If You Can Make an Omelette, Can You Crack an Egg? Probing Zero-Shot Subtask Generalization in Vision-Language-Action Models

Grigorii Guz[1,2] **Giuseppe Carenini**[1] **Mathias Lécuyer**[1] **Michiel van de Panne**[1] **Vered Shwartz**[1,2]
[1] University of British Columbia, [2] Vector Institute for AI
{gguz,carenini,van,vshwartz}@cs.ubc.ca, mathias.lecuyer@ubc.ca

Abstract: Recent robotic vision-language-action (VLA) models have shown impressive zero- and few-shot capabilities when deployed in unseen environments and robot morphologies. While natural language is a convenient way to specify tasks, it remains unclear how reliably VLAs can follow previously unseen language instructions after adaptation to new domains. This capability is particularly important for multi-task settings where collecting data and finetuning models for each potential task is impractical. To investigate this, we evaluate how well VLAs finetuned on a set of high-level tasks (place block in drawer, stack blocks) perform on the constituent low-level subtasks (grasp block, lift grasped block), and compare this to models finetuned directly on those subtasks. This evaluation protocol isolates unseen instruction understanding from the model's physical task execution capabilities. We find that even for bigger VLAs, the performance gap between high-level vs. subtask finetuning does not shrink consistently. Overall, our results indicate that beyond model scaling, fine-grained robot data annotation and appropriate data collection protocols are crucial for improving the multi-task capabilities of existing robotic VLA policies[1].

Keywords: vision-language-action models, multi-task learning, language annotation

1 Introduction

Recently introduced robotic foundational vision-language-action models [VLAs; 1, 2, 3] are demonstrating increasingly strong performance on a range of real-world robotics tasks, while requiring less data to adapt to new tasks, environments, or even new robots. They support arbitrary language inputs as task specifications, which is potentially very convenient for scenarios involving many tasks and language-based human interaction. It may also side-step the need for hand-designed planning primitives by instead relying on abstractions encoded by natural language.

However, since natural language is flexible, the same task can be specified at different levels of abstraction with respect to its intermediate steps. Correspondingly, some robotics tasks like pick-and-place and object rearrangement can be decomposed into intermediate subtasks at different levels of granularity. The decision about which level of abstraction (the selection of primitive skills and appropriate data collection protocol) to use for finetuning the VLA in a given setting may impact its downstream performance. If large language and/or vision-language models (LLM, VLMs) planners are used to generate free-form natural language task decompositions, the generated intermediate steps may differ in their abstraction levels from those selected for VLA fine-tuning. One way to address this is to constrain such planners to the pre-selected primitives [4, 5]. However, in realistic human-robot interaction, users might issue corrections or suggestions [6, 7] corresponding to constituent subtasks of the primitives themselves. For VLAs to be reliable in such settings, it is de-

[1]The code is available here: https://github.com/grig-guz/vla_subtask_generalization.

sirable for them to implicitly encode the temporal compositional structure of tasks they were trained to perform, as reflected by the ability to recognize the language instructions for constituent subtasks. Given that VLAs were already exposed to a multitude of skills and corresponding language instructions during large-scale pretraining [8], while also employing LLMs or VLMs for processing language instructions [1, 9, 3], it may be the case that they already encode such compositional structure.

We test these abilities in simulation, leveraging the datasets from CALVIN and LIBERO, which are two popular multi-task language-conditioned robotic manipulation benchmarks [10, 11]. From those, we selected a subset of tasks (which we treat as HL) that can be decomposed into sequences of 2-4 lower-level (LL) tasks. We finetune a selection of recently released VLAs with HL task demonstrations and compare their performance on the corresponding constituent LL tasks relative to directly finetuning the VLAs on the LL tasks. We analyze how the relative performance is affected by factors such as model scale, the amount of (implicit) LL skill demonstrations within the HL fine-tuning dataset, as well as specificity of language labels. Since the demonstrations for LL tasks are contained within the HL task trajectories, our setup isolates language understanding from control/task execution capabilities of VLAs.

Our results suggest that in terms of constituent LL task performance, finetuning VLAs on HL tasks leads to a substantial performance gap compared to fine-tuning them directly on LL task trajectories, though this gap tends to shrink with increased model scale. A closer analysis reveals that the transfer performance depends on the specific LL task, and even then, sufficient state space coverage is required for VLAs to execute the specified task (as opposed to other instruction-irrelevant tasks). Hence, robust contextual language understanding does not emerge "for free" even with introducing specialized language processing components (e.g. LLMs or VLMs) into VLA architecture for processing language instructions. As a consequence, handling unseen free-form natural language instructions or feedback requires additional targeted supervision and appropriate data collection methods.

2 Related Work

2.1 Robotic Vision-Language-Action Models

The emergence of robotic VLAs is motivated by the observation that large-scale pretraining was shown to improve all-around model capabilities in many fields, most notably computer vision and natural language processing. A recent trend is to aggregate datasets for numerous tasks, task specification modalities, and robot types. Such projects, like Open X-Embodiment [OXE; 8], enable large-scale pretraining of generalist robot policies.

In this work, we employ a subset of recent models of this type, including Octo [1], π_0-FAST [9] and CogACT [3] (see more details in Section 4.1 and Table 10). Due to the high diversity and abundance of pretraining data, these models have (to a varying extent) demonstrated interesting capabilities such as transfer of manipulation skills to unseen objects and robot types, either via finetuning on a small number of examples or completely zero-shot. In this work, we focus on a particular type of generalization - compositional generalization, testing whether the performance of VLAs transfers from high-level tasks (such as "place block in drawer", "stack blocks") to their constituent low-level tasks (such as "grasp block", "lift grasped block"), as well as the effects of language-task labeling specificity.

2.2 Testing Language Understanding in VLAs

Many recent studies in robotic manipulation domain employ language for task specifications and evaluate the robustness of VLA to diverse attributes such as object appearance, lighting conditions and presence of distractor objects [12, 13], as well as generalization of manipulation skills to novel scenes and objects [14, 15]. Here, we review works emphasizing language as the main factor of variation and evaluating several desired aspects of language instruction following behavior in VLAs.

Figure 1: (Left) A frame from the CALVIN environment, scene D. (Right) A frame from the LIBERO environment, kitchen scene.

