine-tuning data.

Put the Carrot On the Yellow Plate (Pick and Place): A longer horizon baseline task testing the model's ability to perform more complex actions. This task is considered in-distribution because the object and instruction are within the VLA fine-tuning data.

Put the Eggplant In the Pan, Then Lift the Fish (Pick, Place, and Lift): A long-horizon, multi-step task requiring the robot to perform both pick-and-place and pick-up actions. This task is borderline in-distribution; the objects and instructions, when considered separately, are in-distribution, but not when combined.

Pick Up the Carrot (Pick Up - T): A pick-up task with natural language instruction given in one of three languages: Spanish ('recoger la zanahoria'), Mandarin ('拿起胡萝卜'), and Hindi ('गाजर उठाओ'). This task is considered out-of-distribution because the instructions are not within the VLA fine-tuning dataset, thereby requiring the model to implicitly translate the task. Two distractor objects are present in each trial to prevent the policy from inferring the task from the observation alone.

Pick Up the Item Above Ash Ketchum (Pick Up - A): AA pick-up task requiring the model to first locate the pop-culture figure 'Ash Ketchum' and lift the correct object. This task is considered out-of-distribution because VLAs are not trained to recognize semantic concepts like 'Ash Ketchum,' thereby requiring the model to leverage its latent world knowledge to succeed. Distractor logos ('NASA') and objects are present. The item located above 'Ash Ketchum' varies in-between trials, as does its location in the environment.

We report the full task success rates for all models in Fig. 5 (see Appendices C.1 to C.6 for additional scoring and experimental details). For the multilingual translation tasks (Pick Up -T), we report the average across all languages, which amounts to ninety trials per model.

4.2.1 Results of Baselines

On the ID pick-up and pick-and-place tasks, OpenVLA performs the best. This high performance is likely explained by fine-tuning on the much larger Open-X-Embodiment dataset, which provides broader coverage of common grasping skills. Nevertheless, the comparable success rates observed for VLM2VLA answer Q2: the 'action as language' approach enables competitive action prediction in a specific robot embodiment.

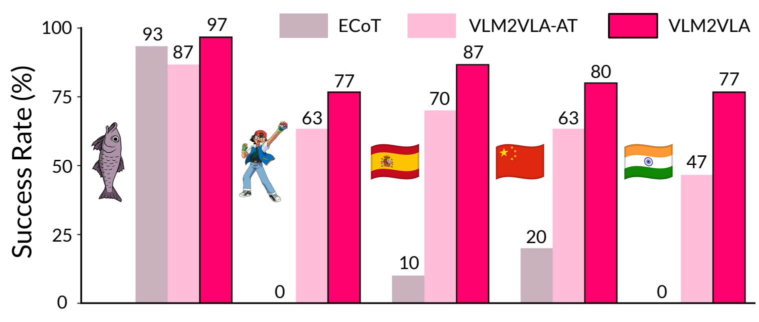

Figure 6 Analysis of task decomposition for OOD manipulation tasks. Points are awarded if the model's task plan correctly identifies the task object, and task destination if present. See Appendix C.3 for additional details.

As task complexity increases, the advantage of a purely reactive policy, like OpenVLA, starts to diminish. In the compositional task 'Put the Eggplant In the Pan, Then Lift the Fish', we found that OpenVLA often successfully completes the first subtask (put the eggplant in the pan), but fails to attempt the second. ECoT, on the other hand, normally generates motion plans to finish both subtasks, but fails to execute them as accurately as OpenVLA. We found that of the three policies, for this task, VLM2VLA not only generates correct hierarchical reasoning, but also executes them more frequently (see Fig. 5 and Fig. 6).

The OOD scenarios most clearly demonstrate the benefits of our knowledge-preserving pipeline. In the multilingual translation experiment (Pick Up - T), our method significantly outperforms both OpenVLA and ECoT. As shown in Fig. 6 and Fig. 7, VLM2VLA frequently translates the non-English language instruction and identifies the correct object; this is a direct application of the multilingual capabilities retained by our model as reported in Section 4.1. While ECoT does achieve a non-zero task success rate, the results presented in Fig. 6 suggest it may be relying on heuristics, such as simply picking up the closest or most salient object in the near vicinity, rather than a genuine understanding of the instruction.

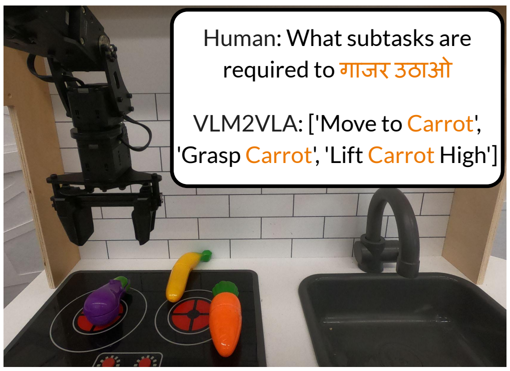

Figure 7 A qualitative demonstration of VLM2VLA's zero-shot multilingual capabilities. Given the language instruction in Hindi ('pick up the carrot'), our model identifies the correct object amidst distractors (eggplant and banana), demonstrating a genuine understanding of the task.

