HYBRID TRAINING FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Using Large Language Models to produce intermediate thoughts, a.k.a. Chain-of-thought (CoT), before providing an answer has been a successful recipe for solving complex language tasks. In robotics, similar embodied CoT strategies, generating thoughts before actions, have also been shown to lead to improved performance when using Vision-Language-Action models (VLAs). As these techniques increase the length of the model's generated outputs to include the thoughts, the inference time is negatively affected. Delaying an agent's actions in real-world executions, as in robotic manipulation settings, strongly affects the usability of a method, as tasks require long sequences of actions. However, is the generation of long chains-of-thought a strong prerequisite for achieving performance improvements? In this work, we explore the idea of Hybrid Training (HyT), a framework that enables VLAs to learn from thoughts and benefit from the associated performance gains, while enabling the possibility to leave out CoT generation during inference. Furthermore, by learning to conditionally predict a diverse set of outputs, HyT supports flexibility at inference time, enabling the model to either predict actions directly, generate thoughts or follow instructions. We evaluate the proposed method in a series of simulated benchmarks and real-world experiments.

1 INTRODUCTION

Despite recent advances in robotics, truly generalist robot policies have long been elusive. Thanks to the joint efforts of collecting large-scale robot data (O'Neill et al., 2024) and making large Vision Language Models (VLM) open-source (Steiner et al., 2024; Tong et al., 2024), we have entered a new era in robotics foundation models. By fine-tuning VLMs on robotic datasets containing actions, we obtain so-called **Vision-Language-Action models (VLAs)** (Kim et al., 2024; Brohan et al., 2023b;a): large policy models that are trained end-to-end to take language instructions and raw camera images as inputs, and output low-level robotic actions.

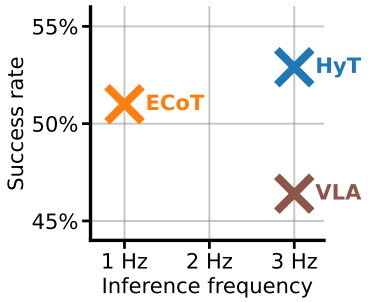


Figure 1: **Hybrid Training (HyT)** of VLAs increases the agent's performance similarly to **ECoT**, but also maintains the same fast inference as standard **VLAs**. Performance refers to the ClevrSkills experiments (9 tasks, 3000 demos) in the Experiments section.

VLAs possess several advantages over previous work, such as multimodal prompting of the agent and the availability of knowledge from the base pre-trained VLM. However, generalization to out-of-distribution (OOD) settings, e.g., task configurations not available in the robotics fine-tuning dataset, remains challenging. Indeed, the knowledge of the agent is vast about general concepts, but remains limited in the robotics settings, where the data distribution is often narrow.

In order to further unleash the capabilities of VLAs with little data, recent works have trained models to predict intermediate outputs, representing the agent's intentions, before predicting actions (Zawalski et al., 2024; Zhao et al., 2025). One notable example is **Embodied CoT (ECoT)** (Zawalski et al., 2024), where the VLA learns to output useful information about the given task in language form (Wei et al., 2023), before generating the actions to execute. This not only has shown to improve performance, but it

also allows humans to more easily interpret the agent's intentions and potentially intervene on them, i.e. correcting the agent's intentions, before action generation. However, due to the intermediate reasoning outputs generated before actions, the action inference frequency of these models can be significantly lower.

The human cognition process from observation to action is hypothesized to leverage the interaction of two systems (Kahneman, 2011). The fast and intuitive **System I** handles most daily tasks, taking control in contexts that our brain judges as unchallenging. The slow and deliberate **System II** is activated when decisions require additional computation, such as comparing options or processing complex information. The tendency of the brain is to delegate as many decisions as possible to System I, to save energy and time. However, in order to do so, humans need to improve their capabilities to deal with complex decisions. This is done by developing a **skilled intuition** (Kahneman & Klein, 2009; Simon, 1992) that allows leveraging previously learned cues to solve familiar tasks, effortlessly.

In this work, we explore the hypothesis that VLA models can similarly develop skilled intuition, when trained with the right objective. Learning from CoT reasoning traces, a model can further internalize knowledge about environments and tasks. Then, at test-time, the model more eagerly recognizes patterns and can leverage such knowledge to generate actions, even in the absence of intermediate thoughts generation. With this hypothesis in mind, we develop a **Hybrid Training (HyT)** framework, which allows the agent to learn from a combination of CoT and actions data.

Hybrid training presents a more flexible learning objective, which encompasses both ECoT and standard VLAs. During training, we implement the HyT objective using a Monte Carlo estimate, consisting of sampling a variety of conditional inputs and outputs with different probabilities. The model learns to predict a set of ouputs, modelling a multitude of conditional action probabilities, which mainly depends on a newly introduced *modality* variable. During test-time, it's the modality variable that allows us to influence the model's generation. By default, the modality variable conditions the model to directly predict actions. This allows VLAs trained with HyT to maintain the same inference time as standard VLAs, while benefitting from training on reasoning traces.

Furthermore, the modality variable can be used for manipulating the VLA into operating in different inference modes. The "act" mode, as mentioned, resembles standard VLA's inference and allows to generate actions directly. In addition, we show a "think" mode, where the VLAs generates intermediate thoughts as in ECoT, and a "follow" mode, where the VLA follows a set of provided intentions, e.g. by a human or an oracle, similarly to the lower level policies in hierarchical systems (Shi et al., 2025; Hafner et al., 2022).

In addition to investigating and validating our proposed HyT framework, on a set of simulated benchmarks (ClevrSkills (Haresh et al., 2024), LIBERO (Liu et al., 2023)) and real-world tasks (on a UFactory xArm 6), we aim to address a fundamental question regarding VLA models:

What is the contribution of CoT techniques to VLAs performance?

2 Related Work

Vision-Language-Action models. Open-source efforts in the robotics field, such as the Open X-Embodiment dataset (O'Neill et al., 2024), have fueled progress in the development of large VLAs (Kim et al., 2024; Black et al., 2024; Wen et al., 2024; Ghosh et al., 2024; Jiang et al., 2023). Recent works have also explored hierarchical VLA architectures (Shi et al., 2025; NVIDIA et al., 2025), showing they can be beneficial for solving open-ended and long-horizon tasks. Our work aims to improve VLA's performance, by improving the way available reasoning annotations and action data is used, independently of the architecture used.

