VLA-OS: Structuring and Dissecting Planning Representations and Paradigms in Vision-Language-Action Models

Chongkai Gao[1] **Zixuan Liu**[1] **Zhenghao Chi**[1] **Junshan Huang**[2] **Xin Fei**[3]

Yiwen Hou[1] **Yuxuan Zhang**[1] **Yudi Lin**[1] **Zhirui Fang**[3] **Zeyu Jiang**[4]

Lin Shao[1]

[1]National University of Singapore [2]University of Science and Technology of China
[3]Tsinghua University [4]Nanyang Technological University

Abstract

Recent studies on Vision-Language-Action (VLA) models have shifted from the end-to-end action-generation paradigm toward a pipeline involving task planning followed by action generation, demonstrating improved performance on various complex, long-horizon manipulation tasks. However, existing approaches vary significantly in terms of network architectures, planning paradigms, representations, and training data sources, making it challenging for researchers to identify the precise sources of performance gains and components to be further improved. To systematically investigate the impacts of different planning paradigms and representations isolating from network architectures and training data, in this paper, we introduce VLA-OS, a unified VLA architecture series capable of various task planning paradigms, and design a comprehensive suite of controlled experiments across diverse object categories (rigid and deformable), visual modalities (2D and 3D), environments (simulation and real-world), and end-effectors (grippers and dexterous hands). Our results demonstrate that: 1) visually grounded planning representations are generally better than language planning representations; 2) the Hierarchical-VLA paradigm generally achieves superior or comparable performance than other paradigms on task performance, pretraining, generalization ability, scalability, and continual learning ability, albeit at the cost of slower training and inference speeds. Video results are in https://nus-lins-lab.github.io/vlaos/.

1 Introduction

Building intelligent and generalizable robots capable of perceiving, reasoning about, and interacting with physical environments remains a persistent challenge in the robotics community [34, 23]. Recent studies have increasingly emphasized the development of foundational models for robot manipulation tasks by training large Vision-Language-Action models (VLAs) on extensive datasets [8, 82, 43, 54, 2, 12, 7, 22]. Different from end-to-end foundation models in computer vision [58, 45, 40] and natural language processing tasks [1, 30, 89], recent studies of VLAs have shifted toward a new paradigm capable of performing task planning and policy learning either simultaneously or sequentially [98, 95, 27, 72, 48, 5, 77, 82]. This shift arises from the inherent complexity of robotic manipulation tasks, which naturally exhibit hierarchical structures involving both high-level task planning and low-level physical interactions [9]. Compared to end-to-end VLAs that only generate

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Embodied World Models for Decision Making.

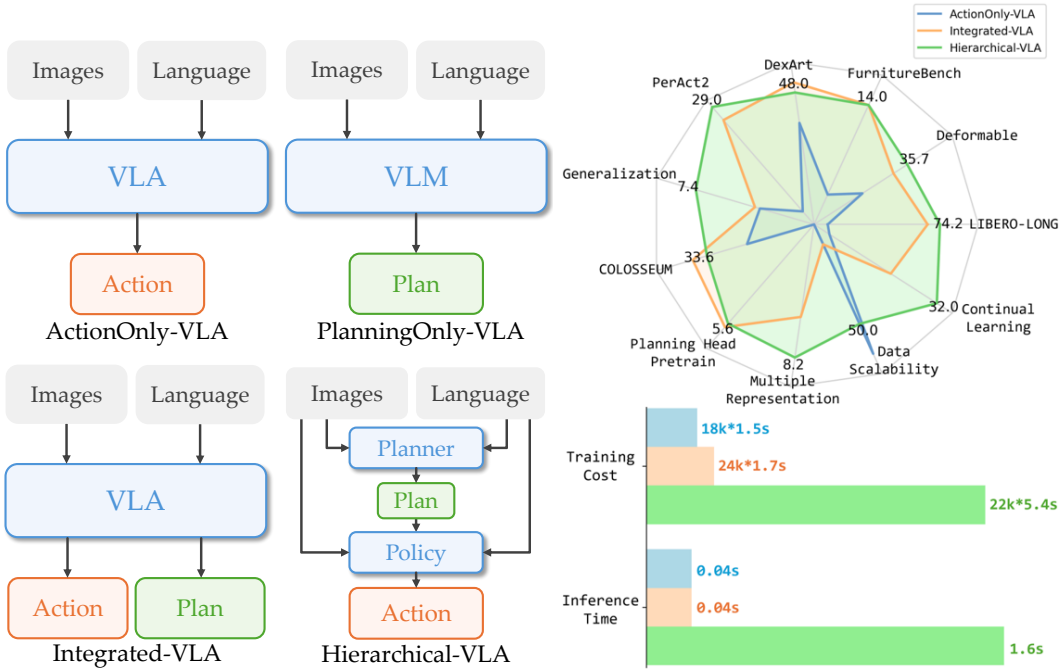


Figure 1: Left: four different VLA paradigms. Note that in this paper, we didn't explore PlanningOnly-VLA since they usually cannot be trained with the provided datasets and perform worse than others. Right: VLA paradigm comparison results. Hierarchical-VLA exhibits a generally better performance than ActionOnly-VLA and Integrated-VLA, while it incurs larger training and inference costs. This motivates future work on improving training and inference algorithms for Hierarchical-VLA models.

actions, these methods demonstrate stronger capabilities in task reasoning and comprehension for long-horizon tasks [104, 82], better success rates [95, 72], and higher sample efficiency [39, 27, 87].

However, current task-planning approaches in VLA are mainly based on intuitive designs and lack fair and systematic comparisons, as these methods vary along multiple dimensions, including network architectures, planning paradigms, data representations, and training data sources. For example, some works [72, 5, 27, 77] use a separate high-level task planning model to generate various task planning representations for a low-level VLA model, while others [82, 95, 98] use a single VLA to generate task planning representations and actions together. Consequently, substantial disagreement remains within the VLA community regarding the appropriate design and effective utilization of task planning. This makes it difficult for researchers to clearly identify which specific component contributes to performance gains or requires further improvement, hindering progress in the field.

Among these challenges, five core questions stand out: 1) **Representation**: What representation should we adopt for task planning and policy learning? Does using multiple representations yield better results, or could they conflict with one another? 2) **Paradigm**: Should we employ a monolithic model that jointly performs task planning and policy learning, or should we opt for a hierarchical paradigm where two separate models handle these tasks independently? 3) **Bottleneck**: Between task planning and policy learning, which presents a greater challenge for current manipulation tasks? 4) **Scalability and Pretraining**: Do VLAs that incorporate task planning preserve the advantageous properties of end-to-end foundation models, such as model and data scalability, as well as benefits derived from pretraining? and 5) **Performance**: Do VLAs employing task planning have better generalization and continual learning ability than end-to-end VLAs? Addressing these questions will provide the community with a clearer understanding of how task planning works in VLA models, and offer empirical evidence and guidance for future developments.

In this work, we aim to answer these questions with systematic and controllable experiments. First, to avoid biases introduced by specific neural network choices, we develop VLA-OS[1] model series:

[1]"OS" stands for "Operating System" and designates that our model family provides unified and organized interfaces of advanced VLA architectures with various planning heads and different paradigms for users.

a unified and composable family of VLA models for general-purpose manipulation tasks capable of different task planning paradigms. Concretely, we designed VLA-OS-A, VLA-OS-I, and VLA-OS-H that correspond to three mainstream VLA paradigms (ActionOnly-VLA, Integrated-VLA, Hierarchical-VLA), respectively, as illustrated in Figure 1. VLA-OS series features a unified, interchangeable VLM backbone that can be directly downloaded from HuggingFace, various plug-and-play planning heads for different representations, and two different action heads both supporting 2D/3D tasks, as shown in Figure 2. We show in Table 1 that VLA-OS exhibits superior performance compared to most existing VLA methods with fewer parameters and without pretraining.

Next, to answer the **representation** question, we annotate three kinds of task reasoning representations, including language reasoning, visual reasoning, and goal images, and conducted exhaustive combinatorial experiments with Integrated-VLA and Hierarchical-VLA models on LIBERO [51] benchmark to identify representations that yield optimal performance. Subsequently, employing the optimal representations identified, we conducted performance comparisons among three VLA paradigms on six benchmarks to answer the **paradigm** question, including rigid body manipulation tasks [51], visual generalization tasks [64], complex long-horizon tasks [32], real-world deformable manipulation tasks, dexterous manipulation tasks [3], and dual-arm manipulation tasks [28]. Furthermore, to answer the **bottleneck** question, we designed a novel set of evaluation metrics tailored to separately assess the performance of task planning and policy learning parts. To answer the **scalability** question, we use LIBERO [51] to test the model and data scalability as well as the effects of pretraining among different paradigms. And lastly, we test the generalization capabilities and continual learning ability of different VLA paradigms to answer the **performance** question.

Our experiments yield three primary findings: 1) Visually grounded planning representations (visual reasoning and image foresight planning) outperform language-based planning representations across multiple dimensions including task performance, generalization, training and speed, and low-level policy execution; 2) Hierarchical-VLA matches or exceeds the performance of Integrated-VLA and ActionOnly-VLA in terms of task performance, generalization, scalability, planning scores, continual learning, and gains from task-planning pretraining, albeit at the expense of increased training cost and slower inference; 3) On LIBERO [51] benchmark tasks, policy learning is consistently more challenging than task planning, regardless of which planning representation is used. We believe that our findings (as well as source codes, annotated datasets, and checkpoints) will provide significant help and guidance for future research within the VLA community and the broader robotics community.

2 Related Works

2.1 VLA Paradigms for Robot Manipulation

Vision-Language-Action Models (VLAs) refer to multi-modal comprehensive models that can handle visual and language inputs and generate robot actions for control. The word “VLA" was first proposed and studied in RT-2 [11], where they train a VLM to output actions as text tokens for robot control. After that, more VLA works are emerging. According to how they incorporate the task planning process, we divide VLAs into four paradigms and introduce each of them as follows.