This trend is even more pronounced in the 'Pick Up the Item Above Ash Ketchum' task, requiring a combination of both open-world semantic knowledge (recognizing a pop-culture figure) and spatial reasoning. Here, VLM2VLA is the only model to achieve a meaningful success rate. This result provides a positive answer to Q3: the foundational world knowledge preserved by our method can be leveraged to solve novel problems zero-shot.

4.2.2 Results of Ablations

Section 4.1 demonstrated that LoRA fine-tuning can mitigate catastrophic forgetting for both our method and the token-based ablation, but does the 'actions as language' representation actually facilitate learning a more effective policy? The results demonstrated in Fig. 5 and Fig. 6 show our approach consistently outperforms the token-based ablation across all manipulation tasks. While VLM2VLA-AT performs competitively on simple ID tasks, its performance degrades as task complexity increases; this is seen most clearly in the OOD setting despite retaining strong VQA capabilities. For instance, Fig. 6 shows VLM2VLA-AT struggles with multilingual commands and scores only half as well as VLM2VLA in the 'Pick Up the Item Above Ash Ketchum' task (achieving a success rate of just 30% to our model's 60%). These findings are suggestive of a disconnect between the VLM's latent world knowledge and the action token assignment achieved after fine-tuning.

5 Conclusion and Discussion

In this work, we present VLM2VLA: a novel VLA training paradigm that addresses the challenge of catastrophic forgetting often experienced when adapting a VLM for robotic control. We demonstrate that by grounding the action prediction problem in natural language, LoRA can be used to train manipulation policies without significantly degrading the backbone VLM's foundational capabilities. Our data-centric approach of treating 'actions as language' obviates the need for major architectural modifications to the VLM or expensive co-training schemes. As our experiments show, the retained latent world knowledge of VLM2VLA enables zero-shot generalization to novel tasks that require open-world semantic reasoning. Ultimately, this work offers one path toward building generalist policies by closing the gap between large-scale foundation models and real-world robotic control.

5.1 Limitations and Future Work

While VLM2VLA demonstrates one promising path toward better leveraging the foundational capabilities of VLMs for generalist policies, our approach has several limitations that present clear avenues for future work.

Inference latency. VLM2VLA generates actions autoregressively, which is inherently slow; in our experiments, the median run-time required for one cycle of action generation was 6.1 seconds, though subject to high variance (see Appendix C.6 for additional inference details). Future work could explore more advanced decoding techniques to accelerate our hierarchical prediction scheme without compromising the VLM's pretrained representations.

Dexterous tasks. Our work has focused on translational end-effector control, which is suitable for many simple pick-and-place tasks. While this design choice afforded strong action labeling from Gemini, it precludes dexterous manipulation tasks which require nuanced action prediction, e.g., rotation. Moreover, the granularity at which we predict motion plans is coarse, and a logical next step is to obtain fine-grained language annotations when constructing our VLA fine-tuning dataset. As vision-language models become more adept at spatial reasoning, we anticipate this re-labeling scheme will enable more precise robotic control.

Cross-embodiment learning. The scope of our work limited training to a specific robot embodiment, thereby preventing the same policy from readily adapting to other robots, especially those which utilize other means of low-level control, such as joint angles, which do not map easily to spatial affordances. Yet, we believe our 'actions as language' relabeling scheme is a promising solution. Using language as a common medium, it ought to be possible to describe all robot actions regardless of embodiment. Therefore, performing VLM2VLA's relabeling scheme could potentially be used to train a cross-embodiment policy.

Improving or eliminating the verifier. Our method currently relies on a separate verifier module to transition between subtasks, further slowing down inference speed. Future work could explore how to better train the base VLM as a verifier, or investigate how the policy could automatically reason about subtask completion given the observation and task description alone, thereby averting the this step altogether.

Larger-scale training. We are working towards scaling our training pipeline with larger robotics datasets. We hypothesize that combining our language-based action representation with large-scale training may unlock zero-shot generalization, open-world semantic reasoning, and instruction following capabilities that go well beyond the state-of-the-art.

Acknowledgments

We are grateful to Dhruv Shah for fruitful conversations in the early stages of this work.

Asher J. Hancock was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2146755. The authors were partially supported by the NSF CAREER Award #2044149, #2107048, the Office of Naval Research (N00014-23-1-2148), and the Sloan Fellowship.

References

[1] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025.

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

[3] Gemini Model Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025.

[4] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model, 2024.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024.

[6] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

[7] Jensen Gao, Suneel Belkhale, Sudeep Dasari, Ashwin Balakrishna, Dhruv Shah, and Dorsa Sadigh. A taxonomy for evaluating generalist robot policies, 2025.

[8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale, 2023.

[9] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[10] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[11] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct: Action reasoning models that can reason in space, 2025.

[12] Asher J. Hancock, Allen Z. Ren, and Anirudha Majumdar. Run-time observation interventions make vision-language-action models more visually robust, 2024.

[13] Zhongyi Zhou, Yichen Zhu, Minjie Zhu, Junjie Wen, Ning Liu, Zhiyuan Xu, Weibin Meng, Ran Cheng, Yaxin Peng, Chaomin Shen, and Feifei Feng. ChatVLA: Unified multimodal understanding and robot control with vision-language-action model, 2025.

[14] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing AI into the physical world, 2025.

[15] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021.

[16] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action model, 2025.

[17] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024.

[18] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied multimodal language model, 2023.

[19] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation, 2025.