Chain-of-Thought (CoT) and reasoning. Generating a chain of thought has shown improved performance in LLMs solving complex reasoning tasks (Wei et al., 2022). Additionally, reasoning has shown notable success using RL with verifiable rewards, coupled with Supervised Finetuning (SFT) on example reasoning traces (DeepSeek-AI et al., 2025; Havrilla et al., 2024). CoT techniques specifically for VLMs (Shao et al., 2024) and VLAs have also been researched (Zawalski et al., 2024; Zhao et al., 2025). In particular, ECoT (Zawalski et al., 2024) shows that embodied thoughts can greatly improve the agent's predictions in robotics, despite the higher inference costs. Our work

grounds on their findings and proposes a method that accomplishes both strong performance and fast inference.

Hybrid reasoning. Recent works have attempted to distill slow thinking capabilities into faster models (Deng et al., 2024; Yu et al., 2024). Closely related to our method is DualFormer (Su et al., 2025), which proposes to train a language model by systematically dropping out reasoning traces. In robotics domain, RFST (Zhu et al., 2024) proposes a hierarchical setup that uses a discriminator to decide whether to switch to the fast or slow system, with the respective model of the chosen mode being then used as the policy. Our work, instead, provides a method to train a single hybrid system that learns to conditionally generate a variety of outputs.

3 PRELIMINARIES

Vision-Language Action (VLA) models are multimodal policies generally trained with imitation learning. A VLA processes language inputs through a Transformer-based LLM architecture (Vaswani et al., 2023; Brown et al., 2020; Team et al., 2024). Language is first "tokenized" into language tokens that are then processed by the LLM. Similarly, VLAs can process visual inputs through a vision encoder, e.g., a vision transformer (Dosovitskiy et al., 2021), that transforms image patches into visual tokens, which are then processed by the LLM.

Given a language description of a task l, the goal of the VLA policy is to solve the task in a given environment. The policy observes the environment through images x, generally captured by a camera in the environment. The policy interacts with the environment using actions a. Through imitation learning, the policy's objective is to learn, at each discrete timestep, the distribution $p(a_t|x_t, l)$ that solves the given task, which is empirically observed from a dataset of demonstrations.

In addition to predicting actions, thinking VLAs, like ECoT (Zawalski et al., 2024), also generate intermediate language outputs. These reasonings are expressed as thoughts τ in language form, predicted by the model. Generally, thoughts include information about the overall plan of action, the current subtask to execute, the location of objects in the image, or the direction of the agent's ongoing motion (Zawalski et al., 2024). Thinking VLA models are trained to predicts the joint probability distribution over actions and thoughts: $p(a_t, \tau_t|x_t, l) = p_\theta(a_t|x_t, l, \tau_t)p_\theta(\tau_t|x_t, l)$.

Thinking VLAs learn a single set of parameters θ to predict both actions and thoughts. Hierarchical VLAs (Shi et al., 2025; NVIDIA et al., 2025) use a two-level hierarchy of models, where one model provides an actionable language instruction for solving the task, while the second model executes the plan. By treating high-level plans and thoughts interchangeably, action prediction in hierarchical VLAs can be modelled as: $p(a_t, \tau_t|x_t, l) = p_{\theta_l}(a_t|x_t, \tau_t)\, p_{\theta_h}(\tau_t|x_t, l)$, where θ_h denotes the parameters of the "high-level" model and θ_l of the "low-level" model.

4 METHOD

VLAs that learn to predict thoughts and actions, like thinking and hierarchical VLAs, have demonstrated improved performance in several works (Shi et al., 2025; NVIDIA et al., 2025; Zawalski et al., 2024; Zhao et al., 2025). Compared to standard VLAs, these models learn to: (i) predict thoughts in language form, and (ii) condition the actions' probability on the generated thoughts. Thoughts in language form generally consist of significantly more tokens than their action counterpart. Thus, generating thoughts at test-time comes at a high inference cost. This can significantly slow down the agent's action execution in the environment.

We hypothesize that the primary benefits of these models arise not from the generated thoughts themselves, but from the knowledge learned by the model through thought prediction and thought-conditioned actions prediction. This suggests that the model refines its capabilities by internalizing the patterns present in the thoughts, akin to the development of intuitive expertise (Kahneman & Klein, 2009). Under this hypothesis, after a learning process that involves thoughts conditioning and thoughts prediction, a VLA should be able to predict actions with higher accuracy, independently of the presence of thoughts as an intermediate output.

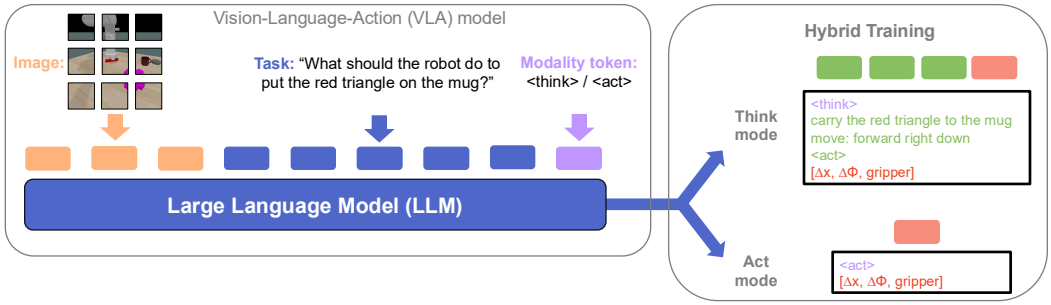


Figure 2: **Hybrid Training (HyT) framework.** Given a set of inputs, on the left, including a modality variable, the VLA model learns to conditionally generate a variety of outputs. Examples for the 'think' and 'act' conditional distributions are presented on the right.

To address the need for agents capable of producing multiple probability distributions within a single model, we introduce a new training strategy, *Hybrid Training* (HyT), designed to integrate structured reasoning with flexible policy learning.

Definition 4.1 (Hybrid Training) *Given a task description l and the current environment observation x_t, the conditional distribution over actions a_t can be expressed as:*

$$p(a_t|x_t, l) = \sum_i \sum_j p_\theta(a_t, \tau^i, m^j|x_t, l) = \sum_i \sum_j p_\theta(a_t, \tau^i|x_t, l, m^j)p(m^j), \tag{1}$$

by marginalizing out thoughts τ and the set of values assumed by the "modality" variable m.