PlanningOnly-VLA These works leverage pretrained LLMs or VLMs to reason and perform task planning without generating the low-level action. They break up the given task into simpler sub-tasks that can be performed by either using a set of pre-trained sub-skills [36, 2, 65, 71, 19], or outputting the parameters of pre-defined motions or cost functions for optimization [49, 73, 38, 37, 78, 57, 25, 26]. The problem is that their VLMs and low-level skills usually cannot be trained with further datasets, which frequently places them at a disadvantage compared to other VLA paradigms capable of training on given datasets [97, 92]. Consequently, we do not include PlanningOnly-VLA in this study.

ActionOnly-VLA These works employ an end-to-end fashion to directly map visual and language inputs to robot actions with a multi-modal network. Pioneering works mainly focus on verifying the effectiveness of large-scale robot learning [10, 11, 59, 75], while later works start to explore different model architectures, training objectives, and extra multi-modal representations and information fusion designs to make this paradigm more effective and efficient [8, 54, 43, 84, 46, 102, 62, 99, 100, 4, 103, 66]. In this work, we design VLA-OS-A for this paradigm by synthesizing several advanced model designs that have been verified to be superior in recent works [47, 8, 4].

Integrated-VLA These works use a single model to perform task planning and policy learning simultaneously. According to whether the action generation process is conditioned on the planning

embeddings or results, they can be further divided into explicit planning and implicit planning. For explicit planning, EmbodiedCoT [95] and CotVLA [98] generate either language-based or goal-image-based embodied chain-of-thought [79] reasoning before generating actions, and the action generation process is conditioned on the embeddings of CoT. For implicit planning, MDT [69] and PIDM [77] use goal image foresight generation loss as an auxiliary objective for planning, while RoboBrain [39] and ChatVLA [104] train VLA with auxiliary task reasoning loss in language representations. Some recent works also seek to use latent action tokens [93, 70, 13, 16, 35] that serve as forward dynamics representations to generate future images as image foresight planning, and decode these latent actions to real actions with another action head. The inputs to the action head are from the VLM encoder, and they do not need the planning heads (decoder) during inference [93, 70, 13, 16] or they only need one planning forward pass [35], so we also see these methods as implicit planning. In this work, we design VLA-OS-I for this paradigm with various plug-and-play planning heads upon VLA-OS-A for different planning representations, and design corresponding variants for both explicit and implicit planning paradigms as VLA-OS-I-I and VLA-OS-I-E.

Hierarchical-VLA These works use two separate models for task planning and policy learning, with no connection or gradient between them. The idea of hierarchical models has always existed in robotics research [25, 26, 87, 14, 80]. RT-H [5] is the first work of this paradigm, where they use two identical VLMs to generate languages and actions respectively. Later works [72, 83, 82] also follow this idea but use different model architectures for task planning and action generation. Other works seek to generate multi-modal planning results for policy learning, such as image flows or trajectories [29, 27, 48], future videos [20, 91], affordance [55, 56], keypose [17], and keypoints [94]. In this work, we design VLA-OS-H for this paradigm.

2.2 VLA Benchmarks and Evaluations

With the rapid advancement of VLA models, benchmarks and evaluation studies for VLA have also experienced significant growth. Given the complexity and multi-dimentionality of robot manipulation tasks and VLA models, different works usually focus on evaluating one or several specific dimensions of VLA. Some works focus on the VLA model designs and training algorithms, such as different model architectures and input and output spaces [47, 88]. Other works aim to build benchmark environments and tasks to evaluate different capacities of current VLA models, such as spatial and visual generalization ability [97], long-horizon task reasoning ability [92], and different training data modalities [104]. In this work, we focus on task planning paradigms for VLA and keep the model architectures the same with systematically designed controllable experiments.

3 VLA-OS Model Family Design

3.1 Preliminaries

We study imitation learning for robot manipulation tasks. Specifically, for each task $\mathcal{T}$, we assume a set of demonstrations $\mathcal{D}_{\mathcal{T}} = \{(o_i^1, a_i^1), (o_i^2, a_i^2), \cdots, (o_i^{T_i}, a_i^{T_i})\}_{i=1}^N$ and a language goal are given, where T_i is the episode length, o is the observation, a is the robot action, and N is the number of demonstrations. We use a history of multi-view images and proprioception information as observations. In this work, we set the image resolution as 224×224. For actions, we use a normalized continuous delta end-effector pose δ_p action space and gripper open/close action σ for training. We also let the policy generate action chunks, i.e., $a_t = ([\delta_p, \sigma]^t, \cdots, [\delta_p, \sigma]^{t+L-1})$. For dexterous hands, we use the delta joint values as the action space. We train the policy with either flow matching [50, 53] loss (for multi-modal demonstration datasets) or L1 loss (for simple and uni-modal demonstration datasets) under the suggestion of previous works [44, 8, 4, 47].

3.2 VLA-OS-A for ActionOnly-VLA Paradigm

VLA-OS-A model series directly generates actions without task planning stages. It is also used as the base model for other paradigms. We design a block-wise causal attention VLA drawing inspiration from [8], as shown in Figure 2. First, a VLM encodes the visual and language inputs, where the vision encoder will encode input image patches and project them into language embedding space with an MLP. Then, we use a separate set of weights as an action head for the robotics-specific tokens (action and proprioception states). The action head is a transformer decoder that has the same number of layers as the LLM, and for each layer, the queries of the proprioception tokens can attend to both the

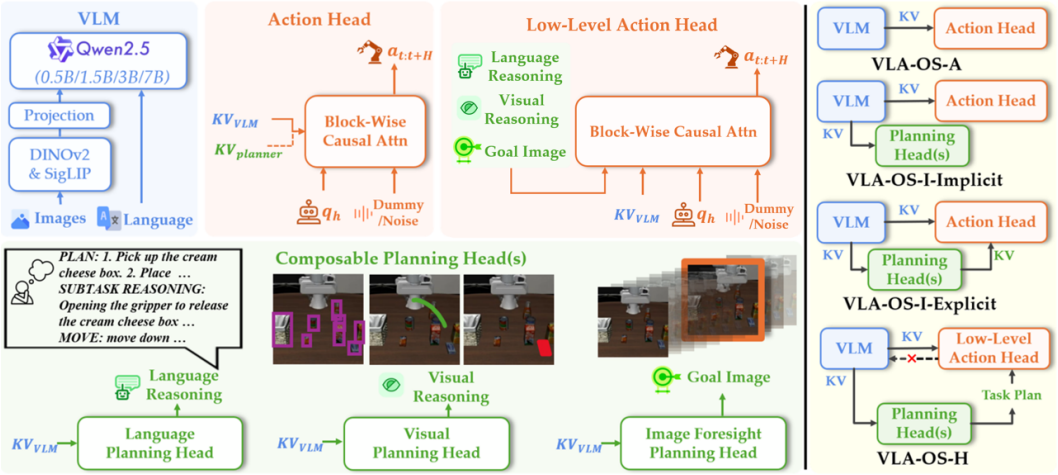


Figure 2: The VLA-OS model family. Left: the VLM and the composable heads. Our VLM has the same architecture with different numbers of parameters. Although we only draw Qwen2.5 here, our code supports any kind of LLM backbone from HuggingFace. Right: four VLA-OS architectures used in our experiments. To minimize the effects of different numbers of parameters in different models, we restrict the number of parameters of all heads to about 5% of the VLM.

keys and values from the LLM and the proprioception keys and values, and the queries of the action tokens can attend to the keys and values from the LLM, the proprioception tokens, and themselves.

Compared to π_0 [8], we make two changes in VLA-OS-A: 1) we use an ensemble of vision encoders (DINOV2 [58]+SigLIP [96]), which is proven to be better than using a single vision encoder [41]; 2) to support the model scalability experiments, we need a set of LLMs with the same structure but have different number of parameters. Thus, we choose Qwen2.5 [89] LLM series with 0.5B, 1.5B, 3B, 7B pretrained checkpoints rather than the original PaliGamma [6]. To make it a VLM, we finetune Qwen2.5 LLMs with the vision encoders and the projector on LLaVa v1.5 [52] data mixture by ourselves. We call our VLA family that uses 0.5B, 1.5B, 3B, 7B LLM backbones with suffixes of -S(mall), -B(ase), -M(iddle), and -L(arge). Detailed information can be found in Appendix C. Note, although we use Qwen2.5 in this work, our codes support any kind of LLM from HuggingFace, which makes VLA-OS highly flexible compared to [8] that is restricted to a specific backbone.

For 3D action head, we also take in multi-view depth images as input, and fuse the multi-view RGBD images to 3D point cloud using camera intrinsics and extrinsics, and inject additional CLIP features onto the point cloud, as in 3D diffuser actor [42]. We also downsample the point cloud with farthest point sampling. Each point from the downsampled point cloud will be seen as a token and these 3D tokens are sent to the action head as additional inputs.

3.3 VLA-OS-I for Integrated-VLA Paradigm

To perform task planning with different kinds of representations, we design three kinds of task planning heads for VLA-OS. We first annotate three kinds of task reasoning datasets corresponding to each planning representation, as shown in Figure 3. Here we only briefly introduce each of them. Details of the data annotation process can be found in Appendix B.

The **language reasoning** data contains 8 different keys [95] for each timestep, including `Task`, `Plan`, `Subtask`, `Subtask Reason`, `Move`, `Move Reason`, `Gripper Position`, and `Object Bounding Boxes`, containing the understanding of the scene and decomposition of the task. The **visual reasoning** data contains spatial semantic information and is more grounded in input images compared to language reasoning. We follow [61, 86] and use location tokens `<loc i>` to represent the i-th bin token from top-left to bottom-right. We use this kind of token to represent `object bounding boxes`, `end-effector flow`, and `target object affordance` as the visual planning representations. The **image foresight reasoning** data is a third-person view image at the K-th future step as the short-horizon goal image.