[20] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3d vision-language-action generative world model, 2024.

[21] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. DexVLA: Vision-language model with plug-in diffusion expert for general robot control, 2025.

[22] NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots, 2025.

[23] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, Lars Liden, and Jianfeng Gao. Magma: A foundation model for multimodal ai agents, 2025.

[24] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models, 2025.

[25] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024.

[26] Zhongyi Zhou, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. ChatVLA-2: Vision-language-action model with open-world embodied reasoning from pretrained knowledge, 2025.

[27] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025.

[28] Shresth Grover, Akshay Gopalkrishnan, Bo Ai, Henrik I. Christensen, Hao Su, and Xuanlin Li. Enhancing generalization in vision-language-action models by preserving pretrained representations, 2025.

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[30] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. OneTwoVLA: A unified vision-language-action model with adaptive reasoning, 2025.

[31] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025.

[32] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models, 2025.

[33] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina

Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as I can, not as I say: Grounding language in robotic affordances, 2022.

[34] Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized planning in PDDL domains with pretrained large language models, 2023.

[35] Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections, 2024.

[36] Jaden Clark, Suvir Mirchandani, Dorsa Sadigh, and Suneel Belkhale. Action-free reasoning for policy generalization, 2025.

[37] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models, 2022.

[38] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language, 2022.

[39] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language, 2024.

[40] Soroush Nasiriany, Sean Kirmani, Tianli Ding, Laura Smith, Yuke Zhu, Danny Driess, Dorsa Sadigh, and Ted Xiao. RT-Affordance: Affordances are versatile intermediate representations for robot manipulation, 2024.

[41] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation, 2025.

[42] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control, 2023.

[43] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged robot vision-language-action model for navigation, 2025.

[44] Peiyuan Zhi, Zhiyuan Zhang, Yu Zhao, Muzhi Han, Zeyu Zhang, Zhitian Li, Ziyuan Jiao, Baoxiong Jia, and Siyuan Huang. Closed-loop open-vocabulary mobile manipulation with gpt-4v, 2025.

[45] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning, 2025.

[46] Xiang Li, Cristina Mata, Jongwoo Park, Kumara Kahatapitiya, Yoo Sung Jang, Jinghuan Shang, Kanchana Ranasinghe, Ryan Burgert, Mu Cai, Yong Jae Lee, and Michael S. Ryoo. LLaRA: Supercharging robot learning data for vision-language policy, 2025.

[47] Dantong Niu, Yuvan Sharma, Giscard Biamby, Jerome Quenum, Yutong Bai, Baifeng Shi, Trevor Darrell, and Roei Herzig. LLARVA: Vision-action instruction tuning enhances robot learning, 2024.

[48] Huang Fang, Mengxi Zhang, Heng Dong, Wei Li, Zixuan Wang, Qifeng Zhang, Xueyun Tian, Yucheng Hu, and Hang Li. Robix: A unified model for robot interaction, reasoning and planning. *arXiv preprint arXiv:2509.01106*, 2025.

[49] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert

Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A large-scale in-the-wild robot manipulation dataset, 2025.

[50] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale, 2024.

[51] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic VLMs: Investigating the design space of visually-conditioned language models, 2024.

[52] Selma Liliane Wanna, Agnes Luhtaru, Ryan Barron, Jonathan Salfity, Juston Moore, Cynthia Matuszek, and Mitch Pryor. Let's talk about language! investigating linguistic diversity in embodied AI datasets. In *1st Workshop on Safely Leveraging Vision-Language Foundation Models in Robotics: Challenges and Opportunities*, 2025.

[53] David Snyder, Asher James Hancock, Apurva Badithela, Emma Dixon, Patrick Miller, Rares Andrei Ambrus, Anirudha Majumdar, Masha Itkina, and Haruki Nishimura. Is your imitation learning policy better than mine? policy comparison with near-optimal stopping, 2025.

[54] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer Reinforcement Learning, 2025.

Appendix

A Data Curation

A.1 Prompting

In this section, we showcase our Gemini prompting strategy to generate language-annotated robot trajectories at scale. Our example is specific to the WidowX 250S robot embodiment, focusing solely on translational degrees of freedom. All text within the prompt is invariant across robot trajectories, except for words/phrases designated in color, which are trajectory-dependent.

Gemini Inputs and Prompt Template

Inputs:

Gemini is provided with the robot trajectory data $\tau = \{(o_t, a_t)\}_{t=0}^{T}$ consisting of RGB observations o_t and actions a_t controlling the relative position of the robot's end-effector. Additionally, Gemini is provided with the main-task instruction L. We combine all actions and instruction into a text file called trajectory_log_content.

Gemini Prompt

Attached are images of a robot performing a task, and the real actions in a text file called trajectory_log_content. You are programming a 4-DoF robot arm (x,y,z,gripper). Positive x points forward (negative x points backward) with respect robot's end-effector. Positive y points left (negative y points right) with respect to the robot's end effector. Positive z points up (negative z points down) with respect to the robot's end-effector. The gripper takes a binary value of 1 for open and 0 for closed.

Your task is to analyze the language instruction found in the trajectory log. Crucially, first think step-by-step and explain your reasoning for each action phase necessary to complete the instruction given all the images and their ordering during the trajectory. Be short and concise. Output your reasoning as a structured JSON file.