The hybrid training formulation allows us to describe a singla VLA model, with parameters θ, that learns different conditional action distributions, mainly depending on a modality variable.

4.1 Hybrid Training Implementation

The HyT formulation presented in Eq. 1 generalizes the definition of previous VLAs and enables the possibility to combine multiple objectives into a single model. In particular, leveraging the insights from other VLA models, we can use the hybrid training framework to conditionally learn three distributions:

$$p(a_t|x_t, l) = \underbrace{p_\theta(a_t|x_t, l, m^a)p_\theta(m^a)}_{\text{act}} + \underbrace{p_\theta(a_t|x_t, l, \tau_t)p_\theta(\tau_t|x_t, l, m^\tau)p_\theta(m^\tau)}_{\text{think}} + \underbrace{p_\theta(a_t|x_t, \tau_t, m^f)p_\theta(m^f)}_{\text{follow}}. \tag{2}$$

This way the agent will learns to generate different outputs depending on the modality variable value $m \in \{m^a, m^\tau, m^f\}$. The meaning of each distribution is defined as follows:

- **"Act" distribution**: similarly to standard VLAs, it instructs the model to directly predicts actions. It follows from $p_\theta(\tau = \emptyset|m^a) = 1$.
- **"Think" distribution**: similarly to ECoT, it instructs the model to first predict intermediate thoughts and then generate actions.
- **"Follow" distribution**: similarly to a low-level policy in a hierarchical system, it instructs the model to closely follow provided thoughts/instructions for action prediction. It follows from $p_\theta(a_t|x_t, \tau_t, m^f) = p_\theta(a_t|x_t, l, \tau_t, m^f)$ (conditional independence) and $p_\theta(\tau = \emptyset|m^f) = 1$.

Referring to our hypothesis, combining the three distributions should enable the agent to learn to output actions directly (act distribution) while also learning to predict thoughts and to follow instructions. The learning objective for the model can be described as:

$$\min_\theta \mathcal{L}_{\text{hyt}}(\theta) = w_\alpha \mathcal{L}_{\text{act}}(\theta) + w_\tau \mathcal{L}_{\text{think}}(\theta) + w_f \mathcal{L}_{\text{follow}}(\theta) \tag{3}$$

where all the terms $\mathcal{L}_\square$ are negative log-likelihood losses with respect to the corresponding outputs (actions or thoughts and actions)[1].

[1]For action prediction, the loss is negative log-likelihood in case the actions are discretized and predicted by the LLM. Alternatively, the loss can be an L1 or L2 loss, for continuous action prediction.

While one could compute the loss directly as the weighted sum of the three terms, for each datapoint, this would reduce variability in the batches, as the model would contain the same thoughts and actions multiple times in the same batch. In order to avoid this, we propose to use a Monte Carlo estimate of the objective in Equation 3. Rather than being used as loss coefficients, the weights w_a, w_τ, w_f define the probabilities of sampling the corresponding inputs and outputs for the model during training. This way, at each batch sampled during training, the model receives modality tokens, actions and thoughts, with probabilities that are defined by the coefficients.

To summarize, in order to train a VLA with HyT, one needs to specify: (i) the probability distributions to be learned, (ii) a set of values to use for the modality variable, and (iii) a set of coefficients/probabilities for sampling data during training. In this work, we adopt the distributions defined in Eq. 2, with modality variables defined as simple texts, such as "$< act >$" or "$< think >$", and with the set of coefficients $\{w_a : 0.25, w_\tau : 0.5, w_f : 0.25\}$. A simplified diagram illustrating HyT (with only two kinds of outputs) is shown in Figure 2.

4.2 Inference Time

At test-time, the VLA is tasked to predict actions that solve a task. Thanks to HyT, it is possible to predict actions directly, leveraging the model's "act" distribution. In order to do so, we provide the model with the task description, the environments inputs, e.g. a camera image, and the modality variable $m^a =< act >$ which instructs the model to directly output actions. This enables the model to leverage the more extensive knowledge coming from learned thoughts and actions, without incurring any additional inference costs compared to standard VLAs.

While the main goal of this work is to show that HyT enables higher performance at high control frequency, the flexibibility of the HyT framework also enables the possibility to instruct the agent to *operate in different "modes"*, i.e. conditionally predict different forms of outputs. Other than using the VLA in 'act' mode, it is indeed possible to use the model in 'think' or 'follow' modes.

Q: What's the value in using other inference modalities?

While we expect no major performance difference, e.g. when operating the model in 'think' mode rather than in 'act' mode, additional inference modes support greater flexibility in the use of the VLA. For example, the 'think' mode can be used to read the VLA's intentions, for greater interpretability. Instead, the 'follow' mode enables to provide a set of more fine-grained embodied instructions for the agent to follow, such as 'move to the left' or 'pick up the object below the gripper'. Note that, as also shown in the ECoT work (Zawalski et al., 2024), the 'thought' mode can also enable instruction-following, by overriding the agent's intentions. We verify whether these modalities can be useful with HyT in the Experiments section.

Q: How to operate the VLA in different modalities?

Generally, we set the modality variable to $m^a =< act >$, enforcing the model to directly predict actions. If we want to use a different mode, e.g. to predict intermediate thoughts, we can use the $m^\tau =< think >$ variable. In practice, we observed that models trained with HyT always diligently attend to the modality token and generate their outputs accordingly.

In this work, we do not explore the possibility of switching the modality token dynamically during executions, as we observed that, after HyT training, different operating modes have different generative behaviors, but similar performance. The modality variable is thus set at the beginning of a task execution and not updated during the episode. Further understanding and studying the tradeoffs and potential advantages that could derive from a switching mechanism between different modalities is left for future work.

5 Experiments

We validate the proposed HyT method in a series of simulated benchmarks, which we use to extensively assess the framework's capabilities, and on a set of real-world tasks, which demonstrates the practical utility of the approach.