We then design language planning head, visual planning head, and image foresight planning head for each kind of representation, as shown in Figure 2. All of them are transformers that have the same number of layers with the LLM backbone, and use the block-wise causal attention mechanism to acquire the keys and values from each layer of the LLM backbone as conditions. The language

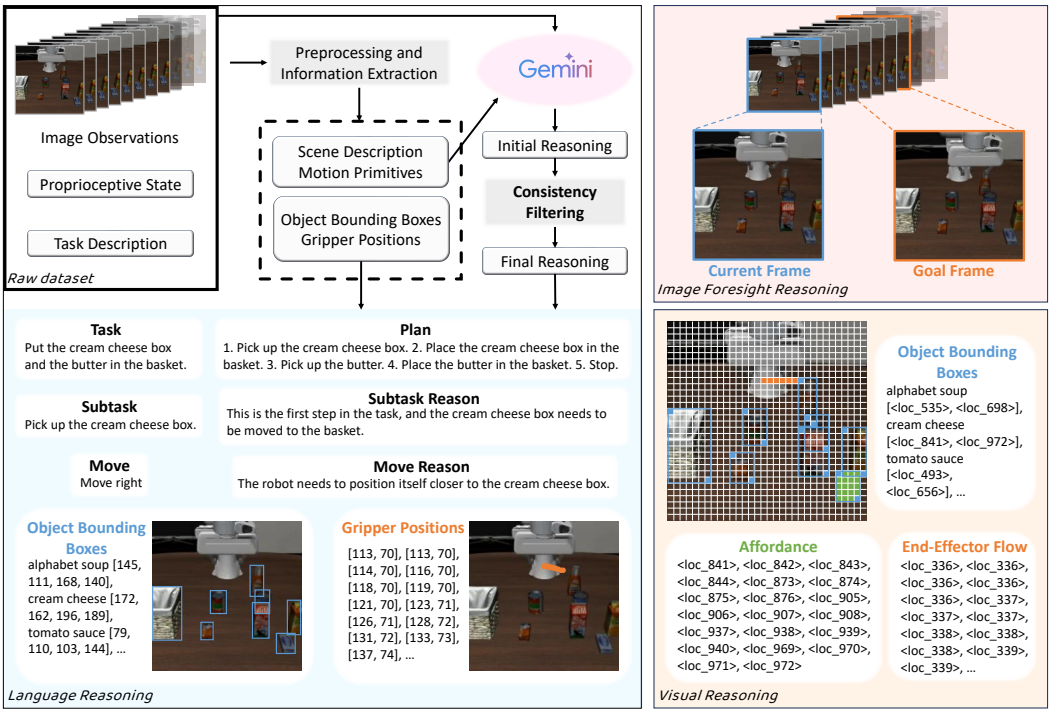


Figure 3: The formats and contents of the language reasoning dataset, the visual reasoning dataset, and the image foresight reasoning dataset in this work. We use various vision-language models for data annotation. We illustrate the language reasoning data annotation process on the top left part.

planning head uses the LLM's tokenizer for decoding, whereas the visual planning head uses an extended tokenizer vocabulary to predict location tokens. The image foresight planning head is an autoregressive image generation model similar to the recent SOTA image generator [31]. It auto-regressively generates the image in a coarse-to-fine paradigm proposed by VAR [76]. The language and visual planning heads are trained with cross-entropy loss, while the image foresight planning head is trained with the special loss in [31].

For all three planning heads, there are two kinds of ways to use them: 1) implicit planning: the action head is independent of the planning heads, i.e., the planning heads serve as auxiliary losses for the VLA training and will not be executed during inference. This may help the model avoid planning accumulation error and improve the inference speed; 2) explicit planning: the action head also attends to the keys and values from each layer of the planning heads, and during inference, the VLA must first perform task planning before generating actions. This may help solve complex tasks in a chain-of-thought [79, 95, 98] manner.

3.4 VLA-OS-H for Hierarchical-VLA Paradigm

This model uses two networks for task planning and policy learning respectively. As shown in Figure 2, we use the VLM together with planning heads for task planning, and modify the action head to an encoder-decoder transformer for policy learning. This action head can take as input the images, proprioception observations, and the planning representations to generate actions. To keep the comparison fair, we make the layer of the encoder and decoder of the action head half of the other two VLA-OS paradigms. We also give frozen image features from AM-Radio [67] and language features from Qwen2.5 [89] for the inputs of the action head to compensate for deficiencies in visual and linguistic features not captured by the VLM. Training details are in Appendix C.

4 Experiments and Findings

In this section, we perform systematic and controllable experiments with the VLA-OS model series on various manipulation tasks shown in Figure 4 to answer the research questions in Section 1. Detailed experimental settings are in Appendix C. All models are trained on 8×NVIDIA A100 80G GPUs. The continual learning experiments are in Appendix C.

Figure 4: Benchmarks used in our evaluations, including LIBERO [51] and FurnitureBench [32] for 2D rigid body manipulation experiments, The COLOSSEUM [64] for 3D and generalization evaluation, real-world deformable object manipulation tasks (fold the handkerchief, unfold the jean, and straighten the rope), DexArt [3] for dexterous tasks, and PerAct2 [28] for dual-arm tasks.

4.1 Sanity Check of VLA-OS

Before investigating different VLA paradigms for our research questions, we first verify the correctness and basic performance of our VLA-OS models to serve as a foundational sanity check. We train VLA-OS-A-S on four suites from LIBERO [51] (LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, LIBERO-Long) from scratch with L1 loss and compare them with Diffusion-Policy [18], fine-tuned OpenVLA [43], fine-tuned CoT-VLA [98], fine-tuned DiT Policy [33], and the state-of-the-art methods: fine-tuned π_0 [8] and its variant π_0-FAST [62]. Results are shown in Table 1.

Table 1: Sanity check. Success rates on four LIBERO benchmarks. Baseline results are from their papers [43, 8, 44]. Our results are the average of top-3 checkpoints averaged over 20 rollouts for each task suite. **Bold** indicates the best result except SOTA, and underline indicates comparable result.

	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long	Average
Diffusion Policy [18] (scratch)	78.3	92.5	68.3	50.5	72.4
OpenVLA [43] (fine-tuned)	84.7	88.4	79.2	53.7	76.5
CoT-VLA [98] (fine-tuned)	87.5	91.6	87.6	**69.0**	81.1
DiT Policy [33] (fine-tuned)	84.2	96.3	85.4	63.8	82.4
π_0-FAST [62] (fine-tuned)	**96.4**	**96.8**	88.6	60.2	85.5
VLA-OS-A-S (scratch, ours)	87.0	96.5	**92.7**	66.0	**85.6**
π_0 [8] (fine-tuned, SOTA)	96.8	98.8	95.8	85.2	94.2

We can see that VLA-OS-A-S performs better (+13.2%) than Diffusion Policy (trained from scratch) and the fine-tuned OpenVLA model (+9.1%), CoT-VLA (+4.5%), and DiT Policy (+3.2%), and is comparable to fine-tuned π_0-FAST (+0.1%). Although our model is worse than the SOTA method, these results sufficiently demonstrate that our model design is excellent and competitive. Note that VLA-OS-A-S is **trained from scratch** and utilizes **only a 0.5B LLM backbone**.

Finding 1: For downstream tasks, larger VLA models trained on large-scale datasets do not necessarily outperform smaller models that are trained from scratch. Model architectures and algorithmic designs are still important at the current moment.

4.2 Planning Representation Experiments

To explore which representation is better for task planning and policy learning, we perform comprehensive experiments with language planning (**L**), visual planning (**V**), image foresight planning (**IF**), and their combinations on LIBERO-LONG [51] benchmark that contains 10 long-horizon tasks with 50 demonstrations in each task for VLA-OS-I and VLA-OS-H. The best representation will be used as the default representation for all later experiments. Table 2 shows the results.

Finding 2: For Integrated-VLA paradigm, implicit planning can yield a positive performance gain, whereas explicit planning incurs a significant performance degradation when trained from scratch.

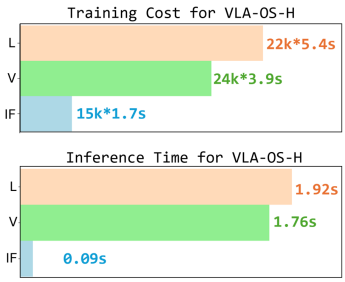
Analysis: The implicit planning paradigm leverages various auxiliary task planning objectives as additional losses for training, and during inference, there is no difference between it and ActionOnly-VLA, thus it brings performance improvement. This shows that **using task planning as auxiliary losses** can improve the performance. However, the explicit planning paradigm will have to first complete the entire planning process before the action head generation during inference, and this will bring severe

Table 2: Different planning representation comparison on LIBERO-Long. All results are the average of top-3 checkpoints averaged over 20 rollouts. Numbers in parentheses indicate the change relative to the result of VLA-OS-A in Table 1.