Notably, Gao et al. [16] organized many of the aforementioned factors as well as their interactions into a single conceptual framework.

First, due to linguistic variability across different people (lexical choice, grammatical structure, etc.), one attractive property is robustness to semantically equivalent paraphrases of seen instructions. The study by Parekh et al. [17] tested how well the VLAs generalize at test time to paraphrases of language instructions observed during training. They find that LLMs within VLA architectures enable robust generalization to instruction paraphrases for in-distribution tasks, and that augmenting the training set annotations with paraphrases improves the success rate on unseen longer-horizon tasks constituted by previously seen tasks. Besides the differences in the target VLA abilities, they focus on the impact of different neural architecture components on generalization. Hence, they train the VLAs from scratch, without considering the effect of large-scale pretraining, as we do here.

Another useful capability is generalization to language instructions specifying unseen long-horizon tasks with intermediate steps consisting of previously seen tasks. Parekh et al. [17], Haresh et al. [18] test this by finetuning VLAs on a set of lower-level tasks and evaluating on progressively longer-horizon tasks. They find that the VLAs' performance degrades with increase in task horizon length. Another line of work proposes test-time adaptation [4] and data annotation [19, 20] techniques for improving long-horizon task generalization.

Like [17, 18], we evaluate how well VLAs generalize to new tasks specified with language instructions. However, we focus on effects of paraphrasing and fine-grained labeling on higher-to-lower level task generalization, which to our knowledge was not explored before for VLAs. Further, we utilize human teleoperation data instead of data from scripted policies [17, 18], which we elaborate on in §3.1.

3 Datasets and Evaluation

We use the CALVIN and LIBERO benchmarks for our experiments (§3.1). For each environment, we define a new task inventory consisting of original high-level (HL) tasks along with our decomposition into low-level (LL) tasks (§3.2). We relabeled the datasets according to task structure (§3.3). Finally, we modified the original evaluation protocols in each benchmark to enable more fine-grained error analysis (§3.4).

3.1 The Benchmarks

CALVIN [10] and LIBERO [11] are multi-task robotic manipulation benchmarks which can support evaluation of natural language understanding capabilities of robotic policies. The tasks included in these benchmarks involve rearrangement of rigid objects and manipulation of simple articulated objects. We selected these benchmarks for two reasons. First, the provided task types are appropriate for our analysis since they are straightforward to decompose into intermediate subtasks which themselves can be labeled with language instructions. Second, these benchmarks contain sufficient amounts of multi-task language-annotated robot data collected from humans. Since the

VLAs that we experiment with (§4.1) were pretrained largely or entirely on human teleoperation data [8, 21, 22], we assumed that LL task transfer will be more consistent if the HL trajectories are also from humans, hence closer to the pre-training distribution.

3.2 Task Decomposition

For our experiments, we selected a collection of tasks that can be treated as high-level and another set of tasks to be low-level. To guide our choice, we assumed the main structural relation between the HL and LL tasks to be that 1) the instruction for each HL and LL task can be expressed as a single-clause English sentence, and that 2) each HL task instruction can also be expressed as a semantically equivalent (in the given scene setup) multiple-clause English sentence, where each clause is itself an instruction for some LL task. For example, the HL task "heat up the frying pan on the stove" can be expressed as "turn on the stove and put the frying pan on the stove", where each clause in the latter is an instruction for a LL task.

For CALVIN, we created these sets by selecting a subset of original CALVIN's tasks (treated as HL) and implementing a few additional LL tasks to decompose the selected HL ones. See Figure 3 for the dataset trajectory distribution for the resulting HL and LL tasks, Tables 4, 5 for pre/post-conditions for HL and LL tasks respectively, Table 6 for specification of the exact decomposition of HL tasks into LL tasks and Table 7 for language instructions.

For LIBERO, we selected a subset of tasks from the original LIBERO-90 and LIBERO-10 splits for which the success condition included at least two predicates (e.g. the "heat up the frying pan on the stove" task from above). Those tasks were treated as HL and tasks involving individual predicates as LL. See Table C.1 for resulting tasks and Table C.2 for corresponding language instructions.

3.3 Data Collection and Task Labelling

The dataset from the CALVIN benchmark was assembled with "play data" collection protocol, meaning that the humans teleoperating the robot were performing arbitrary self-initiated tasks in a single episode for an extended period of time. To annotate the resulting data with language instructions, the original automatic data labeling procedure in CALVIN slides 2-second windows over the recorded dataset, and assigns language labels to those windows based on the task evaluator logic (more details in §3.4). We extend this procedure to annotate each window based on both HL and LL task criteria. For a given window, we check that both the HL task and corresponding LL task sequence (see Table 6) are completed according to pre-conditions and post-conditions for those tasks (see Tables 4 and 5). That way, each demonstrations in the resulting dataset is annotated with 1) a single HL task instruction 2) LL task segment boundaries and corresponding instructions. Overall, we collect 75 trajectories per HL task, for a total of 1,350 trajectories. The resulting HL and LL task trajectory counts are shown on Figure 3.

For LIBERO, the authors relied on the more standard episodic data collection method, where the human teleoperating the robot was told which task to perform in each episode. To segment the trajectories for the selected tasks, we relied on the original LIBERO task evaluators. We iterate over every timestep in a HL task trajectory, and once one of the HL task's predicates (§3.2) is satisfied, we label the segment with the corresponding LL instruction and treat the next timestep as the beginning of a LL new trajectory. This way, each HL trajectory is split into multiple LL segments. The resulting HL dataset contains 6 tasks with up to 50 trajectories per task, for a total of 234 trajectories (the original LIBERO dataset contains 50 trajectories per task, but some of them did not satisfy the benchmark's task checkers), while the LL dataset contains 9 tasks and 468 trajectories. The trajectory counts are shown on Figure 4.

3.4 Evaluation Protocols

During the episode, the language instruction $l \in \mathcal{L}$ for the current task and $o_t \in \mathcal{O} = \mathbb{R}^{H \times W \times 3}$ RGB camera observation are provided to the robot at every timestep t. In response, a policy π :

$\mathcal{O} \times \mathcal{L} \rightarrow P(\mathcal{A})$ infers a continuous action $a_t \in \mathcal{A} = \mathbb{R}^7$. This proceeds until the robot completes the given task, or a timeout is reached.