Then, after you've explained your reasoning, give the actions with respect to (dx, dy, dz, gripper) that the robot must execute to achieve this goal given the orientation described previously, associating each action or group of actions with the natural language decomposition you decided on. Store these actions in JSON format as a list of dictionaries, where each dictionary has 'step_description' (string) and 'actions' (list of lists representing [dx, dy, dz, gripper]). Make sure your actions match what is provided in the attached .txt file. For example:

```
{{
"step_description": "Grasp the Pepper",
"actions": [
      [0.010, 0.026, 0.002, 1.0],
      [0.017, 0.033, 0.008, 1.0],
      [0.003, 0.007, -0.018, 1.0],
      [-0.006, 0.0, -0.015, 1.0],
      [0.003, 0.002, 0.001, 0.0]
    ]
}}
```

Here is the trajectory log content: trajectory_log_content

In summary, Gemini 2.5 [3] is given the entire trajectory $\tau = \{(o_t, a_t)\}_{t=0}^{T} \in \mathcal{D}_{\text{rob}}$ and main-task instruction L as context, as well as the robot's coordinate system and sign convention in the base frame. For example, the Bridgev2 dataset [50] utilizes the 6-DoF WidowX 250S robotic arm, where movement in the positive 'x' direction corresponds to 'forward', negative movement in the 'y' directions corresponds to 'right,' and so forth. Since all action sequences are given as relative movements of the robot's end-effector, the aforementioned coordinate system provides strong spatial grounding for Gemini to construct high-quality language annotations and determine the most appropriate timesteps to carve the trajectory into sub-trajectories. To address potential concerns about auto-labeling errors, we manually inspected a random subset of Gemini's generated data for accuracy in subtask decomposition, motion-planning, and action prediction.

Initially, we solely used Gemini 2.5 Pro, but later switched to Gemini 2.5 Flash to save on costs. In total, generating the entire dataset costed roughly $900. Finally, we provide an example of single language-annotated trajectory from the Bridgev2 dataset. We additionally include the first and last observations of the robot trajectory for reference.

Figure 8 Key observations throughout the original, robot trajectory, which are provided to Gemini as context.

(a) Observation: 0 (Initial) **(b)** Observation: 10 **(c)** Observation: 20 **(d)** Observation: 28 (Final)

Gemini Language Annotation Example

Output:

Below is the structured JSON output generated by Gemini for the task of L = 'open the silver pot'. Gemini first decomposes the trajectory into sequence of steps, then performs a chain-of-thought reasoning to generate a motion plan. Finally, it copies the corresponding actions to each step.

```
[
  {
    "STEP_DESCRIPTION": "Move to Lid Handle",
    "REASONING": "The robot must first navigate its end-effector to be directly above
        the lid handle. This requires moving downwards (-dz) and adjusting the
        horizontal position (likely backward -dx and right -dy based on the initial
        position and target). The gripper must remain open (1) for grasping.",
    "MAIN_MOVEMENTS": "Primarily backward (-dx), right (-dy), and downward (-dz)
        movements. Gripper remains open (1).",
    "ACTIONS": [
      [ 0.0,  0.0,  0.0,  1.0 ],
      [ -0.002,  -0.002,  -0.013,  1.0 ],
      [ -0.023,  -0.007,  -0.005,  1.0 ],
      [ -0.018,  -0.006,  0.022,  1.0 ],
      [ -0.018,  -0.005,  -0.003,  1.0 ],
      [ -0.018,  -0.006,  -0.005,  1.0 ],
      [ -0.015,  -0.014,  -0.014,  1.0 ],
      [ -0.004,  -0.010,  -0.007,  1.0 ],
      [ 0.0,  -0.005,  -0.003,  1.0 ]
```

```
    ]
  },
  {
    "STEP_DESCRIPTION": "Grasp Lid Handle",
    "REASONING": "Once positioned correctly, the robot needs to grasp the handle.
        This involves closing the gripper (0). There might be a final small downward
        adjustment (-dz) just before or during the grasp.",
    "MAIN_MOVEMENTS": "Close gripper (0), minor downward adjustment (-dz).
        Gripper changes to 0.",
    "ACTIONS": [
      [ 0.002,  -0.007,  -0.010,  0.0 ]
    ]
  },
  {
    "STEP_DESCRIPTION": "Lift Lid",
    "REASONING": "After securing the handle, the robot must lift the lid vertically
        (+dz) to clear the base of the pot. It also needs to move it slightly away
        from the pot base, involving some horizontal movement (likely forward +dx
        and left +dy). The gripper remains closed (0) to hold the lid.",
    "MAIN_MOVEMENTS": "Primarily upward (+dz), forward (+dx), and left (+dy).
        Gripper remains closed (0).",
    "ACTIONS": [
      [ 0.007,  0.0,  -0.001,  0.0 ],
      [ 0.002,  0.0,  0.0,  0.0 ],
      [ -0.002,  -0.002,  0.0,  0.0 ],
      [ -0.004,  0.0,  0.002,  0.0 ],
      [ 0.0,  0.004,  0.004,  0.0 ],
      [ 0.015,  0.014,  0.017,  0.0 ],
      [ 0.023,  0.007,  0.020,  0.0 ]
    ]
  },
  {
    "STEP_DESCRIPTION": "Move Lid to Placement Spot & Lower",
    "REASONING": "The goal is to open the pot, implying the lid needs to be placed
        somewhere else. This involves horizontal movement (forward +dx, left +dy)
            towards the placement spot and then lowering the lid (-dz). The gripper
            stays closed (0).",
    "MAIN_MOVEMENTS": "Primarily forward (+dx), left (+dy), and downward (-dz).
        Gripper remains closed (0).",
    "ACTIONS": [
      [ 0.027,  0.008,  -0.003,  0.0 ],
      [ 0.021,  0.017,  -0.007,  0.0 ],
      [ 0.018,  0.007,  -0.011,  0.0 ],
      [ 0.015,  0.007,  -0.010,  0.0 ],
      [ 0.010,  0.004,  -0.014,  0.0 ]
    ]
  },
  {
    "STEP_DESCRIPTION": "Release Lid",
    "REASONING": "Once the lid is resting on the surface, the robot must release it
        by opening the gripper (1).",
    "MAIN_MOVEMENTS": "Open gripper (1). Gripper changes to 1.",
    "ACTIONS": [
```

```
      [ 0.003,  0.001,  -0.002,  1.0 ]
    ]
  },
  {
    "STEP_DESCRIPTION": "Retract",
    "REASONING": "Finally, the robot should move its arm away from the lid and
        the pot to a safe, neutral position, typically upwards (+dz) and potentially
            backward (-dx). The gripper remains open (1).",
    "MAIN_MOVEMENTS": "Primarily upward (+dz) and backward (-dx).
        Gripper remains open (1).",
    "ACTIONS": [
      [ -0.002,  -0.002,  0.0,  1.0 ],
      [ -0.005,  -0.005,  0.006,  1.0 ],
      [ 0.0,  -0.003,  0.022,  1.0 ],
      [ -0.010,  0.002,  0.022,  1.0 ],
      [ -0.010,  -0.001,  0.017,  1.0 ],
      [ 0.0,  0.0,  0.001,  1.0 ]
    ]
  }
]
```