	L	V	IF	L+V	L+IF	V+IF	L+V+IF
VLA-OS-I-I	68.0 (↑2.0)	71.0 (↑5.0)	72.5 (↑6.5)	66.7 (↑0.7)	**73.3** (↑7.3)	71.0 (↑5.0)	71.7 (↑5.7)
VLA-OS-I-E	60.5 (↓5.5)	52.5 (↓13.5)	**67.5** (↑1.5)	42.7 (↓23.3)	56.7 (↓9.3)	56.7 (↓9.3)	50.7 (↓15.3)
VLA-OS-H	63.5 (↓2.5)	69.0 (↑3.0)	71.7 (↑5.7)	71.5 (↑5.5)	72.0 (↑6.0)	73.7 (↑7.7)	**74.2** (↑8.2)

	VLA-OS-A	VLA-OS-I	VLA-OS-H
LIBERO-LONG (2D)	66.0	73.3 (↑ 7.3)	**74.2** (↑ 8.2)
The COLOSSEUM (3D)	34.4	**35.7** (↑ 1.3)	35.3 (↑ 0.9)
Deformable (Real-World)	28.5	**35.4** (↑ 6.9)	33.6 (↑ 5.1)
FurnitureBench	11.0	**14.0** (↑ 3.0)	**14.0** (↑ 3.0)
DexArt	45.0	**49.0** (↑ 4.0)	48.0 (↑ 3.0)
PerAct2	21.0	28.0 (↑ 7.0)	**29.0** (↑ 8.0)
Generalization	6.1	6.2 (↑ 0.1)	**7.4** (↑ 1.3)
Planning Head Pretraining	–	79.1 (↑**5.8**)	79.8 (↑5.6)

(a) Success rates of different VLA paradigms on more benchmarks, as well as the generalization and task planning pretraining experiments. All results are averaged over 20 rollouts among 3 best checkpoints.



(b) Training cost and inference time for different representations.

Figure 6: More results for different paradigms and inference time and training cost for different representations. Results of the right figure are calculated from the LIBERO-LONG benchmark.

planning accumulation error issues. Typically, the length of planning tokens significantly exceeds that of action tokens (approximately 2000 vs. 8), which will exacerbate the accumulation error issue than purely with action tokens. Additionally, the embeddings from every layer of the planning head are fed into the action head, affecting its internal representations. Meanwhile, the action head does not receive raw visual or language inputs. It only receives embeddings from the VLM and planning heads, which makes it lack the necessary error-correction capability. Instead, Hierarchical-VLA will not only take in the raw visual observation and language instruction as inputs, but also confine the planning accumulation errors exclusively to the explicit representation level, rather than allowing them to propagate into the deeper embedding layers.

For qualitative comparisons, we show in Figure 5 an example that when VLA-OS-H uses the same planning heads as VLA-OS-I-E where there are some planning errors, it can correct the behavior while VLA-OS-I-E cannot.

Finding 3: Visually grounded planning representations work better than language planning representations, and also have faster inference speed and smaller training cost.

From the results in Table 2, we can see that visual planning and image foresight planning are better than language planning (↑5.75 v.s. ↑2.0 for VLA-OS-I-I and ↑4.35 v.s. ↓2.5 for VLA-OS-H). We also illustrate the inference speed and training cost in Figure 6b (introduced in Section 4.5) to show the speed and cost advantages of visually grounded planning representations.

Finding 4: When employing multiple planning representations concurrently, Hierarchical-VLA outperforms Integrated-VLA paradigms.

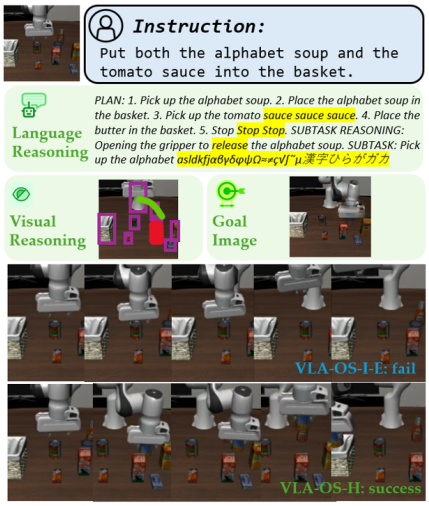


Figure 5: Comparison between VLA-OS-I-E and VLA-OS-H with the same planning errors. The three planning representations shown in this figure all have small planning errors (highlighted).

4.3 More Performance, Generalization, and Benefit from Planning Head Pretraining

To further compare different planning paradigms, we perform additional experiments to explore their performance on: 1) more manipulation benchmarks including 3D manipulation tasks [64], real-world deformable tasks, furniture assembly tasks [32], dexterous manipulation tasks [3], and dual-arm manipulation tasks [28]; 2) generalization ability; and 3) benefits from planning head pretraining.

For 1), in COLOSSEUM, we train and test on the *No-Perturbation* setting. For 2), we use THE COLOSSEUM and train on *No-Perturbation* but test on *ALL-Perturbation* setting, including changes in color, texture, size of objects, table-tops, backgrounds, lighting, distractors, physical properties, and camera poses. For 3), a lot of literature [27, 48, 13, 72, 94, 91] claim that the primary advantage of using task planning in VLA rather than ActionOnly-VLA is that their task-planning components can be trained on large-scale task-agnostic planning data without costly action annotations. Here, we train them on LIBERO-90, a larger dataset with 90 manipulation tasks and 50 demonstrations for each task. We only train the planning components, i.e., the VLM and planning heads. Then we fine-tune the pretrained VLM and planning heads together with the action head on LIBERO-LONG with both the task reasoning and policy learning losses. Results are in Table 6a.

Finding 5: *Integrated-VLA and Hierarchical-VLA outperform ActionOnly-VLA across a broad spectrum of tasks (2D, 3D, simulation, and real-world), with their performances largely comparable.*

Finding 6: *Both Integrated-VLA and Hierarchical-VLA benefit similarly from task-planning pretraining, exhibiting analogous gains in task success rate.*

Finding 7: *Hierarchical-VLA demonstrates the best generalization ability.*

4.4 Separate Investigation of Task Planning and Policy Learning Parts

It is imperative to discern whether task failures arise from the planning component or policy learning. In this part, we use LIBERO-LONG [51] for Integrated-VLA (only for task planning) and Hierarchical-VLA to separately evaluate the task planning part and policy learning part of the model for all three planning representations. For evaluation, we manually divide each long-horizon task into several sub-tasks, and forcibly reset the environment to the initial state of each subtask. Then we compute the average planning correctness $\mathbb{I}$ (0 or 1) of the planning outcomes and execution success rate $\mathbb{S}$ (0 or 1) from the action head across all subtask start points. Thus, for a given task trajectory, we can get **Decomposition Score (DCS)**$= \frac{1}{T}\sum_{t=1}^{T}\mathbb{I}(p_t)$ and **Instruction Following Score (IFS)**$= \frac{1}{T}\sum_{t=1}^{T}\mathbb{S}(a_t^{seq})$, where T is the total sub-task number and p_t and a_t^{seq} are the planning outcomes and actions generated at subtask t. Note for Hierarchical-VLA, we give the ground truth planning results when testing IFS. Results are shown in Table 3.

Table 3: Separate evaluation of task planning and policy learning modules for different paradigms and representations. Results are averaged from 20 episodes for each task in LIBERO-LONG.

	L		V		IF	
	DCS	IFS	DCS	IFS	DCS	IFS
VLA-OS-I-I	0.79	–	0.83	–	0.92	–
VLA-OS-H	0.81	0.84	0.86	0.93	0.94	0.90

Finding 8: *Hierarchical-VLA performs better than Integrated-VLA in task planning.*

Finding 9: *Visually grounded planning representations are easier for low-level policy to follow.*

4.5 Training Cost and Inference Speed

We report the inference speed and training cost for different paradigms and planning representations. The training cost is calculated by multiplying the total training steps by the per-step time on LIBERO-LONG with 8× A100 NVIDIA GPUs. Results are shown in Figure 1 and 6b.

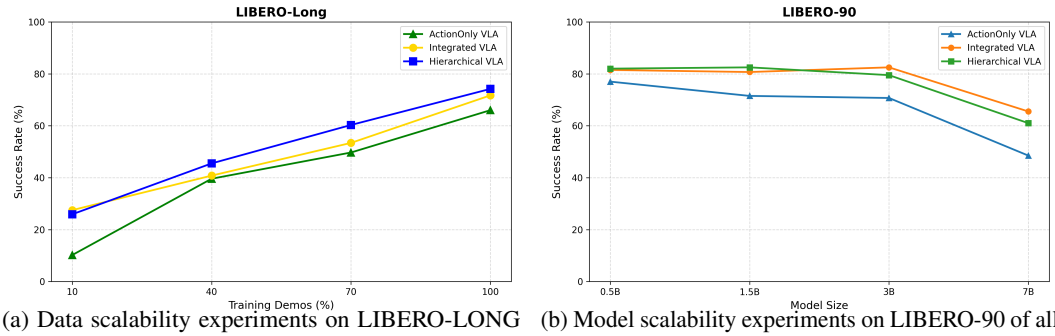
Finding 10: *The autoregressive property of the language-planning representation head is the principal cause of its higher training cost and slower inference speeds.*

4.6 Data and Model Scalability Experiments

In this part, we perform experiments for data and model scalability of different VLA paradigms. For data scalability, we use LIBERO-LONG [51], a dataset with 10 tasks with a total of 500 demonstrations. We use 10%, 40%, 70%, and 100% of the data to train on three VLA paradigms with the model size S. For model scalability, we use LIBERO-90, a dataset with 90 tasks and 4,500 demonstrations, for the experiment with all training data. We choose Qwen-2.5 LLM backbone with parameters of 0.5B, 1.5B, 3B, and 7B for experiments. Results are shown in Figure 7.

Finding 11: *The performance of all VLA paradigms improves as the amount of action-labeled demonstration data increases, i.e., all VLA paradigms have the data scalability.*

Finding 12: *For tasks trained from scratch with roughly 5,000 demonstrations, the LLM backbone should be limited to 0.5B parameters, or keep the total model size under 1B parameters.*



(a) Data scalability experiments on LIBERO-LONG with different planning paradigms with 0.5B LLM backbone. Success rates are calculated among 20 evaluation episodes among the 3 best checkpoints.

(b) Model scalability experiments on LIBERO-90 of all VLA paradigms. Success rates are calculated among 20 evaluation episodes among the 3 best checkpoints. We select the best checkpoint before 100k steps.