For CALVIN, we perform evaluation with 5 tasks per episode. In the beginning of each episode, we sample the initial state and a physically consistent random sequence of 5 tasks. The policy receives instructions for one task at a time - once it is completed, the environment provides the instruction for the next task. As the main evaluation metric, in §5.1 we report the average number of tasks completed per episode (out of 5). We use this setup since it was the original evaluation protocol in CALVIN, as well as because some of the tasks require certain preconditions to be satisfied beforehand. For example, "lift the grasped block" task requires the robot to grasp one of the blocks first, so this task is always preceded by e.g. "grasp the red block" task in the evaluation sequence. The pre-conditions and post-conditions for each task are outlined in the Appendix B.1.

For LIBERO, the selected tasks can be executed independently from one another, so we follow the standard evaluation protocol with one task per episode and report the success rates per task and across tasks in §5.2.

New evaluation protocol. The original task evaluators in both CALVIN and LIBERO benchmarks consider it a success if the robot eventually completed the target task, *even if it performed other tasks first*. This can undesirably reward a brute force policy, for example rewarding a robot for completing the task "grasp the red block", by grasping every block in the environment. To address this, we modify the task evaluators to terminate an episode if the robot completes an irrelevant task (to the one currently given).

In our analysis (§5), we use both evaluator versions to establish whether the task due to misunderstanding of a given instruction that leads to completing an irrelevant task vs the failure to complete any task at all resulting in timeout. We refer to the corresponding evaluation protocols as **Hard** evaluation vs **Easy** evaluation.

4 Experimental Setup

4.1 Models

We used the Octo-Small, Octo-Base [1], π_0-FAST [9] and CogAct [3] VLAs for evaluation. This choice covers a variety of model parameter sizes (from 27M to 7.6B), to investigate the effects of model scale. Further, to explore the effects of large-scale robotics data pretraining, we also finetune a version of Octo-Base with random initialization for all parameters except T5 language encoder. Most VLAs follow the transformer-based encoder-decoder architecture, where the input images and language instruction are encoded into a sequence of tokens in the action decoder's input space[2] (see Table 10). A more detailed overview of each model is in Appendix D.2.

4.2 Finetuning

For all models, we perform full model finetuning for up to 50k iterations, using only language instructions as goal specifications. The finetuned parameters include image encoders, multimodal decoders and action heads, for all models considered. We perform validation after every 5k iterations by evaluating the models on a small number of episodes with the **Easy** evaluation setting, and select the best-performing model for subsequent analysis. See Appendix D for details on preprocessing and hyperparameters.

4.3 Evaluation

As in the original CALVIN benchmark, we evaluate the models' ability to execute random sequences of tasks of length 5. We use 1,000 episodes for all evaluations. As metrics, we report the average

[2]To our knowledge, there are no VLAs that feature any architectural inductive biases to facilitate compositional language understanding, so this could not be considered as a factor in the selection of VLAs.

Model	LL ft → LL instr	HL ft → LL instr	Conj ft → LL instr
Easy evaluation			
Octo-B (R)	2.3 ± 0.8	-1.12 ± 0.07	-0.77 ± 0.09
Octo-S	3.7 ± 0.22	-2.07 ± 0.14	-1.62 ± 0.28
Octo-B	3.77 ± 0.13	-2.04 ± 0.05	-1.82 ± 0.14
Pi-0 Fast	3.71 ± 0.08	-1.12 ± 0.05	-0.73 ± 0.1
CogAct	3.5 ± 0.13	-0.78 ± 0.23	-0.38 ± 0.24
Hard evaluation			
Octo-B (R)	2.17 ± 0.72	-1.33 ± 0.06	-1.06 ± 0.11
Octo-S	3.48 ± 0.17	-2.04 ± 0.1	-1.65 ± 0.27
Octo-B	3.56 ± 0.16	-2.04 ± 0.02	-1.84 ± 0.14
Pi-0 Fast	3.43 ± 0.13	-1.4 ± 0.07	-1.04 ± 0.19
CogAct	3.3 ± 0.11	-1.13 ± 0.14	-0.91 ± 0.09

Table 1: LL task performance across different finetuning settings on CALVIN. The first column - average number of tasks completed per episode by LL finetuned models (out of 5). The remaining columns - transfer gaps between LL and that column's finetuning setting. Top - **Easy** evaluation setting - an episode continues until the target task is completed or the timeout is achieved. Bottom - **Hard** evaluation setting - an episode terminates if the model completes a non-target task.

number of tasks that the robot completes per episode (out of 5), as well as per-task success rates. For compactness, we arrange semantically similar tasks into groups (see Appendix D.1) and report average per-group success rates.

For LIBERO, we execute one task per episode and evaluate each task for 50 episodes. We report the success rates for individual tasks and average success rate across all tasks.

5 Experiments

Our analysis aims to uncover if VLAs trained on a set of HL tasks can also perform the constituent LL tasks in a zero-shot manner. We finetune all models on the same observation-action data but with 3 different types of language annotations. `HL finetuning` is the base case; it refers to training on trajectories and instructions corresponding to HL tasks. `Conj finetuning` setup uses the same trajectories as in HL finetuning, but with replacing the HL instructions by single "recipe" consisting of the corresponding conjunctions of LL instructions ("rotate red block left" → "grasp red block, then rotate the grasped block left, then ungrasp the block"). For both settings, we perform validation (§4.2) with HL tasks but with same instruction types as used for finetuning. Finally, for `LL finetuning` we use LL task trajectories (segments of HL trajectories) with corresponding LL task instructions and use LL tasks for validation.