General setup. Across all experiments, the task description is provided in the same format, i.e. *"What should the robot do to {task}?"*. Image inputs are 224×224 RGB images from

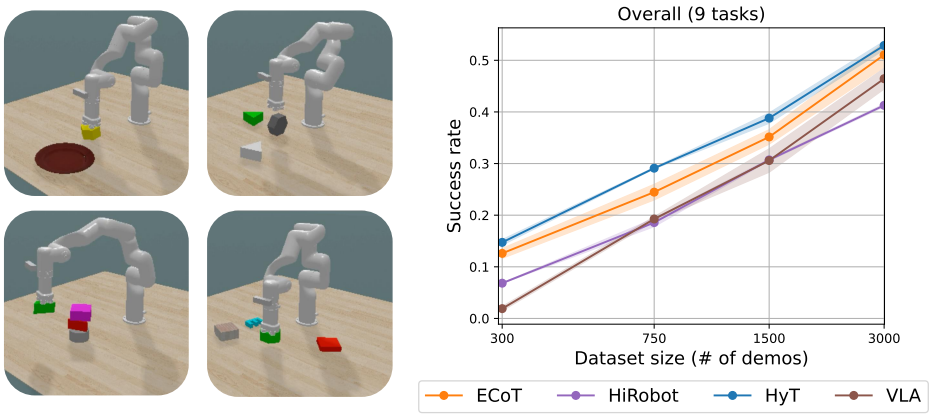


Figure 3: **ClevrSkills benchmark** Aggregated performance on 9 ClevrSkills environments (examples shown on the left). Shaded areas indicate standard errors.

a camera pointing at the robot's workspace. The action space is 7-dimensional and defined as: $[\Delta \mathbf{x}, \Delta \boldsymbol{\phi}, \text{gripper}]$. The end-effector position $\mathbf{x}$ and orientation $\boldsymbol{\phi}$ are controlled in delta space, while the gripper pose is in absolute value.

5.1 CLEVRSKILLS BENCHMARK

The ClevrSkills benchmark (Haresh et al., 2024) is based on the ManiSkill2 manipulation environments (Gu et al., 2023). The environment includes an oracle solver to collect demonstrations, and reasoning traces from the oracle, which we can use to discriminate subtasks and extract thoughts. ClevrSkills also adopts a vacuum gripper and mostly uses simple shapes and objects, isolating the challenges of manipulating complex objects, and focussing on planning and generalization instead.

Baselines. In this Section, we focus on comparing the HyT training methodology with other training paradigms, namely the standard VLA training (Kim et al., 2024), ECoT-like thinking VLA training (Zawalski et al., 2024), and HiRobot-like hierarchical VLA training (Shi et al., 2025).

Dataset. Using the oracle, we collect a diverse dataset that spans three main task groups: Place At, Place OnTop, and Stack Tower, with several variations in terms of object types and numbers. The overall dataset is made of 3000 trajectories. For all the thoughts, we adopt the same format, which includes the current subtask and the high-level motion, i.e. coarse direction instructions. This simple definition has proven effective in early stages of our analysis and in related work (Shi et al., 2025). A more detailed dataset description is given in the Appendix.

Training. For all approaches, we start training the VLA from the PaliGemma-2 VLM model with 3B parameters (Steiner et al., 2024), which is based on the Gemma-2 LLM (Team et al., 2024) and on the SigLIP vision encoder (Zhai et al., 2023). For action prediction, actions are tokenized using a set of 256 discrete bins, and predicted by the LLM (Kim et al., 2024). We perform full-finetuning of the model with a batch size of 32 and a learning rate of $2e-5$, using the Adam optimizer (Kingma & Ba, 2017). For the HiRobot approach, we train two distinct PaliGemma-2 models (Shi et al., 2025).

Evaluation. During evaluation, we run the agent in the environment for 100 evaluation episodes. At each episode, objects and positions are resampled randomly (but consistently among evaluation runs), making no evaluation environment exactly the same as any of the training examples. The agent is, thus, required to generalize actions to new settings in order to succeed. For each model, we evaluate 3 checkpoints taken at epochs 5, 7 and 10 during training. In general, we found no strong correlation between validation losses and performance on the task. Though, with additional training after 10 epochs, models tend to start overfitting.

Q: How does HyT performance scales with data, compared with other training paradigms?

With this experiment, we aim to verify the hypothesis that VLAs using HyT can develop intuitive 'thinking'. If the hypothesis is correct, we expect increased performance compared to standard VLAs, thanks to the knowledge internalized by training on reasoning traces.

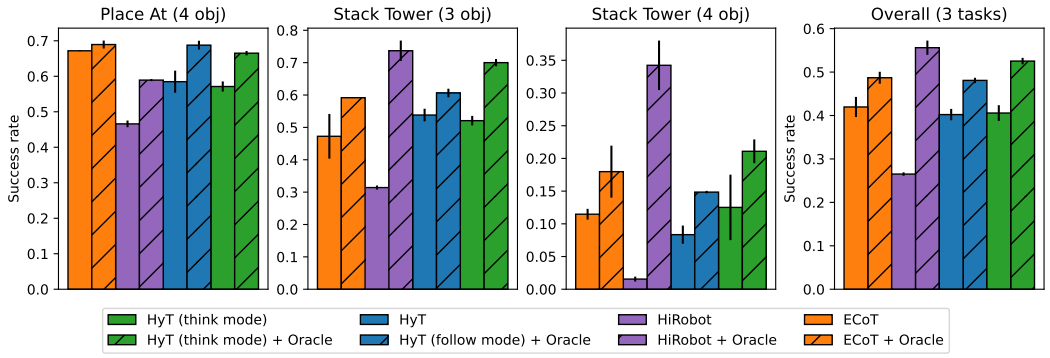


Figure 4: **Instruction-following and other inference modalities.** Comparing performance when using own generated thoughts (default) or following instructions generated by an oracle.

As imitation learning performance can greatly vary with the number of demonstrations, we evaluate models trained with different amounts of demonstrations: 300, 750, 1500, 3000. Overall performance is summarized in Figure 3 and detailed performance per task is provided in Appendix. The most evident result is that the hypothesis behind this work is confirmed: the models trained with HyT not only outperform standard VLAs, but they also generally perform better than models trained with the ECoT and HiRobot recipes, at all data scales.

Among the baselines ECoT performs (second) best at all data scales. Hierarchical VLAs have higher performance than standard VLAs with smaller amount of demos, but their performance increase is slower than for other methods, eventually being outperformed by standard VLAs from 1500 demos on. In terms of inference time, we found that standard VLAs and HyT output actions at around 3Hz on A100 GPUs (4 models acting in parallel). ECoT models are $3\times$ slower. HiRobot hierarchical generation is, overall, $4\times$ slower.