A.2 Data Preparation

After Gemini splits each trajectory into sub-trajectories, we found it helpful in policy training to chunk the actions along a single dimension to have larger magnitudes; in particular, we found that without this step, VLM2VLA often provided negligible action predictions, even though its predicted motion-plans appears sensible.

In this work, we used a threshold along a single dimension, without sign change, of 2.5 centimeters and an absolute threshold of 5 centimeters. We include an example of the data for training Gemma-3 before and after this post-processing step below:

Original Sub-Trajectory Action Data

Inputs:

The model is provided with the subtask 'Grasp the Yellow Pepper' and image file: obs_0.jpg.

Human Prompt

You are programming a 4-DoF robot arm. Based on the image (obs_0.jpg) and subtask 'Grasp the Yellow Pepper', output the sequence of low-level [x, y, z, gripper] actions needed to accomplish the task.

Model Output

```
[[ 0.000,  0.000,  0.000, 1.0],
 [-0.002,  0.000, -0.007, 1.0],
 [ 0.000, -0.004, -0.016, 1.0],
 [ 0.002, -0.002, -0.014, 1.0],
 [ 0.003,  0.000, -0.008, 1.0],
 [ 0.002,  0.000, -0.011, 1.0],
 [ 0.000,  0.000, -0.005, 1.0],
 [ 0.000,  0.000, -0.007, 1.0],
 [ 0.000,  0.000, -0.006, 1.0],
 [ 0.001, -0.003, -0.003, 0.0]]
```

Post-Processed Sub-Trajectory Action Data

Inputs:

The model is provided with the subtask 'Grasp the Yellow Pepper' and image file: obs_0.jpg

Human Prompt

You are programming a 4-DoF robot arm. Based on the image (obs_0.jpg) and subtask 'Grasp the Yellow Pepper', output the sequence of low-level [x, y, z, gripper] actions needed to accomplish the task.

Model Output

```
[[-0.002, -0.004, -0.023, 1.0],
  [0.007, -0.002, -0.045, 1.0],
  [0.0, 0.0, -0.006, 1.0],
  [0.001, -0.003, -0.003, 0.0]]
```

We further augment our dataset with two additional sets of data. First we construct positive and negative examples of subtask completion, each consisting of a tuple $(\bar{o}_i, \bar{o}_j, l_i)$ with j distinct from i: the robot successfully completed subtask i if and only if $j = i + 1$. In principle, this data could train the VLM as the verifier, but we found our model unable to reliably perform this step. We leave addressing this shortcoming as future work.

Secondly, we train the model to predict which direction (e.g., left, right, or none) the robot should move to complete a given subtask, using the same movement threshold of 2.5 centimeters. While we do not query our VLA in this manner at run-time, we found this auxiliary training signal beneficial for downstream performance.

A.3 Actions as Tokens

For the action representation ablation, we cast the digits 0-9 to the tenth least likely tokens under the Gemma-3-12B-IT model. We describe this mapping in the table provided in Table 2. In addition, we provide an example of decoded model output under this mapping, identical to the one provided above.

Table 2 Mapping of digits to the least likely tokens under the Gemma-3 model.