Figure 7: Data and model scalability experiments across different VLA paradigms.

5 Conclusion and Limitation

We provide a systematic investigation across different VLA paradigms and task planning representations through various kinds of manipulation tasks. Experiments show the superiority of visually grounded planning representations and the Hierarchical-VLA paradigm. Main findings include:

1. The time has not yet come to scale up VLA model sizes. Model architectures and training algorithms still matter.
2. Visually grounded representations (visual and image foresight) are better than language representations in terms of success rates, low-level following, and continual learning.
3. Integrated-VLA and Hierarchical-VLA outperform ActionOnly-VLA on task performance and generalization ability, but incur faster forgetting.
4. Integrated-VLA and Hierarchical-VLA perform comparably on task performance and Planning Head Pretraining, but Hierarchical-VLA generalizes better, has better task-planning performance, and performs better when using multiple planning representations.
5. All VLA paradigms have data scalability. For tasks trained from scratch with roughly 5,000 demonstrations, the LLM backbone should be limited to 0.5B parameters, or keep the total model size under 1B parameters.

We believe our findings offer meaningful insights that can inform future research in VLA and the broader robotics community. We recommend following research directions based on our findings:

1. Why are visually grounded representations better than language? The point is, language representations already contain enough information for doing the task.
2. In both explicit v.s. implicit and Hierarchical v.s. Integrated comparisons, reducing the influence (or coupling) of action head training on VLM improve the performance. We suppose this is because of the gradient conflicts between action training and planning training. So, how to verify this, why they conflict with each other, and how to avoid the gradient conflict between them are interesting questions.
3. How to design network architectures to extract information from VLM effectively? The current KV cache mechanism is good, but it restricts the layer of the planning and action heads to match the layer number of the LLM backbone.
4. How to design faster planning heads for autoregressive planning heads?
5. How to design better low-level action heads with better planning-following ability?
6. How to construct large-scale task planning datasets for VLA? How to transfer current datasets to task planning datasets?

The limitations of this paper are: 1) despite the VLA-OS family encompassing a wide array of task planning paradigms for VLA, there remain several designs and variants that we have not yet covered, such as using latent actions [93, 13] for image generation rather than VAR [76, 31]-like generator in VLA-OS, video generation for planning [91, 20], and scene flow for planning [27, 81]; 2) we didn't explore embodiment transfer, sim2real transfer, and 2D to 3D transfer problems for VLA; 3) our training dataset remains limited to fewer than 10,000 trajectories, and we have not yet investigated the research questions that arise from pretraining on larger datasets such as the OXE [60] dataset.

6 Acknowledgment

We thank Zhixuan Xu for his valuable discussion and his guidance for drawing the pictures.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[3] Chen Bao, Helin Xu, Yuzhe Qin, and Xiaolong Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21190–21200, 2023.

[4] Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint. `https://ai.stanford.edu/blog/minivla/`, 2024.

[5] 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. *arXiv preprint arXiv:2403.01823*, 2024.

[6] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

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

[8] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[9] Michael Brady. *Robot motion: Planning and control*. MIT press, 1982.

[10] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

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

[13] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.xxxxx*, 2025.

[14] Haonan Chen, Junxiao Li, Ruihai Wu, Yiwei Liu, Yiwen Hou, Zhixuan Xu, Jingxiang Guo, Chongkai Gao, Zhenyu Wei, Shensi Xu, et al. Metafold: Language-guided multi-category garment folding framework via trajectory generation and foundation model. *arXiv preprint arXiv:2503.08372*, 2025.

[15] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *European Conference on Computer Vision*, pages 74–91. Springer, 2024.

[16] Xiaoyu Chen, Junliang Guo, Tianyu He, Chuheng Zhang, Pushi Zhang, Derek Cathera Yang, Li Zhao, and Jiang Bian. Igor: Image-goal representations are the atomic control units for foundation models in embodied ai. *arXiv preprint arXiv:2411.00785*, 2024.

[17] Yangtao Chen, Zixuan Chen, Junhui Yin, Jing Huo, Pinzhuo Tian, Jieqi Shi, and Yang Gao. Gravmad: Grounded spatial value maps guided action diffusion for generalized 3d manipulation. *arXiv preprint arXiv:2409.20154*, 2024.

[18] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[19] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

[20] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

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

[22] FigureAI. Helix: A vision-language-action model for generalist humanoid control. `https://www.figure.ai/news/helix`, 2025. Accessed: 2025-02-20.

[23] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, page 02783649241281508, 2023.

[24] Chongkai Gao, Haichuan Gao, Shangqi Guo, Tianren Zhang, and Feng Chen. Cril: Continual robot imitation learning via generative and prediction model. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6747–5754. IEEE, 2021.

[25] Chongkai Gao, Yizhou Jiang, and Feng Chen. Transferring hierarchical structures with dual meta imitation learning. In *Conference on Robot Learning*, pages 762–773. PMLR, 2023.

[26] Chongkai Gao, Zekun Li, Haichuan Gao, and Feng Chen. Iterative interactive modeling for knotting plastic bags. In *Conference on Robot Learning*, pages 571–582. PMLR, 2023.

[27] Chongkai Gao, Haozhuo Zhang, Zhixuan Xu, Zhehao Cai, and Lin Shao. Flip: Flow-centric generative planning for general-purpose manipulation tasks. *arXiv preprint arXiv:2412.08261*, 2024.

[28] Markus Grotz, Mohit Shridhar, Yu-Wei Chao, Tamim Asfour, and Dieter Fox. Peract2: Benchmarking and learning for robotic bimanual manipulation tasks. In *CoRL 2024 Workshop on Whole-body Control and Bimanual Manipulation: Applications in Humanoids and Beyond*, 2024.

[29] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023.

[30] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[31] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. *arXiv preprint arXiv:2412.04431*, 2024.

[32] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *The International Journal of Robotics Research*, page 02783649241304789, 2023.

[33] Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. *arXiv preprint arXiv:2503.19757*, 2025.

[34] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Hao-Shu Fang, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023.

[35] Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024.

[36] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

[37] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[38] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[39] Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, et al. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. *arXiv preprint arXiv:2502.21257*, 2025.

[40] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024.

[41] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024.

[42] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

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

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

[45] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

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

[47] Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024.

[48] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, et al. Hamster: Hierarchical action models for open-world robot manipulation. *arXiv preprint arXiv:2502.05485*, 2025.

[49] 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. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[50] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

[52] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[53] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

[54] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[55] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *arXiv preprint arXiv:2308.10901*, 2023.

[56] 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. *arXiv preprint arXiv:2411.02704*, 2024.

[57] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

[58] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

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

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

[61] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023.

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

[63] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[64] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

[65] Dicong Qiu, Wenzong Ma, Zhenfu Pan, Hui Xiong, and Junwei Liang. Open-vocabulary mobile manipulation in unseen dynamic environments with 3d semantic maps. *arXiv preprint arXiv:2406.18115*, 2024.

[66] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[67] Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12490–12500, 2024.

[68] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[69] Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. *arXiv preprint arXiv:2407.05996*, 2024.

[70] Dominik Schmidt and Minqi Jiang. Learning to act without actions. *arXiv preprint arXiv:2312.10812*, 2023.

[71] Rutav Shah, Albert Yu, Yifeng Zhu, Yuke Zhu, and Roberto Martín-Martín. Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation. *arXiv preprint arXiv:2410.06237*, 2024.

[72] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

[73] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[74] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[75] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[76] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[77] Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. *arXiv preprint arXiv:2412.15109*, 2024.

[78] Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. Llm^ 3: Large language model-based task and motion planning with motion failure reasoning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12086–12092. IEEE, 2024.

[79] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[80] Zhenyu Wei, Zhixuan Xu, Jingxiang Guo, Yiwen Hou, Chongkai Gao, Zhehao Cai, Jiayu Luo, and Lin Shao. D (r, o) grasp: A unified representation of robot and object interaction for cross-embodiment dexterous grasping. *arXiv preprint arXiv:2410.01702*, 2024.

[81] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.

[82] Junjie Wen, Minjie Zhu, Yichen Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Chengmeng Li, Xiaoyu Liu, Yaxin Peng, Chaomin Shen, et al. Diffusion-vla: Scaling robot foundation models via unified diffusion and autoregression. *arXiv preprint arXiv:2412.03293*, 2024.

[83] 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. *arXiv preprint arXiv:2502.05855*, 2025.

[84] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[85] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[86] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4818–4829, 2024.

[87] Zhixuan Xu, Chongkai Gao, Zixuan Liu, Gang Yang, Chenrui Tie, Haozhuo Zheng, Haoyu Zhou, Weikun Peng, Debang Wang, Tianrun Hu, et al. Manifoundation model for general-purpose robotic manipulation of contact synthesis with arbitrary objects and robots. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10905–10912. IEEE, 2024.

[88] Feng Yan, Fanfan Liu, Liming Zheng, Yufeng Zhong, Yiyang Huang, Zechao Guan, Chengjian Feng, and Lin Ma. Robomm: All-in-one multimodal large model for robotic manipulation. *arXiv preprint arXiv:2412.07215*, 2024.

[89] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[90] Huanrui Yang, Lin Duan, Yiran Chen, and Hai Li. Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization. *arXiv preprint arXiv:2102.10462*, 2021.

[91] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 1 (2):6, 2023.

[92] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

[93] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

[94] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[95] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[96] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023.

[97] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. *arXiv preprint arXiv:2412.18194*, 2024.

[98] 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. *arXiv preprint arXiv:2503.22020*, 2025.

[99] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[100] Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

[101] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[102] Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. *arXiv preprint arXiv:2501.10105*, 2025.