Q: How can we use the other inference modalities enabled by HyT and how do they perform?

As described in the Method section, HyT enables the VLA to be used in multiple inference modalities, other than the 'acting' mode, directly generating actions. This can be useful in situations where interpretability or the ability to follow fine-grained instructions are useful. In order to verify empirically that these modalities are actually usable, we perform a study on the instruction following capabilities of the models on a restricted set of tasks.

In order to test instruction following from external sources, we provide the agent with a set of "Oracle thoughts", which we extract using the code from the the ClevrSkills' oracle. Oracle thoughts replace the agent's thoughts only during moving subtasks, i.e. "move to location X". This is because the oracle has precise conditions for picking and placing, compared to the VLAs. A learned policy not always satisfies them, while still being successful, causing the agent to get stuck.

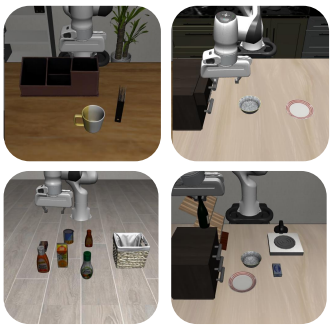
We use the models trained on the multitask dataset with the largest size (3000 demos) and present results in Figure 4 for three of the most complex tasks in the ClevrSkills benchmark adopted. HyT performance is shown in all inference modalities: 'act' mode (default, in blue), 'think' mode (in green), and 'follow' and 'think' modes with oracle thoughts (barred). First, we can observe that for all the methods tested, following instructions from an oracle, rather than self-predicted thoughts, can improve the performance of the agent. For HyT, this means that the additional inference modalities ('think' and 'follow') can actually be useful in such contexts.

Furthermore, we observe that, without oracle thoughts, models trained with HyT perform similarly in 'act' and 'think' modes, corroborating the idea that, with HyT, in the evaluated settings, intermediate thought generation at test-time may be unnecessary. Nonetheless, for future work, it would be worth verifying whether this finding holds for tasks requiring more complex embodied reasoning.

5.2 LIBERO Benchmark

The LIBERO benchmark is one of the most adopted in the VLA community (Zheng et al., 2025; Ghosh et al., 2024; Kim et al., 2024; Zhao et al., 2025; Huang et al., 2025; Pertsch et al., 2025; Lee

Figure 5: **LIBERO benchmark** Aggregated performance on 4 LIBERO task suites (examples shown on the left). Mean performance evaluated on 100 evaluation episodes.



LIBERO	Spatial	Object	Goal	Long	Avg.
Finetuned from VLA					
TraceVLA (Zheng et al., 2025)	84.6	85.2	75.1	54.1	74.8
Octo (Ghosh et al., 2024)	78.9	85.7	84.6	51.1	75.1
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5
CoT-VLA (Zhao et al., 2025)	87.5	91.6	87.6	69.0	81.1
ThinkAct (Huang et al., 2025)	88.3	91.4	87.1	70.9	84.4
π_0-FAST (Pertsch et al., 2025)	**96.4**	96.8	88.6	60.2	85.5
MolmoAct (Lee et al., 2025)	87.0	95.4	87.6	77.2	86.6
Finetuned from VLM					
VLA-OFT (Kim et al., 2025)	94.2	**97.8**	91.4	84.8	92.1
HyT (ours)	94.0	97.2	**96.2**	**89.4**	**93.7**

et al., 2025; Kim et al., 2025). The agent is evaluated on 4 suites of tasks: Spatial, Object, Goal and Long (10).

In LIBERO, oracle thoughts are not available. Thus, to extract CoT, we first use the simulator to obtain labelled object bounding boxes and high-level motion primitives, including the gripper changes. Then, we feed this information step-wise, along with the task description, to an LLM that generates a plan made of subtasks and associate subtasks with temporal steps. More information about the thoughts extraction and structure can be found in the Appendix.

Q: Can HyT be employed in combination with different VLA designs and how does performance compare to other VLAs in the literature?

Previous work on LIBERO has shown that there are some important choices in the VLA design that are crucial to obtain high performance, such as the adoption of action chunking (Lee et al., 2025; Zhao et al., 2025; Pertsch et al., 2025; Kim et al., 2025). For this reason, we chose to implement HyT in combination with the OFT fine-tuning recipe (Kim et al., 2025), which employs action chunking and continuous actions prediction (L1 head).

For both the VLA-OFT model and HyT, we fine-tune the model starting from the Prismatic VLM (Karamcheti et al., 2024). This allows us to better leverage the language pre-training of the VLM, as also shown in (Zawalski et al., 2024). Hyperparameters and training settings are the same as in (Kim et al., 2025). Results are presented in Figure 5.

Compared to other fine-tuning recipes, the OFT strategy leverages data efficiently, outperforming all other baselines in all suites of tasks but LIBERO Spatial. Training the model with HyT allows the model to increase performance even further, especially in the most complex suites of tasks: Goal and Long. This shows that HyT can be successfully combined with well-established VLA designs, potentially improving state-of-the-art performance.

We also tested fine-tuning VLA-OFT and HyT starting from the OpenVLA pre-trained VLA model. However, in this settings, we found no difference in performance (overall success: 95.3%±0.1), as both methods nearly saturate the benchmark's performance. One plausible conclusion is that HyT and other Chain-of-Thought (CoT) strategies, by leveraging additional thought data during training, can compensate for the lack of robotics pre-training or the scarcity of fine-tuning data, as simlarly shown in the other experimental results in this work.

5.3 Real-world Experiments

One of the major benefits of the HyT approach is the capability to achieve higher performance while retaining lower inference time. This is particularly useful in real-world use-cases, where faster execution is important to carry out tasks quickly and to increase the perceived quality of the agent.

For our real-world experiments, we collected a dataset comprising 320 trajectories using a robotic setup featuring an UFactory xArm 6 with a flexible two-fingered gripper, operating on a white tabletop. The agent observes the environment through RGB images captured by a RealSense D435

Table 1: **Real-world experiments.** Success rates with standard error on the real-world tasks (on the left). Additional details about experimental settings are provided in the Appendix.