5.1 Results for CALVIN Environment

Average number of completed tasks. Table 1 compares the average number of LL tasks completed for both evaluation settings (see §3.4). First, all models (except the randomly initialized Octo-B (R)) perform similarly when finetuned with LL data annotation, but there is a drop in performance when finetuning with HL or Conj annotations. However, the performance gap tends to shrink for the bigger models, especially CogAct. Second, all models (except the randomly initialized Octo) exhibit same or better LL task transfer when finetuned with Conj instructions than HL instructions. This is likely due to to the fact that `Conj finetuning` models were exposed to the same vocabulary as present in LL task instructions. However, the performance gap with `LL finetuning` is still sizable, meaning that fine-grained annotation does fully not alleviate the need for trajectory segmentation. Third, compared to **Hard** evaluation setting, the **Easy** evaluation settings shows higher absolute LL task performance and, for the bigger models, smaller performance gap between different finetuning regimes. Thus, a sizable proportion of failures occur due to execution of irrelevant tasks.

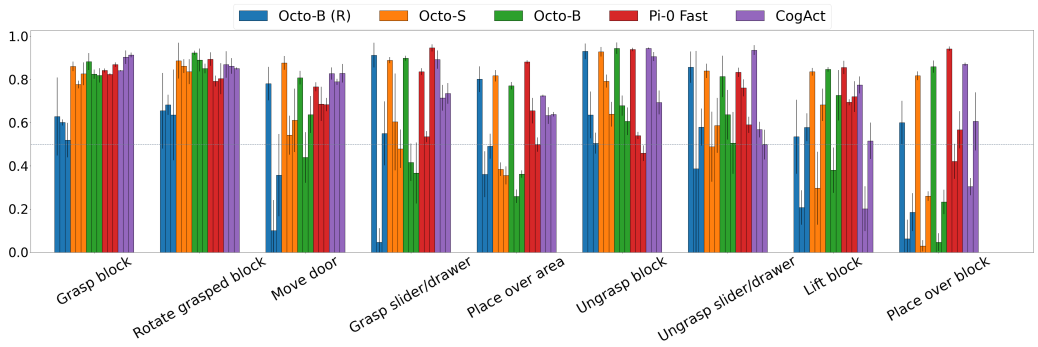

Figure 2: Success rates for individual LL tasks from CALVIN, **Hard** evaluation setting. Each bar group corresponds to a single LL task. Within each group, the three bars for each model (same color) correspond to LL, HL and Conj finetuning (same order as columns in Table 1).

Task-Specific Performance. Success rates for individual CALVIN tasks are shown in Figure 2. There, the bigger models are more likely to have similar per-task performance across all three finetuning configurations, showing more robustness to changes in data annotation. However, the transfer gap is very pronounced for the last five tasks. Two possible reasons for models' decreased performance when changing the data annotation style are 1) lack of data, or 2) these tasks cannot be learned from longer trajectories without segmentation. To disentangle these factors, we employ a bootstrap-like procedure. Due to resource limitations, we only perform this experiment with the Octo-S model.

Let $\mathcal{T}_{LL}$ denote the set of all LL tasks. We repeatedly subsample the finetuning dataset with full HL trajectories. With each subsample, we 1) finetune the model on those HL trajectories (without LL segmentation) and evaluate the per-task performance with **Hard** setting, and 2) record the number of unique implicit demonstrations for each LL task within those HL trajectories in each subsample. With LL task counts and LL task performance scores across many subsamples, for each LL task $\tau \in \mathcal{T}_{LL}$, we apply linear regression $sr_\tau = \beta_{intercept} + \beta_\tau c_\tau$ with that task's implicit LL segment counts c_τ as features and predict the success rate sr_τ on that LL task. Then, we examine the resulting regression coefficients - if for the target task τ we have $\beta_\tau > 0$, then the model is indeed learning the LL task from unsegmented demonstrations. We perform this subsampling-finetuning procedure with `HL` and `Conj finetuning` settings and report the p-values from the t-test for $\beta_\tau > 0$.

Task	**HL ft**	**Conj ft**
Grasp block	0.05	0.0
Rotate block	0.0	0.0
Move door	0.15	0.73
Grasp slider/drawer	0.22	0.74
Place over area	0.11	0.98
Ungrasp block	0.32	1.0
Ungrasp slider/drawer	0.99	0.53
Lift block	0.39	0.0
Place over block	0.99	0.0

Table 2: Regressing the success rate sr_τ of each LL task τ on counts c_τ of its implicit demonstrations in the bootstrap samples of the HL task trajectories. The entries are p-values for $\beta_\tau > 0$.

The results in Table 2 show the success rates for "Grasp block" and "Rotate block" improve significantly with more implicit demonstrations across both finetuning settings. As seen from Figure 2, for these two tasks the performance is most similar across all finetuning settings. However, for 'Lift block" and "Place over block" tasks, we see that the extra implicit demonstrations are only useful with `Conj finetuning` setting, which is also reflected in the large gap between these settings in Figure 2. For all cases, the results show no significant benefit from access to more implicit LL demonstrations.

Model	LL ft → LL instr	HL ft → LL instr	Conj ft → LL instr
Easy evaluation			
Octo-B (R)	0.38 ± 0.01	$+0.13 \pm 0.07$	$+0.09 \pm 0.06$
Octo-S	0.51 ± 0.02	$+0.14 \pm 0.03$	$+0.18 \pm 0.04$
Octo-B	0.53 ± 0.00	$+0.09 \pm 0.06$	$+0.07 \pm 0.05$
Pi-0 Fast	0.47 ± 0.01	$+0.10 \pm 0.06$	$+0.09 \pm 0.04$
CogAct	0.83 ± 0.04	-0.03 ± 0.01	-0.02 ± 0.07
Hard evaluation			
Octo-B (R)	0.35 ± 0.01	-0.10 ± 0.02	-0.13 ± 0.02
Octo-S	0.32 ± 0.01	-0.06 ± 0.02	-0.05 ± 0.02
Octo-B	0.28 ± 0.04	-0.04 ± 0.01	-0.04 ± 0.03
Pi-0 Fast	0.30 ± 0.05	-0.06 ± 0.01	-0.07 ± 0.01
CogAct	0.35 ± 0.03	-0.04 ± 0.02	-0.04 ± 0.03

Table 3: Average LL task success rates across different finetuning settings on LIBERO. The first column - average task success rate. The remaining columns - transfer gaps between LL and that column's finetuning setting. Top - **Easy** evaluation setting - an episode continues until the target task is completed or the timeout is achieved. Bottom - **Hard** evaluation setting - an episode terminates if the model completes a non-target task.