Digit	Token	Token ID
9	<unused6142>	262044
8	<unused6141>	262043
7	<unused6140>	262042
6	<unused6139>	262041
5	<unused6138>	262040
4	<unused6137>	262039
3	<unused6136>	262038
2	<unused6135>	262037
1	<unused6134>	262036
0	<unused6133>	262035

Post-Processed Sub-Trajectory Action Data Under Action Tokenization

Model Output

```
[[-<unused6133>.<unused6133><unused6133><unused6135>, -<unused6133>.<unused6133>
<unused6133><unused6137>, -<unused6133>.<unused6133><unused6135><unused6136>, <u
nused6134>.<unused6133>],
 [<unused6133>.<unused6133><unused6133><unused6140>, -<unused6133>.<unused6133><
unused6133><unused6135>, -<unused6133>.<unused6133><unused6137><unused6138>, <un
used6134>.<unused6133>],
 [<unused6133>.<unused6133>, <unused6133>.<unused6133>, -<unused6133>.<unused613
3><unused6133><unused6139>, <unused6134>.<unused6133>],
 [<unused6133>.<unused6133><unused6133><unused6134>, -<unused6133>.<unused6133><
unused6133><unused6136>, -<unused6133>.<unused6133><unused6133><unused6136>, <un
used6133>.<unused6133>]]
```

B VQA Experiment Details

We build our evaluation pipeline using the open-source framework `lmms-eval`[1], which provides standardized implementations of multimodal benchmark datasets, evaluation metrics, and inference wrappers for a wide range of models. We report results for our VLM2VLA model (finetuned with Gemma-3-12B backbone), two strong baselines (OpenVLA and ECoT, both Prismatic-7B backbone), and the original instruction-tuned Gemma-3-4B and Gemma-3-12B models.

MolmoAct and $\pi_{0.5}$ are designed to handle inputs from multiple cameras (e.g., exterior and wrist views). However, since standard VQA benchmarks provide only a single image per sample, we use the multi-view vision-language model to process single-image VQA benchmarks by utilizing only the primary image input from each sample and masking the other image inputs. We use the $\pi_{0.5}$-base checkpoint for all $\pi_{0.5}$ experiments.

C Robot Manipulation Tasks and Detailed Results

C.1 Experiments and Manipulation Evaluation Metrics

As described in Section 4.2, we evaluate all generalist robot policies on two in-distribution tasks, one borderline in-distribution task, and two out-of-distribution tasks. Each policy received a total of 30 roll-outs for each

[1] https://github.com/EvolvingLMMs-Lab/lmms-eval

task. In this section, we provide additional details on the scoring rubric used to evaluate each method, giving special care to fairly evaluate planning and non-planning policies.

Pick Up the Carrot: The robot's goal is to navigate to and lift up the carrot. There are no distractor objects present. This experiment is considered within distribution because it is part of the Bridgev2 dataset.

Between trials, the carrot's initial position varied between three possibilities: (i) directly below the robot's gripper, (ii) directly in front of the robot, (iii), in front and to the left or right of the robot. The robot's initial position was constant between trials.

Scoring: Partial credit (1 out of 2) is assigned if the robot makes contact with the carrot.

Put the Carrot On the Yellow Plate: The robot's goal is to grasp the carrot, move to the plate, and successfully release it on the plate. There are no distractor objects present. This experiment is considered within distribution because it is part of the Bridgev2 dataset.

Between trials, the carrot's initial position varied between two possibilities: (i) directly in front of the robot and (ii) in front and to the left or right of the robot. The robot's initial position was constant between trials.

Scoring: Partial credit (1 out of 2) is assigned if the robot makes contact with the carrot.

Put the Eggplant In the Pan, Then Lift the Fish: The robot's goal is to grasp the eggplant, move and release it in the pan, then navigate to and successfully lift the fish. This task is considered borderline in-distribution because the individual subtasks are within distribution, but the combination is not.

The eggplant is located to the right of the fish between all trials, which is located in front of and to the right of the robot's initial condition; all starting positions for the eggplant, pan, fish, and robot are invariant across trials.

Scoring: For non-planning policies (OpenVLA), partial credit (1 out of 5) is awarded if the robot makes contact with the eggplant, (2 out of 5) if the robot places the eggplant in the pan, (3 out of 5) if the robot moved toward the fish, (4 out of 5) if the robot contacts the fish, and (5 out of 5) if the robot lifts the fish.

For planning policies (ECoT, VLM2VLA), partial credit (1 out of 5) is awarded if the policy generates a subtask decomposition or motion-plan with the correct primitives: grasping the eggplant, moving to the pan, lifting the fish. The remaining partial credit is awarded the same as for non-plannign policies: (2 out of 5) if the robot makes contact with the eggplant, (3 out of 5) if the robot places the eggplant in the pan, (4 out of 5) if the robot contacts the fish, and (5 out of 5) if the robot lifts the fish.

Pick Up the Carrot (Multilingual Translation): The robot's goal is to navigate to and lift up the carrot. Instructions are given in one of three languages: Spanish ('recoger la zanahoria'), Mandarin ('拿起胡萝卜'), and Hindi ('गाजर उठाओ'). This task is considered out of distribution becuase the language commands are not within the Bridgev2 dataset.

Between trials, the carrot's initial position varied between three possibilities: (i) directly below the robot's gripper, (ii) directly in front of the robot, (iii), in front and to the left or right of the robot. The robot's initial position was constant between trials. Two distractor objects (banana and eggplant) are present in each trial in an attempt to prevent the policy from inferring the task from the observation alone.