	OpenVLA	HyT
In-distribution tasks	**52%** ±10	**72%** ±9
banana in green bowl	70% ±15	70% ±15
red cube in brown bag	50% ±20	100% ±0
tomato left of lettuce	60% ±22	60% ±22
zucchini in front of green cube	0% ±0	50% ±25
Out-of-distribution tasks	**29%** ±9	**54%** ±10
rubber duck in green bowl	40% ±15	20% ±13
mushroom in brown bag	0% ±0	100% ±0
purple die left of lettuce	0% ±0	50% ±25
zucchini in front of red hexagon	75% ±22	75% ±22
Overall	**41%** ±7	**63%** ±7

camera, positioned at a corner of the table. For the models, we start from the pre-trained OpenVLA model with 7B parameters (Kim et al., 2024) and compare directly to it. We perform LoRA fine-tuning with rank 32 (Hu et al., 2021), a batch size of 32 and a learning rate of $5e-4$, using the Adam optimizer (Kingma & Ba, 2017).

Our evaluation spans two categories of tasks: in-distribution and out-of-distribution. The in-distribution set includes tasks for which the dataset contains at least 10 demonstrations. For the out-of-distribution set, we modify the in-distribution tasks with alterations, such as different objects or placements, ensuring the agent encounters novel scenarios not present in the training data. See the Appendix for additional details.

The results, shown in Table 1, show that HyT overall outperforms OpenVLA, especially in out-of-distribution tasks. From a qualitative perspective, we notice that OpenVLA and HyT have similar flaws, e.g., they tend to pick objects with the wrong orientation. However, HyT tends to be more precise when reaching picking and placing positions, e.g. it never reached for the wrong object while OpenVLA did, eventually leading to a noticeable performance gap.

6 Discussion

Thinking strategies for VLAs (Zawalski et al., 2024; Shi et al., 2025) have shown important benefits in terms of performance and interpretability over standard VLAs, with the drawback of slower inference. In this work, we support the idea that the thinking process can be "internalized" by the model, developing some form of expert intuition (Kahneman & Klein, 2009). We proposed the Hybrid Training framework, which enables the possibility of learning from thoughts, for higher performance, while also being able to predict actions directly for faster inference, as empirically validated through simulated and real-world experiments. To conclude, we attempt to answer the question: *What is the contribution of CoT techniques to VLAs performance?*

From analyzing the HyT framework, our understanding is that learning to generate CoT and learning to predict actions from CoT improves the agent's understanding of the environment. This is closely related to how auxiliary losses have been used for imitation and reinforcement learning (Yarats et al., 2020; Srinivas et al., 2020), improving the learning dynamics of the model and generalization performance. We do not exclude that thoughts generation might be useful at test-time in more complex settings, but this would require evaluating on tasks that require advanced reasoning capabilities, which are currently sparse and rarely adopted in the robotics literature.

One limitation of CoT, hierarchical and HyT methods is that they require additional human labelling, providing thoughts for the agent to learn from. In this respect, one advantage of HyT approach is that it does not require the CoT to be present for the whole dataset. Instead, given that HyT relies on sampling for approximating its objective, thoughts can be sampled just for the trajectories that contain them. Automated reasoning approaches to extract useful thoughts in a self-supervised manner would be a promising direction to explore in future work (DeepSeek-AI et al., 2025).

ETHICS STATEMENT

Robotic agents that are interpretable could have a positive societal impact, as they allow users to read and/or verify the agent's intention. Intervenability can both have positive and negative social impact, as it allows users to correct the agent's intentions, but it could also allow malicious users to steer the agent's behavior towards unethical behaviors.

REPRODUCIBILITY STATEMENT

In order to ensure reproducibility, we made use of open-source simulation benchmarks (ClevrSkills, LIBERO) and open-source models (Paligemma, Prismatic, OpenVLA). In the main text and/or in the Appendix, we state all the important hyperparameters for fine-tuning the models using our approach and replicate our results. We also provide details about all the tasks, chains-of-thoughts, and models' prompts employed.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023a. URL https://arxiv.org/abs/2307.15818.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023b. URL https://arxiv.org/abs/2212.06817.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,

Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step, 2024. URL https://arxiv.org/abs/2405.14838.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills, 2023. URL https://arxiv.org/abs/2302.04659.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels, 2022. URL https://arxiv.org/abs/2206.04114.

Sanjay Haresh, Daniel Dijkman, Apratim Bhattacharyya, and Roland Memisevic. Clevrskills: Compositional language and visual reasoning in robotics, 2024. URL https://arxiv.org/abs/2411.09052.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning, 2025. URL https://arxiv.org/abs/2507.16815.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts, 2023. URL https://arxiv.org/abs/2210.03094.

Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

Daniel Kahneman and Gary Klein. Conditions for intuitive expertise: a failure to disagree. *Am Psychol*, 64(6):515–526, September 2009.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models, 2024. URL https://arxiv.org/abs/2402.07865.

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.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan

Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2024. URL https://arxiv.org/abs/2310.08864.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning, 2024. URL https://arxiv.org/abs/2403.16999.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

Herbert A Simon. What is an "explanation" of behavior? *Psychological science*, 3(3):150–161, 1992.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020. URL https://arxiv.org/abs/2004.04136.

Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. Paligemma 2: A family of versatile vlms for transfer, 2024. URL https://arxiv.org/abs/2412.03555.

DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces, 2025. URL https://arxiv.org/abs/2410.09918.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Ziteng Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal llms, 2024. URL https://arxiv.org/abs/2406.16860.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation, 2024. URL https://arxiv.org/abs/2409.12514.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images, 2020. URL https://arxiv.org/abs/1910.01741.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1, 2024. URL https://arxiv.org/abs/2407.06023.

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.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025. URL https://arxiv.org/abs/2503.22020.

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, 2025. URL https://arxiv.org/abs/2412.10345.

Minjie Zhu, Yichen Zhu, Jinming Li, Junjie Wen, Zhiyuan Xu, Zhengping Che, Chaomin Shen, Yaxin Peng, Dong Liu, Feifei Feng, and Jian Tang. Language-conditioned robotic manipulation with fast and slow thinking, 2024. URL https://arxiv.org/abs/2401.04181.