5.2 Results for LIBERO Environment

Average success rate across tasks. The results are shown in Table 3. First, we consider the **Easy** evaluation setting. In comparison to CALVIN, there is now a difference across different VLAs on `LL finetuning`, which can be attributed to smaller finetuning dataset size. Further, `LL finetuning` does not lead in LL task performance - other finetuning configurations have similar success rates. To investigate this, consider the results for **Hard** evaluation setting. Notice that the success rates dropped severely compared to the **Easy** evaluation, and that there is now no performance difference between the VLAs. Such a drastic difference is due to bias in the LIBERO dataset. For example, for the HL task "turn on stove and put frying pan on it", all demonstrations start with the robot turning on the stove and then moving the pan. All other HL LIBERO tasks have the same bias in LL task ordering within their demonstrations. As a result, we found that all VLAs complete the preceding tasks first, independent of the current instruction. This pattern can be seen by comparing per-task success rates for **Easy** (Figure 5) and **Hard** (Figure 6) settings - for some tasks the performance is identical across both settings, for others the success rates are near zero for **Hard** evaluation.

This issue is not present for CALVIN as it used a different data collection protocol. The CALVIN dataset was built with play data collection method, where the data is collected in one long episode such that human can begin executing any task at any moment. Hence, each task has more uniform coverage of the environment's state space. On the other hand, the demonstrations for LIBERO environment were collected episodically, and due to LL task order bias in HL task trajectories, the initial states of each episode were only covered by data for a single LL task. This can result in unexpected VLA policy behavior even when learning tasks from gold standard trajectory segmentations.

6 Conclusion

We investigated the low-level subtask transfer capabilities of a set of recently released VLAs. While we find evidence that large-scale pretraining of these models and model size indeed improve the performance, the results were not consistent across all tasks, language annotation schemes and data collection regimes. This suggests that current VLAs do not implicitly encode the compositional task structure during finetuning. As such, robust deployment in settings with flexible language-based user interaction may require specialized data collection and annotation protocols balancing multiple levels of task abstraction.

References

[1] D. Ghosh, H. R. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, Q. Vuong, T. Xiao, P. R. Sanketi, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024. URL https://doi.org/10.15607/RSS.2024.XX.090.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

[3] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, X. Wang, B. Liu, J. Fu, J. Bao, D. Chen, Y. Shi, J. Yang, and B. Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024. URL https://arxiv.org/abs/2411.19650.

[4] V. Myers, C. Zheng, O. Mees, K. Fang, and S. Levine. Policy adaptation via language optimization: Decomposing tasks for few-shot imitation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=qUSa3F79am.

[5] L. X. Shi, Z. Hu, T. Z. Zhao, A. Sharma, K. Pertsch, J. Luo, S. Levine, and C. Finn. Yell at your robot: Improving on-the-fly from language corrections. In *Robotics: Science and Systems*, 2024. URL https://doi.org/10.15607/RSS.2024.XX.025.

[6] H. Liu, A. Chen, Y. Zhu, A. Swaminathan, A. Kolobov, and C.-A. Cheng. Interactive robot learning from verbal correction, 2023.

[7] L. X. Shi, brian ichter, M. R. Equi, L. Ke, K. Pertsch, Q. Vuong, J. Tanner, A. Walling, H. Wang, N. Fusai, A. Li-Bell, D. Driess, L. Groom, S. Levine, and C. Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=lNVHg9npif.

[8] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. Ben Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. Di Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano,

P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

[9] 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, 2025. URL https://arxiv.org/abs/2501.09747.

[10] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[11] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=xzEtNSuDJk.

[12] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3153–3160, 2024. doi:10.1109/ICRA57147.2024.10611331.

[13] Z. Wang, Z. Zhou, J. Song, Y. Huang, Z. Shu, and L. Ma. Towards testing and evaluating vision-language-action models for robotic manipulation: An empirical study, 2024. URL https://arxiv.org/abs/2409.12894.

[14] J. Gao, A. Xie, T. Xiao, C. Finn, and D. Sadigh. Efficient data collection for robotic manipulation via compositional generalization. In *RSS 2024 Workshop: Data Generation for Robotics*, 2024. URL https://openreview.net/forum?id=PqnSt9Hwyv.

[15] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pISLZG7ktL.

[16] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh. A taxonomy for evaluating generalist robot policies, 2025. URL https://arxiv.org/abs/2503.01238.

[17] A. Parekh, N. Vitsakis, A. Suglia, and I. Konstas. Investigating the role of instruction variety and task difficulty in robotic manipulation tasks. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19389–19424, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi:10.18653/v1/2024.emnlp-main.1080. URL https://aclanthology.org/2024.emnlp-main.1080/.

[18] S. Haresh, D. Dijkman, A. Bhattacharyya, and R. Memisevic. Clevrskills: Compositional language and visual reasoning in robotics. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=64sZtFSOh6.

[19] N. Blank, M. Reuss, M. Rühle, Ö. E. Yağmurlu, F. Wenzel, O. Mees, and R. Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=EdVNB2kHv1.

[20] L. Smith, A. Irpan, M. G. Arenas, S. Kirmani, D. Kalashnikov, D. Shah, and T. Xiao. Steer: Flexible robotic manipulation via dense language grounding, 2024. URL https://arxiv.org/abs/2411.03409.

[21] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2024. URL https://arxiv.org/abs/2403.12945.

[22] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[23] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), Jan. 2020. ISSN 1532-4435.

[24] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

[25] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, T. Unterthiner, D. Keysers, S. Koppula, F. Liu, A. Grycner, A. Gritsenko, N. Houlsby, M. Kumar, K. Rong, J. Eisenschlos, R. Kabra, M. Bauer, M. Bošnjak, X. Chen, M. Minderer, P. Voigtlaender, I. Bica, I. Balazevic, J. Puigcerver, P. Papalampidi, O. Henaff, X. Xiong, R. Soricut, J. Harmsen, and X. Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

[26] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

[27] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and

K. Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

[28] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[29] 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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yAzN4tz7oI.