Scoring: For non-planning policies (OpenVLA), partial credit (1 out of 2) is awarded if the robot makes contact with the carrot. For planning policies (ECoT, VLM2VLA), partial credit (1 out of 2) is awarded if the policy's planning included the correct item 'carrot' *and* the robot makes contact with the carrot.

Pick Up the Item Above Ash Ketchum: The robot's goal is to navigate to and lift up the object situated above the pop-culture figure 'Ash Ketchum.' This task is considered out of distribution because the picture of 'Ash Ketchum' is not present in the Bridgev2 dataset.

Between trials, the picture of 'Ash Ketchum' varied between being either to the left or right of the robot's starting position, which remained invariant across trials. The target object is placed a few centimeters above the picture to preclude partial observability.

Scoring: For non-planning policies (OpenVLA), partial credit (1 out of 2) is awarded if the robot makes contact with the correct object. For planning policies (ECoT, VLM2VLA), partial credit (1 out of 2) is awarded if the policy's planning included the correct item *and* the robot makes contact with the item.

C.2 Hardware Details

All real-world policy evaluations were performed on a 6-DoF WidowX 250S robotic arm in a toy kitchen environment, as prescribed in [50]. Images were captured with a Realsense D435 camera mounted on the right side of the robot to provide a third-person point of view. The initial conditions of task and distractor objects between trials were varied to ensure an accurate appraisal of policy performance. Like the environment, all objects are drawn from the Bridgev2 dataset to minimize out-of-distribution failures. To prevent damage to our setup, a safety filter was applied to all policies evaluated in this work. The filter prevented further execution of actions which would drive the robot into the environment, e.g., zero out downward or forward commands if the end-effector was contacting the counter or wall.

C.3 Task Decomposition Evaluation Metrics

In this section, we describe the scoring procedure used to generate Fig. 6. The objective was to quantify each policy's ability to interpret and break down a variety of user instructions. Performance was measured by analyzing the textual output of each model's chain-of-thought reasoning, or subtask decomposition, whenever available.

For each task considered from Appendix C.1, a model's reasoning was considered successful if the model's textual output contained a specific set of required keywords. The evaluation was case-insensitive and manually checked to ensure all policies received a fair appraisal. The keywords and synonyms are defined as follows:

eggplant: purple, aubergine

carrot: orange, gajar

pan: (no synonyms defined)

fish: (no synonyms defined)

We now specify the scoring rubric for each experiment which required analysis:

Put the Eggplant In the Pan, Then Lift the Fish: success required the simultaneous presence of synonyms for 'eggplant,' 'pan,' and 'fish.'

Pick Up the Carrot (Multilingual Translation): success required the presence of a carrot synonym in the model's output. Interestingly, some subtask decompositions from VLM2VLA occasionally contained the word 'gajar,' the Hindi synonym for carrot, demonstrating direct translation abilities of our method. This phenomena was not observed for Spanish nor Mandarin inputs.

Pick Up the Item Above Ash Ketchum: success required identifying the correct object for that specific trial. The required keyword was either a synonym of carrot *or* eggplant, depending on the ground truth object for that trial.

C.4 Test-Time Prompting

In this section, we describe the prompting strategies used in VLM2VLA to elicit grounded subtask predictions, spatial reasoning for motion-planning, and translational action prediction.

First, given the initial image, we prompt VLM2VLA to break the main language instruction into a series of subtasks.

Subtask Prediction Prompt

Describe the sequence of remaining high-level steps required to complete the overall task 'main task', starting from the current state. Give your output as a list. Be specific and do not skip steps. Here is an example for 'put the pot in the sink': ['Move Down to Pot', 'Grasp Pot', 'Lift Pot High', 'Move Pot Left to the Sink', 'Lower Pot to Sink', 'Release']. Start with 'Move to'

For a given subtask, we prompt VLM2VLA with the current observation and subtask.

Motion-Planning Prompt

Given the image, reason about what high-level actions the robot arm should take to complete the task "subtask". +dx is forward, -dx is backward, +dy is left, -dy is right, +dz is up, -dz is down, 1 is gripper open, 0 is gripper closed. Provide concise, accurate spatial reasoning.

Once a motion-plan is generated, VLM2VLA is prompted with that reasoning, current observation, and subtask.

Action Generation Prompt

You are programming a 4-DoF robot arm (x,y,z,gripper). +dx is forward, -dx is backward, +dy is left, -dy is right, +dz is up, -dz is down, 1 is gripper open, 0 is gripper closed. Based on the subtask 'subtask' and the reasoning 'motion-plan reasoning', output the sequence of low-level [dx, dy, dz, gripper] actions needed as a python list of lists. Be concise in direction of movement. Example format: [[dx1,dy1,dz1, grip1], [dx2,dy2,dz2, grip2]]. PROVIDE ONLY THE PYTHON LIST.

All outputs from VLM2VLA are generated using nucleus (top-p) sampling with p=0.95. We use distinct temperature settings for each generation stage: 0.1 for motion planning and 0.5 for action prediction. For the initial sub-task decomposition stage, the temperature is set to 0.5 for in-distribution tasks and 1.0 for all other scenarios. The aforementioned hyperparameters were empirically found to work well.