A APPENDIX

A.1 CLEVRSKILLS EXPERIMENTS

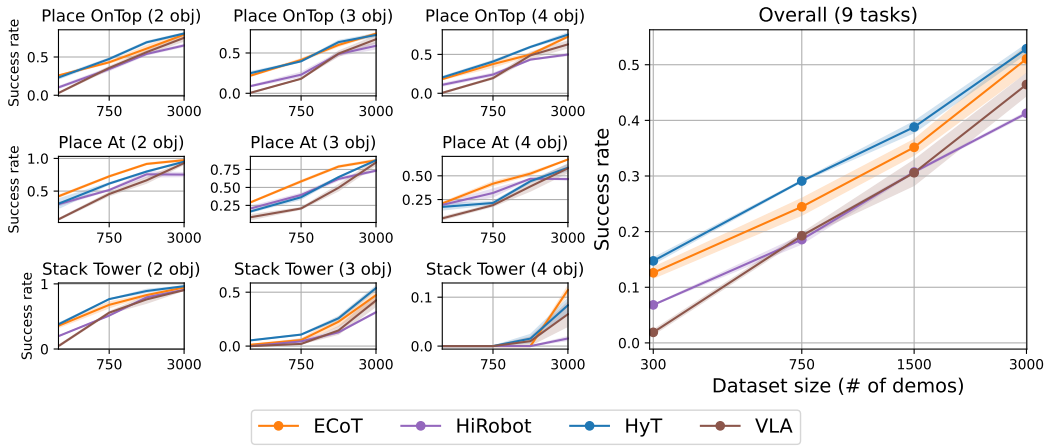


Figure 6: **Detailed results on the ClevrSkills benchmark.**

Tasks definition The tasks are defined as follows:

- *Place At* tasks evaluate the agent's spatial understanding. The tasks require the agent to bring an object to a location specified in relation to another object location, i.e. left/right/behind/in front of the object. The objects used are mainly simple-shaped objects of different colors as in (Jiang et al., 2023).
- *Place OnTop* tasks evaluate the agent's understanding of object interactions, as they require placing an object on top of another in a stable position. The set of objects used contains simple shapes, but also more complex objects from YCB, such as mugs, wooden blocks, bowls, etc.
- *Stack Tower* evaluates the long-horizon capabilities of the agent, as it requires multiple "place on top" with simple object shapes, where the order of the objects in the stack is defined in the prompt.

For each task, we train and test the agent with three variants, containing varying number of objects. In *Place OnTop* and *Place At* the number of additional objects is only distracting or increasing the amount of clutter in the scene, e.g. making placing objects on the table harder. For the *Stack Tower* tasks more objects also require more actions, as the agent should stack all the objects present in the scene.

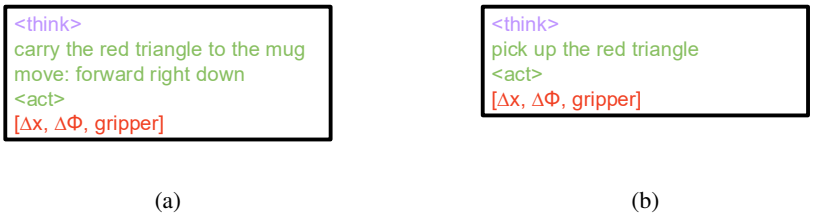
Training details. The training for all experiments and approaches is done using PyTorch DDP (Paszke et al., 2017) on 4 A100 GPUs. Inference requires less than 20GB VRAM and is performed on a multi-instance A100 for simulated environments.

Evaluation details At each episode, objects and positions are resampled randomly (but consistently among evaluation runs), making no evaluation environment exactly the same as any of the training examples. The agent is, thus, required to generalize actions to new settings in order to succeed. For each model, we evaluate 3 checkpoints taken at epochs 5,7 and 10 during training. In general, we found no strong correlation between validation losses and performance on the task. Though, with additional training after 10 epochs, models tend to start overfitting.

Dataset definition. For each task group, we collected a set of 1000 demo trajectories composed as follows: 250 demos in the 2 objects task, 500 demos in the 3 objects task, 250 demos in the 4 objects task. Then, to train agents with different sizes of the dataset, we subsample fixed subsets of trajectories from each dataset. For the multitask dataset, the task group datasets are subsampled and then aggregated.

Extracting thoughts. In ClevrSkills' demonstrations (Haresh et al., 2024) a variety of solvers is applied to the task, following a pre-specificied order - the oracle's plan. Each solver takes executes one subtask from the overall plan and the solver parameters can be recovered in the demonstrations. In order to create thoughts, we can use the ClevrSkills library to transform each subtask from the solver

into a natural language instruction, e.g. "Move to X" or "Pick up Y". Then, for moving instructions - e.g. "Move to..." and "Carry Z to..." - we extract the motion direction of the agent. Similarly to (Shi et al., 2025), this is obtained at each timestep by computing the distance between the current end-effector position and the position at the end of the motion subtask. This is then transformed into language, in the form of "left/right", "forward/backward" and "up/down" instructions, as in (Zawalski et al., 2024). Finally, for distances that are less than 1 cm the agent receives a "close" moving instruction. Two examples of thoughts and actions, in the HyT format, are provided in Figure 7.



(a) (b)

Figure 7: Examples of thoughts and actions for a moving (a) and a non-moving (b) subtasks, expressed in language format following the format used for HyT.

Gripper information. The ClevrSkills benchmark adopts a vacuum-gripper robot which creates a vacuum in the gripper's suction cups to maintain contact with objects. Compared to a fingers-based gripper, a suction gripper's state, i.e. open or closed, cannot be easily assessed through images. Thus, in order for the agent to be able to perform grasping actions reliably, we concatenate the state of the gripper in language form to the task description (or to the thoughts for the hierarchical low-level VLA). The gripper state information we pass to the agent is: (i) whether the gripper suction cups are making contact with something, (ii) whether the gripper was on, grasping something, or not. There's no need for the gripper information in the real-world experiments, as we use a fingers-based gripper, where the gripper state is visually observable.

A.2 LIBERO EXPERIMENTS

CoT extraction. In order to generate plans and subtasks we use the Gemma-2 9B model.