[103] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[104] Zhongyi Zhou, Yichen Zhu, Minjie Zhu, Junjie Wen, Ning Liu, Zhiyuan Xu, Weibin Meng, Ran Cheng, Yaxin Peng, Chaomin Shen, et al. Chatvla: Unified multimodal understanding and robot control with vision-language-action model. *arXiv preprint arXiv:2502.14420*, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our introduction and abstract clearly reflect the paper's contribution.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitation is in the last paragraph of this paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical results in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the training data, machine, and setting in the Experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

We provide the source code in the supplementary materials. We will release the dataset and model checkpoints at the camera-ready state.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the training details in Appendix **??**.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the testing process (i.e., how many checkpoints in how many evaluation episodes). Although we do not show the error bar, we believe our statement is clear enough.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the computer resources in the Experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our research does not have a very important societal impact since it is for the academic research.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No such dataset.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: This paper's datasets are all from open-sourced datasets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All of the training data annotated in this paper is shown in the data annotation section.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing involved in this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No crowdsourcing involved in this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research is not LLM.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Benchmarks and Dataset Details

A.1 VLM Pretraining Dataset

The LLM backbones we choose are Qwen-2.5 [89] series. Since they are not VLM, we first pretrain it to VLM with LLaVa v1.5 [52] data mixture, which consists of two subsets used for a multi-stage training pipeline. The first subset consists of a 558K sample mixture of examples sourced from various captioning datasets, while the second consists of 665K multimodal instruct tuning examples comprised of synthetic data generated in [52], as well as examples from existing vision-language training sets. According to the conclusion from Prismatic-VLMs [41], we only use the first subset to train the VLM in a single-stage optimization procedure, that is, directly fine-tuning all parameters. We implement the training code with PyTorch using Fully Sharded Data Parallel (FSDP [101]) and BF16 mixed precision and train the VLM with 2 epochs for all Qwen2.5 model types (0.5B, 1.5B, 3B, and 7B). The training hyperparameters are shown in Table 4.

Table 4: Training hyperparameters of VLM for Qwen2.5 LLM.

Hyperparameter	Value
Batch Size	64
Max Gradient Norm	1.0
Weight Decay	0.1
Learning Rate	2e-5
Optimizer	AdamW
Scheduler	Warmup & Cosine Decay
Warmup Ratio	0.03

A.2 LIBERO Dataset

The LIBERO Dataset [51] contains five subsets: LIBERO-Spatial, LIBERO-Object, LIBERO-GOAL, LIBERO-LONG, and LIBERO-90. The first four subsets contain 10 tasks for each of them, with 50 demonstrations for each task. The last subset contains 90 tasks with also 50 demonstrations for each task. All tasks have a language instruction that describes the task. We use the two camera views for all subsets (the agentview and eye-in-hand view). We use a resolution of 224×224 for each view of the image. The action space is 7-dim, containing 6-dim $\delta x, \delta y, \delta z, \delta roll, \delta pitch, \delta yaw$ and 1-dim gripper open/close. We use a history length of 2 and a future action length of 8.

Following OpenVLA [43], we further clean up the original LIBERO datasets by:

- We filter out all "no-op" actions from the dataset, i.e., actions that have near-zero magnitude in the translation and rotation components and do not change the state of the robot's gripper.
- We replay all demonstrations in the corresponding simulation environments and filter out the demonstrations that fail to complete the task (as determined by the environments' success criteria).

A.3 The COLOSSEUM Dataset

For 3D manipulation tasks and generalization experiments, we use The Colosseum [64] as our task benchmark. This benchmark contains 20 single-arm manipulation tasks in simulation. Each task has various variants such as lighting, distractors, physical properties perturbations, and camera pose. The cameras in this benchmark are depth cameras, so we can get the depth map and then get the point cloud observations by fusing all cameras. We follow 3D-DA [42] to preprocess the 3d observations to point cloud tokens. Then we send the point cloud tokens to the action head (or the low-level action head) as additional inputs, together with the original multi-view images. This makes the action heads have the 3D-aware property. For each task, we have 100 demonstrations. The action space is 8-dim, containing 3-dim $\delta x, \delta y, \delta z$ and 4-dim $\delta w, \delta q_x, \delta q_y, \delta q_z$ as the delta quaternion for rotation, and 1-dim gripper open/close. We use a history length of 2 and action length of 8.

A.4 The Real-World Deformable Manipulation Dataset

For deformable object manipulation tasks, we design three real-world deformable object manipulation tasks: unfold the jeans, fold the handkerchief, and straighten the rope, as shown in Figure 4. We use two camera views for these tasks, where a third-view camera is mounted on another X-Arm, and an eye-in-hand-view camera is mounted on the main X-Arm, as shown in Figure 8. We collect 100 demonstrations for each task with human teleoperation. The cameras we use are RealSense D435i. We freeze the rotation of the X-Arm, so the action space is 4-dim: 3-dim $\delta x, \delta y, \delta z$ and 1-dim gripper open/close. The average horizon of these tasks is 214 steps. We also use an observation history length of 2 and a future action length of 8.

(a) The jeans unfold task. (b) The handkerchief folding task. (c) The rope straightening task.

Figure 8: The real-world deformable object manipulation tasks.

A.5 The DexArt Dataset

We use the DexArt [3] benchmark for dexterous manipulation tasks. This benchmark contains four dexterous manipulation tasks built on the Sapien [85] simulator, including *turn on the faucet*, *open the laptop*, *lift the bucket*, and *open the toilet*. The original benchmark is a reinforcement learning benchmark, and they provide the official trained policy checkpoint. We load these checkpoints and collect 100 demonstrations for each task. We use one camera view for each task.

A.6 The FurnitureBench Dataset

For long-horizon complex manipulation tasks, we choose FurnitureBench [32] as our task benchmark. This benchmark provides corresponding simulation environments called FurnitureSim, and it provides demonstrations for four tasks: *cabinet*, *lamp*, *one-leg*, and *round-table*. Each task has 100 demonstrations. The action space is 8-dim, containing 3-dim $\delta x, \delta y, \delta z$ and 4-dim $\delta w, \delta q_x, \delta q_y, \delta q_z$ as the delta quaternion for rotation, and 1-dim gripper open/close. We use three camera views as input.

A.7 The PerAct2 Dataset

For dual-arm manipulation tasks, we choose PerAct2 [28] as our task benchmark. We use five tasks in this benchmark: *handover item*, *lift ball*, *put bottle in fridge*, *straighten rope*, and *sweep to dustpan*. As in The Colosseum, we make this benchmark a 3D task benchmark. Each task has 100 demonstrations. The action space is 22-dim, where 16-dim is for the dexterous hand joint values and 6-dim is for the end-effector. For this dataset, we do not use the image foresight planning.

A.8 The Real-World Rigid-Body Manipulation Dataset

To further verify our conclusions in the real-world setting, we design 5 manipulation tasks in a single-arm manipulation setting, as shown in Figure 9. We collect 50 demonstrations for each task. The action space is 7-dim.

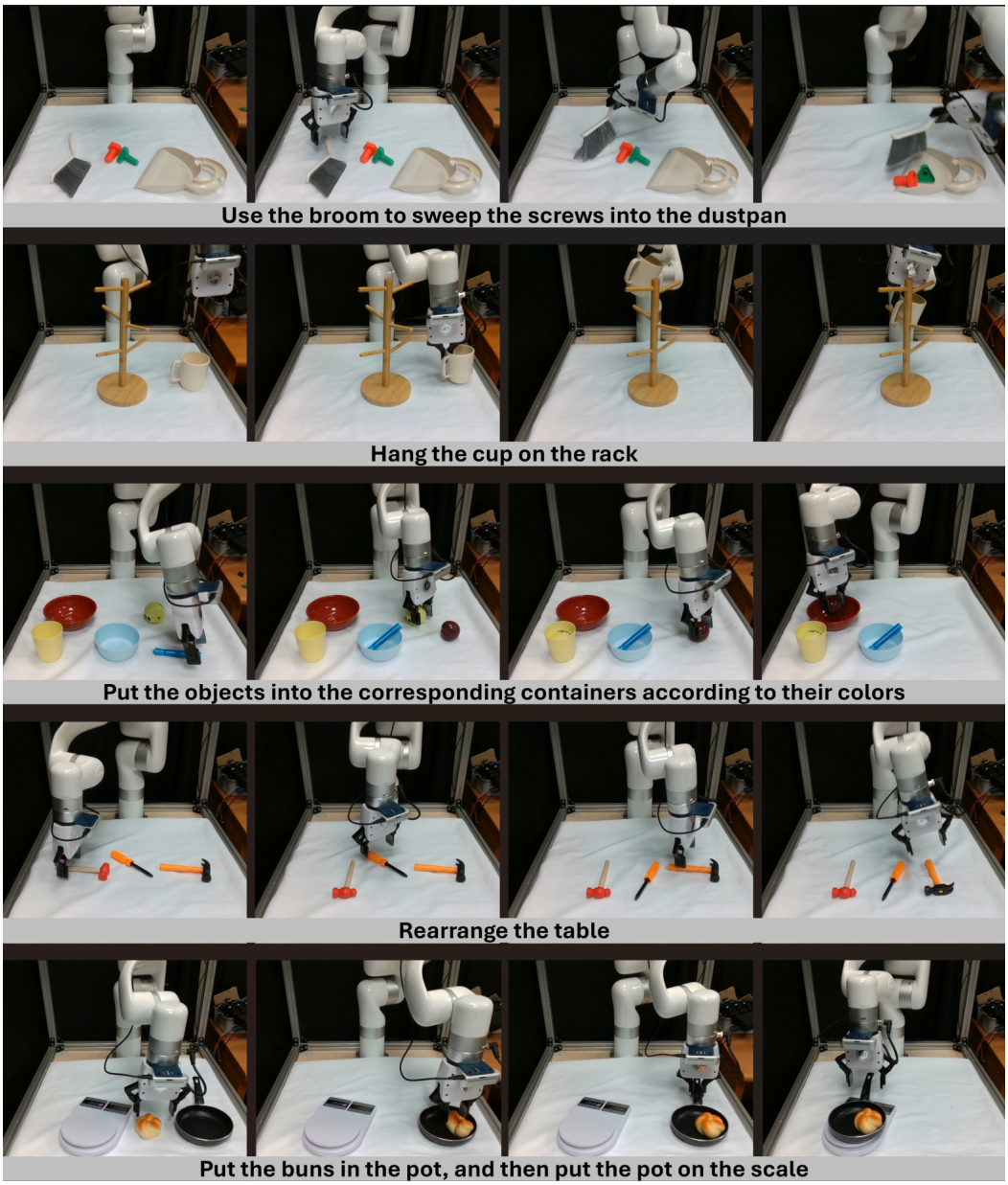


Figure 9: Real world manipulation tasks.