A Additional Figures

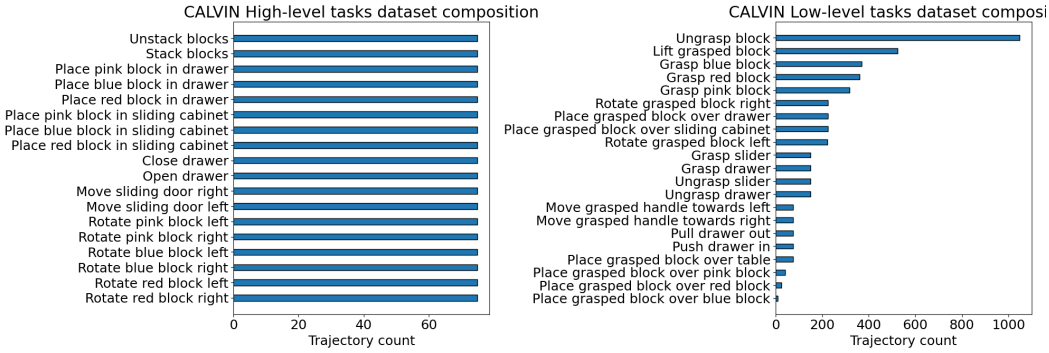

Figure 3: Per-task counts of trajectories in the CALVIN dataset, HL vs. LL labelling.

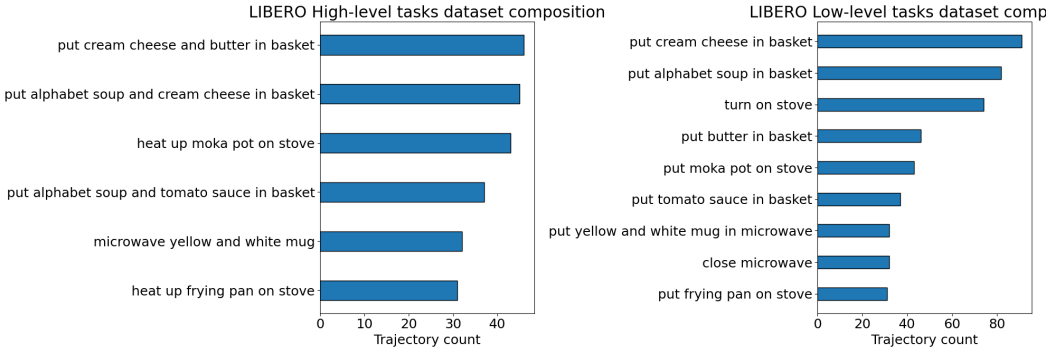

Figure 4: Per-task counts of trajectories in the LIBERO dataset, HL vs. LL labelling.

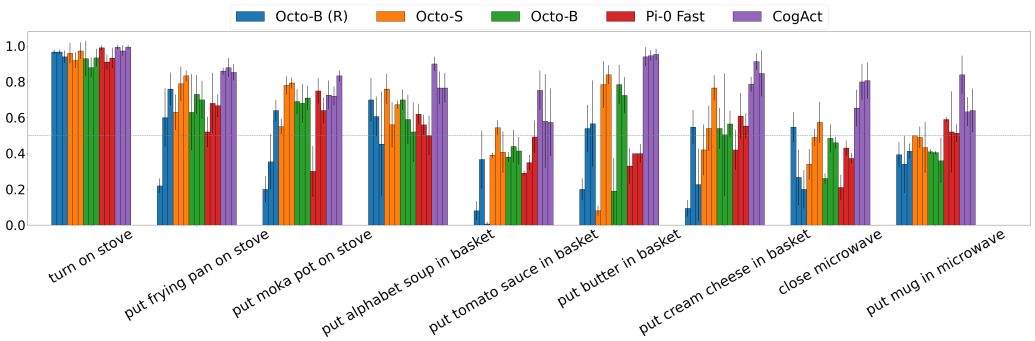

Figure 5: Success rates for individual LL tasks from LIBERO, **Easy** evaluation setting. Each bar group corresponds to a single LL task. Within each group, the three bars for each model (same color) correspond to LL, HL and Conj finetuning (same order as columns in Table 3).

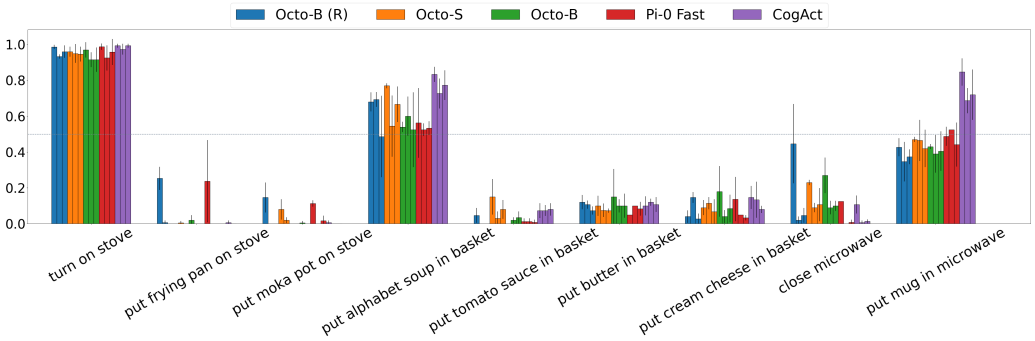

Figure 6: Success rates for individual LL tasks from LIBERO, **Hard** evaluation setting. Each bar group corresponds to a single LL task. Within each group, the three bars for each model (same color) correspond to LL, HL and Conj finetuning (same order as columns in Table 3).

B CALVIN Task specifications

B.1 Task pre- and post-conditions

Task name	Preconditions	Postconditions
Rotate the X left/right	$Robot.contacts = \emptyset$ All blocks are stationary	$Robot.contacts = \emptyset$ All blocks are stationary X rotated by D degrees
Move slider door left/right	$Robot.contacts = \emptyset$ All blocks are stationary Door is in opposite position	$Robot.contacts = \emptyset$ All blocks are stationary Slider door was moved left/right
Open/close the drawer	$Robot.contacts = \emptyset$ All blocks are stationary Drawer is in opposite position	$Robot.contacts = \emptyset$ All blocks are stationary Drawer was opened/closed
Place X in Y	$Robot.contacts = \emptyset$ All blocks are stationary X not in Y	$Robot.contacts = \emptyset$ All blocks are stationary X in Y
Stack blocks	$Robot.contacts = \emptyset$ All blocks are stationary X_1 block is not on X_2 block	$Robot.contacts = \emptyset$ All blocks are stationary X_1 block is on X_2 block
Unstack blocks	$Robot.contacts = \emptyset$ All blocks are stationary X_1 block is on X_2 block	$Robot.contacts = \emptyset$ All blocks are stationary X_1 block is not on X_2 block

Table 4: HL task semantics. $X \in \{$ red block, blue block, pink block$\}$. $Y \in \{$drawer, sliding cabinet$\}$.