For plan generation we use the following prompt:

```
I want to provide a robotic arm with a short list of high-level steps to
    complete the task: '{instruction}'.
The robotic arm is operating in a scene that contains the following
    objects:
{object_1}, {object_2}, ..., {object_n}

Break down the instruction into a Python list of high-level steps. These
    steps describe the coarse primitive actions that the robot needs to
    take to complete the task.

It is important not to invent subtasks unless they are explicitly
    required by the instruction.
For example, if the task does not mention opening or closing a drawer, do
     not include those actions.
After listing the steps, please provide a brief explanation for each one
    of the subtasks.
```

Listing 1: Prompt for LLM to generate high-level robotic steps

For inferring step-wise subtasks we use the following prompt:

```
I want to analyze each step my robot took to accomplish the task: '{
    instruction}'.
To complete this task, it followed these subtasks in order:
{subtask 1}
{subtask 2}
...
{subtask n}

Below is a mapping from each timestep to the motion the robot executed.
    The final item in each entry represents the object closest to the
    gripper. This information is used solely for reasoning purposes and
    is not part of the motion itself:
{...}

Please provide a Python dictionary that maps each of the subtasks (as
    strings) to the integer timestep at which that subtask begins. The
    mapping should include only the subtasks listed above. Since the
    subtasks are sequential, later subtasks must correspond to later
    timesteps. Each subtask must be assigned a single timestep, and the
    first subtask should always begin at step 0.
```

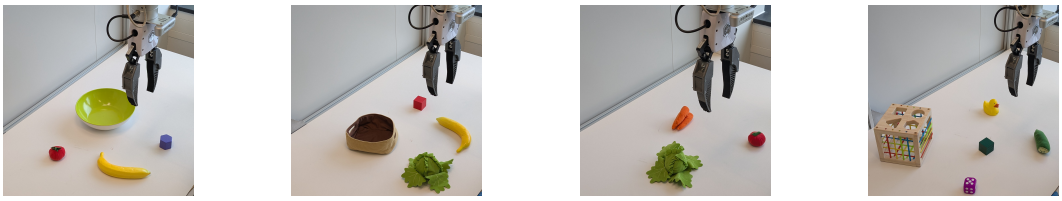
Listing 2: Prompt for LLM to associate subtasks with steps in an episode

The final CoTs look like this:

```
PLAN: locate the cabinet, position above the middle drawer, open the
    drawer
VISIBLE OBJECTS: akita_black_bowl_1: [138,139,140,150], cream_cheese_1:
    [157,147,170,166], wine_bottle_1: [104,96,116,122], plate_1:
    [99,171,141,196], wooden_cabinet_1: [0,94,81,212], flat_stove_1:
    [161,87,213,137], wine_rack_1: [13,27,95,123]
SUBTASK REASONING: the robot needs to move into the correct spatial
    location relative to the cabinet
SUBTASK: position above the middle drawer
MOVE: move left
GRIPPER POSITION: [106, 149]
```

Listing 3: CoT for LIBERO experiments

A.3 Real-world experiments



(a) (b) (c) (d)

Figure 8: The setup for some of the real-world tasks: (a) banana in green bowl (b) red cube in brown bag (c) zucchini in front of green cube (d) tomato left of lettuce.

Dataset preparation. The dataset for real-world experiments is collected via human tele-operation, using a VR set and a controller to map the human motions and intentions, e.g., closing the gripper, to the robot. After collecting the trajectories, we apply some basic pre-processing operations. First, we take the sequences collected at 60 Hz, and we subsample them at 10 Hz. Then, we extract actions in the format $[\Delta x, \Delta\phi, \text{gripper}]$ from the end-effector and gripper positions of the demonstrations. Finally, we filter out no-motion operations, i.e. having Δx and $\Delta\theta$ equal to zero.

Extracting thoughts. For extracting the subtasks to use in the thoughts, we use a simple heuristic-based approach. This is based on the assumption that all the tasks in the dataset are in the format 'place the {obj1} {position} the {obj2}', e.g. 'place the {zucchini} {in front of} the {green cube}'. We identify the position when the robot closes the gripper as the *grasping position*, and the position when the robot opens the gripper as the *releasing position*. Then, we identify the following 3 keyframe moments: (i) the moment when the agent reaches within 3 cms distance from the grasping position, (ii) the grasping moment, (iii) the moment when the agent reaches within 3 cms distance from the releasing position. Finally, the subtasks are defined in the trajectory as follows:

- *Move to the* {*obj1*}: between the start of the trajectory and keyframe (i);
- *Pick up the* {*obj1*}: between keyframe (i) and keyframe (ii);
- *Move* {*position*} *the* {*obj2*}: between keyframe (ii) and keyframe (iii);
- *Place object* {*position*} *the* {*obj2*}: between keyframe (iii) and the end of the trajectory.

Also, to have the same thought structure as in the ClevrSkills' experiments, we concatenate a 'move: {direction}' to the moving subtasks, where the direction is computed using the end-effector position (see Section A.1).

Evaluation. During evaluation, we limit the execution time to 150 agent steps, which is $\sim$ 2 minutes per trajectory, considering 3 actions/second from the model and a control loop running at $\sim$ 2Hz for stable motions. In case of failed grasps, we allow up to 3 attempts to the agent, before assessing the outcome of the episode. In the following, you find a detailed description of how each in-distribution task is executed and randomized. For out-of-distribution tasks, we follow the same procedures, but we swap one of the main actors in the scene (e.g. the object to pick or the placing target). See Figure 8) for reference. Note: fruits are stuffed toys, while vegetables are hard foam models.

- *Place the banana in the green bowl (10 trials).* The scene includes a green bowl and three objects: a banana, a tomato, and a blue hexagon, each with predefined positions. We conduct five trials with the banana starting in one location, and five more from a different location. In each location, the banana appears in three orientations: vertical (2/5 trials), horizontal (2/5 trials), and diagonal (1/5 trial).
- *Place the red cube in the brown bag. (6 trials).* The setup features a brown bag and three objects: a red cube, a banana, and a lettuce leaf, all placed at specific locations. We run two trials for each location of the red cube. Its orientation varies between being aligned with the robot's base and being tilted.
- *Place the tomato left of the lettuce (5 trials).* The scene contains three objects: a lettuce leaf, a carrot, and a tomato. Their positions are randomized within defined regions, though the overall layout remains consistent. We perform five trials with these randomized placements. A trial is only considered successful if the tomato ends up on the table, near the lettuce, and to its left.
- *Place the zucchini in front of the green cube (4 trials).* The environment includes five objects: a shape sorting box, a zucchini, a purple die, a rubber duck, and a green cube. Object positions are randomized within certain bounds, while maintaining the general layout. We carry out four trials with these randomized setups. Success is defined as the zucchini being placed on the table, close to and in front of the green cube.