B Reasoning Dataset Annotation

B.1 Language Reasoning Dataset

This dataset contains language-based planning results for the task that understands the scene and decomposes the task, as used in [72, 92, 104, 5]. We design a unified language planning format and structure applicable to all manipulation tasks with 8 different keys, including `Task`, `Plan`, `Subtask`, `Subtask Reason`, `Move`, `Move Reason`, `Gripper Position`, and `Object Bounding Boxes`. For example, for the task *open the top drawer of the cabinet*, the reasoning data should be:

```
TASK: Open the top drawer of the cabinet. PLAN: 1. Approach the
cabinet. 2. Locate the top drawer. 3. Locate and grasp the drawer
handle. 4. Open the drawer. 5. Stop. VISIBLE OBJECTS: akita black bowl
 [100, 129, 133, 155], plate [17, 131, 56, 158], wooden cabinet [164,
75, 224, 175] SUBTASK REASONING: The top drawer has been located; the
robot now needs to position itself to grasp the handle. SUBTASK:
Locate and grasp the drawer handle. MOVE REASONING: Moving left aligns
 the robot's end effector with the drawer handle. MOVE: move left
GRIPPER POSITION: [167, 102, 166, 102, 165, 102, 164, 102, 162, 102,
161, 102, 160, 102, 158, 102, 156, 102, 154, 102, 153, 102, 151, 102,
149, 102, 147, 102, 145, 102, 143, 102]
```

Similarly to EmbodiedCoT [95], we provide an overview of our data labeling pipeline in Figure 3. To obtain a comprehensive understanding of the scene, we first query the Prismatic-7B VLM [41], which outputs a detailed scene description. Next, we derive low-level motion primitives by analyzing the robot's proprioceptive state across a 10-step prediction horizon, assuming a static camera viewpoint, and translating these movement traces into a set of pre-defined action templates (e.g., "move left", "move up"). To construct the full reasoning trace, we use Gemini1.5 [74] to synthesize higher-level plans. Given the task instruction, scene description, and step-wise movements, Gemini1.5 generates a structured plan that includes a sequence of sub-tasks, as well as the specific sub-task relevant to each step. Additionally, it provides concise justifications for both the movement taken and the associated sub-task.

However, during experiments, we observed that the quality of the generated reasoning, referred to as *initial reasoning* in Figure 3, was often suboptimal, exhibiting two major issues. First, there was inconsistency in the planning outputs: even for the same task, the language descriptions of sub-tasks varied significantly. This stems primarily from the inherent randomness in responses from large language models such as Gemini1.5. Second, we found a mismatch between the generated plans and the actual trajectories. This issue was particularly pronounced in complex, long-horizon tasks (e.g., FurnitureBench [32]), where the provided inputs—task instruction, scene description, and step-wise movement primitives—were insufficient for the model to infer coherent and accurate planning steps. As a result, the low quality of the *initial reasoning* posed challenges for training the planning head, as the model struggled to learn meaningful mappings from observations to such plannings.

To address these issues, we applied a filtering and refinement process to the *initial reasoning*. Specifically, for each task, Gemini or human experts selected and edited the task descriptions and high-level plans produced by Gemini to ensure consistency across episodes of the same task. Once a canonical task and plan were established, we prompted Gemini again to regenerate the step-wise reasoning under this fixed structure. This process yielded the *final reasoning* in Figure 3, which aligns better with the trajectories and provides more coherent supervision for training the planning head.

In addition to the reasoning generated by Gemini, we also incorporate object bounding boxes and gripper positions into the final annotations. For real-world data, we adopt a labeling strategy similar to EmbodiedCoT [95], leveraging vision-language models to annotate object locations from visual inputs. For simulation data, we exploit the availability of camera intrinsics and extrinsics to project 3D gripper positions into 2D image coordinates. Object bounding boxes can also be directly extracted using simulator-provided segmentation masks, enabling efficient and accurate annotation of the visual scene.

Finally, we represent language planning in the following format:

- Task: A concise natural language description of the goal the robot needs to achieve.

- Plan: A high-level sequence of steps to accomplish the task, typically numbered and described in imperative language.
- Subtask: A mid-level action derived from the plan, typically one step at a time, to be executed next.
- Subtask Reason: A rationale explaining why the current subtask is necessary or meaningful in context.
- Move: A specific low-level movement command to guide the robot toward completing the subtask.
- Move Reason: A justification of the chosen movement, often grounded in spatial alignment or task constraints.
- Gripper Position: A list of 2D coordinates that define the intended trajectory or position of the robot's gripper in image space. This often reflects the gripper's pixel-level alignment with the target object.
- Object Bounding Boxes: A list of objects currently detected in the scene, each annotated with a bounding box in pixel coordinates $[x_1, y_1, x_2, y_2]$

B.2 Visual Reasoning Dataset

This dataset will generate visual representations in the language format for task planning. Compared to pure language-based representations, these visual representations have better spatial semantic information and are more grounded in the input images, which are used in recent multi-modal learning works [61, 86]. In this work, we use three keys, including `object bounding boxes`, `end-effector flow`, and `target object affordance` as the visual planning representations.

As shown in Figure 3, we use discrete location tokens on the input image to represent visual planning results. For an image with width W and height H, we evenly divide both the width and height into P segments each, thus we use $P \times P$ discrete bins to represent the visual pictures and each bin consists of $(W/P) \times (H/P)$ pixels. We use a new location token `<loc i>` to represent the i-th bin token from top-left to bottom-right, and increase the tokenizer's word vocabulary to add these bin tokens. For bounding boxes, we use the top-left and bottom-right bins to represent them. For end-effector flows, we use a sequence of bins to represent them. For affordances, we use a region of bins to represent the target regions. In this work, $W = H = 224$, and $P = 32$, i.e., each bin consists of 7×7 pixels. For example, for the task *Put the cream cheese box and the butter in the basket*, the visual reasoning data should be:

```
VISUAL OBJECT BBOXES: alphabet soup <loc_500, loc_632>, cream cheese <
loc_353, loc_452], tomato sauce <loc_461, loc_624>, ketchup <loc_341,
loc_503>, orange juice <loc_538, loc_767>, milk <loc_563, loc_791>,
butter <loc_684, loc_783>, basket <loc_448, loc_775>. VISUAL EE FLOW:
loc_387, loc_387, loc_387, loc_419, loc_419, loc_419, loc_419, loc_419
, loc_419, loc_419, loc_419, loc_451, loc_451, loc_451, loc_451,
loc_451>. VISUAL AFFORDANCE: loc_354, loc_355, loc_356, loc_386,
loc_387, loc_388, loc_418, loc_419, loc_420>
```

Specifically, given a manipulation task $\mathcal{T}$ consisting of N steps $(i.e. 1, 2, ..., N)$, we take the following steps to generate visual-based planning representations $\{\mathcal{V}_i^{box}, \mathcal{V}_i^{flow}, \mathcal{V}_i^{afford}\}_{i=1}^N$:

1. **Object Bounding Boxes:** We first get instance semantic maps $S = \{S_i\}_{i=1}^N \in \mathcal{R}^{N \times H \times W}$ from the simulation engine to compute binary masks for each object in each frame. Next, we sequentially apply $cv2.morphologyEx()$ to reduce noise and reconnect fragmented regions, $cv2.findContours()$ to detect object contours, and $cv2.boundingRect()$ to compute the rectangular bounding box for each detected object. Finally, we annotate the location token of the top-left and bottom-right bins for each bounding box. The final bounding box visual annotation for task $\mathcal{T}$ can be formulated as $\left\{\mathcal{V}_i^{box}\right\}_{i=1}^N$, where $\mathcal{V}_i^{box} = \left\{(loc_j^{tl}, loc_j^{br})\right\}_{j=1}^{m_i}$.
2. **End-effector Flow:** The end-effector flow visual annotation is obtained by directly labeling the location tokens corresponding to the gripper positions in the language-based planning representation. Formally, the end-effector flow annotation for task $\mathcal{T}$ can be formulated as $\left\{\mathcal{V}_i^{flow}\right\}_{i=1}^N$, where $\mathcal{V}_i^{flow} = loc_i^{gripper}$.

3. **Object Affordance:** The object affordance is represented as a heatmap centered on the target object to be fetched. We first identify the target object by detecting changes in all bounding boxes (e.g. shifts in location or variations in size). Next, we employ the pretrained SAM2 [68] model to infer a precise object mask within the target bounding box. Finally, we compute a Gaussian heatmap centered at the gripper position within the object mask to model the affordance. Location tokens corresponding to regions with affordance values exceeding a predefined threshold are then annotated in a top-left to bottom-right order. The final object affordance annotation for task $\mathcal{T}$ can be formulated as $\left\{\mathcal{V}_i^{afford}\right\}_{i=1}^{N}$, where $\mathcal{V}_i^{afford} = \{loc_j\}_{j=1}^{n_i}$.