Task name	Preconditions	Postconditions
Grasp the X	$Robot.contacts = \emptyset$ All blocks are stationary	All other blocks are stationary. $\lvert X.contacts \cap Robot.contacts \rvert \geq 2$ $\lvert Robot.contacts \setminus X.contacts \rvert = 0$ X is on the ground.
Grasp the slider door	$Robot.contacts = \emptyset$	$\lvert X.contacts \cap Slider.contacts \rvert \geq 2$ Slider door not moved too much
Touch the drawer	$Robot.contacts = \emptyset$	$\lvert X.contacts \cap Drawer.contacts \rvert \geq 1$ Drawer not moved too much
Ungrasp the block	Some X is grasped All other blocks are stationary	$Robot.contacts = \emptyset$ All blocks are stationary X not moved too much in x, y dir X not rotated too much around z axis
Ungrasp the Y	Robot is touching/grasping Y	$Robot.contacts = \emptyset$ Y was not moved too much
Place grasped block over X_2/table/Y	Some X_1 is grasped and lifted $X_1 \neq X_2$ X_1 is not over target surface All other blocks are stationary	Same X_1 is grasped and lifted X_1 is over target surface All other blocks are stationary
Rotate the grasped block left/right	Some X is grasped All other blocks are stationary	Same X is grasped X rotated around z axis by D
Move the slider door left/right	Robot is grasping the slider door	Robot is grasping the slider door Slider door moved by D left/right
Pull/push the drawer	Robot is touching the drawer	Robot is touching the drawer Drawer was pushed/pulled by D
Lift the grasped block	Some X is grasped All other blocks are stationary. X is on the ground	Same X is grasped All other blocks are stationary. X is lifted by D.

Table 5: LL task semantics. $X \in \{$ red block, blue block, pink block$\}$. $Y \in \{$drawer, sliding cabinet$\}$.

B.2 HL $\rightarrow$ LL task decomposition

Task name	LL tasks
Rotate the X block left/right	Grasp the X block, Rotate the grasped block left/right, ungrasp the block
Move the slider door left/right	Grasp the slider, move the slider door left/right, ungrasp the slider
Open the drawer	Grasp the drawer, Pull the drawer, Ungrasp the drawer
Close the drawer	Grasp the drawer, push the drawer, Ungrasp the drawer
Place grasped block in sliding cabinet/drawer	Place the gripper over sliding cabinet/drawer, ungrasp the block
Stack block	Grasp the X_1 block, lift the grasped block, place gripper over X_2 block, ungrasp the block
Unstack block	Grasp the X_1 block, lift the grasped block, place gripper over table, ungrasp block

Table 6: HL task decompositions in terms of LL tasks. $X \in \{\text{red}, \text{blue}, \text{pink}\}$.

B.3 Task prompts

Task name	Prompt
Grasp the X block	grasp the X block
Grasp the drawer	grasp the drawer handle
Grasp the slider	grasp the slider handle
Ungrasp the block	ungrasp the block
Ungrasp the drawer hanle	ungrasp the drawer handle
Ungrasp the slider handle	ungrasp the slider handle
Place the grasped block over the X block/table/drawer/slider	place the grasped block over X block/ table surface/drawer/slider
Rotate the grasped block left/right	rotate the grasped block 90 degrees to the left/right
Move the slider door left/right	move the grasped handle towards the left/right
Pull/push the drawer	pull the grasped handle out/push the grasped handle in
Lift the grasped block	lift the grasped block
Rotate the X block left/right	rotate the X block 90 degrees to the left/right
Move the slider door left/right	move the sliding door to the left/right
Open the drawer	open the drawer
Close the drawer	close the drawer
Place grasped block in sliding cabinet	store the grasped block in the sliding cabinet
Place grasped block in drawer	store the grasped block in the drawer
Stack block	stack blocks on top of each other
Unstack block	remove the stacked block

Table 7: Prompts for LL tasks (first group) and HL tasks (second group). $X \in \{\text{red}, \text{blue}, \text{pink}\}$.

C LIBERO Task specifications

C.1 Task decompositions

HL Task name	LL tasks
Turn on the stove and put the frying pan on it	Turn on the stove, Put the frying pan on the stove
Turn on the stove and put the moka pot on it	Turn on the stove, Put the moka pot on the stove
Put the yellow and white mug in the microwave and close it	Put the yellow and white mug in the microwave, Close the microwave
Put both the alphabet soup and the cream cheese box in the basket	Put the alphabet soup in the basket, Put the cream cheese box in the basket
Put both the alphabet soup and the tomato sauce in the basket	Put the alphabet soup in the basket, Put the tomato sauce in the basket
Put both the cream cheese box and the butter in the basket	Put the cream cheese box in the basket, Put the butter in the basket

Table 8: HL task decompositions in terms of LL tasks.

C.2 Task prompts

Task name	Prompt
Put the frying pan on stove	put the frying pan on the stove
Put the moka pot on the stove	put the moka pot on the stove
Turn on the stove	turn on the stove
Close the microwave	close the microwave
Put the yellow and white mug in the microwave	put the yellow and white mug in the microwave
Put the alphabet soup in the basket	put the alphabet soup in the basket
Put the cream cheese box in the basket	put the cream cheese box in the basket
Put the butter in the basket	put the butter in the basket
Put the tomato sauce in the basket	put the tomato sauce in the basket
Turn on the stove and put the frying pan on it	heat up the frying pan on the stove
Turn on the stove and put the moka pot on it	heat up the moka put on the stove
Put the yellow and white mug in the microwave and close it	microwave the yellow and white mug
Put both the alphabet soup and the cream cheese box in the basket	put both the alphabet soup and the cream cheese box in the basket
Put both the alphabet soup and the tomato sauce in the basket	put both the alphabet soup and the tomato sauce in the basket
Put both the cream cheese box and the butter in the basket	put both the cream cheese box and the butter in the basket

Table 9: Prompts for LL tasks (first group) and HL tasks (second group).