We use the default setting of greedy decoding for both the OpenVLA and ECoT baselines. The task prompts varied slightly between policies to maximize performance of each model on a given task. While all models received near identical commands, we followed the prompt formats suggested by [4, 45] for the baselines, and used our own for VLM2VLA and VLM2VLA-AT (see Appendix C.4).

C.5 Verifier Prompting

To improve robustness of the policy, we operate in closed-loop fashion with a verifier $V : \mathcal{O} \times \mathcal{O} \times \mathcal{L} \times \mathcal{L} \to \mathcal{L}$, where $\mathcal{O}$ is the RGB image space and $\mathcal{L}$ is the language space; at the end of each action-generation cycle, we query the verifier with the observations before ($\bar{o}_i$) and after ($\bar{o}_{i+1}$) action execution ($\bar{a}_i$), as well as the current (l_i) and next (l_{i+1}) subtask. The verifier $V(\bar{o}_i, \bar{o}_{i+1}, l_i, l_{i+1})$ will reason if the current subtask was completed successfully, given the observations and the next subtask. The output from the verifier dictates the subtask used in the next action-generation cycle, which continues until all N subtasks are completed. In this work, we utilize Gemini 2.5 Pro [3] as the verifier, albeit one may train the model itself to do this step.

In this section, we describe the prompting strategies used in this work to query the external verifier if the subtask was successfully completed.

Verifier Prompting Strategy

Inputs:

After having executed action chunk $\bar{a}_i$, Gemini is provided with two observations, $(\bar{o}_i,\ \bar{o}_{i+1})$ and the current and next subtask $(l_i,\ l_{i+t})$.

Verifier Instructions

You are a precision-oriented robot inspector. Your primary job is to evaluate the two images ($\bar{o}_i$, $\bar{o}_{i+1}$) if the 'Current Subtask' (l_i) was executed with enough precision to guarantee the success of the 'Next Subtask' (l_{i+1}).

CRITICAL RULE: Judge the Precondition.

A 'move to' action is ONLY successful if the final position is perfectly aligned for the subsequent action, i.e., executing a grasp as the next step will result in a successful grasp. A 'close enough' position is a FAILURE if it jeopardizes the next step.

For a 'grasp' subtask to succeed, the gripper MUST be centered directly above the object's grasp point in the preceding 'move' step. Any significant offset or misalignment in the 'after' image is a failure.

Provide your analysis in the specified JSON format, following the turn-based examples.

Expected Model Output (JSON)


```
{
  "success": true,
  "confidence": "High",
  "reasoning": "This is a successful grasp because the
  second image clearly shows the robot has successfully
  grasped the eggplant."
}
```


C.6 Inference Latency

We conducted a series of rollouts and measured the wall-clock time for a single inference cycle, which we defined as the generation of both the mid-level reasoning trace and low-level action prediction. All experiments were conducted on an NVIDIA A100 GPU. We collected data across 30 evaluation runs and report the summary statistics in Table 3.

Table 3 Inference Latency Statistics (N=30 trials)

Statistic	**Value**
Median	6.1 [s]
Mean (Average)	10.5 [s]
Standard Deviation	14.3 [s]
Interquartile Range (IQR)	5.0 - 6.7 [s]
Minimum	3.8 [s]
Maximum	48.8 [s]

The median of 6.1 seconds and small interquartile range demonstrate VLM2VLA is capable of fast inference. However, the high standard deviation of 14.3 seconds warrants discussion. Our analysis of the output logs revealed that 1) approximately 10% of cases triggered our retry mechanism due to bad output formatting from our model, causing additional run-time and 2) a small subset of trials exhibited unusually long run-times (> 45 seconds), suggesting a computational bottleneck.

These results highlight the importance of developing fast decoding schemes for low-latency performance in real-world scenarios, which we reserve for future work.

D Training Details

We fine-tune the Gemma-3-12B-IT [1] model using the TRL library [54]. Training is managed with Hugging Face's Accelerate framework, utilizing a DeepSpeed ZeRO Stage 2 configuration for efficient memory usage across multiple GPUs. We employ parameter-efficient fine-tuning (PEFT) using low-rank adaptation (LoRA) to update the model weights.

To obtain the final checkpoint, we train the model on a single node with 4 NVIDIA A100 GPUs. The final model was fine-tuned for one epoch, which we found sufficient for this work. In total, our model required approximately 300 GPU hours. The key hyper-parameters used for this training run are described in Table 4; notably, we found that using a small effective batch size worked well.

Table 4 VLM2VLA Training Hyper-parameters

Hyperparameter	**Value**
Training Strategy	
Base Model	Gemma-3-12B-IT
Frameworks	TRL, Accelerate, DeepSpeed (ZeRO Stage 2)
Fine-Tuning Method	PEFT (LoRA)
Precision	bfloat16 (BF16)
LoRA (PEFT) Configuration	
Rank (r)	16
Alpha (α)	32
Target Modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `down_proj`, `gate_proj`
Optimization	
Optimizer	AdamW
Learning Rate	5e-5
LR Scheduler	Linear Decay
Adam Beta1	0.9
Adam Beta2	0.999
Adam Epsilon	1e-8
Dataloader Configuration	
Global Batch Size	1
Per-Device Batch Size	1
Gradient Accumulation Steps	2
Effective Global Batch Size	8
Max Sequence Length	1024