B.3 Image Foresight Reasoning

Image Foresight (IF) reasoning dataset will imagine a future goal frame as the most general representation for task planning. There is no special effort here to label the goal image. We just select the future image from the trajectory.

Here we want to introduce more about the image generation head. In this work, we use an image generation head for planning based on [31]. It auto-regressively generates the image in a coarse-to-fine paradigm proposed by [76]. Given an input image, it iteratively quantizes the residual image following a coarse-to-fine resolution schedule $\{(h_k, w_k)\}_{k=1}^{K}$. It also applies a technique called **B**itwise **S**elf-**C**orrection (BSC) to mitigate the performance gap between training and testing caused by teacher-forcing training.

Formally, inside each quantization iteration k, the tokenizer does the following steps:

1. **Calculate and Quantize Residual:** It computes the difference between the original raw feature $\boldsymbol{F}$ and the reconstructed flipped feature from the previous iteration ($\boldsymbol{F}_{k-1}^{flip}$). This residual is then interpolated to the current resolution (h_k, w_k) and quantized following [90] to produce tokens at the current resolution $\boldsymbol{R}_k = \mathrm{quantize}(\mathrm{down}(\boldsymbol{F} - \boldsymbol{F}_{k-1}^{flip}, (h_k, w_k)))$.
2. **Apply Random Flipping For BSC:** A random flipping operation ($\mathrm{Random_Flip}(\cdot)$) is applied to the quantized residual $\boldsymbol{R}_k$ based on a probability p. This results in the flipped residual $\boldsymbol{R}_k^{flip} = \mathrm{Random_Flip}(\boldsymbol{R}_k, p)$.
3. **Reconstruct Flipped Feature:** The algorithm reconstructs the cumulative flipped feature $\boldsymbol{F}_k^{flip}$ up to the current iteration. It does this by interpolating *all* previously generated flipped residuals ($\boldsymbol{R}_i^{flip}$ for i from 1 to k) to the original image resolution (h, w) and sums them together: $\boldsymbol{F}_k^{flip} = \sum_{i=1}^{k} \mathrm{up}(\boldsymbol{R}_i^{flip}, (h, w))$.

During inference, generation starts from a global conditioning signal, for example, the text embedding in a T2I generation setting. Notably, it generates *all* tokens of a resolution at once, distinguishing this method from the raster-scanning generation paradigm.

We select [31] as our image generation head based on three primary advantages. Firstly, it surpasses the state-of-the-art diffusion-based models [15, 21, 63] in performance on academic benchmarks and in human preference evaluations. Secondly, [31] achieves lower inference latency compared to prevalent diffusion models, a critical requirement for embodied planning within the hierarchical VLA framework. Third, our experiments indicate that the training loss of [31] serves as a stronger predictor of the final quality of foresight image generation while necessitating fewer hyperparameter adjustments, such as the noise scheduling required by the diffusion models. In practice, when the loss drops below 0.1, it indicates that the training is complete.

C VLA-OS Model Details and Continual Learning Experiments

C.1 Action Head Details

The action head for all VLA-OS models is in the same architecture, with only different numbers of layers. It is a block-wise causal attention transformer with the same number of layers as the LLM backbone, as introduced in Section 3.2. Let $[KV_1, \cdots, KV_n]$ be the KV tokens from the LLM, $[t]$ be the denoising timestep embedding token, $[q]$ be the proprioceptive token, and $[a_t, \cdots, a_{t+H-1}]$ be the action token, sequentially. The tokens in each block can attend to itself and blocks before it, but cannot attend to blocks after it. The hyperparameters of the action head are shown in Table 5.

Table 5: Hyperparameter of the action head transformer.

	Layer Number	Hidden Size	Dropout	Head	Non-Linear Func
Action Head S	24	512	0.1	8	GELU
Action Head S	28	512	0.1	8	GELU
Action Head S	36	512	0.1	8	GELU
Action Head S	28	512	0.1	8	GELU

The low-level action head used for VLA-OS-H is also a transformer. It has a separate convolutional neural network (CNN) for encoding the input images, the visual planning images, and the image foresight image. For other parts, it keeps the same setting as the normal action head.

C.2 Planning Head Details

All three planning head transformers share the same network structure (the VAE encoder and decoder of the image foresight planning head are frozen). The planning head takes as input the keys and values from each layer of the LLM backbone. The hyperparameters of the planning head are shown in Table 6.

Table 6: Hyperparameter of the planning head transformer.

	Layer Number	Hidden Size	Dropout	Head	Non-Linear Func
Action Head S	24	512	0.1	8	GELU
Action Head S	28	512	0.1	8	GELU
Action Head S	36	512	0.1	8	GELU
Action Head S	28	512	0.1	8	GELU

C.3 Training Loss Details

The action heads can be trained with either L1 behavior cloning loss, or the flow matching loss. L1 loss is shown to be better than L2 MSE loss for VLA [43, 44]. The L1 loss is:

$$\mathcal{L}_{L1}(\theta) = \mathbb{E}_{s,a\in\mathcal{D}}\big|\pi_\theta(s) - a\big|. \tag{1}$$

The flow matching loss is:

$$\mathcal{L}_{FM} = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),t\sim U(0,1),s,a\sim\mathcal{D}}||\pi_\theta(x_t, t|s) - u_t||_2^2, \tag{2}$$

where $x_t = (1-t)\epsilon + ta$ and $u_t = \frac{d}{dt}x_t = a - \epsilon$.

The planning head losses for the language planning head and the visual planning head are the standard next-token prediction loss. The loss for the image foresight planning head follows the original paper [31].

C.4 Training Details of Hierarchical-VLAs

The training process of Hierarchical-VLAs can have multiple choices, since they inherently incorporate two models that are not connected by backward gradients. In this work, we aim to reuse

the trained model to the greatest extent possible to reduce the cost of repeated training. Thus, for Hierarchical-VLAs, we first borrow the trained VLMs backbone as well as the planning heads from Integrated-VLAs, and finetune them on the planning datasets to get the high-level models for the Hierarchical-VLAs. Later, for the low-level model, we extract the Keys and Values from the high-level LLM backbone and use them as the embedding of the input visual and language signals and send them to each layer of the low-level action head. For the planning outputs from the high-level planning heads, we use different models to encode them: we use a frozen Qwen-2.5 7B [89] model to encode the language planning outputs to get the sentence embeddings, a common Convolutional Neural Network (with 6-channel inputs of the current 3-channel image and a 3-channel visual planning results) to encode the visual planning outputs, and a common Convolutional Neural Network (with 3-channel inputs of the goal image) to encode the image foresight planning outputs. The gradient of the low-level action head will not go backward through the high-level VLM backbone.

C.5 Continual Learning Experiments of Different VLA Paradigms

Continual learning for robot imitation learning [51, 24] measures the degree to which the VLA model forgets old tasks when continuously learning new tasks. In this part, we test the continual learning capacities of three paradigms and three representations on 10 tasks of LIBERO-LONG sequentially. We only use Sequential Finetuning (SEQL) as our lifelong learning algorithm. We use the original metrics from LIBERO [51], including forward transfer (FWT), negative backward transfer (NBT), and area under the success rate curve (AUC). Denote $c_{i,j,e}$ as the agent's success rate on task j when it learned over $i-1$ previous tasks and has just learned e epochs ($e \in 0, 2, \cdots, 20$) on task i. Let $c_{i,i}$ be the best success rate over all evaluated epochs e for the current task i (i.e., $c_{i,i} = \max_e c_{i,i,e}$). Then, we find the earliest epoch e_i^* in which the agent achieves the best performance on task i (i.e., $e_i^* = \arg\min_e c_{i,i,e_i} = ci, i$), and assume for all $e \geq e_i^*, c_{i,i,e} = ci, i$. Given a different task $j \neq i$, we define $c_{i,j} = ci, j, e_i^*$. Then the three metrics are defined as follows:

$$
\begin{aligned}
\mathrm{FWT} &= \sum_{k\in[K]} \frac{\mathrm{FWT}_k}{K}, \quad \mathrm{FWT}_k = \frac{1}{11} \sum_{e\in\{0\ldots50\}} c_{k,k,e}, \\
\mathrm{NBT} &= \sum_{k\in[K]} \frac{\mathrm{NBT}_k}{K}, \quad \mathrm{NBT}_k = \frac{1}{K-k} \sum_{\tau=k+1}^{K} \left(c_{k,k} - c_{\tau,k}\right), \\
\mathrm{AUC} &= \sum_{k\in[K]} \frac{\mathrm{AUC}_k}{K}, \quad \mathrm{AUC}_k = \frac{1}{K-k+1} \left(\mathrm{FWT}_k + \sum_{\tau=k+1}^{K} c_{\tau,k}\right).
\end{aligned}
\tag{3}
$$

Results are shown in Table 7 and Table 8.

Table 7: Continual learning results on LIBERO-LONG of three different VLA paradigms. The VLA-OS-I and VLA-OS-H are trained with three planning representations together.

	FWT(↑)	NBT(↓)	AUC(↑)
VLA-OS-A	0.71	**0.32**	0.25
VLA-OS-I	0.75	0.43	0.29
VLA-OS-H	**0.80**	0.45	**0.32**

Table 8: Continual learning results on LIBERO-LONG of three different planning representations (Language (L), Visual (V), and Image Foresight (IF)) on VLA-OS-I.

	FWT(↑)	NBT(↓)	AUC(↑)
L	0.72	0.47	0.26
V	0.74	0.40	**0.28**
IF	**0.75**	**0.39**	0.27

Finding 13: VLA paradigms with task planning (Integrated-VLA and Hierarchical-VLA), compared to the non-planning paradigm (ActionOnly-VLA), achieve higher forward transfer but incur faster forgetting.

Finding 14: Visually grounded planning representations deliver superior forward transfer and exhibit slower forgetting relative to language-based planning representations.