D Additional experiment details

D.1 CALVIN Task grouping

When reporting the results, we grouped the tasks in the following way:

HL tasks. Rotate block = [rotate red block left, rotate red block right, rotate blue block left, rotate blue block right, rotate pink block left, rotate pink block right]. Slider/drawer = [open drawer, close drawer, move slider left, move slider right]. Place block = [place red block into drawer, place blue block into drawer, place pink block into drawer, place red block into slider, place blue block into slider, place pink block into slider]. Stack/Unstack blocks = [stack blocks, unstack blocks].

LL tasks. Grasp block = [grasp red block, grasp blue block, grasp pink block]. Rotate block = [rotate grasped block left, rotate grasped block right]. Move door = [move grasped slider left, move grasped slider right, pull the drawer, push the drawer]. Grasp slider/drawer=[grasp slider, grasp drawer]. Place over area = [place grasped block over table, place grasped block over slider, place grasped block over drawer]. Ungrasp block = [ungrasp block]. Ungrasp slider/drawer = [Ungrasp slider, ungrasp drawer]. Lift block = [lift grasped block]. Place over block = [place grasped block over red block, place grasped block over blue block, place grasped block pink red block].

D.2 Overview of used models

Model	Pretraining dataset	Image encoder	Language encoder	Action decoder	Action model
Octo 27, 93M	800k OXE	CNN	T5-base (frozen)	Transformer	Diffusion
π_0-FAST 3.6B	Proprietary + OXE	PaliGemma-3B	PaliGemma-3B	PaliGemma-3B	Categorical (FAST tokenization)
CogAct 7.6B	400k OXE	DinoV2 + SigLIP-400M	Llama-2 tokenizer	Llama2	Diffusion

Table 10: Overview of the VLAs used in our experiments.

Octo. Octo [1] comes with 27 million (Octo-Small) and 93 million (Octo-Base) parameter versions, making it the smallest of the OXE-pretrained models, from which 800k trajectories were used for pre-training. The language inputs are processed with the T5-base language model [23], while the image observations and goals are passed through convolutional networks and then flattened into image patch-based vectors. The resulting sequence of language and image patch embeddings and an extra readout token are processed in the transformer, and its outputs at readout token position are sent to the action head. The action distribution is modelled with denoising diffusion model with action chunking [24] (multiple actions predicted at a time).

Octo-Random Baseline. We also finetune a randomly initialized (i.e. not pre-trained) Octo-Base architecture on our finetuning dataset, in order to establish the effects of large-scale pretraining on LL task understanding. We randomly initialize all parameters except for the language model, which is initialized to T5 as in Octo.

π_0-FAST. π_0-FAST [9] is an intermediate-scale model with 3.5B total parameters, pretrained on 809M frames/10k hours of data, mostly from a closed-source dataset, but with 9.1% of those frames from the OXE, BridgeV2 [22] and DROID [21] datasets. It leverages PaliGemma [25] VLM for processing input images and language instructions. PaliGemma itself consists of SigLIP-400M [26], which projects input images into a flat sequence of tokens, which are concatenated with language instruction tokens, a window of future actions tokenized with custom FAST action tokenizer, and all together passed through Gemma-2B [27] language model. Some of the tokens in Gemma-2B

are reserved for continuous actions, so during inference, only language instructions and images are given as input, with the model generating action tokens autoregressively. After the model generated the stop-token, the preceding action token sequence is converted into continuous actions.

CogAct. CogAct [3] comes with 7 billion parameters and is the largest VLA that we analyze. It is pretrained on 400k trajectories or 22.5M frames from OXE. It first converts the input images with SigLIP-400M and DinoV2 into a set of visual tokens, after which these tokens are concatenated with language instruction tokens processed by a LLama-2 tokenizer [28], as well as a readout token. The resulting sequence is processed with LLama-2-7B. Then, the outputs at the readout token are used for iteratively denoising a sequence of future actions with a transformer decoder. As in Octo, the action distribution is modelled with a diffusion model.

D.3 Other models considered

We also experimented with OpenVLA [2] and RDT [29]. OpenVLA only supports predicting a single at a time, and in our 30HZ control setting was prone to getting "stuck" in certain state space regions where it was predicting low-norm actions. We tried filtering the dataset based on action norm, but it did not meaningfully improve the performance. RDT, another recently released VLA, also showed limited initial performance, was also excluded due to the fact that it contained CALVIN as part of its large-scale pretraining dataset mixture, which would conflict with our new CALVIN task definitions.

D.4 Inference

For all models, at a timestep t we generate H future actions $a_{t:t+H}$ based on image observation o_t and language instruction l, and then execute these actions consecutively before sampling a new set of actions at timestep $t + H$. This is referred to as action chunking inference+execution strategy, with H being the horizon hyperparameter. We use $H = 10$ for all models except for Octo, for which we use $H = 4$ as recommended by authors.

The authors of CogAct proposed Adaptive Action Ensemble, which is a strategy aimed at reducing jerkiness/improving smoothness of inferred trajectories. During initial experiments, we found that it to negatively affect the final performance, so we opted for standard action chunking.

D.5 Preprocessing

For each model, we use the same image and action preprocessing logic as in the original papers.

D.6 Hyperparameters

Hyp. name	**Octo**	**π_0-FAST**	**CogACT**
Batch size	256	128	128
Optimizer	AdamW	AdamW	AdamW
Learning rate	3e-4	2.5e-5	2e-5
LR schedule	Inv. square root	Cosine	None
Weight decay	0.01	1e-10	0
Observation window	2	1	1
Image resolution	256x256	224x224	224x224
Training act. horizon	10	10	15
Finetuned params	All except T4	All	All
Action horizon	10	10	10
Diffusion sampling steps	20	-	8

Table 11: Finetuning (first group) and inference (second group) hyperparameters for each model